

# Dynamics of Continuous, Discrete & Impulsive Systems

Series A: Mathematical Analysis

**Editor-in-Chief**

**Xinzhi Liu, University of Waterloo**

*Special Issue on*

*Advances in Neural Networks--Theory and Applications*

*Part 1*

DCDIS 14(S1) 1-502 (2007)

ISSN 1201-3390

---

**Watam Press • Waterloo**

# A Neural Network Growing Algorithm that Ensures Monotonically Non Increasing Error

Walter H. Delashmit  
 WHD Consulting, Justin Texas

Michael T. Manry  
 University of Texas at Arlington, Arlington, Texas

AMS subject classifications:: 40-04, 41-04, 90-04

**Abstract::** Since neural network training is very sensitive to the choice of the initial weights and thresholds, multilayer perceptron training error usually fails to be a monotonically non-increasing function of the number of hidden units. A new network growing algorithm was developed based on a dependently initialized network structure in which weights and thresholds from a well-trained smaller network are used to initialize a larger network. It is shown that this technique yields an error curve that is a monotonically non-increasing function of the number of hidden units. Theoretical results for this technique are presented as well as experimental simulation results.

## 1 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 an increasing number of hidden units. These “ad hoc” solutions do not significantly mitigate the problems with MLP training.

In this paper we address solutions to these problems. In section 2, the basics of MLPs are presented. In section 3, a formulation of the problem is presented. In section 4, dependently initialized networks are defined and properties of these networks are developed. In section 5, simulation results are presented for a variety of data sets. A summary of the paper is presented in section 6. References are presented in section 7.

## 2 Multilayer Perceptron Fundamentals

MLP neural networks consist of units arranged in layers. Each layer is composed of nodes and each node connects to every node in the subsequent layer. 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. For every connection from one node to another node, weights modify the outputs from the preceding layer. The input layer distributes the inputs to the subsequent layers and use 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 subsequent node on a subsequent layer has the original input multiplied by a weight with a threshold added and

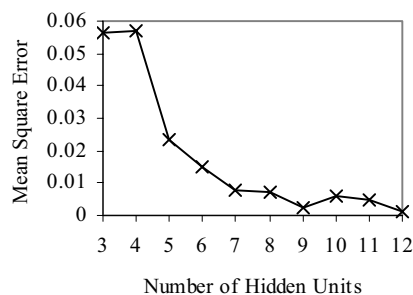


Figure 1. Typical MLP nonmonotonic error curve

is then is passed through an activation function that may be linear or nonlinear (hidden units). A typical three-layer network is shown in Fig. 2. Only three layer MLPs will be considered in this dissertation since these networks have been shown to approximate any continuous function [1-3]. 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 Fig. 2 to simplify the diagram.

The training data consists of a set of  $N_v$  training patterns ( $x_p, t_p$ ) where  $p$  represents the pattern number. In Fig. 2,  $x_p$  corresponds to the  $N$ -dimensional input vector of the  $p$ th training pattern and  $y_p$  corresponds to the  $M$ -dimensional output vector from the trained network for the  $p$ th pattern. For ease of notation and analysis, thresholds on the hidden units and output units are handled by augmenting the input vector by assigning the value of one to the augmented vector component denoted by  $x_p(N+1)$ . The output and input units have linear activations. The input to the  $j$ th hidden unit,  $net_p(j)$  is expressed by

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

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

$$O_p(j) = f(\text{net}_p(j)) \quad (2)$$

The nonlinear activation is typically chosen to be the sigmoidal function

$$f(\text{net}_p(j)) = \frac{1}{1 + e^{-\text{net}_p(j)}} \quad (3)$$

In (1), the  $N$  input units are represented by the index  $k$  and  $w(j,k)$  denotes the weights connecting the  $k$ th input unit to the  $j$ th hidden unit. Similarly, when subscripts are used on the weight functions such as,  $w_{ij}$  the second subscript is the source node and the first subscript is the destination node.

