Hidden Layer Training via Hessian Matrix Information

Changhua Yu, Michael T. Manry, Jiang Li

Department of Electrical Engineering
University of Texas at Arlington, TX 76019
E-mail: changhua_yu@uta.edu; manry@uta.edu; li@wcn.uta.edu

Abstract

The output weight optimization-hidden weight optimization (OWO-HWO) algorithm for training the multilayer perceptron alternately updates the output weights and the hidden weights. This layer-by-layer training strategy greatly improves convergence speed. However, in HWO, the desired net function actually evolves in the gradient direction, which inevitably reduces efficiency. In this paper, two improvements to the OWO-HWO algorithm are presented. New desired net functions are proposed for hidden layer training, which use Hessian matrix information rather than gradients. A weighted hidden layer error function, taking saturation into consideration, is derived directly from the global error function. Both techniques greatly increase training speed. Faster convergence is verified by simulations with remote sensing data sets.

Introduction

The multi-layer perceptron (MLP) is widely used in the fields of signal processing, remote sensing, and pattern recognition. Since back propagation (BP) was first proposed for MLP training (Werbos 1974), many researchers have attempted to improve its convergence speed. Techniques used to improve convergence include second order information (Battiti 1992, Mollor 1997), training network layer by layer (Wang and Chen 1996, Parisi et al. 1996), avoiding saturation (Yam and Chow 2000, Lee, Chen and Huang 2001) and adapting the learning factor (Magoulas, Vrahatis and Androulakis 1999, Nachtsheim 1994).

Algorithms like Qprop (Fahlman 1989), conjugate gradient (Fletch 1987, Kim 2003), Levenberg-Marquardt (LM) (Hagan and Menhaj 1994, Fletch 1987) often perform much better than BP. In these algorithms, the essential difference is the weight updating strategy. The convergence speeds are quite different when the weights are modified in the gradient direction, a conjugate direction or the Newton direction. As to which one is better, this depends on the nature of the application, the computational load and other factors. Generally, gradient methods perform worst. While the Newton method performs best, it requires more computation time.

Chen (Chen, Manry and Chandrasekaran 1999) 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 the training error. The algorithm modifies the hidden weights based on the minimization of the MSE between a desired and the actual net function, as originally proposed by Scalero and Tepedelenlioglu (Scalero and Tepedelenlioglu 1992). Although, OWO-HWO greatly increases the training speed, it still has room for improvement (Wang and Chen 1996) because it uses the delta function, which is just the gradient information, as the desired net function change. In addition, HWO is equivalent to BP applied to the hidden weights under certain conditions (Chen, Manry and Chandrasekaran 1999).

In this paper, a Newton-like method is used to improve hidden layer training. First, we review OWO-HWO training. Then, we propose new desired hidden layer net function changes using Hessian matrix information. Next, we derive a weighted hidden layer error function from the global training error function, which de-emphasizes error in saturated hidden units. We compare the improved training algorithm with the original OWO-HWO and LM algorithms with simulations on three remote sensing training data sets.

The OWO-HWO Algorithm

Without loss of generality, we restrict our discussion to a three layer fully connected MLP with linear output activation functions. First, we describe the network structure and our notation. Then we review the OWO-HWO algorithm for our MLP.

Fully Connected MLP Notation

The network structure is shown in Fig. 1. For clarity, the bypass weights from input layer to output layer are not shown. The training data set consists of N_v training patterns $\{(\boldsymbol{x}_p, \boldsymbol{t}_p)\}$, where the pth input vector $\boldsymbol{x}_p$ and the pth desired output vector $\boldsymbol{t}_p$ have dimensions N and M, respectively. Thresholds in the hidden and output layers are handled by letting $x_{p,(N+1)}=1$. For the jth hidden unit, the net input net_{pj} and the output activation O_{pj} for the pth training pattern are

