

# NEW LEARNING FACTOR AND TESTING METHODS FOR CONJUGATE GRADIENT TRAINING ALGORITHM

Tae Kim<sup>1</sup>, Michael T. Manry<sup>1</sup>, and Javier Maldonado<sup>2</sup>

<sup>1</sup>Department of Electrical Engineering

University of Texas at Arlington

Email; [tkim@ieee.org](mailto:tkim@ieee.org), [manry@uta.edu](mailto:manry@uta.edu)

<sup>2</sup>Chihuahua Institute of Technology, Av. Tecnológico 2909, Chihuahua, Chih., México, 31310

## ABSTRACT

The conjugate gradient method has advantages over backpropagation in the training of artificial neural networks. Unlike previous investigators who have obtained learning factors using computationally expensive iterative line searches, we obtain the optimal learning factor in one step. We validate the learning factor with several tests, and analyze the input bias problem. Examples confirm the usefulness of improved conjugate gradient.

## 1. INTRODUCTION

Many people have used conjugate gradient (CG) artificial neural network (ANN) training because of its advantages as a second-order optimization method. They applied it to applications such as temporal differences learning, [11] image processing, [12, 16] optimal modeling, [13], agricultural prediction, [14] and power system control [15]. In CG, accurate calculation of the learning factor is critical for the conjugacy of direction vectors, and conjugacy is critical for finding the error function minimum [2, 3].

The learning factor, which exactly minimizes the error function in a given iteration, is the optimal learning factor (OLF). Investigators have used the line search technique [7 – 10] and other methods [17, 18] to obtain approximate OLFs. Unfortunately, these methods are computationally expensive, and it is not clear when the learning factors produced are accurate.

In this paper, we find a closed form expression for the OLF in the multilayer perceptron (MLP) and methods for validating it. In section 2, we review the structure and notation used in this paper. A brief summary of the CG algorithm is given in section 3. A Taylor series expansion of the error function is used to develop our OLF approach in section 4. Tests for validating OLF methods are developed in section 5. Examples illustrating the tests are also given.

## 2. STRUCTURE AND NOTATION

In this section, we discuss the structure and notation of the three layered MLP used for our research. In Fig. 2.1, the MLP is shown. The subscript  $p$  denotes the pattern number in the training data. Thus  $x_{pi}$  represents the  $i$ th input element of the  $p$ th training patterns, where  $1 \leq i \leq N$ .  $x_{pN+1}$  which has a value of 1 allows weights to replace thresholds in the hidden and output layer.

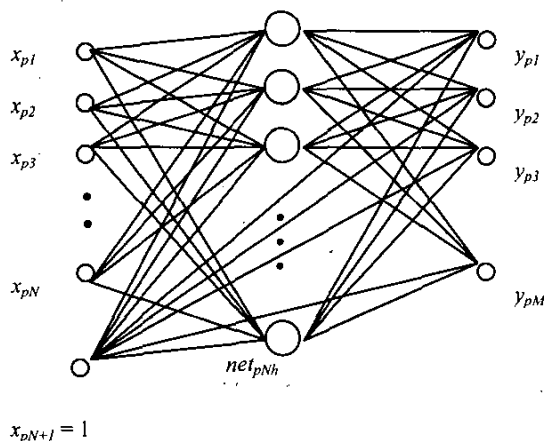


Figure 2.1. Multilayer Perceptron Structure

$N$ ,  $N_h$ , and  $M$  are the numbers of input, hidden, and output nodes respectively.  $y_{pk}$  denotes the output of the  $k$ th node in the output layer, and  $net_{pj}$  denotes the  $j$ th hidden node net function, all for the  $p$ th training pattern.

