

Evaluation and Improvement of Two Training Algorithms

Tae-Hoon Kim, Jiang Li and Michael T. Manry
University of Texas at Arlington
Department of Electrical Engineering
Arlington, TX 76019
Telephone:(817) 272-3469
FAX: (817) 272-2253
Email: manry@uta.edu

Abstract

Two effective neural network training algorithms are output weight optimization – hidden weight optimization and conjugate gradient. The former performs better on correlated data, and the latter performs better on random data. Based on these observations and others, we develop a procedure to test general neural network training algorithms. Since good neural network algorithm should perform well for all kinds of data, we develop alternation algorithms, which have runs of the different algorithms in turn. The alternation algorithm works well for both kinds of data.

1. Introduction

The Conjugate gradient (CG) algorithm occasionally performs well in artificial neural network (ANN) training [1 – 4]. Another training algorithm, which often performs well, is output weight optimization – hidden weight optimization (OWO-HWO) [5]. Experimentation has shown that CG performs better on random data and OWO-HWO performs better on correlated data. Based on these observations, we develop a set of desired training algorithm properties in section 2. In section 3 we show that both algorithms lack some of the desired properties. In section 4 we construct a new algorithm that has all the desired properties. In section 5, we present numerical results. In section 6, we conclude this paper.

2. Desired training algorithm properties

Many investigators have described Multilayer Perceptron (MLP) training algorithm properties, both good and bad. In general it is convenient for the algorithm to: (1) work equally well for all types of data, (2) be invariant to

linear transformation of the input vectors, and (3) approach theoretically attainable values of training error, such as Cramer-Rao-MAP bounds [7].

The desired properties of training algorithms are:

(P1) The algorithm should work well for both correlated and random data. By correlated data, we mean (a) that inputs and desired outputs are correlated, and (b) inputs may be correlated with each other.

(P2) The algorithm should be invariant to input biases and variances.

(P3) For large enough networks, memorization should take place during training. Thus, zero is our lower bound on training error.

If an algorithm fails to memorize data sets having a small number of patterns, it can be considered to have a problem. The upper bound on pattern storage [6] for a neural net is denoted by

$$S_u = \frac{N_w}{M} \quad (2.1)$$

where the number of network parameters N_w is

$$N_w = (N+1)N_h + (N+N_h+1)M \quad (2.2)$$

for a fully connected MLP, which includes bypass weights. Here N , N_h , and M respectively denote number of inputs, hidden units and outputs.

2.1 Data file types

Following properties (P1) and (P3), four kinds of data can be prepared for testing the algorithms. In random data, input and desired output vectors have statistically independent elements, generated by a random number generator. In random memorization (RM) data, the number of training patterns, N_v , satisfies the condition $N_v = S_u$. In random non-memorization (RN) data, $N_v \gg S_u$.

Correlated training data can take many forms. We assume that elements of the input vector are correlated,

and that correlations exist between input and desired output vector elements. In correlated memorization (CM) data, N_v satisfies the condition $N_v = S_u$. In correlated non-memorization (CN) data, $N_v \gg S_u$.

2.2 Procedure to evaluate training algorithms

The proposed steps for evaluating a training algorithm are as follows:

Step 1: Determine whether or not the algorithm is invariant to input variances and biases. Invariance is the desired outcome.

Step 2: Test the algorithm on RM and RN data. It is desirable that the training error for RM data is much less than the training error of RN data.

Step 3: Test the algorithm on CM and CN data. It is desirable that the training error for CM data is much less than the training error of CN data.

3. Evaluation of training algorithms

In this section, we evaluate the CG and OWO-HWO training algorithms following the steps above.