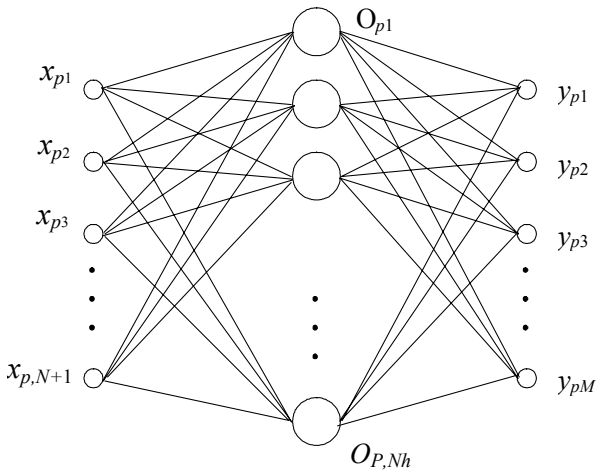


Fig. 1 The network structure

$$net_{pj} = \sum_{n=1}^{N+1} w_h(j,n) \cdot x_{pn}$$
$$O_{pj} = f\left(net_{pj}\right) \tag{1}$$

Here x_{pn} denotes the nth element of $\boldsymbol{x}_p$ and $w_h(j, n)$ denotes the weight connecting the nth input unit to the jth hidden unit. N_h is the number of hidden units. The activation function f is sigmoidal

$$f\left(net_{pj}\right) = \frac{1}{1+e^{-net_{pj}}} \tag{2}$$

The kth output y_{pk} for the pth training pattern is

$$y_{pk} = \sum_{n=1}^{N+1} w_{oi}(k,n) \cdot x_{pn} + \sum_{j=1}^{N_h} w_{oh}(k,j) \cdot O_{pj} \tag{3}$$

$w_{oi}(k, n)$ denotes the weight connecting the nth input node to the kth output unit and $w_{oh}(k, j)$ denotes the weight connecting the jth hidden unit to the kth output unit. In batch mode training, the overall performance of a feed-forward network, measured as MSE, can be written as

$$E = \frac{1}{N_v} \sum_{p=1}^{N_v} \sum_{k=1}^{M} \left[t_{pk} - y_{pk}\right]^2 \tag{4}$$

where t_{pk} denotes the kth element of the pth desired output vector.

Review of OWO-HWO

As the output units have linear activation functions in this paper, the OWO procedure can be realized by solving linear equations (Chen, Manry and Chandrasekaran 1999), which result when gradients of E with respect to the output weights, $w_{oi}(k, n)$ and $w_{oh}(k, j)$, are set to zero.

In HWO, the hidden weights $w_h(j, n)$ 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 the jth hidden unit and pth pattern, the desired net function net_{pjd} is constructed as (Chen, Manry and Chandrasekaran 1999):

$$net_{pjd} = net_{pj} + Z \cdot \Delta net_{pj} \cong net_{pj} + Z \cdot \delta_{pj} \tag{5}$$

Z is the learning rate and δ_{pj} is the jth hidden unit delta function defined as:

$$\delta_{pj} = f'\left(net_{pj}\right) \sum_k \delta_{opk} w_{oh}(k,j) \tag{6}$$

where δ_{opk} is the delta function of the kth output unit,

$$\delta_{opk} = t_{pk} - y_{pk} \tag{7}$$

The hidden weights are updated as

$$w_h(j,n) \leftarrow w_h(j,n) + Z \cdot e(j,n) \tag{8}$$

where $e(j, n)$ is a direction vector element. The weight changes are derived using

$$net_{pj} + Z \cdot \delta_{pj} \cong \sum_{n=1}^{N+1} \left[w_h(j,n) + Z \cdot e(j,n)\right] \cdot x_{pn} \tag{9}$$

Therefore,

$$\delta_{pj} \cong \sum_{n=1}^{N+1} e(j,n) \cdot x_{pn} \tag{10}$$

The error of (10) for the jth hidden unit is measured as

$$E_\delta(j) = \frac{1}{N_v} \sum_{p=1}^{N_v} \left[\delta_{pj} - \sum_{n=1}^{N+1} e(j,n) \cdot x_{pn}\right]^2 \tag{11}$$

Linear equations resulting from (11) can be solved for $e(j, n)$ by the conjugate gradient method. After finding the learning factor Z, the hidden weights are updated as in (8). After applying OWO and HWO to the network, the procedures are repeated for as many iterations as necessary.