Except for the input nodes, multiple weighted terms are input to nodes. These combined terms, called net functions, are expressed in Eq.s (2.1) and (2.2). Here,  $w(j,i)$  represents the weight connecting the  $i$ th input unit to the  $j$ th hidden unit and  $w_o(k,i)$  is the weight connecting the  $i$ th input to the  $k$ th output unit and  $w_{oh}(k,j)$  connects the  $j$ th hidden unit to the  $k$ th output unit.  $w(j,N+1)$  represents the threshold to the  $j$ th hidden unit and  $w_o(k,N+1)$  represents the threshold to the  $k$ th output unit. For the hidden units and output units respectively,

$$net_{pj} = \sum_{i=1}^{N+1} w(j,i) \cdot x_{pi} \quad (2.1)$$

$$y_{pk} = \sum_{i=1}^{N+1} w_o(k,i) \cdot x_{pi} + \sum_{j=1}^{N_h} w_{oh}(k,j) \cdot f(net_{pj}) \quad (2.2)$$

Nodes other than hidden ones have linear activation functions. The sigmoid function is used in hidden units. Outputs are calculated using inputs, hidden unit outputs and

thresholds. Thus the number of contributions is  $N+N_h+1$ . Since output nodes don't have activation functions, it is not necessary to express  $y_{pk}$  using the *net* symbol.

### 3. CONJUGATE GRADIENT TRAINING

CG is an unconstrained optimization technique used to minimize the nonnegative error function  $E(\mathbf{w})$  of an ANN with respect to  $\mathbf{w}$  which represents all the networks weights and thresholds [1, 2]. Let  $N_w$  be the number of elements in the weight vector  $\mathbf{w}$ . In CG, we use the gradient vector  $\mathbf{g} = (g_1, g_2, \dots, g_{N_w})'$  which is  $\mathbf{g} = \partial E / \partial \mathbf{w}$  and the direction vector  $\mathbf{p} = (p_1, p_2, \dots, p_{N_w})'$  to obtain the weights for the ANN. In the first iteration,  $\mathbf{p}$  equals  $-\mathbf{g}$ . Then for next  $N_w-1$  iterations,  $\mathbf{p}$  is found as

$$\mathbf{p} \leftarrow -\mathbf{g} + B_1 \cdot \mathbf{p} \quad (3.1)$$

where

$$B_1 = \frac{E_g(i)}{E_g(i-1)} \quad (3.2)$$

and where  $E_g(i)$  and  $E_g(i-1)$  denote the gradient vectors' energies in the current and previous iterations respectively. In each iteration, the learning factor  $Z$  is found by solving

$$\frac{dE(\mathbf{w} + Z \cdot \mathbf{p})}{dZ} = 0 \quad (3.3)$$

for  $Z_{OLF}$ . Then the weights are updated as

$$\mathbf{w} \leftarrow \mathbf{w} + Z_{OLF} \cdot \mathbf{p} \quad (3.4)$$

### 4. OPTIMAL LEARNING FACTOR FOR CG

There are many ways to obtain the learning factor  $Z_{OLF}$ . A common one is to use the line search method. Using a Taylor series expansion of  $E(\mathbf{w}+Z \cdot \mathbf{p})$ , a third degree expansion of  $E(Z)$  is

$$E(\mathbf{w} + Z \cdot \mathbf{p}) \cong A_0 + A_1 \cdot Z + A_2 \cdot Z^2 + A_3 \cdot Z^3 \quad (4.1)$$

where

$$A_j = \frac{1}{j!} \cdot \frac{\partial^j E}{\partial Z^j} \Big|_{Z=0} \quad (4.2)$$

Depending on whether the second degree or the third degree approximations of the error are used,  $Z_{OLF}$  can be obtained using Eq. (3.3) either as

$$Z_{OLF} = \frac{-A_1}{2 \cdot A_2} \quad (4.3)$$

or

$$Z_{OLF} = \frac{-A_2 \pm \sqrt{A_2^2 - 3 \cdot A_1 \cdot A_3}}{3 \cdot A_3} \quad (4.4)$$

Two possible solutions are available from Eq. (4.4). From the second derivative test,  $\partial^2 E(Z) / \partial Z^2$  should be greater than zero when  $E(Z)$  is at the local minimum. The second derivative of  $E(Z)$  at  $Z_s$ , which are the solution points of Eq. (3.3) is expressed as

$$\frac{\partial^2 E}{\partial Z^2} \Big|_{Z=Z_s} \cong 2A_2 + 6A_3 \cdot Z = \pm 2\sqrt{A_2^2 - 2 \cdot A_1 \cdot A_3} \quad (4.5)$$