3.1 Evaluation of CG:

As an experiment we applied CG to biased and unbiased RM data. The multilayer perceptron (MLP) structure was 8-36-1 and the training error for each case is shown in Fig 1. We see that CG memorized only the unbiased data. In

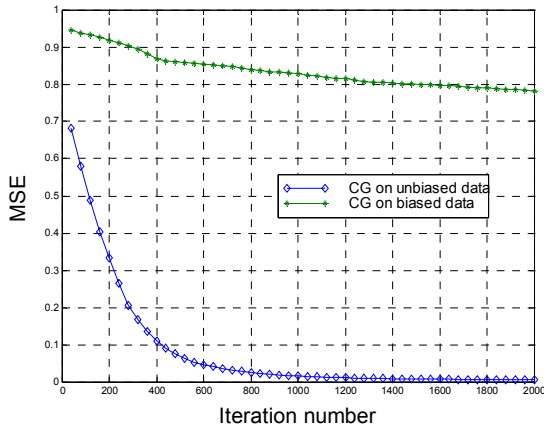


Fig 1. CG applied to biased and unbiased RM data

the following we show that gradients are affected by biases. Assuming linear output activation, the output layer delta function for the k_{th} output is

$$\delta_{po}(k) = 2 \cdot (t_{pk} - y_{pk}) \quad (3.1)$$

and the hidden layer delta functions is

$$\delta_{ph}(j) = O_j \cdot (1 - O_j) \cdot \sum_{k=1}^M \delta_{po}(k) \cdot w_o(k, N + j + 1) \quad (3.2)$$

let's express the input as a sum of the i th input mean m_i and a zero-mean input z_{pi} as

$$x_{pi} = z_{pi} + m_i \quad (3.3)$$

In the following, we show why the gradients are different for biased and unbiased training data:

(1) Input to hidden weight gradients for the unbiased and biased cases are respectively

$$\frac{-\partial E_p}{\partial w(j, i)} = \delta_{ph}(j) \cdot z_{pi} \quad (3.4)$$

$$\frac{-\partial E_p}{\partial w(j, i)_b} = \delta_{ph}(j) \cdot (z_{pi} + m_i) = \frac{-\partial E_p}{\partial w(j, i)} + \delta_{ph}(j) \cdot m_i \quad (3.5)$$

(2) Input to output weight gradients for unbiased and biased cases are respectively

$$\frac{-\partial E_p}{\partial w_o(k, i)} = \delta_{po}(k) \cdot z_{pi} \quad (3.6)$$

$$\frac{-\partial E_p}{\partial w_o(k, i)_b} = \delta_{po}(k) \cdot (z_{pi} + m_i) = \frac{-\partial E_p}{\partial w_o(k, i)} + \delta_{po}(k) \cdot m_i \quad (3.7)$$

(3) Hidden to output weight gradients for unbiased and biased cases are respectively

$$\frac{-\partial E_p}{\partial w_o(k, j)} = \delta_{po}(k) \cdot O_j \quad (3.8)$$

$$\frac{-\partial E_p}{\partial w_o(k, j)_b} = \delta_{po}(k) \cdot O_j = \frac{-\partial E_p}{\partial w_o(k, j)} \quad (3.9)$$

From equations (3.4–3.9), we observe that input connected weight gradients for unbiased and biased data are different. Thus the performance of CG is dependent on input biases. A similar result holds for input variances.

3.2 Evaluation of OWO-HWO

In OWO-HWO we alternately, solve linear equations for output weights and solve linear equations for hidden weight changes. Therefore input biases should have no effect on training. In the following, we evaluate OWO-HWO training and show how the performance of OWO-HWO compares to that of CG on RM and CM data. The RM data set RV369.tr, generated by a random number

generator, contains 369 patterns. The MLP structure is 8-36-1 so that it satisfies $N_v = S_u$. The training error for each algorithm is shown in Fig. 2. We observe that memorization takes place for CG algorithm, but not for

