Fast Generation of a Sequence of Trained and Validated Feed-Forward Networks

Pramod L. Narasimha[1], Walter Delashmit[2], Michael Manry[1], Jiang Li[3] and Francisco Maldonado[4]
[1]Dept. of Electrical Engineering, University of Texas at Arlington, Arlington, TX 76010
[2]Lockheed Martin Missiles and Fire Control, Dallas, TX 75265
[3]Department of Radiology, Clinical Center, National Institutes of Health, Bethesda, MD, 20892
[4]Williams Pyro, 2721 White Settlement Rd., Fort Worth, TX 76107
L_pramod79@yahoo.com, Walter.Delashmit@lmco.com, Manry@uta.edu, lij3@cc.nih.gov, frankmald@hotmail.com

Abstract

In this paper, three approaches are presented for generating and validating sequences of different size neural nets. First, a growing method is given along with several weight initialization methods, and their properties. Then a one pass pruning method is presented which utilizes orthogonal least squares. Based upon this pruning approach, a one-pass validation method is discussed. Finally, a training method that combines growing and pruning is described. In several examples, it is shown that the combination approach is superior to growing or pruning alone.

Introduction

According to the structural risk minimization (SRM) principle, a sequence of learning machines of increasing size should be produced, and trained as well as possible. The machine with the smallest validation error is the best compromise between the training error and the complexity of the network. If this principal is followed with neural nets, the final training error as a function of the number of hidden units, $E_f(N_h)$, will be monotonically nonincreasing. Sequences of networks are produced though growing methods and pruning methods. In growing methods (Delashmit 2003), one can design a set of different size networks in an orderly fashion, each with one or more hidden units than the previous network (Fahlman et. al. 1990) (Chung & Lee 1995). A drawback of growing methods is that the network can get trapped in local minima and they are also sensitive to initial conditions.

In pruning methods, a large network is trained and then less useful nodes or weights are removed (Hassibi et. al. 1993) (LeCun et.al. 1990) (Sakhnini, Manry, & Chandrasekaran 1999). Some pruning algorithms remove less useful hidden units using the Gram-Schmidt procedure as reported by Kaminsky and Strumillo (1997) for Radial Basis Functions and Maldonado et.al. (2003) for the multilayer perceptron (MLP). Unfortunately, if one large network is trained and pruned, the resulting error versus N_h curve is not minimal for smaller networks. In other words, it is possible, though unlikely, for each of the hidden units to be equally useful after training.

In this paper, we investigate three approaches for generating and validating sequences of different size neural nets. First, a growing method is described, along with analyses of weight initialization methods. Then a pruning method is presented which requires only one pass through the training data. A one-pass validation method is presented, which is based upon pruning. Next, a method that combines growing and pruning is presented. Results show that this third approach overcomes the shortcomings of growing or pruning alone.

Multilayer Perceptron

Figure 1 depicts feed-forward MLP, having one hidden layer with N_h nonlinear units and an output layer with M linear units. For the pth pattern, the jth hidden unit's net function and activation are

$$net_{pj} = \sum_{i=1}^{N=1} w(j,i) \cdot x_{pi} \qquad (1)$$

for $1 \le p \le N_v$, $1 \le j \le N_h$ and

$$O_{pj} = f(net_{pj}) = \frac{1}{1+e^{-net_{pj}}} \qquad (2)$$

Here the threshold of the jth node is handled by letting $x_{p,N+1}$ be one. Weight w(i, j) connects the ith input to the jth hidden unit. The kth output for the pth pattern is,

$$y_{pk} = \sum_{i=1}^{N+1} w_o(k,i) \cdot x_{pi} + \sum_{i=1}^{N_h} w_o(k, j+N+1) \cdot O_{pj} \qquad (3)$$

where $1 \le k \le M$. The first sum in (3) describes the effect of bypass weights connecting inputs and outputs. There are N_v training patterns denoted by $\{(\mathbf{x}_p, \mathbf{t}_p)\}_{p=1}^{N_v}$, where each pattern consists in an input vector x_p and a desired output vector t_p. For the pth pattern, the N input values are x_{pi}, $(1 \le i \le N)$ and the M desired output values are t_{pk} $(1 \le k \le M)$. Example training algorithms are Levenberg

Marquart (Fun & Hagan 1996), backpropagation, Output Weight Optimization – Hidden Weight Optimization (Yu et.al., 2005), and Genetic Algorithms (Arena et al. 1992).