Improved Hidden Layer Training

In the original HWO algorithm, the desired hidden layer net function change $Z \cdot \Delta net_{pj}$ uses the gradient, i.e. the delta function, which inevitably slows down the optimization procedure due to its steepest descent direction. Here, we use a Newton-like method to derive a new desired net function, which exploits second order information from a Hessian matrix.

The New Desired Net Function

As we know, when E is minimized by the optimal value of the jth hidden layer net function net^*_{pj}, we have:

$$\frac{\partial E}{\partial net^*_{pj}} = 0 \tag{12}$$

As in the derivation of Newton-like methods, we have

$$\frac{\partial E}{\partial net^*_{pj}} \approx \frac{\partial E}{\partial net_{pj}} + \sum_{k=1}^{N_h} \frac{\partial^2 E}{\partial net_{pj} \partial net_{pk}} \cdot \left(net^*_{pj} - net_{pj}\right) \tag{13}$$

Now, the new direction of desired net function change with element $\Delta net^*_{pj} = net^*_{pj} - net_{pj}$ can be found by:

$$\left[\Delta net_{p1}^{*} \quad \cdots \quad \Delta net_{pj}^{*} \quad \cdots \quad \Delta net_{pN_h}^{*}\right]^T = -\mathbf{H}^{-1}\cdot\mathbf{g}$$

$$= -\begin{bmatrix} \frac{\partial^2 E}{\partial net_{p1}^2} & \cdots\cdots & \frac{\partial^2 E}{\partial net_{p1}\partial net_{pN_h}} \\ \vdots & \cdots\cdots & \vdots \\ \frac{\partial^2 E}{\partial net_{pj}\partial net_{p1}} & \cdots\cdots & \frac{\partial^2 E}{\partial net_{pj}\partial net_{pN_h}} \\ \vdots & \cdots\cdots & \vdots \\ \frac{\partial^2 E}{\partial net_{pN_h}\partial net_{p1}} & \cdots\cdots & \frac{\partial^2 E}{\partial net_{pN_h}^2} \end{bmatrix}^{-1} \cdot \begin{bmatrix} \frac{\partial E}{\partial net_{p1}} \\ \vdots \\ \frac{\partial E}{\partial net_{pj}} \\ \vdots \\ \frac{\partial E}{\partial net_{pN_h}} \end{bmatrix} \tag{14}$$

The elements of the Hessian matrix **H** and the gradient matrix **g** are calculated through the chain rule as:

$$g(i) = \frac{(-2)}{N_v} f'_{pi} \sum_{k=1}^{M} \left[t_{pk} - y_{pk}\right] \cdot w_{oh}(k,i) = -\frac{2}{N_v}\delta_{pi} \tag{15}$$

$$h(i,j) = \frac{\partial^2 E}{\partial net_{pi}\partial net_{pj}} = \frac{2}{N_v} f'_{pi} f'_{pj} \sum_{k=1}^{M} w_{oh}(k,i) w_{oh}(k,j) \tag{16}$$

$$h(i,i) = \frac{\partial^2 E}{\partial net_{pi}^2} = \left\{\frac{2}{N_v}\left(f'_{pi}\right)^2 \sum_{k=1}^{M} w_{oh}^2(k,i)\right\} + g(i)\cdot\frac{f''_{pi}}{f'_{pi}} \tag{17}$$