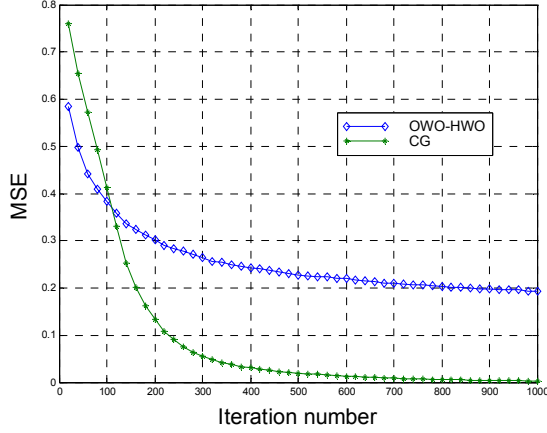


Fig. 2: OWO-HWO & CG applied on RM data

OWO-HWO, and CG outperforms OWO-HWO. Unbiased CM data set power14.tra is obtained from TU Electric Company in Texas. It has 14 inputs and one output. The first ten input features are the last ten minutes' average power load in megawatts for the entire TU Electric utility,

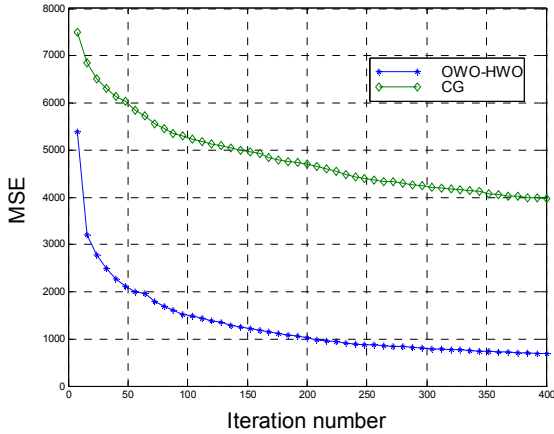


Fig. 3: OWO-HWO & CG applied to CM data

which covers a large part of north Texas. The output is the forecast power load fifteen minutes in the future from the current time. All powers are originally sampled every fraction of a second, and averaged over 1 minute to reduce noise. We chose the structure 14-10-1 for this data. The original data contains 1768 patterns, we randomly pick 175 patterns so as it satisfies the memorization condition

$N_v = S_u$. The training error for each algorithm is given in Fig. 3. We see that OWO-HWO outperforms CG even though both algorithms fail to completely memorize. We want to mention that the target output values in Power14.tra are large hence the resulting MSE is large. Obviously, both algorithms alone cannot satisfy all the desired properties.

4. Alternation algorithms

OWO-HWO and CG have different performance on different types of data, Aiming to satisfy all the desired properties, we need to combine these two algorithms in some way. When one of the algorithms learns slowly, we switch to the other algorithm. Two alternating algorithms, which perform well on all four data types, are developed here.

4.1 Fixed Alternation (FA)

This algorithm has fixed I_f and I_{oh} throughout the training, where I_f and I_{oh} are the numbers of iterations for CG and OWO-HWO applied in each training cycle respectively. A training cycle includes I_f iterations of CG followed by I_{oh} iterations of OWO-HWO. In FA case, we set $I_f = I_{oh}$ for each training cycle. The numbers I_f and I_{oh} are determined heuristically. When a certain number of iterations makes the algorithm get stuck in a local minimum, we use either a bigger or smaller number of iterations.

4.2 Adaptive Alternation (AA)

The adaptive alternation algorithm varies the numbers of iterations I_f and I_{oh} for each training cycle depending on their performances in the previous training cycle. The algorithm, which has the larger error decrease in the previous training cycle, is given more iterations in the next training cycle. For the first training cycle, we run both algorithms for a fixed amount of iterations and calculate the training error decrease for each algorithm, i.e. D_f and D_{oh} . Then I_f and I_{oh} for the next training cycle are calculated as follows:

$$I_f \leftarrow I_t \cdot \frac{\frac{D_f}{I_f}}{\frac{D_f}{I_f} + \frac{D_{oh}}{I_t}} = \frac{I_t^2}{I_f} \cdot \frac{D_f}{D_f + D_{oh}} \quad (4.1)$$

$$I_{oh} \leftarrow I_t \cdot \frac{\frac{D_{oh}}{I_{oh}}}{\frac{D_f}{I_t} + \frac{D_{oh}}{I_{oh}}} = \frac{I_t^2}{I_{oh}} \cdot \frac{D_{oh}}{D_f + D_{oh}} \quad (4.2)$$

where I_t is the total number of iterations for both algorithms in the previous training cycle. This process of equations (4.1) and (4.2) goes on until the end of training. An important property of AA is that the computational resources are allocated to the better performing algorithm.

