

New Training Algorithms for Dependently Initialized Multilayer Perceptrons

Walter H. Delashmit
 Lockheed Martin
 Missiles and Fire Control
 Dallas, TX 75265

Michael T. Manry
 University of Texas at Arlington
 Electrical Engineering Department
 Arlington, TX 76011

Abstract—Due to the chaotic nature of multilayer perceptron training, training error usually fails to be a monotonically nonincreasing function of the number of hidden units. New training algorithms are developed where weights and thresholds from a well-trained smaller network are used to initialize a larger network. Methods are also developed to reduce the total amount of training required. It is shown that this technique yields an error curve that is a monotonic nonincreasing function of the number of hidden units and significantly reduces the training complexity. Additional results are presented based on using different probability distributions to generate the initial weights.

I. INTRODUCTION

When multilayer perceptrons (MLPs) with different numbers of hidden units are trained, the networks often have a training error that is not a monotonically nonincreasing function of the number of hidden units (N_h) as shown in Fig. 1. Thus, investigators often design many networks and save the one with the minimum mean square error (MSE). Alternate approaches adjust the initial seeds to control the values of the assigned random numbers. This attempts to avoid the problem by finding a set of seeds with decreasing MSE for increasing number of hidden units. These “ad hoc” solutions do not mitigate chaotic MLP training.

In this* paper we address solutions to these problems. In section II, a formulation of the problem is presented. In section III, dependently initialized networks are defined and properties of these networks are developed. In section IV, techniques are presented for training new hidden units for these networks along with simulation results. Results using different initial weight probability distributions are presented in section V with a summary of the paper in section VI.

II. PROBLEM FORMULATION

MLP training [1] is inherently dependent on network initialization. Assume that a set of different size MLPs (different N_h values) are to be designed for a given training data set [2]. Let S_{N_h} be the set of all MLPs for that set having N_h hidden units and let $E_{int}(N_h)$ denote the corresponding

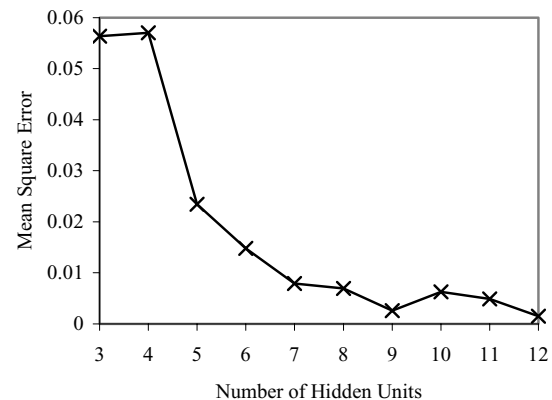


Fig. 1. Typical MLP nonmonotonic error curve.

training error of an initial network that belongs to S_{N_h} . Similarly, let $E_f(N_h)$ denote the corresponding final training error of a well-trained network. Let N_{hmax} be the maximum number of hidden units for which networks are designed. Using this notation the design goal can be specified as follows:

Goal: Choose a set of initial networks from $\{S_0, S_1, S_2, \dots, S_{N_{hmax}}\}$ such that

$$E_{int}(0) \geq E_{int}(1) \geq E_{int}(2) \geq \dots \geq E_{int}(N_{hmax}) \quad (1)$$

and train the network to minimize $E_f(N_h)$ such that

$$E_f(0) \geq E_f(1) \geq E_f(2) \geq \dots \geq E_f(N_{hmax}). \quad (2)$$

An axiom follows that supports this goal.

Axiom: If $E_f(N_h) \geq E_f(N_h-1)$, then the network having N_h hidden units is useless since the training resulted in a larger more complex network with a larger or the same training error. It should be noted that adding one hidden unit adds one nonlinear activation, one threshold and $N+M$ weights where N is the number of inputs and M is the number of outputs of the MLP. In general, we want to avoid useless networks to reduce the amount of computation and reduce the training error.

* This work was supported in part by the Advanced Technology Program of the State of Texas under grant 003656-0129-2002.

