

Finding Optimal Neural Network Basis Function Subsets Using the Schmidt Procedure.

F. J. Maldonado¹, M. T. Manry², and Tae-Hoon Kim²

¹Chihuahua Institute of Technology, Av. Tecnológico 2909, Chihuahua, Chih., México, 31310. E-mail: fmalдона@platon.itchihuahua.edu.mx

²University of Texas at Arlington, Electrical Engineering Dept., Nedderman Hall, Rm 501, 416 Yates Street Arlington, TX 76010. E-mail: manry@uta.edu

Abstract- In designing feed forward neural networks, one often trains a large network and then prunes less useful hidden units. In this paper, two non-heuristic pruning algorithms are derived from the Schmidt procedure. In both, orthonormal systems of basis functions are found, ordered, pruned, and mapped back to the original network. In the first algorithm, the orthonormal basis functions are found and ordered one at a time. In optimal pruning, the best subset of orthonormal basis functions is found for each size network. Linear dependency of basis functions is considered and computational cost is analyzed. Simulation results are given.

I. INTRODUCTION

Two common approaches for obtaining neural network structures are growing methods and pruning methods. New hidden units are added during the training process [2][7] in growing methods. A drawback of growing methods is that the network can get trapped in local minima and they are sensitive to initial conditions. In pruning methods, a large network is trained and then less useful nodes or weights are removed [1][4][6][9]. Kaminsky and Strumillo [12] have developed a pruning algorithm for Radial Basis Function networks that is based upon the Schmidt procedure. Unfortunately, their procedure is difficult to apply to fully connected multilayer perceptrons (MLPs), which have weights connecting the input and output layers. Pruning methods for fully-connected MLPs are developed in this paper.

The organization of this paper is as follows. In section 2 the MLP structure and notation are given. In section 3 a modified Schmidt process is presented as well as necessary weight transformations. Sections 4 and 5 explain basic pruning and the linear dependency condition respectively. Optimal pruning is presented in section 6. Results are given in section 7. Final comments and conclusions are given in section 8.

II. MULTILAYER PERCEPTRON STRUCTURE

A feedforward MLP, having one hidden layer with N_h nonlinear units and an output layer with M linear units is depicted in Figure 1.

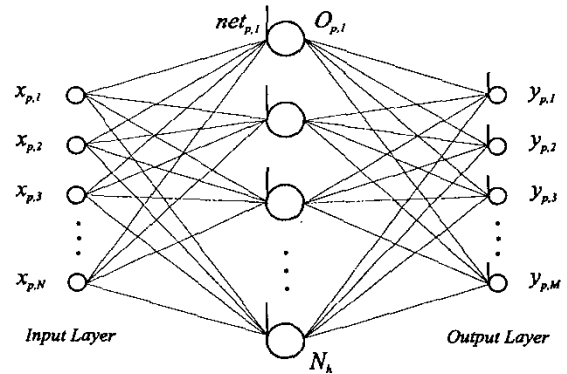


Fig. 1. MLP with one hidden layer

The net value ($net_{p,j}$) and the output value ($O_{p,j}$) for the j^{th} -hidden unit for the p^{th} training pattern are defined as $O_{p,j} = f(net_{p,j})$ with

$$net_{p,j} = \sum_{i=1}^{N+1} w(j,i) \cdot x_{p,i} \quad 1 \leq p \leq N_p, \quad 1 \leq j \leq N_h. \quad (1)$$

Here the threshold of the j^{th} node is handled by letting $x_{p,N+1}$ be one. Weight $w(j,i)$ connects the i^{th} input to the j^{th} hidden unit. For the MLP the most common activation functions are the sigmoid function for hidden layers and linear functions for the output layer. In a two layer MLP the j^{th} output in the hidden layer is given by,

$$O_{p,j} = f(net_{p,j}) = 1 / (1 + e^{-net_{p,j}}) \quad (2)$$

The k^{th} output for the p^{th} pattern is,

$$y_{p,k} = \sum_{i=1}^{N+1} w_o(k,i) \cdot x_{p,i} + \sum_{j=1}^{N_h} w_o(k, j+N+1) \cdot O_{p,j}, \quad (3)$$