5. Performance comparisons

After running CG, OWO-HWO and AA algorithms on a lot of unbiased training data, we observed that AA performed best in most of correlated training data but second best in random training data. Contrary to what we hoped, AA didn't perform significantly better than FA. In half of the training data, FA performed a little bit better. Regular switching of the algorithms was enough to let the

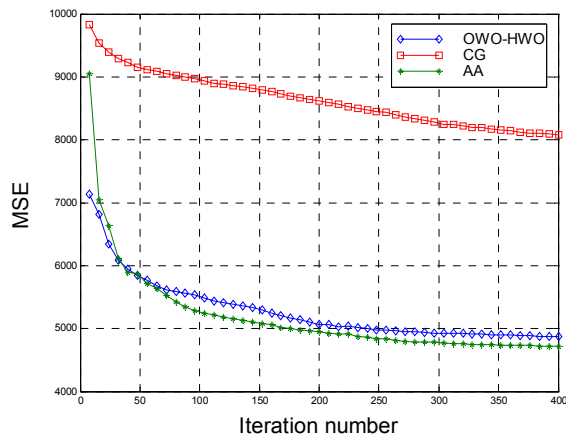


Fig 4. Algorithm performance on CN power14.tra

algorithm get out of local minimum. In this section, we show typical plots of the comparisons. All the algorithms ran on normalized data, and for some of the data, we used KLT transformations to compress the inputs.

First, the algorithms were run on CN data set Power14.tra, which is the original Power14.tra data set with 1768 patterns so as it satisfy non-memorization condition $N_v \gg S_u$. The MLP structure is 14-10-1. For AA, an initial training cycle consists of 20 iterations of CG and 20 iterations of OWO-HWO. We can see that AA performs best (Fig. 4).

Second, the algorithms were run on CM data set F17.tra, which is for onboard *Flight Load Synthesis* (FLS) in helicopters. In FLS, we estimate 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 tub 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 of Fort Worth. We chose a 15-17-9 MLP structure for F-17.tra. The input vectors have been replaced by their KLT transformations, which compressed the dimension of the input vectors from 17 to 15. The original data set contains 2823 patterns, in order to meet memorization condition $N_v = S_u$, we randomly picked 63 patterns so as it satisfies equation (2.1). Figure 5 shows that AA performs well. We want to mention that the target output values in F17.tra are large hence the resulting MSE is large.

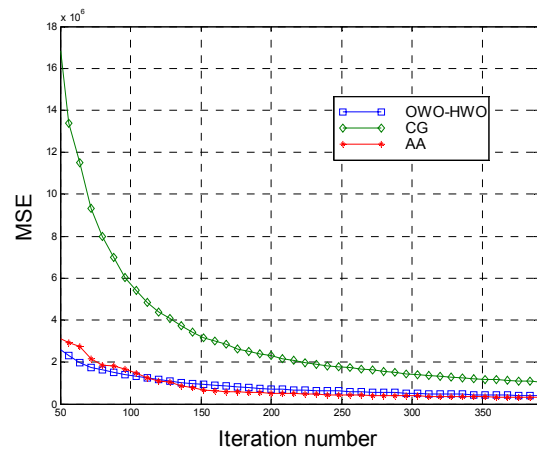


Fig 5 Algorithm performance on CM F17.tra

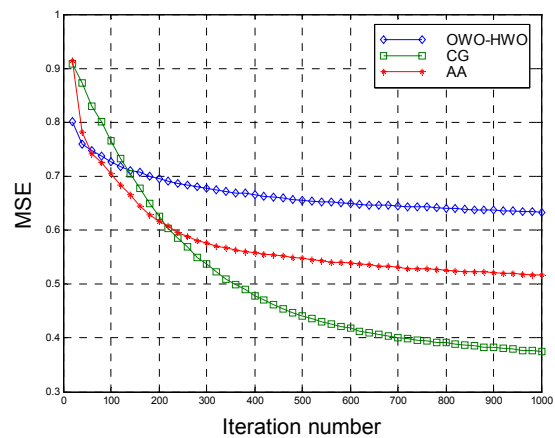


Fig. 6 Algorithm performance on RN RV1000.tra

Third, the algorithms were run on RN data set RV1000.tra, which is generated by a random generator. It consists of 8 inputs and 1 output, 1000 patterns. All

