Recent Developments in Multilayer Perceptron Neural Networks

Walter H. Delashmit
Lockheed Martin Missiles and Fire Control
Dallas, Texas 75265
walter.delashmit@lmco.com
walter.delashmit@verizon.net

Michael T. Manry
University of Texas at Arlington
Arlington, TX 76010
manry@uta.edu

Abstract

Several neural network architectures have been developed over the past several years. One of the most popular and most powerful architectures is the multilayer perceptron. This architecture will be described in detail and recent advances in training of the multilayer perceptron will be presented.

Multilayer perceptrons are trained using various techniques. For years the most used training method was back propagation and various derivatives of this to incorporate gradient information. Recent developments have used output weight optimization-hidden weight optimization (OWO-HWO) and full conjugate gradient methods. OWO-HWO is a very powerful technique in terms of accuracy and rapid convergence. OWO-HWO has been used with a unique "network growing" technique to ensure that the mean square error is monotonically non-increasing as the network size increases (i.e., the number of hidden layer nodes increases). This "network growing" technique was trained using OWO-HWO but is amenable to any training technique. This technique significantly improves training and testing performance of the MLP.

1.0   Introduction

Several neural networks have been developed and analyzed over the last few decades. These include self-organizing neural networks [1, 2], the Hopfield network [3, 4], radial basis function networks [5-6], the Boltzmann machine [7], the mean-field theory machine [8] and multilayer perceptrons (MLPs) [9-17].

MLPs have evolved over the years as a very powerful technique for solving a wide variety of problems. Much progress has been made in improving performance and in understanding how these neural networks operate. However, the need for additional improvements in training these networks still exists since the training process is very chaotic in nature.

2.0   Structure and Operation of Multilayer Perceptron Neural Networks

MLP neural networks consist of units arranged in layers. Each layer is composed of nodes and in the fully connected networks considered in this paper each node connects to every node in subsequent layers. Each MLP is composed of a minimum of three layers consisting of an input layer, one or more hidden layer(s) and an output layer. The above definition ignores the degenerate linear multilayer perceptron consisting of only an input layer and an output layer. The input layer distributes the inputs to subsequent layers. Input nodes have linear activation functions and no thresholds. Each hidden unit node and each output node have thresholds associated with them in addition to the weights. The hidden unit nodes have nonlinear activation functions and the outputs have linear activation functions. Hence, each signal feeding into a node in a subsequent layer has the original input multiplied by a weight with a threshold added and then is passed through an activation function that may be linear or nonlinear (hidden units). A typical three-layer network is shown in figure 1. Only three layer MLPs will be considered in this paper since these networks have been shown to approximate any continuous function [18-20]. For the actual three-layer MLP, all of the inputs are also connected directly to all of the outputs. These connections are not shown in figure 1 to simplify the diagram.
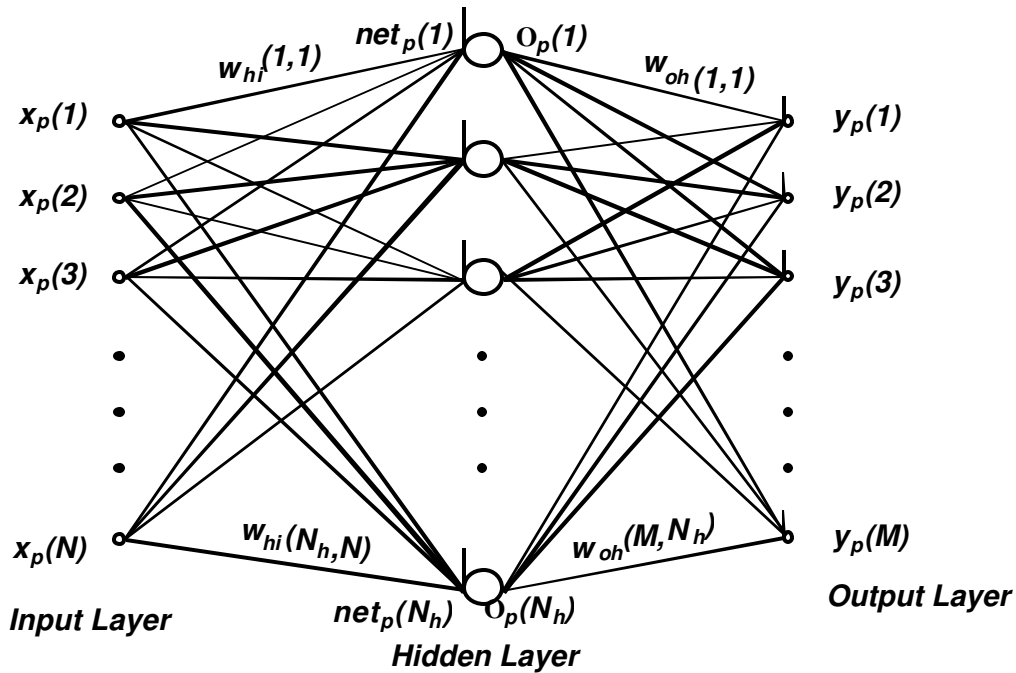