where $1 \leq k \leq M$. There are N_p training patterns denoted by $\{ (x_p, t_p) \}_{p=1}^{N_p}$ where each pattern consists in an input vector x_p and a desired output vector t_p . For the p^{th} pattern, the N input values are $x_{p,i}$ ($1 \leq i \leq N$) and the M desired output values are $t_{p,k}$ ($1 \leq k \leq M$). The mapping error for the p^{th} pattern is

$$E_p = \sum_{k=1}^M [t_{p,k} - y_{p,k}]^2 \quad (4)$$

In order to train a neural network, for one epoch the mapping error for the i^{th} output unit is defined as

$$E(i) = \frac{1}{N_v} \sum_{p=1}^{N_v} [t_{pi} - y_{pi}]^2 \quad (5)$$

The Mean Square Error (MSE) is

$$E = \sum_{i=1}^M E(i) = \frac{1}{N_v} \sum_{p=1}^{N_v} E_p \quad (6)$$

III. SCHMIDT PROCEDURE FOR NEURAL NETS

Rewriting equation (3), we can get the output of the network as

$$y_i = \sum_{k=1}^{N_u} w_o(i, k) \cdot x_k \quad (7)$$

where $x_k = O_p(k, N-1)$ for $N+1 < k \leq N_u$ where N_u is the total number of units equal to $N + N_h + 1$. In equation (7), the signals x_k are the raw basis functions for producing y_i .

The normal Gram-Schmidt procedure [11] is a recursive process that requires obtaining scalar products between raw basis functions and orthonormal basis functions. The disadvantage in this process is that it requires one pass through the training data to obtain each new basis function. In this section a more useful form of the Schmidt process is reviewed, which will let us express the orthonormal system in terms of autocorrelation elements.

The m^{th} orthonormal basis function x_m' , can be expressed as

$$x_m' = \sum_{k=1}^m a_{mk} x_k \quad (8)$$

From equation (8), for $m = 1$, the first basis function is obtained as

$$x_1' = \sum_{k=1}^1 a_{1k} x_k = a_{11} x_1 \quad (9)$$

$$a_{11} = 1/\|x_1\| = 1/r(1,1)^{1/2} \quad (10)$$

where

$$r(i, j) = \langle x_i, x_j \rangle = \left(\frac{1}{N_v} \right) \sum_{p=1}^{N_v} x_{pi} x_{pj} \quad (11)$$

For values of m between 2 and N_u , c_i is first found for $1 \leq i \leq m-1$ as

$$c_i = \sum_{q=1}^i a_{iq} r(q, m) \quad (12)$$

Then m coefficients b_k are obtained as,

$$\begin{cases} b_k = -\sum_{i=k}^{m-1} c_i a_{ik} & 1 \leq k \leq m-1 \\ b_m = 1 \end{cases} \quad (13)$$

Finally for the m^{th} basis function the new a_{mk} coefficients (for $1 \leq k \leq m$) are found as

$$a_{mk} = \frac{b_k}{\left[r(m, m) - \sum_{i=1}^{m-1} c_i^2 \right]^{1/2}} \quad (14)$$

Equating y_i in (7) to

$$y_i = \sum_{q=1}^{N_u} w_o(i, q) x_q' \quad (15)$$

where the weights in the orthonormal system are

$$w_o(i, q) = \sum_{k=1}^q a_{qk} \langle x_k, t_i \rangle = \sum_{k=1}^q a_{qk} c(i, k) \quad (16)$$

and using (8) we obtain output weights for the system as

$$w_o(i, k) = \sum_{q=k}^{N_u} w_o(i, q) a_{qk} \quad (17)$$

Substituting (15) into (5) we obtain $E(i)$ in the orthonormal system as

$$E(i) = \left\langle \left(t_i - \sum_{k=1}^{N_u} w_o(i, k) x_k' \right), \left(t_i - \sum_{q=1}^{N_u} w_o(i, q) x_q' \right) \right\rangle \quad (18)$$

If we decide to use the first N_{hd} hidden units in our original network, the training error is

$$E(i) = \langle t_i, t_i \rangle - \sum_{k=1}^{N+1+N_{hd}} (w_o(i, k))^2 \quad (19)$$

Modifying (17) the output weights would be

$$w_o(i, k) = \sum_{q=k}^{N+1+N_{hd}} w_o(i, q) a_{qk} \quad (20)$$

IV. ORDERED PRUNING

The purpose of pruning is to eliminate less useful inputs and hidden units which (1) have no information relevant for estimating outputs or (2) are linearly dependent on inputs or hidden units that have already been orthonormalized. In section 3, no attempt is made to order the hidden units according to their usefulness. In this section we modify the Schmidt procedure so that during pruning useless basis functions x_m' are eliminated.