Here, f_{pi} is shorthand for $f\left(net_{pi}\right)$ as described in (2), so f'_{pi} and f''_{pi} are the first and second derivatives respectively. In the algorithms that update hidden weights directly by Newton methods, the main trouble is the large dimension of the Hessian matrix. For example, for the three-layer MLP in fig. 1, the Hessian matrix size will be $\left((N+1)\cdot N_h\right)\times\left((N+1)\cdot N_h\right)$. The dimension of **H** in (14) is much smaller, only $N_h \times N_h$ since the hidden unit net functions are treated as weights to be changed. However, calculating its inverse is still not trivial. Both explicit computation (Bishop 1992) and alternative methods (Moller 1993) have been presented in the literature. However, these methods are still computationally intensive for large networks, and it is not certain whether the extra computation speeds up convergence (Magoulas, Vrahatis and Androulakis 1999). Therefore, we just use the diagonal elements of **H** in equation (14) to calculate the desired net function change, yielding

$$\Delta net_{pi}^{*} = \frac{\delta_{pi}}{\left(f'_{pi}\right)^2 \sum_{k=1}^{M} w_{oh}^2(k,i) + \delta_{pi}\cdot\left(\frac{f''_{pi}}{f'_{pi}}\right)} \tag{18}$$

The use of Δnet_{pi}^{*} is detailed in the following subsection.

A Weighted Hidden Layer Error Function

In the following, we derive a new hidden layer error function to replace $E_\delta(j)$ in (11) directly from the global MSE. Thus, hidden layer optimization really decreases global errors. In addition, this new error function takes hidden unit saturation into consideration.

If we substitute equations (3) into (4), we can rewrite the total MSE as

$$E = \frac{1}{N_v}\sum_{p}\sum_{k=1}^{M}\left[t_{pk} - \sum_{j=1}^{N_h} w_{oh}(k,j) O_{pj} - \sum_{n=1}^{N+1} w_{oi}(k,n) x_{pn}\right]^2 \tag{19}$$

During the HWO procedure, the desired net function change is $Z\cdot\Delta net_{pj}^{*}$, while the actual change is $Z\cdot\sum_{n=1}^{N+1} e(j,n)x_{pn}$ as in (9). Here Z is the learning factor. So the corresponding desired hidden unit output O_{pjd} and the actual one $\overline{O}_{pj}$ could be approximated by using Taylor series as

$$O_{pjd} = f\left(net_{pjd}\right) \approx O_{pj} + Z\cdot f'_{pj}\cdot\Delta net_{pj}^{*} \tag{20}$$

$$\overline{O}_{pj} = f\left(\overline{net}_{pj}\right) \approx O_{pj} + Z\cdot f'_{pj}\cdot\sum_{n=1}^{N+1} e(j,n)x_{pn} \tag{21}$$

We denote the kth output caused by the inputs and O_{pjd} as:

$$T_{pk} = \sum_{n=1}^{N+1} w_{oi}(k,n)\cdot x_{pn} + \sum_{j=1}^{N_h} w_{oh}(k,j)\cdot O_{pjd} \tag{22}$$

After HWO, then the actual total error can be rewritten as:

$$E = \frac{1}{N_v}\sum_{p}\sum_{k=1}^{M}\left[t_{pk} - T_{pk} + T_{pk} - \sum_{n=1}^{N+1} w_{oi}(k,n)x_{pn} - \sum_{j=1}^{N_h} w_{oh}(k.j)\overline{O}_{pj}\right]^2$$