Figure 1. Typical three-layer multilayer perceptron neural network.

The training data consists of a set of N_v training patterns $(\mathbf{x}_p, \mathbf{t}_p)$ where p represents the pattern number. In figure 1, $\mathbf{x}_p$ corresponds to the N-dimensional input

vector of the pth training pattern and $\mathbf{y}_p$ corresponds to the M-dimensional output vector from the trained network for the pth pattern. For ease of notation and analysis, thresholds on hidden units and output units are handled by assigning the value of one to an augmented vector component denoted by $x_p(N+1)$. The output and input units have linear activations. The input to the jth hidden unit, $net_p(j)$, is expressed by

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

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

$$O_p(j) = f\left(net_p(j)\right) \qquad (2)$$

The nonlinear activation is typically chosen to be the sigmoidal function

$$f\left(net_p(j)\right) = \frac{1}{1+e^{-net_p(j)}} \qquad (3)$$

In (1) and (2), the N input units are represented by the index k and $w_{hi}(j,k)$ denotes the weights connecting the kth input unit to the jth hidden unit.

The overall performance of the MLP is measured by the mean square error (MSE) expressed by

$$E = \frac{1}{N_v} \sum_{p=1}^{N_v} E_p = \frac{1}{N_v} \sum_{p=1}^{N_v} \sum_{i=1}^{M} \left[t_p(i) - y_p(i)\right]^2 \qquad (4)$$

where

$$E_p = \sum_{i=1}^{M} \left[t_p(i) - y_p(i)\right]^2$$

E_p corresponds to the error for the pth pattern and t_p is the desired output for the pth pattern. This also allows the calculation of the mapping error for the ith output unit to be expressed by

$$E_i = \frac{1}{N_v} \sum_{p=1}^{N_v} \left[t_p(i) - y_p(i)\right]^2 \qquad (5)$$

with the ith output for the pth training pattern being expressed by

$$y_p(i) \; = \; \sum_{k=1}^{N+1} w_{oi}(i,k) \cdot x_p(k) + \sum_{j=1}^{N_h} w_{oh}(i,j) \cdot O_p(j) \qquad (6)$$

In (6), $w_{oi}(i,k)$ represents the weights from the input nodes to the output nodes and $w_{oh}(i,j)$ represents the weights from the hidden nodes to the output nodes.

2.1   Net Control

Some of the problems in MLP training are resolved by using the net-control algorithm of Olvera [21] that makes network weight behavior more stable. Sigmoid activations have small gradients when the input net function is large. This results in slow, ineffective training. In addition, large input values can dominate training and reduce the effects of smaller inputs. This can occur even when smaller inputs are more important to the overall performance of the system. Net control reduces these problems by scaling and shifting net functions so that they do not generate small gradients and do not allow large inputs to mask the potential effects of small inputs. This process adjusts each net function mean and standard deviation to a user defined standard deviation σ_{hd} and mean m_{hd}. The choice of these values of the standard deviation and mean are arbitrary and are the same for each net function. 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 jth hidden unit net function ($1 \le j \le N_h$) and the ith-augmented input ($1 \le i \le N+1$), the input-to-hidden unit weights are adjusted as