Let $j(m)$ be an integer valued function that specifies the order in which raw basis functions x_k are processed into orthonormal basis functions x_k' . Then x_m' is to be calculated from $x_{j(m)}$, $x_{j(m-1)}$ and so on. This function also defines the structure of the new hidden layer where $1 \leq m \leq N_u$ and $1 \leq j(m) \leq N_u$. If $j(m) = k$ then the m^{th} unit of the new structure comes from the k^{th} unit of the original structure.

Given the function $j(m)$, and generalizing the orthonormalization procedure described in section III, the m^{th} orthonormal basis function is described as

$$x_m' = \sum_{k=1}^m a_{mk} x_{j(k)} \quad (21)$$

Initially, x_1' is found as $a_{11} x_{j(1)}$ where

$$a_{11} = 1/\|x_{j(1)}\| = 1/r(j(1), j(1))^{1/2} \quad (22)$$

For $2 \leq m \leq N_u$, we first perform

$$c_i = \sum_{q=1}^i a_{iq} r(j(q), j(m)) \quad (23)$$

for $1 \leq i \leq m-1$. Second, we set $b_m = 1$ and get

$$b_k = -\sum_{i=k}^{m-1} c_i a_{ik} \quad (24)$$

for $1 \leq k \leq m-1$. Lastly, we get coefficients a_{mk} as

$$a_{mk} = \frac{b_k}{\left[r(j(m), j(m)) - \sum_{i=1}^{m-1} c_i^2 \right]^{1/2}} \quad (25)$$

for $1 \leq k \leq m$. $w_o'(i, k)$ is found as

$$w_o'(i, m) = \sum_{k=1}^m a_{mk} c(i, j(k)) \quad 1 \leq i \leq M \quad (26)$$

The goal of ordered pruning is to find the function $j(m)$ which defines the structure of the hidden layer. Here it is assumed that the original basis functions are linearly independent i.e. the denominator of equation (25) is not zero.

Since we want the effects of inputs and the constant "1" to be removed from orthonormal basis functions, the first $N+1$ basis functions are picked as,

$$j(m) = m \quad \text{for } 1 \leq m \leq N+1 \quad (27)$$

The selection process will be applied to the hidden units of the network. We now define notation that helps us specify the set of candidate basis function to choose in a given iteration. First, define $S(m)$ as the set of indices of chosen basis functions where m is the number of units of the current network (i.e. the one that the algorithm is processing). Then $S(m)$ is given by

$$S(m) = \begin{cases} \{\emptyset\} & \text{for } m=0 \\ \{j(1), j(2), \dots, j(m)\} & \text{for } 0 < m \leq N_u \end{cases} \quad (28)$$

Starting with an initial linear network having 0 hidden units, where m is equal to $N+1$, the set of candidate basis functions is clearly $S^c(m) = \{1, 2, 3, \dots, N_u\} - S(m)$, which is $\{N+2, N+3, \dots, N_u\}$. For $N+2 \leq m \leq N_u$, we obtain $S^c(m-1)$. For each trial value of $j(m) \in S^c(m-1)$ we perform operations (23), (24), (25), and (26). Then $P(m)$ is

$$P(m) = \sum_{i=1}^M [w_o'(i, m)]^2 \quad (29)$$

The trial value of $j(m)$ that maximizes $P(m)$ is found. Assuming that $P(m)$ is maximum when testing the i^{th} element, then $j(m) = i$. $S(m)$ is updated as

$$S(m) = S(m-1) \cup \{j(m)\} \quad (30)$$

Then for the general case the candidate basis functions are, $S^c(m-1) = \{1, 2, 3, \dots, N_u\} - \{j(1), j(2), \dots, j(m-1)\}$ with $N_u - m + 1$ candidate basis function. By using equation (29) after testing all the candidate basis function, $j(m)$ takes its value and $S(m)$ is updated according to equation (30). Defining N_{hd} as the desired number of units in the hidden layer, the process is repeated until $m = N+1 + N_{hd}$. Then the orthonormal weights are mapped to normal weights according to equation (20). Considering the final value of function $j(m)$ row reordering of the original input weights matrix is performed for generating the right O_{pj} values when applying equation (7). After reordering the rows,

because only the N_{hd} units are kept then the remaining units (O_{pj} with $N_{hd} < j \leq N_h$) are pruned by deleting the last $N_h - N_{hd}$ rows.