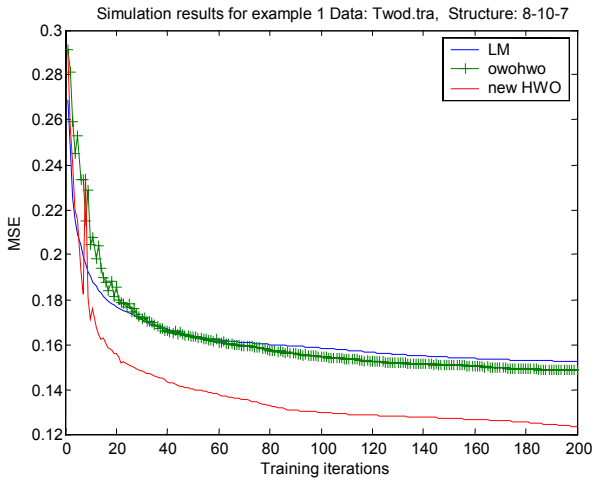
$$= \frac{1}{N_v}\sum_{p}\sum_{k=1}^{M}\left\{\left[t_{pk} - T_{pk}\right] + \sum_{j=1}^{N_h} w_{oh}(k,j)\left[O_{pjd} - \overline{O}_{pj}\right]\right\}^2 \tag{23}$$

If we assume that $[t_{pk}\text{-}T_{pk}]$ is uncorrelated with $\left[\Delta net_{pj}^{*} - \sum_{n} e(j,n)x_{pn}\right]$, and use equations (20) and (21), equation (23) becomes:

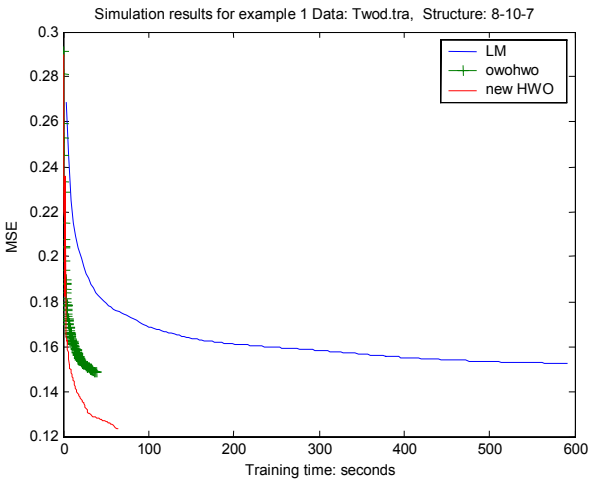
$$E \approx \frac{1}{N_v}\sum_{p}\sum_{k=1}^{M}\left[t_{pk} - T_{pk}\right]^2 + \frac{1}{N_v}\sum_{p}\sum_{k=1}^{M}\left\{\sum_{j=1}^{N_h} w_{oh}(k,j) f'_{pj} Z\left[\Delta net_{pj}^{*} - \sum_{n=1}^{N+1} e(j,n)x_{pn}\right]\right\}^2 \tag{24}$$

We can evaluate (24) further because it is reasonable to assume that the values $\left\{w_{oh}(k,j)f'_{pj}\left[\Delta net_{pj}^{*} - \sum_{n} e(j,n)x_{pn}\right]\right\}$ associated with different hidden units are uncorrelated with each other. This assumption yields

$$E \approx \frac{1}{N_v}\sum_{p}\sum_{k=1}^{M}\left[t_{pk} - T_{pk}\right]^2 + \frac{1}{N_v}\sum_{p}\sum_{k=1}^{M}\sum_{j=1}^{N_h} w_{oh}^2(k,j) Z^2\left(f'_{pj}\right)^2\left[\Delta net_{pj}^{*} - \sum_{n=1}^{N+1} e(j,n)x_{pn}\right]^2 \tag{25}$$



(a)



(b)

Fig. 2 Results for *Twod.tra*, Structure: 8-10-7

Since Z, $[t_{pk}\text{-}T_{pk}]$ and $w_{oh}(k,j)$ are constant during the HWO procedure, minimizing E in the current iteration is equivalent to minimizing

$$E_\delta(k)=\frac{1}{N_v}\sum_p\left(f'_{pj}\right)^2\left[\Delta net^*_{pj}-\sum_{n=1}^{N+1}e(j,n)x_{pn}\right]^2 \qquad (26)$$

with respect to $e(j, n)$. This hidden layer error function successfully de-emphasizes error in saturated hidden units by inserting the square of the derivative of the activation function. When net_{pj} is in the sigmoid's linear region, errors between net_{pj} and net_{pjd} receive large weight in (26).