$$w_{hi}(j,i) \leftarrow \frac{w_{hi}(j,i) \cdot s_{hd}}{s_h(j)} \qquad (7)$$

with the hidden unit thresholds adjusted using

$$w_{hi}(j,N+1) \leftarrow w_{hi}(j,N+1) + m_{hd} - \frac{m_h(j) \cdot s_{hd}}{s_h(j)} \qquad (8)$$

2.2   Training of Multilayer Perceptron Neural Networks

Training [22-24] of MLPs is an area of much research and is a very important task in developing useful network architectures. Existing training techniques include back-propagation, output weight optimization-hidden weight optimization (OWO-HWO), the full conjugate gradient method and many others.

Backpropagation Training

Backpropagation training [25] was introduced by Werbos [26] with an independent development by Parker [27]. Another discussion is contained in Rumelhart and

McClelland [28]. In backpropagation training, all the weights and thresholds are updated using

$$w(j, i) \leftarrow w(j, i) + Z \cdot \frac{-\partial E_p}{\partial w(j, i)} \tag{9}$$

where Z is a constant learning factor. By using the chain rule, the gradients can be expressed by

$$\frac{\partial E_p}{\partial w(j, i)} = -d_p(j) \cdot O_p(i) \tag{10}$$

where

$$d_p(j) = \frac{-\partial E_p}{\partial net_p(j)} \tag{11}$$

is called the delta function. The calculation of the delta functions for the output units and the hidden units respectively [1] are expressed by

$$d_p(k) = f'(net_p(j)) \cdot (t_p(k) - O_p(k))$$
$$d_p(j) = f'(net_p(j)) \sum_n d_p(n) w(n, j) \tag{12}$$

where f is the first derivative of the activation function and n is the index of units in subsequent layers that are connected to the jth unit in the previous layer.

Output Weight Optimization-Hidden Weight Optimization

Output weight optimization hidden weight optimization (OWO-HWO) [12, 29-31] is an improved technique for updating the multilayer perceptron weights. Linear equations are used to solve for the output weights in OWO. Separate error functions for each hidden unit are used and multiple sets of linear equations are solved to determine the weights connecting to the hidden units [29, 30] in HWO. The desired jth net function $net_{pd}(j)$ for the pth pattern is

$$net_{pd}(j) = net_p(j) + Z \cdot d_p(j) \tag{13}$$

where $net_p(j)$ and Z is the learning factor. Desired weight changes $w_d(j,i)$ are found by solving

$$d_p(j) \cong \sum_{i=1}^{N+1} w_d(j, i) \cdot x_p(i) \tag{14}$$

These equations are approximately solved unit-by-unit for the desired weight changes $w_d(j,i)$. The hidden weights, $w(j,i)$ are updated as

$$?w(j, i) = Z \cdot w_d(j, i) \tag{15}$$

The total change in the error function E for a given iteration due to the changes in all hidden weights is approximately

$$? E \cong -Z \frac{1}{N_v} \sum_{j=1}^{N_h} \sum_{p=1}^{N_v} d_p^2(j) \tag{16}$$

When the learning factor Z is positive and small enough to make the approximation in (14)-(16) valid, the ΔE sequence is non positive and the sequence of training errors for the kth iteration is nonincreasing. Since nonincreasing sequences of nonnegative real numbers converge, the sequence of training errors converges [32].

When the error surface is highly curved, the approximation in (14)-(16) may not be valid for some iterations resulting in increases in E. For these cases, the algorithm restores the previously calculated network and reduces the learning factor Z. After this step is repeated a finite number of times, the E sequence will again be decreasing since the error surface is continuous [32]. This results in the convergence of the sequence of E values.

Full Conjugate Gradient Method

The full conjugate gradient training method (FCG) [12, 29-30] is applied to all of the weights and thresholds in the MLP. The weight vector $\mathbf{w}$ of dimension N_w is formed from these weights and thresholds. An optimal learning factor is used to update all of the weights at the same time.