III. DEPENDENTLY INITIALIZED NETWORKS

For dependently initialized (DI) networks a series of different size networks are designed with each subsequent network having one or more hidden units than the previous network. DI networks are useful for performing a thorough analysis of network performance versus network size.

The flowchart for the basic DI network algorithm is shown in Fig. 2. These networks [2] build on previously well-trained networks. For DI networks, the numerical values of the common subset of the initial weights and thresholds for the larger network are the final weights and thresholds from the well-trained smaller network. To design a DI network requires an initial non-DI network of N_h hidden units. Initial starting networks can have any value for N_h less than N_{hmax} and can be designed using several techniques. After designing the initial network, each subsequent network is a DI network. The final weights, w_f , for the well-trained smaller network of size N_{h-p} are used as the initial weights, w_{int} , for the larger DI network of size N_h . In Fig. 2, $RN(ind+)$ is an array of random numbers used to store the initial

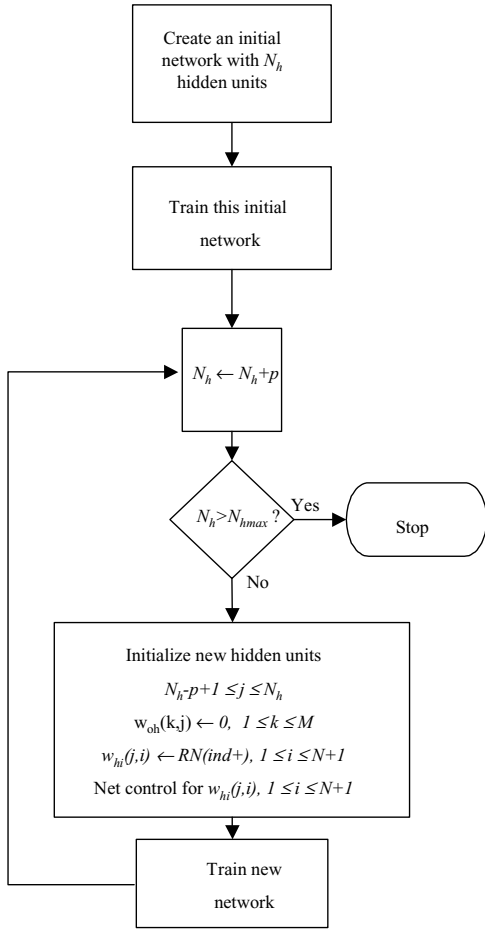


Fig. 2. DI network design flowgraph.

weights and thresholds with the index of $ind+$ used to indicate that the next random number from this array is chosen for the specified weights or thresholds being initialized.

The net control step in Fig. 2 changes the random initial weights and thresholds in order to prevent activation derivatives close to zero. Net control is implemented by setting the hidden unit net function mean and standard deviation to desired values m_{hd} and σ_{hd} . This requires determining the mean $m_h(j)$ and the standard deviation $\sigma_h(j)$ of each hidden unit's net function. For the j th hidden unit net function ($1 \leq j \leq N_h$) and the i th-augmented input ($1 \leq i \leq N+1$), the input-to-hidden unit weights are adjusted as

$$w_{hi}(j,i) \leftarrow \frac{w_{hi}(j,i) \cdot \sigma_{hd}}{\sigma_h(j)} \quad (3)$$

with the hidden unit thresholds adjusted using

$$w_{hi}(j,N+1) \leftarrow w_{hi}(j,N+1) + m_{hd} - \frac{m_h(j) \cdot \sigma_{hd}}{\sigma_h(j)}. \quad (4)$$