Simulation

The proposed new OWO-HWO algorithm was verified using three remote sensing training data sets. Its performance was compared to the original OWO-HWO and the LM algorithm. Our simulations were carried out on a 733Mhz Pentium III, Windows 2000 using the Visual C++ 6.0 compiler.

Training data set *Twod.tra* 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.

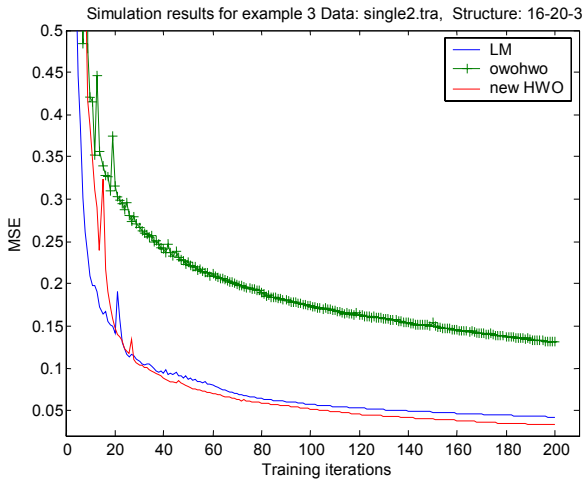
This data set has 8 inputs, 7 outputs, and 1768 patterns. We use 10 hidden units in a three-layer MLP. All the algorithms are trained for 200 iterations. From the simulation results of figure 2, the new algorithm has the fastest convergence speed while LM is the slowest for this data, in terms of both iteration number and time.

Single2.tra consists of 16 inputs, 3 outputs and 5992 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 first 16 inputs represent the simulated back scattering coefficient measured at 10, 30, 50 and 70 degrees at both vertical and horizontal polarization. The remaining 8 are various combinations of ratios of the original eight values. These ratios correspond to those used in several empirical retrieval algorithms (Fung, Li and Chen 1992).

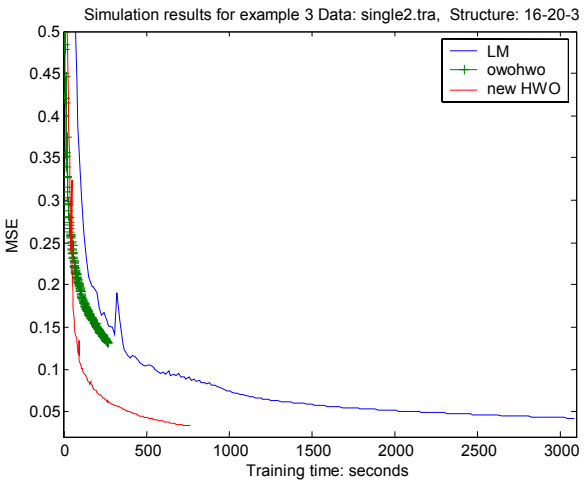
We chose the MLP structure 16-20-3 and trained the network for 200 iterations for all algorithms. From the simulation results of figure 3, the new algorithm performs much better than the original OWO-HWO and almost the same as LM in terms of iterations. In terms of MSE versus time, the new algorithm is still much better than LM. The original OWO-HWO is also better than LM in terms of time although worse in terms of iterations.

Inputs for training data set *OH7.TRA* (Oh, Sarabandi and Ulaby, 1992) are VV and HH polarization at L 30, 40 deg, C 10, 30, 40, 50, 60 deg, and X 30, 40, 50 deg. The corresponding desired outputs are $\Theta=\{\sigma,l,m_v\}^T$, 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, 15,000 training patterns. We used a 20-20-3 network and trained the network for 200 iterations for all three algorithms. The simulation results of figure 4 show that the advantage of the new algorithm over the other two is overwhelming and that the original OWO-HWO is also better than LM. We also show the testing results in figure 5 for this data set. The minimum testing error of the original OWO-HWO is 1.537589, which is not shown here for clarity purpose.