Since only the positive part of Eq. (4.5) can be used, the resulting solution for optimal  $Z$  is

$$Z_{OLF} = \frac{-A_2 + \sqrt{A_2^2 - 3 \cdot A_1 \cdot A_3}}{3 \cdot A_3} \quad (4.6)$$

Whenever the argument of the square root in Eq. (4.6) is negative, we use the solution from Eq. (4.3). Since the correctly obtained direction vector should point toward the trough of the error function, the optimal learning factor should be greater than zero. When the solution from either Eq.s (4.3) or (4.6) turns out to be negative, we use  $-\mathbf{g}$  as a new direction vector and obtain the optimal learning factor again from Eq.s (4.3) or (4.6). The optimal learning factor of Eq. (4.6) is more accurate but it is computationally more expensive.

Traditional line search or curve fitting methods start with a user given step size  $\varepsilon$  and look for three points  $Z_n, Z_{n+1}$ , and  $Z_{n+2}$  defined as

$$E(Z_n) = E(\mathbf{w} + n \cdot \varepsilon \cdot \mathbf{p}) \quad (4.7)$$

which satisfy

$$E(Z_n) \geq E(Z_{n+1}) \text{ and } E(Z_{n+2}) \geq E(Z_{n+1}) \quad (4.8)$$

Then a parabolic function is fit through the points  $Z_n, Z_{n+1}$ , and  $Z_{n+2}$  and the minimum  $Z_{n+4}$  of the parabolic function is obtained. Among the points  $Z_n, Z_{n+1}, Z_{n+2}$ , and  $Z_{n+4}$ , a new set of three points satisfying Eq. (4.8) is found. From the new set, a second and subsequent parabolic fits are performed. This process is continued until a point close to the minimum of  $E(Z)$  is found [1, 2].

From Eq. (4.7) and (4.8), we see that our Taylor's Series approach has two advantages. First, it requires only one pass through the training data, in contrast to the curve fitting approach, which requires many passes. Second, our approach doesn't require a step size  $\varepsilon$  to be chosen.

In order to obtain the optimal learning factor  $Z_{OLF}$ , the error function needs to be expressed in terms of all the weights, weights change vector elements, and the learning factor  $Z$  as in the following. These extended forms are used for the derivation of  $Z_{OLF}$ . Let  $e(j,i)$  denote the change in weight  $w(j,i)$ . Similarly,  $e_o(k,i)$  denotes the change in weight  $w_o(k,i)$ ,  $Net_{pj}$  the change in  $net_{pj}$ ,  $e_{oh}(k,j)$  the change in weight  $w_{oh}(k,j)$ , and  $f$  represents the activation function.

The error function  $E$  may be written in terms of the learning factor as

$$E(Z) = \frac{1}{N_v} \sum_{p=1}^{N_v} \sum_{k=1}^M [t_{pk} - y_{pk}]^2 \quad (4.9)$$

$$y_{pk} = \sum_{i=1}^{N+1} [w_o(k,i) + Z \cdot e_o(k,i)] \cdot x_{pi}$$

$$+ \sum_{j=1}^{N_h} [w_{oh}(k, j) + Z \cdot e_{oh}(k, j)] \cdot f(\text{net}_{pj} + Z \cdot \text{Net}_{pj}) \quad (4.10)$$

where

$$\begin{aligned} \text{net}_{pj} &= \sum_{i=1}^{N+1} w(j, i) \cdot x_{pi} \\ \text{Net}_{pj} &= \sum_{i=1}^{N+1} e(j, i) \cdot x_{pi} \end{aligned} \quad (4.11)$$

Utilizing either Eq. (4.3) or (4.6), we can obtain  $Z_{OLF}$ . Further details in the calculations of coefficients  $A_j$  are given in the Appendix.