The initialization technique shown in Fig. 2 applies whether the network of size N_{h-p} is the initial starting non-DI network or a previously trained DI network. For the new DI network the new initial output weights are initialized with values of zero. The initial error, E_{int} , for the larger network with N_h hidden units, N_v training patterns, M desired outputs (t_p) for the p th pattern and $N+1$ inputs (x_p) for the p th pattern is

$$E_{int}(N_h) = \frac{1}{N_v} \sum_{p=1}^{N_v} E_p = \frac{1}{N_v} \sum_{p=1}^{N_v} \sum_{i=1}^M [t_p(i) - y_{p,int}(i, N_h)]^2 \quad (5)$$

$$y_{p,int}(i, N_h) = \sum_{k=1}^{N+1} w_{f,oi}(i, k) \cdot x_p(k) + \sum_{j=1}^{N_h-p} w_{f,oh}(i, j) \cdot O_{p,f}(j) + \sum_{j=N_h-p+1}^{N_h} w_{int,oh}(i, j) \cdot O_{p,int}(j) \quad (6)$$

$$y_{p,int}(i, N_h) = y_{p,f}(i, N_h-p) = \sum_{k=1}^{N+1} w_{f,oi}(i, k) \cdot x_p(k) + \sum_{j=1}^{N_h-p} w_{f,oh}(i, j) \cdot O_{p,f}(j) \quad (7)$$

where $y_{p,int}(i, N_h)$ denotes the i th output of an initial network having N_h hidden units and $y_{p,f}(i, N_h-p)$ corresponds to the final well-trained network having N_h-p hidden units. Here, $w_{f,oi}(i, k)$ and $w_{f,oh}(i, j)$ are the final weights from input-to-output and hidden unit-to-output, respectively, for the well-trained smaller network with N_h-p hidden units. Outputs of the j th hidden unit, $O_{p,f}(j)$ and $O_{p,int}(j)$, are defined similarly. The j th hidden unit net function with weights from input-to-hidden unit (w_{hi}) is

$$net_p(j) = \sum_{k=1}^{N+1} w_{hi}(j,k) \cdot x_p(k) \quad 1 \leq j \leq N_h \quad (8)$$

with the output activation for the p th training pattern, $O_p(j)$, being expressed by

$$O_p(j) = f(net_p(j)). \quad (9)$$

The sigmoidal function is typically chosen for the nonlinear activation.

$$f(net_p(j)) = \frac{1}{1 + e^{-net_p(j)}}. \quad (10)$$

Properties of DI Networks: Based upon the design technique of Fig. 2, DI network properties are:

- (1) $E_{int}(N_h) < E_{int}(N_h-p)$
- (2) The $E_f(N_h)$ curve is monotonic nonincreasing (i.e., $E_f(N_h) \leq E_f(N_h-p)$)
- (3) $E_{int}(N_h) = E_f(N_h-p)$.

Property 1 follows since $E_{int}(N_h)$ is equal to $E_f(N_h-p)$ and $E_f(N_h-p) < E_{int}(N_h-p)$ results from the implementation of the training algorithm since the weights and thresholds during training are not updated unless the error decreases from one iteration to the next (i.e., $E_{iter} < E_{iter-1}$). Property 2 follows from the initialization shown in Fig. 2 and (5) - (7) where the larger network has N_h hidden units and the smaller network has N_h-p hidden units. The hidden units in the larger network do not contribute to the error since the new output weights and thresholds are initialized to zero resulting in $y_{p,int}(i, N_h) = y_{p,f}(i, N_h-p)$. Hence the training error is monotonic nonincreasing since the weights and thresholds for a new iteration are not updated unless $E_{iter} < E_{iter-1}$. Property 3 also follows from the initialization shown in Fig. 2 and (5) - (7) where the output weights and thresholds for the larger N_h network are initialized to zero resulting in $y_{p,int}(i, N_h) = y_{p,f}(i, N_h-p)$.

Typical results for DI networks using a fixed number of iterations are shown in Fig. 3 for a data set consisting of eight inputs, seven outputs, 1,768 training patterns and 1,000 testing patterns. This data is used in the process of inverting the surface scattering patterns from an inhomogeneous layer above a homogeneous half space where both interfaces are randomly rough. These results are monotonic nonincreasing and the testing performance is also very good.