V. LINEAR DEPENDENCY CONDITION

Unfortunately, ordered pruning by itself is not able to handle linearly dependent basis functions. A minor modification is necessary. Assume that raw basis function $x_{j(m)}$ is linearly dependent on previously chosen basis functions, where $j(m)$ denotes an input ($1 \leq m \leq N$) and $j(m)$ has taken on a trial value. Then

$$x_{j(m)} = \sum_{k=1}^{m-1} d_k x_k \quad (31)$$

Now the denominator of a_{mk} in (25) can be rewritten as

$$g = \langle z_m, z_m \rangle^{1/2} \quad (32)$$

where

$$z_m = x_{j(m)} - \sum_{k=1}^{m-1} \langle x_k, x_{j(m)} \rangle x_k \quad (33)$$

Substituting (31) into (33), however, we get

$$\langle x_k, x_{j(m)} \rangle = d_k \quad (34)$$

and z_m and g are both zero.

If $j(m)$ denotes an input, and g satisfies

$$g < \epsilon \quad (35)$$

for $\epsilon = 10^{-27}$, then we perform

$$j(k) = k \quad (36)$$

for $1 \leq k < m$, and

$$j(k) = k+1 \quad (37)$$

for $m \leq k \leq N$. In effect we decrease N by one and let the $j(k)$ function skip over the linearly dependent input. If $j(m)$ denotes a hidden unit, the same procedure is used to determine whether or not $x_{j(m)}$ is useful. If $x_{j(m)}$ is found to be linearly dependent, the current, bad value of $j(m)$ is discarded before a_{mk} is calculated.

VI. OPTIMAL PRUNING

Unfortunately the best subset of N_{hd} hidden units may not be a subset of the best subset of $N_{hd} + 1$ hidden units. In other words it is possible that $S(m) \not\subset S(m+1)$. Therefore ordered pruning is not optimal.

As in the basic pruning case the optimal pruning algorithm transforms the first $N+1$ units (inputs and threshold) into orthonormal basis functions without reordering, so that $j(m) = m$ for $1 \leq m \leq N+1$. Then the first $N+1$ basis functions are generated by obtaining the corresponding a_{nk} coefficients (with $1 \leq n \leq N+1$ and $1 \leq k \leq n$) according to equation (25), (23) and (24). In addition set $S(m)$ ($m = N+1$) is given by equation (28).

Next the algorithm looks for the optimal hidden units of the network, where the unknowns are the values of

$j(N+2)$ to $j(N+1+N_{hd})$. The initial set of candidate basis functions is $S^c(N+1)$ with N_h elements.

Let N_{hd} denote the desired number of hidden units for the algorithm. When testing the candidate basis function set, all subsets of N_{hd} hidden units in $S^c(N+1)$ are evaluated. In optimal pruning there is not a single $j(m)$ function. We have a different one, every time N_{hd} changes.

Defining n_c as the number of orderings of N_{hd} hidden units, the number of networks to test is given by

$$n_c = \binom{N_h}{N_{hd}} \quad (38)$$

For each candidate network the energy is estimated as

$$P_{total} = \sum_{q=1}^{N_{hd}} P(m_q) = \sum_{q=1}^{N_{hd}} \left(\sum_{i=1}^M [w_o(i, m_q)]^2 \right) \quad (39)$$

where each value of $j(m_q)$ will correspond to a hidden unit that forms the network being tested. Here m_q defines the q^{th} element of the combination being tested.

Then the final solution will be the set of hidden units that maximize equation (39). In this way $j(m)$ take its final value and $S(N+1+N_{hd})$ is updated as

$$\begin{aligned} S(N_{hd}) &= S(N+1) \cup \{j(m_1), j(m_2), \dots, j(m_{N_{hd}})\} \\ &= \{j(1), j(2), \dots, j(N+1), \dots, j(N+1+N_{hd})\} \end{aligned}$$

Once that $j(m)$ and a_{nk} ($1 \leq n \leq N+1+N_{hd}$, $1 \leq k \leq n$) are found the final output weights $w_o(i, k)$ are obtained by using equation (20).