Growing Approach

In growing methods, which we denote as Design Methodology 1 (DM-1), we successively train larger networks. A pioneering example is the cascade correlation approach of Fahlman and Lebierre, 1990. In DM-1 algorithms, the final training MSE versus N_h curve, $E_f(N_h)$, along with the validation error versus N_h curve $E_{val}(N_h)$, should help us find the required network.

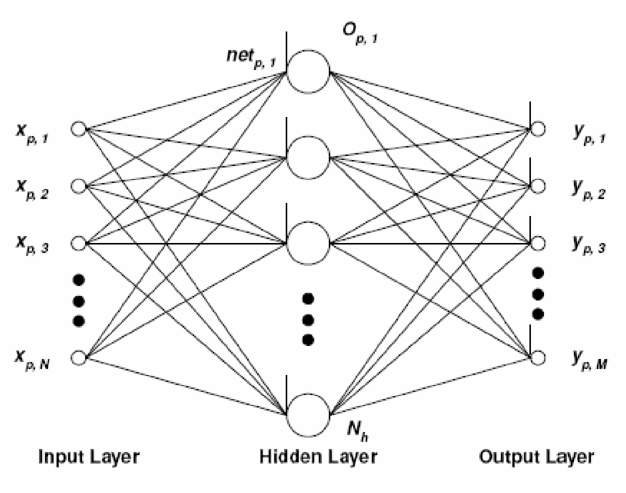

Figure 1: Two layer MLP

MLP training is strongly dependent on the initial weights, so proper initialization is critical. To define the problem being considered, assume that a set of MLPs of different sizes (i.e., different number of hidden units, N_h) is to be designed for a given training data set.

Axiom 1: If $E_f(N_h) > E_f(N_h - 1)$, then training has failed for the network having N_h hidden units, since the larger, more complex network has a larger training error. Monotonic $E_f(N_h)$ curves make smoothly varying $E_{val}(N_h)$ curves more likely, which makes structural risk minimization easier to perform. Three distinct types of network initialization are considered.

Randomly Initialized Networks

When a set of MLPs are Randomly Initialized (RI), they have no initial weights or thresholds in common. 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. Growing of RI networks can be designated as DM-1a.

Theorem 1: If two initial RI networks (1) are the same size, (2) have the same training data set and (3) the training data set has more than one unique input vector, then the hidden unit basis functions are different for the two networks.

A problem with RI networks is that $E_f(N_h)$ is nonmonotonic. That is, for well-trained MLPs, $E_f(N_h)$ does not always decrease as N_h increases since the initial hidden unit basis functions are different. Let $E_i(N_h)$ denote the final training error for an RI network having N_h hidden units, which has been initialized using the i^{th} random number seed based on N_s total seeds. Let $E_{av}(N_h)$ denote the average $E_i(N_h)$, that is

$$E_{av}(N_h) = \frac{1}{N_s}\sum_{i=1}^{N_s} E_i(N_h) \qquad (4)$$

Using the Chebyshev inequality, a bound has been developed on the probability that the average error for N_h hidden units, $E_{av}(N_h)$ is increasing (Delashmit 2003) and that the network with $N_h + 1$ hidden units is useless.

$$P(E_{av}(N_h+1) > E_{av}(N_h)) \leq \frac{\mathrm{var}(E_i(N_h+1)) + \mathrm{var}(E_i(N_h)}{2 \cdot N_s \cdot (m_{av}(N_h) - m_{av}(N_h+1))^2} \qquad (5)$$

Here var() represents the variance and $m_{av}(N_h)$ represent the average MSE for N_h hidden units.

Common Starting Point Initialized Networks

When a set of MLPs are Common Starting Point Initialized with Structured Weight Initialization (CSPI-SWI), the initialization of the weights and thresholds is ordered such that every hidden unit of the smaller network has the same weights as the corresponding hidden unit of the larger network. Input to output weights are also identical. These networks are more likely than RI networks to have monotonic $E_f(N_h)$ curves. Growing with CSPI-SWI networks can be designated as DM-1b.

Theorem 2: If two initial CSPI-SWI networks (1) are the same size and (2) use the same algorithm for processing random numbers into weights, then they are identical.

Theorem 3: If two CSPI-SWI networks are designed, the common subsets of the initial hidden unit basis functions are identical.