### 5. LEARNING FACTOR TESTS

Several problems can lead to the failure of Eq. (3.3) to be true. First, the approximation in Eq. (4.1) can be false. Second, there can be round off errors in calculating the coefficients of Eq. (4.1). Third, the derivations of these coefficients can be in error. The following tests can be utilized to validate the learning factor:

#### ORTHOGONALITY TEST

In CG, the gradient vector  $\mathbf{g}$  must be orthogonal to the previous direction vector  $\mathbf{p}$  when we use the optimal learning factor. Thus the cosine of the angle  $\theta$  between  $\mathbf{p}$  and  $\mathbf{g}$ , should be near zero when we use the good OLF. The cosine of  $\theta$  is expressed as:

$$\cos \theta = \frac{\mathbf{p} \cdot \mathbf{g}}{|\mathbf{p}| \cdot |\mathbf{g}|} = \frac{\sum_{i=1}^{N_s} p_i \cdot g_i}{\sqrt{\sum_{i=1}^{N_s} p_i^2} \cdot \sqrt{\sum_{i=1}^{N_s} g_i^2}} \quad (5.1)$$

In Fig. (5.1), we plot  $\cos \theta$  when  $\mathbf{p}$  is obtained using our OLF. The training data RV400.tra consists of 400 random patterns having 8 inputs and 1 output. The MLP has 4 hidden units. We see that  $\cos \theta$  is confined to less than 0.015.

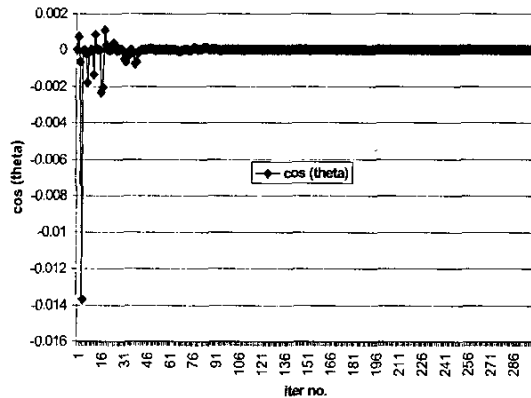


Figure 5.1.  $\cos(\theta)$  values for CG training on RV400.tra.

#### ERROR PREDICTION TEST

Using Eq. (4.1) in section 4, we can predict the error after the weight update. If the predicted error and the actual error significantly disagree, we see that the OLF is in error.  $E_{pre}$  represents the predicted error. For the second degree and third degree series respectively,

$$\begin{aligned} E_{pre} &= A_0 + A_1 \cdot Z_{OLF} + A_2 \cdot Z_{OLF}^2 \\ E_{pre} &= A_0 + A_1 \cdot Z_{OLF} + A_2 \cdot Z_{OLF}^2 + A_3 \cdot Z_{OLF}^3 \end{aligned} \quad (5.2)$$

where  $A_j$  is expressed in Eq. (4.2). Let the relative error  $R_e$  be defined as

$$R_e = \frac{|E_{pre} - E|}{|E|} \quad (5.3)$$

Good OLFs corresponds to  $R_e \leq 0.01$ . In the following, we ran CG for a network having 8 inputs, 4 hidden units, and 1 output. The training data RV400.tra again consisted of 400 random patterns. We see that  $R_e$  is less than  $2.5E-03$ .

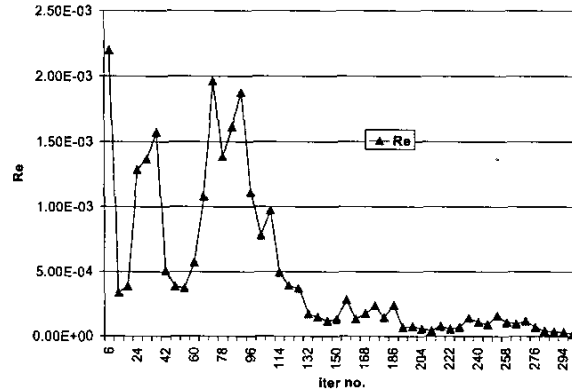


Figure 5.2. Relative error for CG training on RV400.tra.