Let $E_{ord}(N_{hd})$ and $E_{opt}(N_{hd})$ respectively denote the training errors that result from ordered and optimal pruning when a network having N_h hidden units is pruned down to N_{hd} hidden units. The following lemma relates the two error curves.

Lemma: The $E_{ord}(N_{hd})$ and $E_{opt}(N_{hd})$ curves satisfy:

$$E_{ord}(1) = E_{opt}(1)$$

$$E_{ord}(N_{hd}) \geq E_{opt}(N_{hd}) \text{ for } 2 \leq N_{hd} \leq N_h - 1,$$

and

$$E_{ord}(N_h) = E_{opt}(N_h)$$

VII. RESULTS

The pruning algorithms were tested using four sets of training data. In each case we trained a network (100 epochs) and used it as input to the pruning algorithms. Computational cost is quantified by determining the number of multiplications that are required for applying the proposed algorithms.

Twod.tra. This example corresponds to the task of inverting the surface scattering parameters from an inhomogeneous layer above a homogeneous half space [3], where both interfaces are randomly rough. The network has 8 inputs, 7 outputs, 15 hidden units, and the file has 1768 training patterns. The inputs consist of eight theoretical values of back scattering coefficient parameters

at V and H polarization and four incident angles. The outputs were 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 jointly uniform pdf. Results for this data set are given in figures 2 and 3. Figure 3 relates the number of multiplications required for pruning N_h hidden units of the input network down to N_{hd} units. The figure shows the ordered and optimal cases (M_{ord} and M_{opt} respectively). Because of the large numbers of multiplications that are required in optimal pruning, a semilog scale was used for putting both plots together. Optimal pruning performs marginally better than ordered pruning but is computationally expensive.

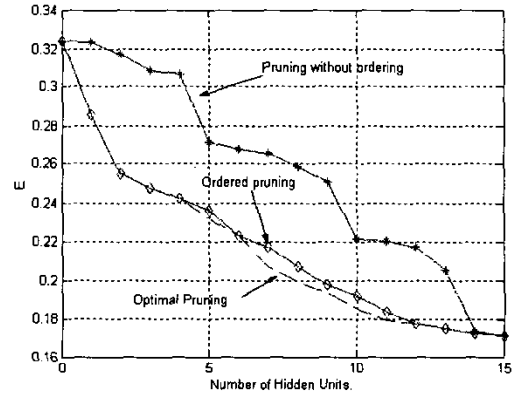


Fig. 2. Inversion of surface scattering parameters.

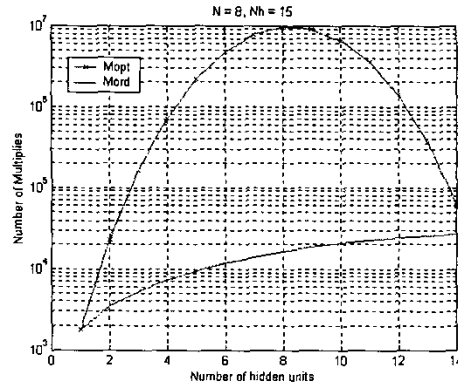


Fig. 3. Number of multiplies for inversion of surface scattering parameters.

Power12.tra. The plot of figure 4 was generated using data obtained from TU Electric Company in Texas where the first ten input features are the last ten minute power load in megawatts for the entire TU Electric utility [5]. The desired output is power load fifteen minutes in the future from the current time. All powers were originally sampled

every fraction of a second, and averaged over 1 minute to reduce noise. The network has 12 inputs, 1 output, 15 hidden units, and the file has 1415 training patterns. The lemma is clearly satisfied. In addition, optimal pruning provides only a very marginal improvement over ordered pruning.