elements have the same variances and their energies are evenly distributed, so we don't need to normalize the data. The MLP structure is 8-36-1. From Fig 6 we can see that AA performs second best.

Fourth, the algorithms were run on 369 patterns of RM data set RV369.tra. The MLP structure of the network is 8-36-1. We can see from Fig.7 that AA performs second best.

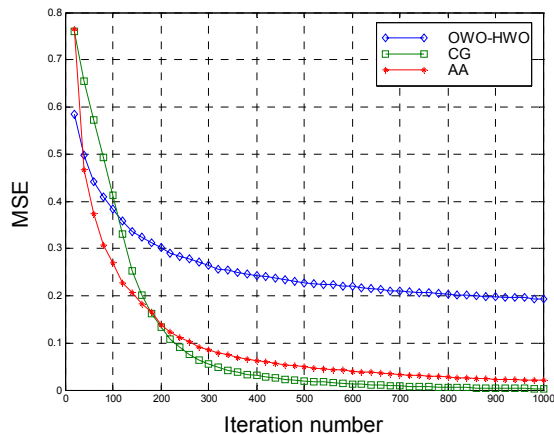


Fig. 7 Algorithm performance on RM RV369.tra

6. Conclusions

In this paper we have introduced a set of desired properties for MLP training algorithms and shown that both CG and OWO-HWO alone lack these properties. Alternation algorithms are developed and run on normalized and KLTed data sets. Among the three algorithms, AA works best for correlated data sets and second best for random data sets. We observe that unless some algorithms are specifically suitable for certain kinds of data, AA performs well. We can say that AA algorithm has the desired properties.

Acknowledgment

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

References

- [1] J.P. Fitch, S.K. Lehman et al., "Ship wake-detection procedure using conjugate gradient trained artificial neural networks," *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 29 Issue: 5, pp. 718-726, Sept. 1991
- [2] E.M. Johansson, F.U. Dowlal and D.M. Goodman, "Backpropagation learning for multilayer feed-forward neural networks using the conjugate gradient method,"

International Journal of Neural Systems, vol. 2, no. 4, pp. 291-301, 1991.

- [3] C. Charalambous, "Conjugate gradient algorithm for efficient training of artificial neural networks," *Circuits, Devices and Systems, IEE Proceedings*, Vol. 139, Issue: 3, pp. 301-310, June 1992.
- [4] E. Barnard, "Optimization for training neural nets," *Neural Networks, IEEE Transactions on*, Vol. 3, Issue: 2, pp. 232-240, March 1992.
- [5] H.H. Chen, M.T. Manry and H. Chandrasekaran, "A Neural Network training Algorithm Utilizing Multiple Sets of linear equations," *Neurocomputing*, vol. 25, no. 1-3, pp. 55-72, April 1999.
- [6] A. Gopalakrishnan, X. Jiang, M.S. Chen and M.T. Manry, "Constructive proof of efficient pattern storage in the multilayer perceptron," *Twenty seventh Asilomar Conference on Signals, Systems & Computers*, vol. 1, pp. 386-390, Nov 1993.
- [7] M.T. Manry, Cheng-Hsiung Hsieh, M.S. Dawson, A.K. Fung and S.J. Apollo, "Cramer Rao Maximum A-Posteriori Bounds on Neural Network Training Error for Non-Gaussian Signals and Parameters," *International Journal of Intelligent Control and Systems*, vol. 1, no. 3, pp. 381-391, 1996.