From the simulation results, it is obvious that the new OWO-HWO algorithm has the fastest convergence speed.

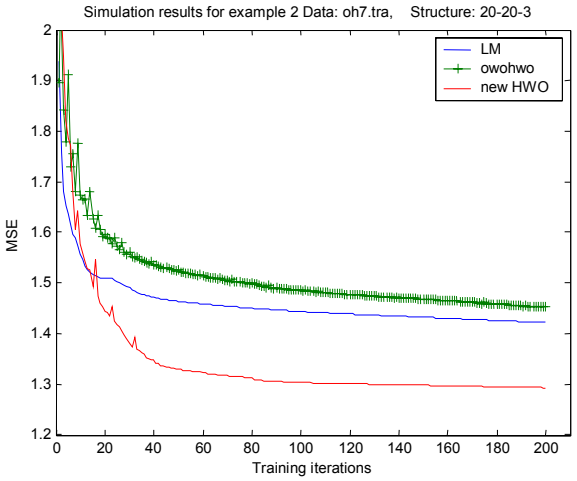


(a)

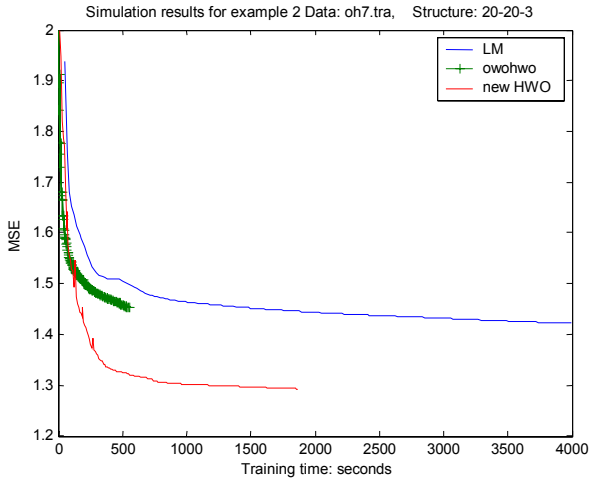


(b)

Fig. 3 Results for *Single2.tra*, Structure: 16-20-3



(a)



(b)

Fig. 4 Results for *oh7.tra*, Structure: 20-20-3

The LM algorithm requires much longer training time due to its computational complexity.

Conclusion

In the proposed algorithm, second order information from the Hessian matrix is used to find the desired hidden layer net function changes, thereby, ensuring better hidden layer training. The new weighted hidden layer error function $E_\delta(j)$ relates hidden weight optimization to the global error function. Training speed is improved because hidden unit saturation is taken into consideration.

In experiments with remote sensing data sets, the new algorithm has much faster training speed than the original OWO-HWO and the LM algorithms. In general, the ratio of testing error to training error is similar to that seen with other training algorithms. Several interesting issues still under investigation include efficient calculation of the

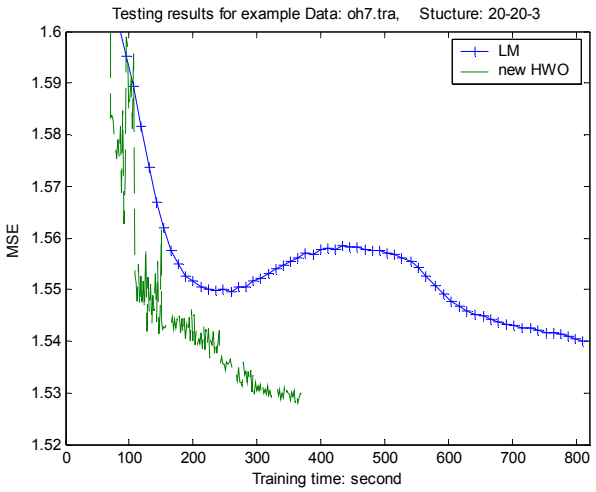


Fig. 5 Testing results for *Oh7.tra*, Structure: 20-20-3

complete Hessian matrix **H**, and more theoretical analysis of the new hidden layer error function.