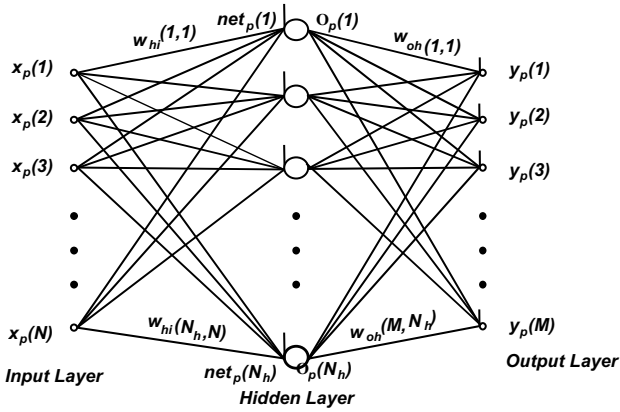


Figure 2. Typical three-layer multilayer perceptron neural network.

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

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

where  $E_p$  corresponds to the error for the  $p$ th pattern and  $t_p$  is the desired output for the  $p$ th pattern. This also allows the calculation of the mapping error for the  $i$ th output unit to be expressed by

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

with the  $i$ th output for the  $p$ th training pattern being expressed by

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

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.

Some of the problems in training multilayer perceptron neural networks have been resolved by using net control processing that was developed by Olvera [4]. This technique was designed to make the network weight behavior more stable. Bounded activations tend to

have small gradients when the average net input function has a large mean. This can result in network training being slow or ineffective. In addition, large input values can dominate the performance and reduce the effects of smaller inputs when significant differences in the mean values of the inputs exist. This occurs even though the inputs with smaller values may actually be more important to the overall performance of the system. Net control aids in controlling these problems by scaling and shifting all net functions so that they do not generate small gradients for bounded activations and do not allow large signals to mask the potential effects of small signals. This process adjusts each net function mean and standard deviation to a user defined standard deviation  $\sigma_{hd}$  and user defined 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  $\sigma_{hd}$ . This requires determining the mean  $m_h(j)$  and the standard deviation  $\sigma_h(j)$  of each of the hidden unit net functions for all patterns. 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

$$w_{hi}(j,i) \leftarrow \frac{w_{hi}(j,i) \cdot \sigma_{hd}}{\sigma_h(j)} \quad (7)$$

with the hidden unit thresholds adjusted using

$$w_{hi}(j,N+1) \leftarrow w_{hi}(j,N+1) + m_{hd} - \frac{m_h(j) \cdot \sigma_{hd}}{\sigma_h(j)} \quad (8)$$

### 3 Problem Formulation

MLP training [5] 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 [6]. 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 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.

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

$$E_{int}(0) \geq E_{int}(1) \geq E_{int}(2) \geq \dots \geq E_{int}(N_{hmax}) \quad (9)$$

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

$$E_f(0) \geq E_f(1) \geq E_f(2) \geq \dots \geq E_f(N_{hmax}) \quad (10)$$

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.

## 4 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. This technique is applicable to networks that are trained using output-weight-optimization/hidden weight optimization (OWO-HWO) [6], the Levinson-Marquadt algorithm [7], backpropagation (BP) [8] and the various modifications to BP.

The flowchart for the basic DI network algorithm is shown in Fig. 3. These networks [9, 10] 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. 3,  $RN(ind+)$  is an array of random numbers used to store the initial weights and thresholds with an index of  $ind+$  used to indicate that the next random number from this array is chosen for the specified specified weights or thresholds being initialized