IV. TRAINING NEW HIDDEN UNITS FOR DI NETWORKS

The previous designs of DI networks for fixed number of iterations were based on training all weights when additional hidden units are added. That procedure typically results in the lowest value for $E_f(N_h)$ since maximum training is obtained for the entire network (i.e., all weights are trained every time new hidden units are added). A computationally

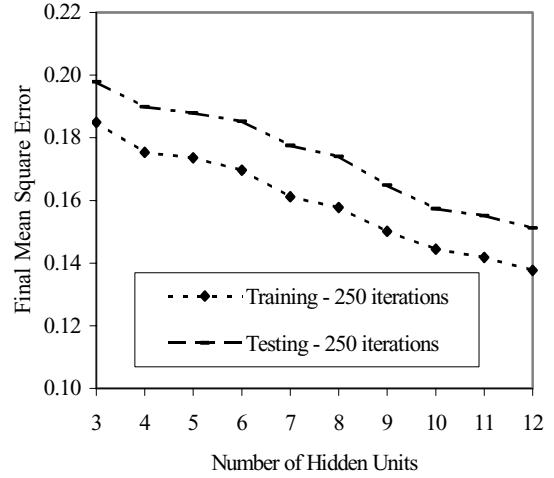


Fig. 3. DI network monotonic error curve for 250 iterations.

more efficient approach to increasing the size of the network is to retain the previous training and only train new hidden units. If conjugate gradient (CG) training [3] is used and the MSE is a quadratic function of all the weights, then the required number of iterations (N_{iter}) equals the total number of weights (N_w) where

$$N_w = N_h(N+M+1) + M(N+1). \quad (11)$$

This technique is implemented by training only the new hidden units until the iteration number (i_t) is greater than a specified fraction, Q , of the number of weights. In actuality, we use output-weight-optimization (OWO)/hidden-weight-optimization (HWO) training [4,5] instead of CG. The two training steps are:

- Step 1:** If $i_t < Q \cdot N_w$, train only new hidden unit weights
- Step 2:** If $i_t \geq Q \cdot N_w$, train all N_w network weights

Thus for $Q = 0.75$ only the new hidden units are trained until i_t is greater than $0.75N_w$. For $Q = 0$ all weights are trained for every iteration.

Results are shown in Fig. 4 for three values of “ Q .” The techniques begin to converge as Q and N_h increase with a larger value of Q requiring a larger value of N_h for satisfactory performance. Values of Q of 0.85 or less gave the same results as for the baseline case of Q equal to zero. This shows a technique for generating improved performance of networks while significantly reducing the number of multiplies required for each training iteration.

The computational savings of this new technique can be found based on the number of multiplies per pattern (N_{mult}) which expresses the number of multiplies a fully trained MLP requires to map an input vector to an output vector. This is expressed as

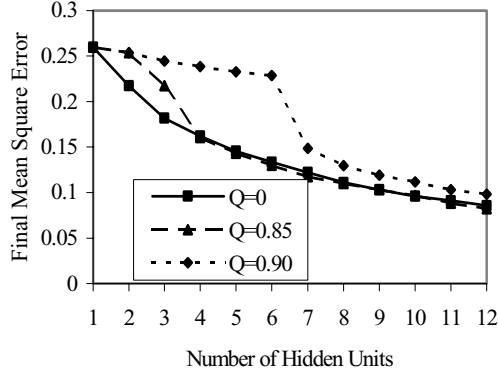


Fig. 4. DI network with reduced training iterations.

$$N_{mult} = N_h(N+M) + MN. \quad (12)$$

Here, we are not counting some multiplies used in training, such as those used in gradient calculations. This also does not count multiplies to process the net functions into sigmoids. That, in theory, can be done with very few multiplies per sigmoid and is insignificant compared to N_{mult} in (12). Since we set the number of iterations (N_{iter}) equal to the total number of weights (N_w) as expressed by (11), an **upper bound** on the total number of multiplies (M_{upper}) per pattern during evaluations of output vectors is expressed by

$$M_{upper} = N_{iter}N_{mult}. \quad (13)$$

When all weights and thresholds are retrained for every iteration M_{upper} is a quadratic function of N_h ,