The goal of the FCG method is to minimize the error function $E(\mathbf{w})$ with respect to $\mathbf{w}$, using the gradient vector $\mathbf{g} = \partial E/\partial \mathbf{w}$ and the direction vector $\mathbf{p}$. For the first iteration, $\mathbf{p} = -\mathbf{g}$. For the subsequent N_w-1 iterations $\mathbf{p}$ is expressed by

$$\mathbf{p} \leftarrow -\mathbf{g} + \frac{E_g(i_t)}{E_g(i_t - 1)} \bullet \mathbf{p} \tag{17}$$

where $E_g(i_t)$ the gradient vector energy in the current iteration and $E_g(i_t-1)$ is the gradient vector energy in the previous iteration. The learning factor Z is calculated by solving

$$\frac{dE(\mathbf{w} + Z \cdot \mathbf{p})}{dZ} = 0 \tag{18}$$

for Z and then the weights are updated using

$$\mathbf{w} \leftarrow \mathbf{w} + Z \bullet \mathbf{p} \tag{19}$$

The optimal learning factor Z can be obtained using a Taylor series expansion of $E(\mathbf{w} + Z \bullet \mathbf{p})$ to obtain the solution of (18).

Comparison of Training Techniques

Back propagation training often results in excessive training time and performance may be poor. Newer algorithms such as FCG and OWO-HWO are often preferred to back propagation.

Kim [33-34] compared FCG and OWO-HWO for many different data types. He showed that FCG performs well on random training data and OWO-HWO performs better than FCG on correlated data. OWO-HWO is also invariant to input biases. In

addition, OWO-HWO is able to reach memorization in a fixed number of iterations. In contrast, FCG is dependent on input biases and thus requires normalized training data. FCG also has problems attaining memorization for correlated training data.

2.3 MLP Design Methodologies

When investigators design neural networks for an application, there are many ways they can investigate the effects of network structure which refers to the specification of network size (i.e., number of hidden units) when the number of inputs and outputs are fixed [35]. A well-organized researcher may design a set of different size networks in an orderly fashion, each with one or more hidden units than the previous network. This approach can be designated as Design Methodology One (DM-1). Alternatively, a researcher may design different size networks in no particular order. This approach can be designated as Design Methodology Two (DM-2). These two design methodologies are significantly different and the approach chosen often is based on the goals set for the network design and the associated performance. In general, the more thorough approach often used for DM-1 may take more time to develop the selected network since the researcher may be interested in trying to achieve a trade-off between network performance and network size. However, the DM-1 approach may produce a superior design since the design can be pursued until the researcher is satisfied that further increases in network size produces diminishing returns in terms of decreasing training time or testing errors. In contrast, DM-2 may be quickly pursued for only a few networks and one of these chosen as an adequate network with fewer comparisons to other networks. It is possible this DM-2 network design could have been significantly improved with just a bit more attention to network design.

3.0 Network Initial Types and Training Problems

MLP training is inherently dependent on the initialization of the networks. Training can be improved by proper network initialization. Three distinct types of networks considered are Randomly Initialized (RI) Networks, Common Starting Point Initialized (CSPI) Networks and Dependently Initialized (DI) Networks [35].

Randomly Initialized Networks: When a set of MLPs are RI, no members of the set have any initial weights and thresholds in common. Practically, this means that the Initial Random Number Seeds (IRNS) of the networks are widely separated. RI networks have no initial weights or thresholds in common for networks of the same or different sizes. These networks are useful when the goal is to quickly design one or more networks of the same or different sizes whose weights are statistically independent of each other. RI networks can be designed using DM-1 or DM-2.

Common Starting Point Initialized Networks: When a set of MLPs are CSPI, each one starts with the same IRNS. These networks are useful when it is desired to make performance comparisons of networks that have the same IRNS for the starting point. CSPI networks can be designed using DM-1 or DM-2.