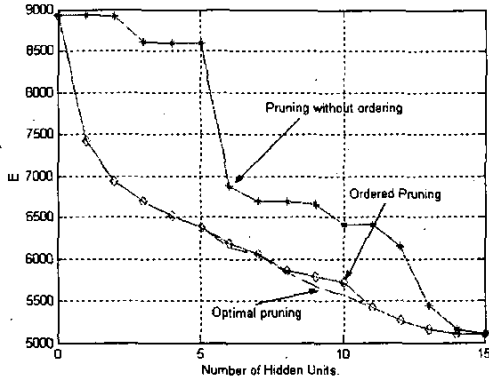


Fig. 4. Electrical power load forecasting.

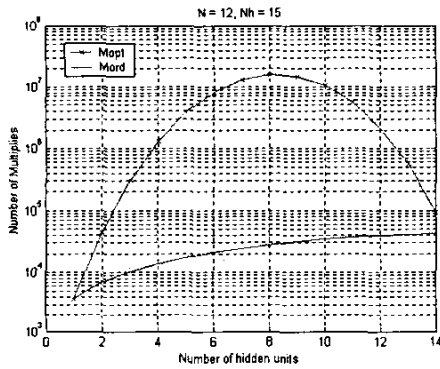


Fig. 5. Number of multiplies for electrical power load forecasting

fmtrain.tra. The second example corresponds to a neural network that performs demodulation of an FM (frequency modulation) signal containing a sinusoidal message [8].

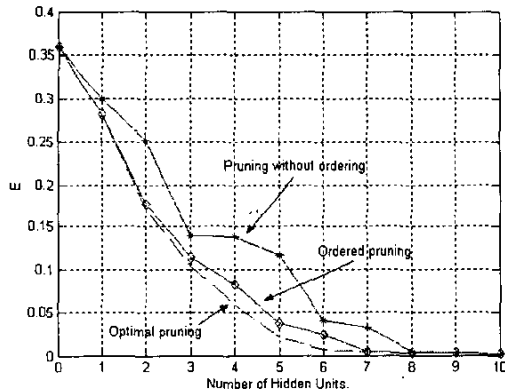


Fig. 6. Demodulation of an FM signal.

The data set was obtained from a program that generates FM modulated sinusoid, which corresponds to the following expression,

$$fm(n) = A_c \cdot \cos[(2\pi f_c n) + A_m \cdot \sin(2\pi f_m n)]. \quad (40)$$

For testing the pruning algorithms a network was generated with 5 inputs, 1 output, and 10 hidden units. The file has 2000 training patterns. From figure 6 and 7 the marginality of optimal pruning is again clear and the lemma is satisfied.

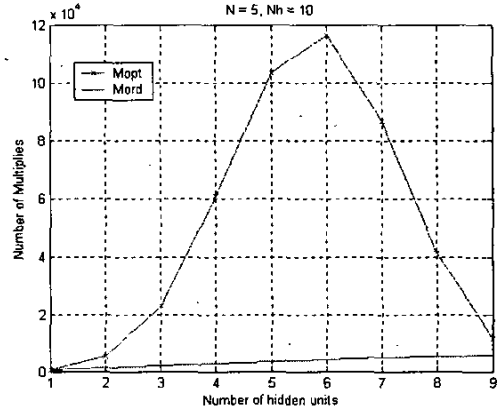


Fig. 7. Number of multiplies for FM case.

Oh7.tra. Finally the last case corresponds to the inversion of radar scattering [10]. The training set contains VV and HH polarization at L 30, 40 deg, C 10, 30, 40, 50, 60 deg, and X 30, 40, 50 deg along with the corresponding unknowns rms surface height, surface correlation length, and volumetric soil moisture content in g/cubic cm. The network has 20 inputs, 3 outputs, 12 hidden units, and the file has 10,453 training patterns. Pruning results and numbers of multiplies are given in figures 8 and 9 respectively

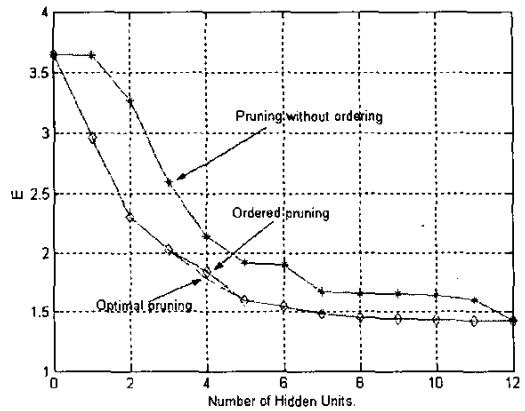


Fig. 8. Inversion of radar scattering parameters.