The above theorems have been proved by Delashmit (2003). Unfortunately, DM-1b, although better than DM-1a, does not guarantee a monotonic $E_f(N_h)$ curve.

Dependently Initialized Networks

Growing with dependently initialized (DI) networks is designated as DM-1c. In DM-1c, a linear network is first trained. Then we successively add N_a hidden units and retrain the network using the method of Yu et. al. (2005), which is a greedy algorithm. This is continued till the number of hidden units equals a user-chosen number.

Properties of DI networks: Let $E_{int}(N_h)$ denote the initial value of error during the training of an MLP with N_h hidden units and let N_{hp} denote the number of hidden units in the previous network, so $N_a = N_h - N_{hp}$. Then:

1. $E_{int}(N_h) < E_{int}(N_{hp})$

2. $E_f(N_h) \leq E_f(N_{hp})$
3. $E_{int}(N_h) = E_f(N_{hp})$

As seen in property 2, DM-1c, produces a monotonically decreasing $E_f(N_h)$ versus N_h curve.

Ordered Pruning

Pruning of a large network, which we denote as Design Methodology 2 (DM-2), is a second common approach to producing a monotonic $E_f(N_h)$ curve. The output of the network in equation 3, can be rewritten as

$$y_i = \sum_{k=1}^{N_u} w_o(i,k) \cdot x_k \tag{6}$$

where $x_k = O_{p,(k-N-1)}$ for $N + 2 \leq k \leq N_u$, where N_u is the total number of units equal to $N + N_h + 1$. In equation (6), the signals x_k are the raw basis functions for producing y_i. The purpose of pruning is to eliminate less useful hidden units, that have no information relevant for estimating outputs or that are linearly dependent on inputs or hidden units that have already been orthonormalized.

Here, we use the Schmidt procedure (Dettman, 1962) to order and remove less useful basis functions, following the approach of Maldonado et.al.(2003). 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 mth unit of the new structure comes from the kth unit of the original structure. Given the function j(m), and generalizing the Schmidt procedure, the mth orthonormal basis function is described as

$$x'_m = \sum_{k=1}^{m} a_{mk} \cdot x_{j(k)} \tag{7}$$

Initially, x'_1 is found as $a_{11} \, . \, x_{j(1)}$ where

$$a_{11} = \frac{1}{\left\| x_{j(1)} \right\|} = \frac{1}{r(j(1), j(1))^{1/2}} \tag{8}$$

Here r(i,j) is the auto-correlation between the ith and jth basis functions. For $2 \leq m \leq N_u$, we first perform

$$c_i = \sum_{q=1}^{i} a_{iq} \cdot r(j(q), j(m)), \text{ for } 1 \leq i \leq m - 1 \tag{9}$$

Second, we set $b_m = 1$ and get

$$b_k = -\sum_{i=k}^{m-1} c_i \cdot a_{ik}\ , \text{ for } 1 \leq k \leq m\text{-}1 \tag{10}$$

Lastly, we get coefficients a_{mk} as

$$a_{mk} = \frac{b_k}{\left[r(j(m), j(m)) - \sum_{i=1}^{m-1} c_i^{\,2} \right]^{1/2}}, \text{ for } 1 \leq k \leq m \tag{11}$$

Then weights in the orthonormal system are found as

$$w'_o(i,m) = \sum_{k=1}^{m} a_{mk} \cdot c(i, j(k)), \text{ for } 1 \leq i \leq M \tag{12}$$

where c(i,j) is the cross-correlation between the jth basis function and the ith desired output. Now, our goal 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 (11) 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,

$$j(m) = m, \text{ for } 1 \leq m \leq N+1 \tag{13}$$

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 in the current network (i.e., the one that the algorithm is of processing). Then S(m) is given by

$$S(m) = \begin{cases} \{\phi\}, & \text{for } m = 0 \\ \{j(1), j(2), \ldots, j(m)\}, & \text{for } 0 < m \leq N_u \end{cases} \tag{14}$$