Dependently Initialized Networks: Here a series of different size networks are designed with each subsequent network having one or more hidden units than the previous network. Larger size networks are initialized using the final weights and thresholds that resulted from training a smaller size network. DI networks are useful when the goal is a thorough analysis of network performance versus size and are most relevant to being designed using DM-1.

3.1 Problems with MLP Training

Problems that have been identified with MLP training include:

(a) Non-monotonic $E_f(N_h)$

For well-trained MLPs, $E_f(N_h)$ should decrease as N_h increases. However, in some

instances this does not occur since the initial hidden unit basis functions are different for RI networks. An example of this phenomenon is illustrated in figure 2 for a typical single initial random number seed case. This can also occur for CSPI networks. When investigators analyze MLP performance versus N_h they often use many seeds so that the MSE averaged over a large number of seeds, as illustrated in figure 3, will be more monotonic non-increasing. Developers may also show performance results using the median error, $E_{f med}(N_h)$, out of a set of seeds (figure 3) or the minimum error, $E_{f min}(N_h)$, out of a set of seeds (figure 4). The problem with using the minimum error is that the minimum for one size network in general will not correspond to the same seed as the minimum for a different size network as illustrated in figure 4.
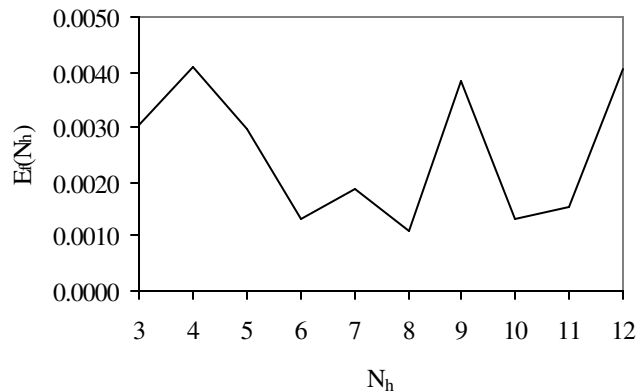

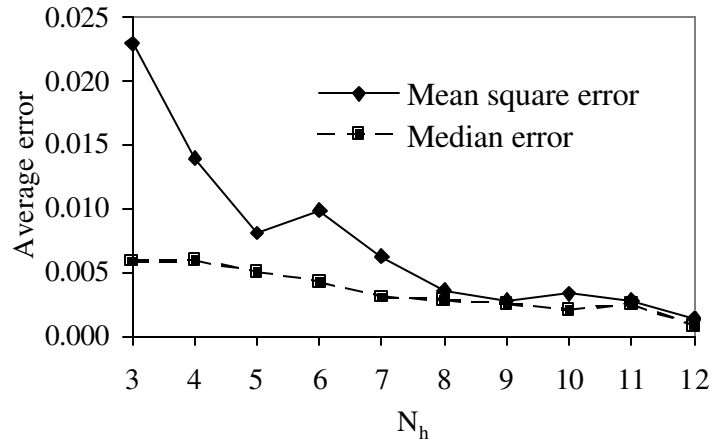Figure 2. Typical single seed mapping error.

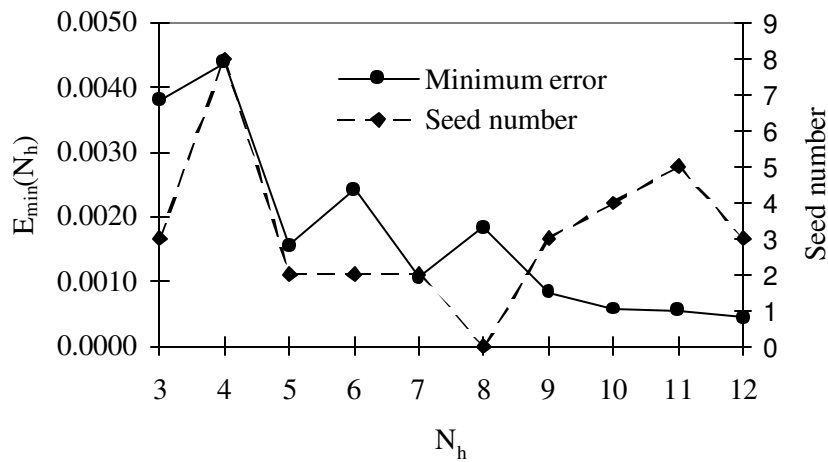Figure 3.   Non-monotonic average error curve.