The initialization technique shown in Fig. 3 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

$$E_{int}(N_h) = \frac{1}{N_v} \sum_{p=1}^{N_v} E_p = \frac{1}{N_v} \sum_{p=1}^{N_v} \sum_{l=1}^M [t_p(i) - y_{p,int}(i, N_h)]^2 \quad (11)$$

$$y_{p,int}(i, N_h) = \sum_{k=1}^{N+1} w_{f,oi}(i, k) \cdot x_p(k) + \sum_{j=1}^{N_h-p} w_{f,oh}(i, j) \cdot O_{p,f}(j) + \sum_{j=N_h-p+1}^{N_h} w_{int,oh}(i, j) \cdot O_{p,int}(j) \quad (12)$$

$$y_{p,int}(i, N_h) = y_{p,f}(i, N_{h-p}) = \sum_{k=1}^{N+1} w_{f,oi}(i, k) \cdot x_p(k) + \sum_{j=1}^{N_h-p} w_{f,oh}(i, j) \cdot O_{p,f}(j) \quad (13)$$

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

$$net_p(j) = \sum_{k=1}^{N+1} w_{hi}(j, k) \cdot x_p(k) \quad 1 \leq j \leq N_h \quad (14)$$

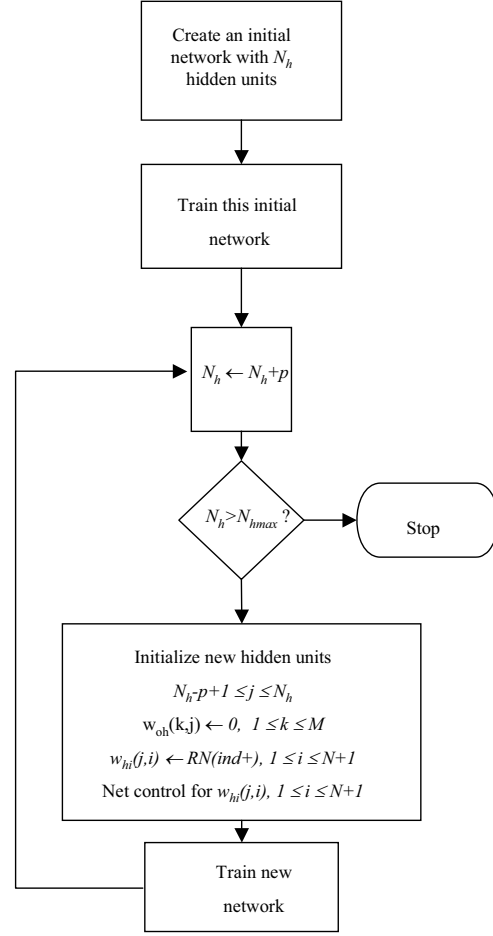


Fig. 3. DI network design flowgraph.

**Properties of DI Networks:** Based upon the design technique of Fig. 3, 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. 3 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 monotonically 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. 3 and (11) - (13) where the output weights and thresholds for the larger  $N_h$  network are initialized to zero resulting in  $y_{p,int}(i, N_h) = y_{p,f}(i, N_{h-p})$ .

## 5 Performance Results for DI Networks

Performance results for DI networks [9] are presented in this section for training with a fixed number of iterations (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 seed, testing results are presented in addition to the training results.

**fm:** This data set [11] is used to perform demodulation of a frequency modulation (FM) signal containing a sinusoidal message. This data set has 5 inputs and 1 output and consists of 1024 training patterns and 1024 testing patterns. The data are generated from the equation

$$r(n) = C_{amp} \cdot \cos[2\pi \cdot n \cdot C_{freq}] + (M_{amp} \cdot \sin(2\pi \cdot n \cdot M_{freq}))$$

where  $C_{amp}$  is the carrier amplitude,  $M_{amp}$  is the message amplitude,  $C_{freq}$  is the carrier frequency and  $M_{freq}$  is the normalized message frequency. In this data set,  $C_{amp} = 0.5$ ,  $M_{amp} = 5.0$ ,  $C_{freq} = 0.1012878$  and  $M_{freq} = 0.01106328$ . In each consecutive pattern,  $n$  is incremented by one.