Starting with an initial linear network having 0 hidden units, where m is equal to N + 1, the set of candidate basis functions is $S^c\{m\} = \{1, 2, 3, \ldots, N_u\} - S(m)$, which is $\{N + 2, N + 3, \ldots, N_u\}$. 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 in equations (9-12). Then P(m) is

$$P(m) = \sum_{i=1}^{M} [w'_o(i,m)]^2 \tag{15}$$

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

$$S(m) = S(m-1) \bigcup \{j(m)\} \tag{16}$$

Then for the general case the candidate basis functions are, $S^c(m - 1) = \{1, 2, 3, \ldots, N_u\} - \{j(1), j(2), \ldots, j(m - 1)\}$ with $N_u - m + 1$ candidate basis function. By using equation (15), after testing all the candidate basis function, j(m) takes its value and S(m) is updated according to equation (16). After the j(m) function is complete, both the original basis functions and the orthonormal ones are ordered. For any given desired number of hidden units N_{hd} , the orthonormal weights are mapped to normal weights according to:

$$w_o(i,k) = \sum_{q=k}^{N_u} w'_o(i,q) \cdot a_{qk} \tag{17}$$

Theorem 4: After performing ordered pruning, the order of the hidden units is stepwise optimal. For any given kth hidden unit, the (k+1)th unit is the one, out of the remaining units, which reduces the MSE the most.

Validation

Given the matrix **A** and the MLP network with ordered hidden units, we wish to generate the validation error versus N_h curve $E_{val}(N_h)$ from the validation data set

$\{(\mathbf{x}_p, \mathbf{t}_p)\}_{p=1}^{N_v}$. For each pattern, we augment the input vector as in the previous sections. So the augmented input vector is $x_p \leftarrow [x^T_p, 1, O^T_p]^T$. Then the augmented vector is converted into orthonormal basis functions by the transformation

$$x'_p(m) = \sum_{k=1}^{m} a_{mk} \cdot x_p(k) \text{, for } 1 \le m \le N_u \qquad (18)$$

In order to get the validation error for all size networks in a single pass through the data, we use the following strategy. Let $y_{pi}(m)$ represent the ith output of the network having m hidden units for the pth pattern, let $E_{val}(m)$ represent the mean square error of the network for validation with m hidden units. First, the linear network output is obtained and the corresponding error is calculated as follows:

$$y_{pi}(0) = \sum_{k=1}^{N+1} w'_o(i,k) \cdot x'_p(k) \text{, for } 1 \le i \le M$$

$$E_{val}(0) \leftarrow E_{val}(0) + \sum_{i=1}^{M} [t_{pi} - y_{pi}(0)]^2 \qquad (19)$$

Then for 1 ≤ m ≤ N_h, the following two steps are performed:

- For 1 ≤ i ≤ M
$$\begin{aligned} y_{pi}(m) = y_{pi}(m-1) + \\ w'_o(i, N+1+m) \cdot x'_p(N+1+m) \end{aligned} \qquad (20)$$
- $$E_{val}(m) \leftarrow E_{val}(m) + \sum_{i=1}^{M} [t_{pi} - y_{pi}(m)]^2 \qquad (21)$$

Apply equations (18-21) for 1 ≤ p ≤ N_v and get the total validation error over all the patterns for each size network. Then these error values should be normalized as

$$E_{val}(m) \leftarrow \frac{E_{val}(m)}{N_v} \text{, for } 0 \le m \le N_h \qquad (22)$$

Thus we generate the validation error versus the network size curve in one pass through the validation data set.

Pruning a Grown Network

The SRM principle requires that $E_f(N_h)$ be as small as possible for each value of N_h. The DM-1c growing approach generates monotonic $E_f(N_h)$ curves. However, hidden units added earlier in the process tend to reduce the MSE more than those added later. Unfortunately, there is no guarantee that an $E_f(N_h)$ sample represents a global minimum., there is no guarantee that the hidden units are ordered properly, and useless hidden units can occur even for small values of N_h. In Design Methodology 3 (DM-3), we attempt to solve these problems by

(1) Performing growing as in DM-1c, and

(2) Performing ordered pruning of the final network.

This forces the grown hidden units to be stepwise optimally ordered. Our method is denoted the "pruning a grown network" approach.

Numerical Results

Three Design Methodologies are compared here for generating sequences of different size networks. In DM-1c, the maximum value of N_h is 20. In DM-2, a single large network with 20 hidden units is trained and then ordered pruning is applied. For DM-3, two methods are used. These are pruning a grown network from the previous section, and the MRAN RBF design approach described in Huang et.al. 2005. Since MRAN does not produce a final network or an $E_f(N_h)$ curve, its results are stated separately, and not plotted. All three methodologies were evaluated using four different data sets.. The plots shown are average MSE versus number of hidden units.