#### DOUBLE LEARNING FACTOR TEST

When  $E(Z)$  is approximated quadratically, the error  $E$  at twice the value of the OLF is very close to the error before the update. Quadratic approximation of  $E(Z)$  is illustrated in Fig. 5.3.

From Fig. 5.3, we see why  $E(0) \approx E(2 \cdot Z_{OLF})$ . When the quadratic approximation of the error is correct, the training error should remain constant when we use  $2 \cdot Z_{OLF}$  as a learning factor. When we use  $Z_{OLF}$  as a learning factor the error should decrease.

In Fig. 5.4, we show the actual running of CG on the network and training data used in tests 1 and 2. As predicted, we see that the training error remains fairly

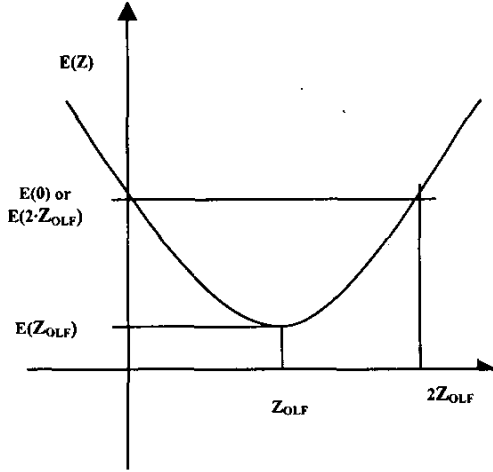


Figure 5.3. Quadratic approximation of the error function in terms of the learning factor

constant when we use  $2 \cdot Z_{OLF}$  as a learning factor. When we use  $Z_{OLF}$  as a learning factor, the error is successfully decreased.

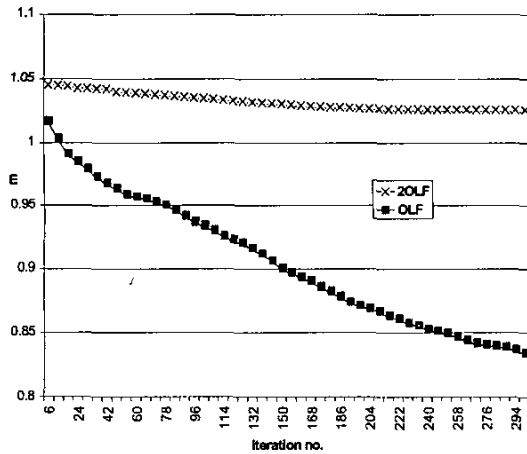


Figure 5.4. Double learning factor test applied to real data

Using the above three learning factor tests, we found out that our OLF works fine. We can apply these learning factor testing methods to other learning factor methods [7 – 10, 17, 18] to see how good their learning factors are.

## 6. THE INPUT BIAS PROBLEM.

It is desirable that MLP training algorithms be invariant to input biases. In this section, however, we show that gradient based training algorithms, including CG, are sensitive to input biases.

Assume that biased and unbiased training data sets  $\{x_p\}$  and  $\{z_p, t_p\}$  respectively are available, where

$$x_p = z_p + m \quad (6.1)$$

Here, the  $N$  - dimensional mean vector  $m$  is approximately

$$m = \frac{1}{N_v} \sum_{p=1}^{N_v} x_p \quad (6.2)$$

Assume that the initial neural nets for the two sets have identical weights. Also assume that hidden layer and output layer thresholds are adjusted so that the networks have identical hidden layer net function and output values for all patterns. During training, the two neural nets have output and hidden unit delta functions calculated as [2]

$$\delta_{po}(k) = 2 \cdot (t_{pk} - y_{pk}) \quad (6.3)$$

$$\delta_{ph}(j) = O_j \cdot (1 - O_j) \cdot \sum_{k=1}^M \delta_{po}(k) \cdot w_o(k, N + j + 1) \quad (6.4)$$