Figure 4. Minimum error curve.

The results shown in figures 2-4 are for a data set consisting of 5 inputs, 1 output and 1024 patterns to demodulate a frequency modulated signal containing a sinusoid (**fm dataset**).

(b) No standard way to initialize and train additional hidden units

As the size of the MLP is increased by adding new hidden units, a consistent technique for initializing these new units as well as the previous hidden units is needed for CSPI networks. However, no widely accepted technique for doing this has been developed.

(c)Net control parameters are arbitrary

Net control [77] is an effective technique for initializing the MLP. Unfortunately, appropriate values for the desired net function means and standard deviations have not been determined in a systematic manner.

(d) No procedure to initialize and train DI networks

DI networks take advantage of previous training on smaller size networks in a systematic manner. Existing techniques for training and initializing these networks are heuristic in nature.

(e) Network linear and nonlinear component interference

Neural network training algorithms spend a significant amount of their time optimizing the linear output weights. Unfortunately, the performance of the MLP can be affected by simultaneous training of the linear and nonlinear components of the mapping. Techniques which mostly concentrate on determining the nonlinear mapping component are lacking. This topic will not be addressed in this paper, but will be addressed in future papers.

4.0   DI Network Development and Evaluation

DI networks [35] were defined above as a potential improvement to RI and CSPI networks. DI networks are designed such that each larger network is initialized with the final weights and thresholds from a previously well-trained smaller network. These networks are designed with DM-1 where a well-organized researcher designs a set of different size networks that achieve acceptable performance bounds. The larger network may have one or more additional hidden units than the smaller network with the additional hidden units being initialized using random numbers as discussed above. It should be noted that this development is based on the technique of designing an initial single network (i.e., single IRNS) and using this to extend to larger networks. Hence, this produces practical networks that can be readily implemented rather than many networks of each size.

4.1   DI Network Basic Algorithm

These networks build on previously well-trained networks to produce improved designs. 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 [36]. To design a DI network requires an initial starting non-DI network of N_h hidden units. Initial starting networks can have any value for an N_h. After designing the initial network, each subsequent network is a DI network. The

final weights and thresholds, w_f, for the well-trained smaller network of size N_{h-p} are used as the initial weights and thresholds, w_{nt}, for the larger DI network of size N_h. This initialization technique applies whether the network of size N_{h-p} is the initial starting non-DI network or a previously trained DI network.

4.2   Properties of DI Networks

Based upon the DI network design technique several very useful properties can be asserted.

(1) $E_{int}(N_h) < E_{int}(N_{h-p})$
(2) The $E_f(N_h)$ curve is monotonic non-increasing (i.e., $E_f(N_h) = E_f(N_{h-p})$).
(3) $E_{int}(N_h) = E_f(N_{h-p})$

Proofs and justifications of these properties will be presented in future papers.

4.3 Performance Results for DI Networks for Fixed Iterations

Performance results for DI networks [36] are presented in this section for training with a fixed number of iterations ($N_{iter} = 250$). These results are for the case where all weights and thresholds are retrained as new hidden units are added to the network. These techniques are applicable to adding any number of new hidden units, but the results are for cases where the number of hidden units is incremented by one for each subsequent network with the initial non-DI network being designed with one hidden unit. Since these networks are designed using a single IRNS, testing results are presented in addition to the training results.