The first data set is for the inversion of surface permittivity. This data has 16 inputs and 3 outputs. The inputs represent the simulated back scattering coefficient measured at 10, 30, 50 and 70 degrees at both vertical and horizontal polarization. The remaining 8 are various combinations of ratios of the original eight values. In figures 2 and 3, training and validation error plots are shown respectively for pruning and growing approaches for inversion of surface permittivity data set. From the figures, it is clear that pruning a grown network is best. The MRAN approach adaptively chose networks with around seventeen hidden units, and gave a validation error of 29.4486, which is much worse than the other methods' results.

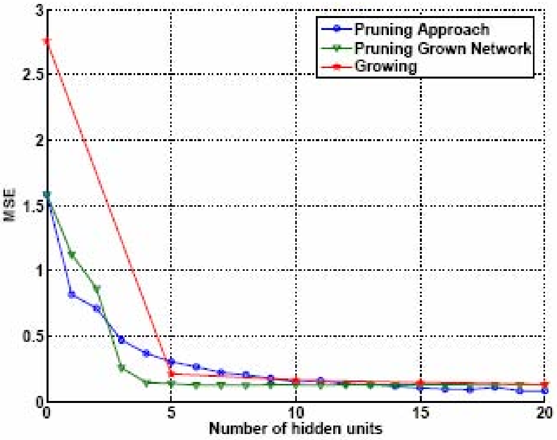

Figure 2: Training error for Surface Permittivity Inversion

The second data set is for inversion of radar scattering from bare soil surfaces. It has 20 inputs and 3 outputs. 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 degrees along with the corresponding unknowns rms surface height, surface correlation length, and volumetric soil moisture content in g/cubic cm. From figures 4 and 5, growing is better than pruning, and pruning a grown network is also very effective. The MRAN approach adaptively chose networks with around five hidden units, and resulted in training and validation MSEs of 7.3346 and 9.8037 respectively, which is much worse.

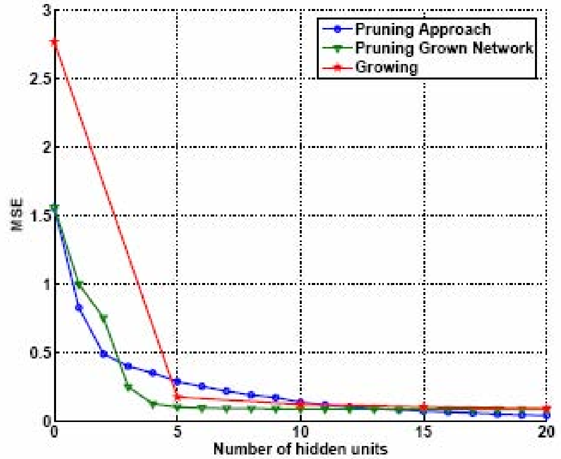

Figure 3: Validation error for Surface Permittivity Inversion

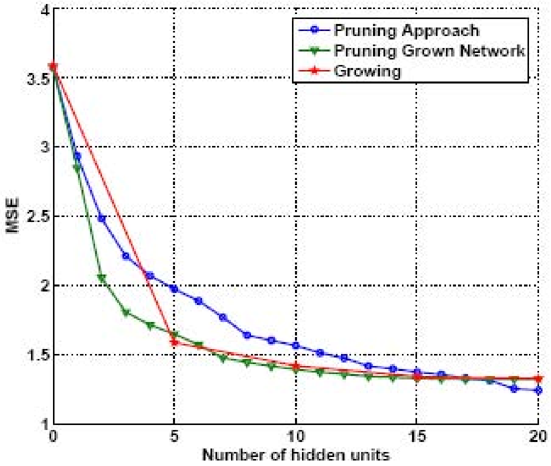

Figure 4: Training error for Inversion of Radar Scattering.