These delta functions are the same for both networks. From Eq. (6.1) the input to hidden weight gradients for the unbiased and biased cases are respectively:

$$\frac{-\partial E_p}{\partial w(j, i)} = \delta_{ph}(j) \cdot z_{pi} \quad (6.6)$$

$$\frac{-\partial E_p}{\partial w(j, i)_b} = \delta_{ph}(j) \cdot (z_{pi} + m_i) = \frac{-\partial E_p}{\partial w(j, i)} + \delta_{ph}(j) \cdot m_i \quad (6.7)$$

The input to output gradients for unbiased and biased data are respectively:

$$\frac{-\partial E_p}{\partial w_o(k, i)} = \delta_{po}(k) \cdot z_{pi} \quad (6.8)$$

$$\frac{-\partial E_p}{\partial w_o(k, i)_b} = \delta_{po}(k) \cdot (z_{pi} + m_i) = \frac{-\partial E_p}{\partial w_o(k, i)} + \delta_{po}(k) \cdot m_i \quad (6.9)$$

Lastly, hidden to output gradients for unbiased and biased data are both equal to

$$\frac{-\partial E_p}{\partial w_o(k, j)} = \delta_{po}(k) \cdot O_j \quad (6.10)$$

From equations (6.6–6.10), we observe that some gradient vector elements are dependent upon the input bias vector  $m$ . Therefore, the gradient vectors for the two initial networks are not proportional to each other, and the weight vectors for the networks diverge during training. Thus the performance of CG is dependent on input biases. We presume that good algorithms perform identically on both unbiased and biased data.

As an example, we apply CG to an MLP with structure 8-36-1, where the training data is unbiased and biased data. We see that the training error reaches a small value for the unbiased data but remains large for the biased data, even after many iterations. CG consistently performs poorly in the presence of input biases.

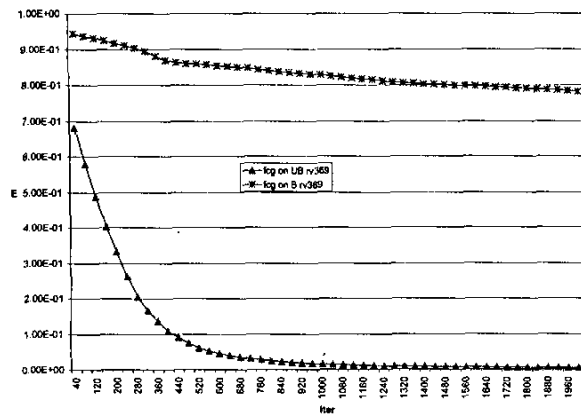


Figure 6.1. CG applied to biased and unbiased random data

## 7. CONCLUSIONS

A new one-pass algorithm was developed for calculating CG optimal learning factors. Tests were developed and demonstrated for validating OLFs. An analysis has shown that gradient-based training algorithms are sensitive to input biases. Better training was demonstrated after biases were removed.

## 8. ACKNOWLEDGEMENT

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

## 9. REFERENCES

[1] Fletcher, R.; *Practical Methods of Optimization*, second edition, New York: Wiley, 1987

[2] S. Haykin, *Neural Networks-A comprehensive foundation*, Prentice-Hall, Inc., NJ, 1999.

[3] M.T. Manry, M.S. Dawson, A.K. Fung, S.J. Apollo, L.S. Allen, W.D. Lyle, and W. Gong, "Fast Training of Neural Networks for Remote Sensing," *Remote Sensing Reviews*, 1994, vol. 9, pp. 77-96

[4] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks* vol. 2, no. 5, pp. 359-366, 1989.

[5] K. Hornik, M. Stinchcombe, and H. White, "Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks," *Neural Networks* vol. 3, no. 5, pp. 551-560, 1990.