Results for the fm data set are shown in Fig. 4. 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 improvements.

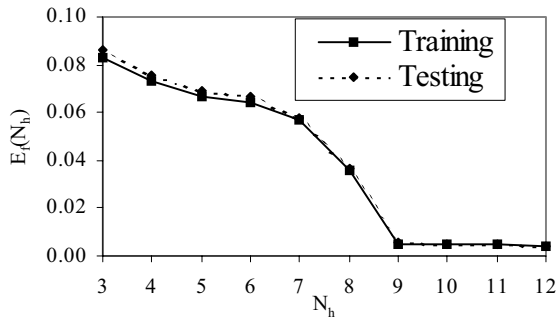


Fig. 4. Training and testing for fm data set.

**twod:** This data set [11] consists of eight inputs and seven outputs and was split into a training file consisting of 1,768 patterns and a testing file consisting of 1,000 patterns. This data is used in the process of inverting the surface scattering parameters from an inhomogeneous layer above a homogeneous half space, where both interfaces are randomly rough. The parameters to be inverted are the effective permittivity of the surface, the normalized root mean square (rms) height, the normalized surface correlation length, the optical depth, and single scattering albedo of an inhomogeneous irregular layer above a homogeneous half space from back scattering measurements. The inputs consist of eight theoretical values of back scattering coefficient parameters at vertical and horizontal polarization and four incident angles. The outputs are the corresponding values of permittivity, upper surface height, lower

surface height, normalized upper surface correlation length, normalized lower surface correlation length, optical depth and single scattering albedo which had a joint uniform probability density function.

Results for the twod data set are shown in Fig. 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.

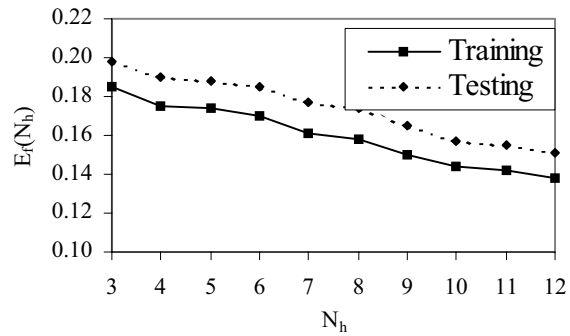


Fig. 5. Training and testing for twod data set.

**oh7:** This data set [11] contains VV and HH polarization at L of thirty and forty degrees, C of ten, thirty, forty, fifty, and 60 degrees and X of thirty, forty, and fifty degrees along with the corresponding unknowns of rms surface height, surface correlation length and volumetric soil moisture content. This data set was randomly split into a training data set consisting of 2,595 patterns and a testing data set consisting of 7,858 patterns.

Results for the oh7 data set are shown in Fig. 6. 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. For this data set, the testing error eventually starts to diverge slightly and values of  $N_h$  greater than five hidden units but the total divergence is less than 4.75%. No significant improvement in testing or training performance is obtained for  $N_h$  greater than five hidden units.

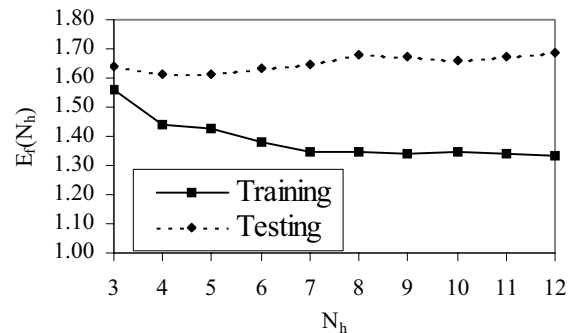


Fig. 6. Training and testing for oh7 data set.