Acknowledgement

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

References

Battiti, R. 1992. First- and Second-order Methods for Learning: between Steepest Descent and Newton's Method. *Neural Computation* 4 (2): 141-166.

Bishop, C. 1992. Exact Calculation of the Hessian Matrix for the Multiplayer Perceptron. *Neural Computation* 4 (4): 494-501.

Chen, H. H.; Manry, M. T.; and Chandrasekaran, H. 1999. A Neural Network Training Algorithm Utilizing Multiple Sets of Linear Equations. *Neurocomputing* 25: 55-72.

Fahlman, S. E. 1989. Faster-learning Variations on Backpropagation: An Empirical Study. In *Proceedings of 1988 Connectionist Models Summer School*, 38-51. San Mateo, Calif.: Morgan Kaufmann.

Fletch, R. 1987. *Practical Methods of Optimization*. John Wiley & Sons.

Fung, A.K.; Li, Z.; and Chen, K.S. 1992. Back Scattering from a Randomly Rough Dielectric Surface. *IEEE Transactions on Geoscience and Remote Sensing* 30 (2): 356-369.

Hagan, M. T., and Menhaj, M. B. 1994. Training Feed-forward Networks with the Marquardt Algorithm. *IEEE Transaction on Neural Networks* 5 (6): 989-993.

Kim, T. H.; Manry, M. T.; and Maldonado, F. J. 2003. New Learning Factor and Testing Methods for Conjugate Gradient Training Algorithm. In *Proceedings of 2003 International Joint Conference on Neural Networks*, 2011-2016. Portland, OR: International Joint Conference on Neural Networks.

Lee, H. M.; Chen, C. M.; and Huang, T. C. 2001. Learning Efficiency Improvement of Back-propagation Algorithm by Error Saturation Prevention Method. *Neurocomputing* 41: 125-143.

Magoulas, G. D.; Vrahatis, M. N.; and Androulakis, G. S. 1999. Improving the Convergence of the Backpropagation Algorithm Using Learning Adaptation Methods. *Neural Computation* 11: 1769-1796.

Moller, M. 1993. A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning. *Neural Networks* 6 (4): 525-533.

Moller, M. 1997. Efficient Training of Feed-forward Neural Networks. Ph.D. diss., Dept. of Computer Science, Aarhus University, Denmark.

Nachtsheim, P. R. 1994. A First Order Adaptive Learning Rate Algorithm for Back Propagation Networks. *IEEE World Congress on Computational Intelligence*: 257 -262.

Oh, Y.; Sarabandi, K.; and Ulaby, F.T. 1992. An Empirical Model and an Inversion Technique for Radar Scattering from Bare Soil Surfaces. *IEEE Transactions on Geoscience and Remote Sensing* 30 (2): 370-381.

Parisi, R., et al. 1996. A Generalized Learning Paradigm Exploiting the Structure of Feedforward Neural Networks. *IEEE Transactions on Neural Networks* 7 (6): 1450-1460.

Scalero, R. S., and Tepedelenlioglu, N. 1992. A Fast New Algorithm for Training Feedforward Neural Networks. *IEEE Transactions on Signal Processing* 40 (1): 202-210.

Wang, G. J., and Chen C. C. 1996. A Fast Multilayer Neural-network Training Algorithm Based on the Layer-by-layer Optimizing Procedures. *IEEE Transactions on Neural Networks* 7 (3): 768-775.

Werbos, P.J. 1974. Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Science. Ph. D. diss., Harvard University, Cambridge, Mass.

Yam, Y. F., and Chow, W. S. 2000. A Weight Initialization Method for Improving Training Speed in Feedforward Neural Network. *Neurocomputing* 30: 219-232.