[6] G. Cybenko, "Approximation by Superpositions of a Sigmoidal Function," *Mathematics of Control, Signals, and Systems*, vol. 2, no. 4, pp. 303-314, 1989

[7] Fitch, J.P.; Lehman, S.K.; Dowla, F.U.; Lu, S.Y.; Johansson, E.M.; Goodman, D.M., "Ship wake-detection procedure using conjugate gradient trained artificial neural

networks," *Geoscience and Remote Sensing*, IEEE Transactions on, vol. 29 Issue: 5, Sept. 1991 pp. 718-726.

[8] Johansson, E. M., Dowla, F.U.; Goodman, D.M., "Backpropagation learning for multilayer feed-forward neural networks using the conjugate gradient method," *International Journal of Neural Systems*, vol.2, no.4 1991 pp. 291-301.

[9] Charalambous, C., "Conjugate gradient algorithm for efficient training of artificial neural networks," *Circuits, Devices and Systems, IEE Proceedings G*, Volume: 139 Issue: 3, June 1992 pp. 301-310

[10] Barnard, E., "Optimization for training neural nets," *Neural Networks, IEEE Transactions on*, Volume: 3 Issue: 2, March 1992 pp 232-240

[11] Falas, T.; Stafylopatis, A.-G., "Temporal differences learning with the conjugate gradient algorithm," *Neural Networks, 2001. Proceedings. IJCNN '01. International Joint Conference on*, 2001 pp. 171-176 vol.1

[12] Yap, K.-H.; Guan, L., "A recursive soft-decision PSF and neural network approach to adaptive blind image regularization," *Image Processing, Proceedings. 2000 International Conference on*, pp. 813-816 vol.3

[13] Campello, R.J.G.B.; Amaral, W.C., "Optimization of hierarchical neural fuzzy models," *Neural Networks. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on*, pp. 8-13 vol.5

[14] Serele, C.Z.; Gwyn, Q.H.J.; Boisvert, J.B.; Pattey, E.; McLaughlin, N.; Daoust, G., "Corn yield prediction with artificial neural network trained using airborne remote sensing and topographic data," *Proceedings. IGARSS 2000. IEEE International*, Volume: 1, pp. 384-386

[15] Lizi Zhang; Jinping Kang; Xianshu Lin; Yinghui Xu, "Application of neural networks trained with an improved conjugate gradient algorithm to the turbine fast valving control," *Proceedings. PowerCon 2000. International Conference on*, pp.1679-1682 vol.3

[16] Kim-Hui Yap; Ling Guan, "A recursive approach to joint image restoration and compensated blur identification," *Neural Networks for Signal Processing X, 2000. Proceedings of the 2000 IEEE Signal Processing Society Workshop*, pp. 567-575 vol.2

[17] Jiang Minghu; Zhu Xiaoyan; Yuan Baozong; Tang Xiaofang; Lin Biqin; Ruan Qiuqi; Jiang Mingyan, "A fast hybrid algorithm of global optimization for feedforward neural networks," *WCCC-ICSP 2000. 5th International Conference on*, Vol. 3, pp. 1609-1612

[18] Moller, M.F., "A scaled conjugate gradient algorithm for fast supervised learning," *Neural Networks* 1993 Vol. 6 pp 525-533

[19] Kim, T., "Development and Evaluation of multiplayer perceptron training algorithms," *dissertation, University of Texas at Arlington* 2001

## 10. APPENDIX

The expression of the error function and its derivatives in terms of all the weights, weights change vector elements, and the OLF is as follows:

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

$$\frac{\partial y_{pk}}{\partial Z} = \sum_{i=1}^{N+1} e_o(k,i) \cdot x_{pi} + \sum_{j=1}^{N_h} \{e_{oh}(k,j) \cdot f(\text{net}_{pj} + Z \cdot \text{Net}_{pj}) + [w_{oh}(k,j) + Z \cdot e_{oh}(k,j)] \cdot f \cdot (1-f) \cdot \text{Net}_{pj}\} \quad (\text{A.2})$$

where

$$f = f(\text{net}_{pj} + Z \cdot \text{Net}_{pj}) \quad (\text{A.3})$$

$$f_0 = f(\text{net}_{pj}) \quad (\text{A.4})$$

$$\frac{\partial^2 y_{pk}}{\partial Z^2} = \sum_{j=1}^{N_h} \text{Net}_{pj} \cdot f \cdot (1-f) \cdot \{2 \cdot e_{oh}(k,j) + \text{Net}_{pj} \cdot (1-2f) \cdot [w_{oh}(k,j) + Z \cdot e_{oh}(k,j)]\} \quad (\text{A.5})$$