fm: Results for the fm data set are shown in figure 5. This DI network has a monotonic non-increasing $E_f(N_h)$ as N_h increases during training. Testing performance for this network is also good being very consistent and tracking the training error very well. These results also show that a value of N_h of nine is the maximum value that results in performance improvement. Additional results for several other datasets will be presented in future papers. Additional results for several other datasets will be presented in future papers.
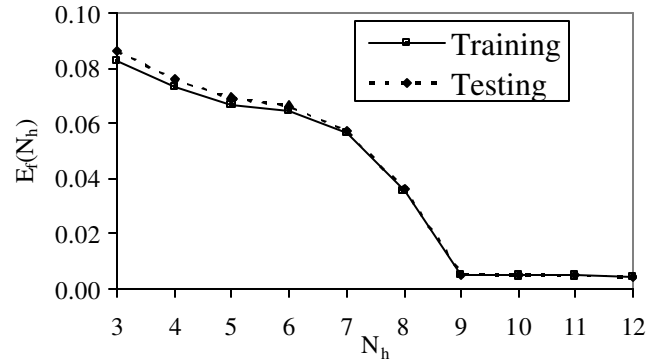
Figure 5. Training and testing for fm data set.

5.0 Conclusions and Future Papers

This paper presented a technique that significantly improves the training performance of multilayer perceptron neural networks. This basically analyzed problems with MLP neural networks developed a dependently initialized training enhancement that ensures that the MSE versus N_h curves are always monotonic non-increasing

Future related papers will discuss a monotonicity analysis developed for MLP networks. In addition investigation into net control improvements will be presented, particularly in regard to the initial network design for DI networks. More quantification of a precise hybrid approach for training both the new and previous hidden units will be . These techniques will also be applied to a separating mean technique where the means of similar units are remove prior to network training. Rules will also be developed to determine when adding additional hidden units is not warranted based on the additional reduction in the $E_f(N_h)$ when the network size increases.

References:

[1]   R. P. Lippman, "An introduction to computing with neural networks," *IEEE ASSP Magazine*, April 1987.

[2]   T. Kohonen, *Self Organization and Associative Memory*, Springer-Verlag, 2nd edition, 1988.

[3]   J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities, *Proceedings of the National Academy of Sciences of the U.S.A.*, pp. 2554-2558, 1982.

[4]   G. Galan-Marin and J. Perez, "Design and analysis of maximum Hopfield networks," *IEEE Transactions on Neural Networks*," vol. 12 no. 2, pp. 329-339, March 2001.

[5]   M. J. D. Powell, "Radial basis functions for multivariable interpolation: A review, "*IMA Conference on Algorithms or the Approximation of Functions*, pp. 143-167, RMCS, Shrivenham, U. K., 1985.

[6]   C. Panchapakesan, M. Palaniswami, D. Ralph and C. Manzie, "Effects of moving the centers in an RBF network," *IEEE Transactions on Neural Networks*, vol. 13, no. 6, pp. 1299-1307, November 2002.

[7]   G. E. Hinton and T. J. Sejnowski, "Learning and relearning in Boltzmann machines," *Parallel Distributed Processing: Explorations in Microstructure of Cognition*, MIT Press, Cambridge, MA., 1986.

[8]   C. Peterson and J. R. Anderson, "A mean field theory learning algorithm for neural networks," *Complex Systems*, 1, pp. 995-1019, 1987.

[9]   H. Chandrasekaran, *Analysis and Convergence Design of Piecewise Linear Networks*, Ph.D. Dissertation, The University of Texas at Arlington, 2000.

[10]   K. K. Kim, *A Local Approach for Sizing the Multilayer Perceptron*, Ph.D. Dissertation, The University of Texas at Arlington, 1996.

[11]   M. S. Chen and M. T. Manry, "Power series analysis of back-propagation neural networks," *Proceedings of IJCNN '91*, Seattle, WA., pp. I-295 to I-300, 1991

[12]   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.

[13]   Z. J. Yang, "Hidden-layer size reducing for multilayer neural networks using the orthogonal least squares method," *Proceedings of the 1997, 36th IEEE Society of Instruments and Control Engineers (SICE) Annual Conference*, pp. 1089-1092, July 1997.

[14]   M. A. Satori and P. J. Antsaklis, "A simple method to derive the bounds on the size to train multilayer neural networks," *IEEE Transactions on Neural Networks*, Vol. 2, No. 4, pp. 467-471, July 1991.