**F24:** This data set [11] consists was randomly split into 1,582 training patterns and 3,163 testing patterns for onboard flight load synthesis in helicopters. There are 24 inputs and 9 outputs for each pattern. This data was obtained from the M430 flight load level survey conducted in Mirabel Canada in early 1995 by Bell Helicopter Textron of Fort Worth.

Results for the F24 data set are shown in Fig. 7. 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.

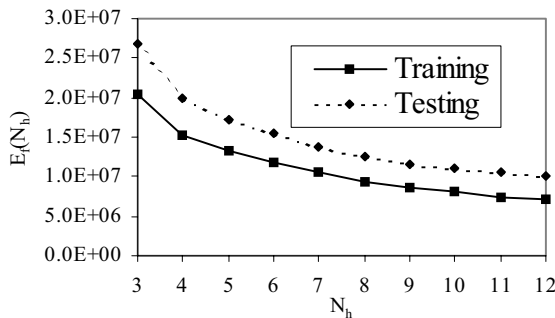


Fig. 7. Training and testing for F24 data set.

**F17:** This data set [11] consists was randomly split into 1,582 training patterns and 3,163 testing patterns for onboard flight load synthesis in helicopters. There are 17 inputs and 9 outputs for each pattern. This data was obtained from the M430 flight load level survey conducted in Mirabel Canada in early 1995 by Bell Helicopter Textron of Fort Worth.

Results for the F17 data set are shown in Fig. 8. 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.

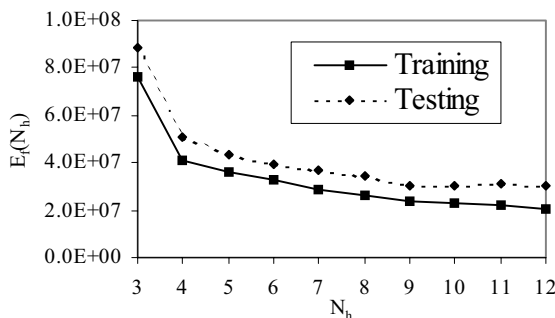


Fig. 8. Training and testing for F17 data set.

## 6 Summary

Dependently initialized networks were introduced and growing and training algorithms were developed for these networks. These networks significantly improve performance in terms of ensuring that the training error is a monotonically non-increasing function of the number of hidden units as the network size increases.

## 7 References

- [1] K. Hornik, M. Stinchcombe and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, Vol. 2, No. 5, pp. 359-366, 1989.
- [2] 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.
- [3] G. Cybenko, "Approximation by superposition of a sigmoidal function," *Mathematics of Control, Signals and Systems*, Vol. 2, No. 4, pp. 303-314, 1989.
- [4] J. Olvera, Monomial Activation Functions for the Multi-Layer Perceptron, M.S. Thesis, The University of Texas at Arlington, 1992.
- [5] W. H. Delashmit and M. T. Manry, "Enhanced Robustness of Multilayer Perceptron Training," *Proceedings of the Thirty-Sixth Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, pp. 1029-1033, 3-6 November 2002.
- [6] H. H. Chen, M. T. Manry, and Hema Chandrasekaran, "A Neural Network Training Algorithm Utilizing Multiple Sets of Linear Equations", *Neurocomputing*, Vol. 25, No. 1-3, pp. 55-72, April 1999.
- [7] Neural Networks Toolbox (4.0) User Guide, The Math Works, Natick, MA 2000.
- [8] 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.
- [9] W. H. Delashmit, "Multilayer Perceptron Structured Initialization and Separating Mean Processing," *Ph.D. Dissertation*, University of Texas at Arlington, May 2003.
- [10] W. H. Delashmit and M. T. Manry, "New Training Algorithms for Dependently Initialized Multilayer Perceptrons," *Proceedings of the Thirty-Seventh Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, pp. 581-585, 9-12 November 2003.
- [11] [http://www-ee.uta.edu/eeweb/IP/training\\_data\\_files.htm](http://www-ee.uta.edu/eeweb/IP/training_data_files.htm)