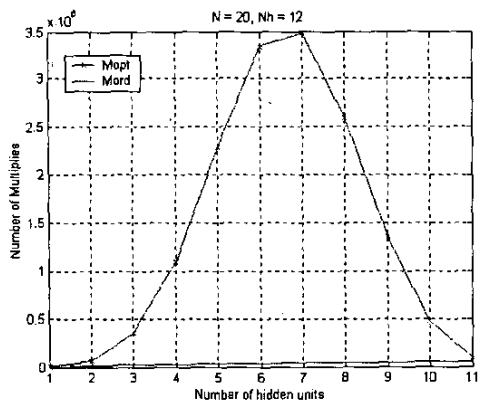


Fig. 9. Number of multiplies for inversion of radar scattering parameters.

From the figures, we see that optimal pruning is better than ordered pruning. Considering the excessive number of multiplies required by optimal pruning and the marginal improvement it provides in the examples that were presented, it is clear that optimal pruning is not practical.

VIII. CONCLUSIONS

Two pruning algorithms for fully-connected MLPs are given in this paper. Although optimal pruning is slightly more effective than ordered pruning, it is too computationally expensive to be useful. The modified orthogonalization procedure is more efficient than normal Gram-Schmidt method and provides a more suitable framework for pruning. For mapping the algorithms performs well. Further work is required for the application of this framework to the classification case.

ACKNOWLEDGEMENT

This work was funded by the Advanced Technology Program of the state of Texas under grant 003656-0129-2001. F. J. Maldonado thanks Chihuahua Institute of Technology for his academic leave from the Image Processing and Neural Networks Laboratory at The University of Texas at Arlington.

REFERENCES

- [1] H. Amin, K. M. Curtis, and B. R. Hayes Gill, "Dynamically pruning output weights in an expanding Multilayer Perceptron Neural Network," *Digital Signal Processing Proceedings, 1997 13th International Conference on DSP*, vol. 2, pp. 991-994, 1997.
- [2] F. L. Chung and T. Lee, "Network-growth approach to design of feedforward neural networks," *IEE Proceedings Control Theory*, 142-5, 486-492 (1995).
- [3] M.S. Dawson, J. Olivera, A.K. Fung and M.T. Manry, "Inversion of surface parameters using fast learning neural networks," *Proc. of IGARSS'92, Houston, Texas, Vol II*, pp 910 - 912, May 1992.
- [4] Y. Hirose, K. Yamashita, and S. Hijiya, "Back-propagation algorithm that varies the number of hidden units," *Neural Networks*, vol. 4, pp. 61-66, 1991.
- [5] K. Liu, S. Subbarayan, R.R.Shoults, M.T. Manry, C.Kwan, F.L. Lewis, and J.Naccarino, "Comparison of very short-term load forecasting techniques," *IEEE Transactions on Power Systems*, vol.11, no.2, pp. 877-882, May 1996.
- [6] Ponnappalli, "A formal selection and pruning algorithm for feedforward artificial network optimization," *IEEE Transactions on Neural Networks*, vol. 10, No. 4, pp. 964-968, 1999.
- [7] R. Reed, "Pruning algorithms - a survey," *IEEE Transactions on Neural Networks*, vol. 4, No. 5, pp. 740-747, 1993.
- [8] K. Röhani and M.T. Manry, "The design of multi-layer perceptrons using building blocks," *Proc. of IJCNN 91, Seattle WA.*, pp. II-497 to II-502.
- [9] I. I. Sakhnini, M. T. Manry, and H. Chandrasekaran, "Iterative improvement of trigonometric networks," *International Joint Conference on Neural Networks (IJCNN '99)*, July 1999.
- [10] Y. K. Sarabandi and F.T. Ulaby, "An empirical model and an inversion technique for radar scattering from bare soil surfaces," in *IEEE Transactions on Geoscience and Remote Sensing*, pp. 370-381, 1992.
- [11] G. Strang, *Linear algebra and its application*. New York: Harcourt Brace, 1988.
- [12] Wladyslaw Kaminski and Pawel Strumillo "Kernell orthonormalization in radial basis function Neural Networks," *IEEE Transactions on Neural Networks*, vol. 8, No. 5, pp. 1177 - 1183, 1997.