[15]   M. S. Chen, *Analysis and Design of the Multilayer perceptron Using Polynomial Basis Functions*, Ph.D. Dissertation, The University of Texas at Arlington, 1991.

[16]   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 and Computers, Vol. 1, pp. 386-390, November 1993.

[17] M. T. Manry, H. Chandrasekaran and C. Hsieh, "Signal processing using the multilayer perceptron," *Handbook of Neural Network Signal Processing*, edited by Y. H. Hu and J. Hwang, pp. 2-1 - 2-29, CRC Press, 2001.

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

[19] 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.

[20] G. Cybenko, "Approximation by superposition of a sigmoidal function," *Mathematics of Control, Signals and Systems*, Vol. 2, No. 4, pp. 303-314, 1989.

[21] J. Olvera, *Monomial Activation Functions for the Multi-Layer Perceptron*, M.S. Thesis, The University of Texas at Arlington, 1992.

[22] V. P. Plagianakos, G. D. Magoulas and M. N. Vrahatis, "Deterministic nonmonotone strategies for efficient training of multilayer perceptrons," *IEEE Transactions on Neural Networks*, vol. 13, no. 6, pp. 1268-1284, November 2002.

[23] H. Chen, "*Neural Network Training and Pruning Using Multiple Regressions*," Ph.D. Dissertation, The University of Texas at Arlington, 1997.

[24] F. J. Maldonado and M. T. Manry, "Optimal pruning of feedforward neural networks based upon the Schmidt procedure," *Thirty-Sixth Annual Asilomar Conference on Signals, Systems and Computers*, pp. 1024-1028, Pacific Grove, CA, 3-6 November 2002.

[25] E. M. Johansson, F. U. Dowla 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.

[26] P. Werbos, *Beyond regression: NewTools for Prediction and Analysis in the Behavioral Sciences*, Ph.D. Dissertation, Committee on Applied Mathematics, Harvard University, Cambridge, MA. 1974.

[27] D. B. Parker, "Learning logic," Invention Report S81-64, File 1, Office of Technology Licensing, Stanford University, 1982.

[28] D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning internal representations by error propagation," in D. E. Rumelhart and J. L. McClelland (Eds.), *Parallel Distributed Processing*, Vol. I, Cambridge, MA, The MIT Press, 1986.

[29] R. Fletcher and C. M. Reeves, "Function minimization by conjugate gradients," *Computer Journal*, Vol. 7, pp. 149-154, 1964.

[30] R. Fletcher, "Conjugate direction methods," in *Numerical Methods for Unconstrained Optimization*, W. Murray editor, Academic Press, pp. 73-86, 1972.

[31] C. Yu and M. T. Manry, "A modified hidden weight optimization algorithm for feed-forward neural networks," *Thirty-Sixth Annual Asilomar Conference on Signals, Systems and Computers*, pp. 1034-1038, Pacific Grove, CA, 3-6 November 2002.

[32] I. S. Sokolnikoff and R. M. Redheffer, *Mathematics of Physics and Modern Engineering*, McGraw-Hill, 2nd edition, 1966.

[33] T. Kim, *Development and Evaluation of Multilayer Perceptron Training Algorithms*, Ph.D. Dissertation, The University of Texas at Arlington, 2001.

[34] T. Kim, J. Li and M. T. Manry, "Evaluation and improvement of two training algorithms," *Thirty-Sixth Annual Asilomar Conference on Signals, Systems and Computers*, pp. 1019-1023, Pacific Grove, CA, 3-6 November 2002.

[35] W. H. Delashmit, *Multilayer Perceptron Structured Initialization and Separating Mean Processing*, Ph.D. Dissertation, University of Texas at Arlington, May 2003.

[36] W. H. Delashmit and M. T. Manry, "Enhanced robustness of multilayer perceptron training," *Thirty-Sixth Annual Asilomar Conference on Signals, Systems and Computers*, pp. 1029-1033, Pacific Grove, CA, 3-6 November 2002.