$$M_{upper} = N_h^2(N+M+1)(N+M) + N_hM(2N(N+M+1)+M) + M^2N(N+1). \quad (14)$$

Based on the new DI network training algorithm described in **Step 1** and **Step 2**, the reduced number of multiplies, M_{red} , using this new algorithm is

$$M_{red} = N_{mult}(N_{iter} - [QN_{iter}]) + (N+M)[QN_{iter}] \quad (15)$$

where the notation $[\cdot]$ designates that the quantity inside the brackets is rounded and it is assumed that each subsequent network consists of only one additional hidden unit. This results in the implementation of **Step 1** adding only $N+M$ multiplies per iteration. In the limit as the number of iterations approaches infinity, which based on the technique used implies that the number of weights and hence the number of hidden units approaches infinity, $[QN_{iter}] \rightarrow QN_{iter}$ yielding a **lower bound** (M_{lower}) on the number of multiplies

$$M_{lower} = (1-Q)N_{mult}N_{iter} + Q(N+M)N_{iter}. \quad (16)$$

In (16), Q equal to zero reduces to the results in (13) and (14) where all weights and thresholds are trained for every

iteration. In (16), Q equal to one would imply that only the hidden units weights and thresholds are trained and the input to output weights and thresholds are not trained. In general, a network trained with a value of Q equal to one would probably not be practical for most situations.

Fig. 5 shows the upper bound, lower bound and actual number of multiplies for Q equal to 0.85 with similar results shown in Fig. 6 for Q equal to 0.9. It can be seen that the number of multiplies used by this new training algorithm approaches the lower bound and is significantly less than the total number of multiplies required to train all weights and thresholds for every iteration. Fig. 7 shows the percentage reduction in the number of multiplies for both values of Q . In analyzing these results, it should be noted that a value of Q of 0.85 produced the same final mean square error for four or more hidden units as was obtained for the case where all weights and thresholds were trained for every iteration.

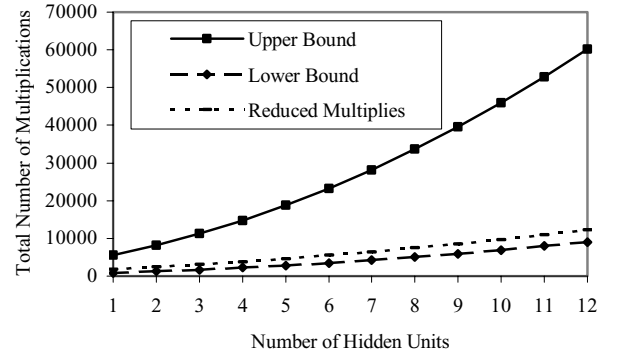


Fig. 5. Reduced computational requirements for $Q = 0.85$.

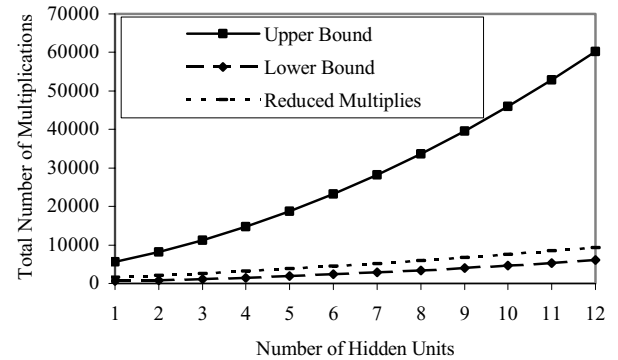


Fig. 6. Reduced computational requirements for $Q = 0.90$.

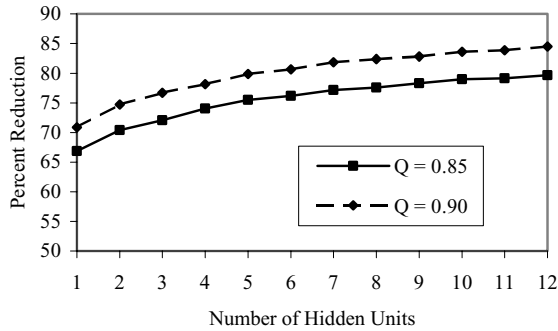


Fig. 7. Percentage reduction in multiplies for new algorithm.