Let

$$X_1 = \{2 \cdot e_{oh}(k,j) + \text{Net}_{pj} \cdot (1-2f) \cdot [w_{oh}(k,j) + Z \cdot e_{oh}(k,j)]\}$$

$$\text{And } X_2 = [w_{oh}(k,j) + Z \cdot e_{oh}(k,j)]$$

Then,

$$\frac{\partial^3 y_{pk}}{\partial Z^3} = \sum_{j=1}^{N_h} \text{Net}_{pj}^2 \cdot f \cdot (1-f) \cdot \{(1-2f) \cdot X_1 + \{\text{Net}_{pj} \cdot X_2 \cdot (-2f) \cdot (1-f) + (1-2f) \cdot e_{oh}(k,j)\}\} \quad (\text{A.6})$$

$$\frac{\partial y_{pk}}{\partial Z} \Big|_{z=0} = \sum_{i=1}^{N+1} e_o(k,i) \cdot x_{pi} + \sum_{j=1}^{N_h} f_0 \cdot \{e_{oh}(k,j) + w_{oh}(k,j) \cdot (1-f_0) \cdot \text{Net}_{pj}\} \quad (\text{A.7})$$

$$\frac{\partial^2 y_{pk}}{\partial Z^2} \Big|_{z=0} = \sum_{j=1}^{N_h} \text{Net}_{pj} \cdot f_0 \cdot (1-f_0) \cdot \{2 \cdot e_{oh}(k,j) + w_{oh}(k,j) \cdot \text{Net}_{pj} \cdot (1-2f_0)\} \quad (\text{A.8})$$

$$\frac{\partial^3 y_{pk}}{\partial Z^3} \Big|_{z=0} = \sum_{j=1}^{N_h} \text{Net}_{pj}^2 \cdot f_0 \cdot (1-f_0) \cdot \{[(1-2f_0) \cdot \{2 \cdot e_{oh}(k,j) + w_{oh}(k,j) \cdot \text{Net}_{pj} \cdot (1-2f_0)\} + \text{Net}_{pj} \cdot (-2f_0) \cdot (1-f_0) \cdot w_{oh}(k,j) + (1-2f_0) \cdot e_{oh}(k,j)]\} \quad (\text{A.9})$$

Now,

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

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

$$\frac{\partial^3 E}{\partial Z^3} = \frac{2}{N_v} \sum_{p=1}^{N_v} \sum_{k=1}^M \left[ 3 \cdot \left( \frac{\partial y_{pk}}{\partial Z} \right) \cdot \left( \frac{\partial^2 y_{pk}}{\partial Z^2} \right) - [t_{pk} - y_{pk}] \cdot \frac{\partial^3 y_{pk}}{\partial Z^3} \right] \quad (\text{A.12})$$

From equations (4.3), (A.7), (A.8), (A.10), and (A.11), we can get second degree learning factor Z. Also from equations (4.5), (A.7), (A.8), (A.9), (A.10), (A.11), and (A.12), we obtain the third degree learning factor. In order to obtain Z, only one pass through the training data is required.

We note here that  $A_1$  can be obtained using the chain rule, the gradient  $\mathbf{g}$ , and the weight change vector  $\mathbf{p}$  as follows. This formula uses the already obtained  $\mathbf{g}$  and  $\mathbf{p}$  to save time for obtaining  $A_1$ .

$$A_1 = \frac{\partial E}{\partial Z} \Big|_{z=0} = \frac{\partial E}{\partial \mathbf{w}} \Big|_{z=0} \cdot \mathbf{p} = \mathbf{g} \cdot \mathbf{p} \quad (\text{A.13})$$