V. INITIAL WEIGHTS PROBABILITY DISTRIBUTIONS

Often Gaussian distributions are used to generate the initial random weights and thresholds for an MLP. However other choices for this probability distribution are possible. Two additional choices were investigated for generating these initial random weights and thresholds. Fig. 8 compares the final MSE values for MLPs where the initial weights and thresholds were generated with (1) a $N(0,1)$ density, (2) a $U(-0.5,0.5)$ density and (3) a $U(0,1)$ density. These results are for the average of ten DI networks so that the results are generalized from what would occur for a single DI network. Initially the weights with the $U(0,1)$ distribution converge faster with the weights with the $N(0,1)$ distribution converging the slowest and the weights with the $U(-0.5,0.5)$ lying between these two values (Fig. 8 and Fig. 9). However, for two or more hidden units, the weights with the $N(0,1)$ distribution begin to yield the lowest final mean square error with the highest final mean square error occurring for the weights with the $U(0,1)$ distribution. This shows that the OWO-HWO training algorithm is robust enough that the random initial weights distributed $U(0,1)$ can be adjusted to produce reasonable performance results, but better performance is obtained with other weight distributions. Training using weights with a $U(0,1)$ distribution has to produce both positive and negative final weights, in general, even though all of the initial values are only positive numbers. The robustness of the training algorithm holds in general for many data sets, but the initial convergence is data set dependent [2].

VI. SUMMARY

Independently initialized networks were introduced and two new training algorithms were described for these networks. These networks significantly improve performance in terms of (1) ensuring that the training error is a monotonically nonincreasing function of the number of hidden units and (2) significantly reducing the number of multiplies required for network training. An analysis of the effects of the initial random seeds was also presented.

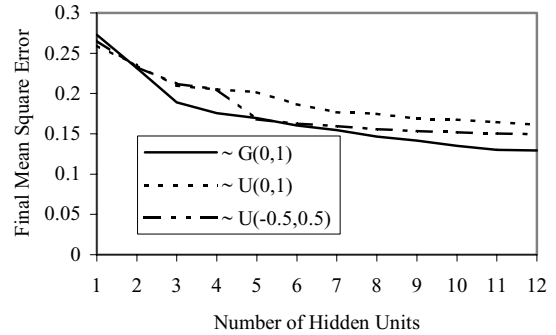


Fig. 8. Initial random weight probability densities.

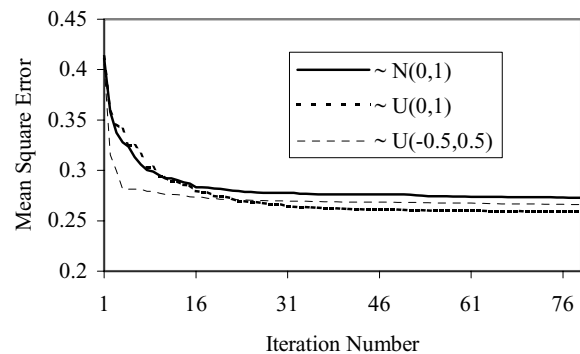


Fig. 9. Convergence versus iteration number.

REFERENCES

1. W. H. Delashmit and M. T. Manry, "Enhanced Robustness of Multilayer Perceptron Training," *Proceedings of the 36th Annual Asilomar Conference on Signals, Systems and Computers*, pp. 1029-1033, November 2002.
2. W. H. Delashmit, "Multilayer Perceptron Structured Initialization and Separating Mean Processing," *Ph.D. Dissertation*, University of Texas at Arlington, May 2003.
3. R. Fletcher and C. M. Reeves, "Function Minimization by Conjugate Gradients," *Computer Journal*, vol. 7, pp. 149-154, 1964.
4. 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.
5. Y. Changhwa and M. T. Manry, "A Modified Hidden Weight Optimization Algorithm for Feedforward Neural Networks," *Proceedings of the 36th Annual Asilomar Conference on Signals, Systems and Computers*, pp. 1034-1038, November 2002.