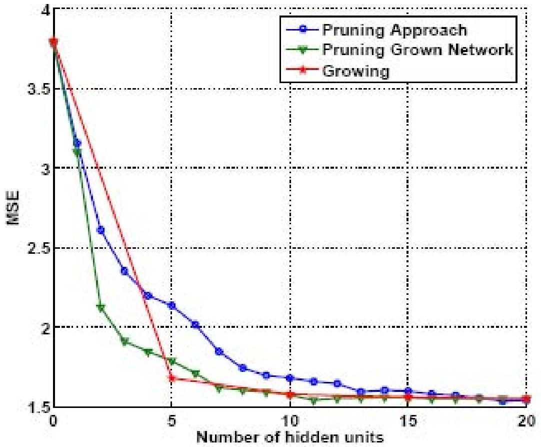

Figure 5: Validation error for Radar Scattering Inversion

The third data set is prognostics data for onboard flight load synthesis (FLS) in helicopters, where we estimate mechanical loads on critical parts, using measurements available in the cockpit. There are 17 inputs and 9 outputs. From figures 6 and 7, pruning is again worse than growing, and pruning a grown network is best. MRAN adaptively chose networks with twenty one hidden units, and resulted in training and validation MSEs of 1.4233 x 10^7 and 2.2803 x 10^9 respectively, which is much worse.

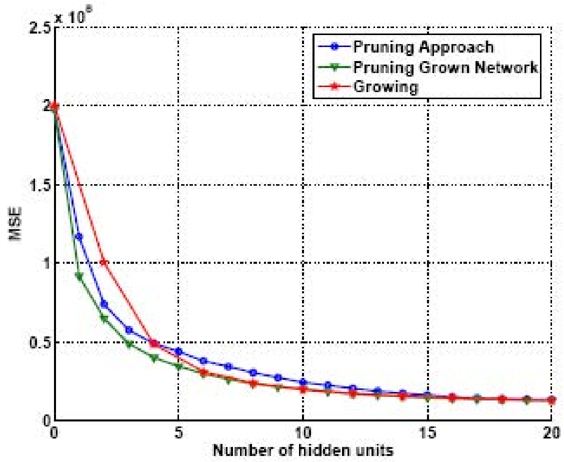

Figure 6: Training error for Prognostics data.

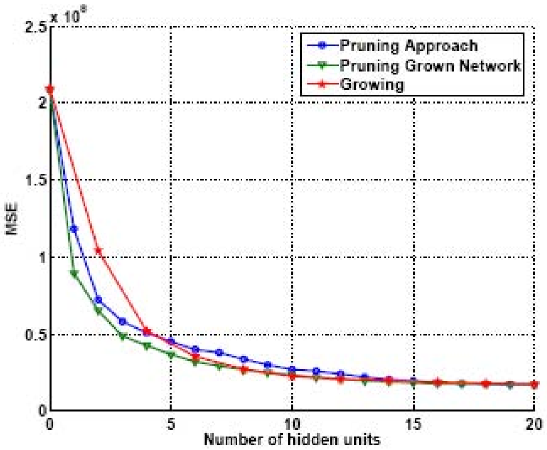

Figure 7: Validation error plots for Prognostics data.

The fourth data set for estimating phoneme likelihood functions in speech, has 39 inputs and 117 outputs. The speech samples are first pre-emphasized and converted into the frequency domain via the DFT. The data is passed through Mel filter banks and the inverse DFT is applied on the output to get Mel-Frequency Cepstrum Coefficients (MFCC). Each of MFCC(n), MFCC(n)-MFCC(n - 1) and MFCC(n)-MFCC(n - 2) would have 13 features, which results in a total of 39 features. The desired outputs are likelihoods for the beginning, middle, and ends of 39 phonemes. From figures 8 and 9, pruning is again worse than growing, and pruning a grown network is best. MRAN adaptively chose networks with twenty hidden units, and resulted in training and validation MSEs of 4.05 x 10^4 and 3.801 x 10^4 respectively, which is much worse.

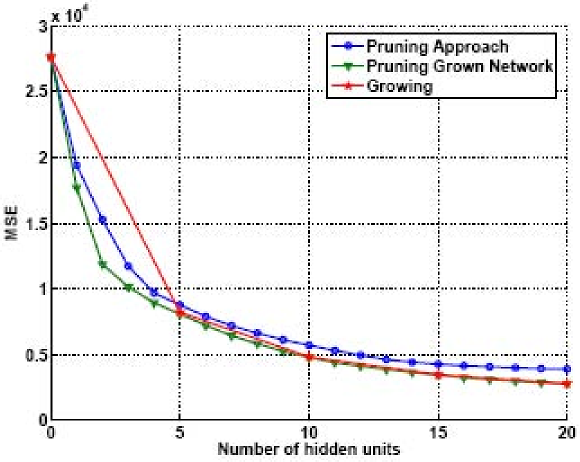

Figure 8: Training error for Speech data.

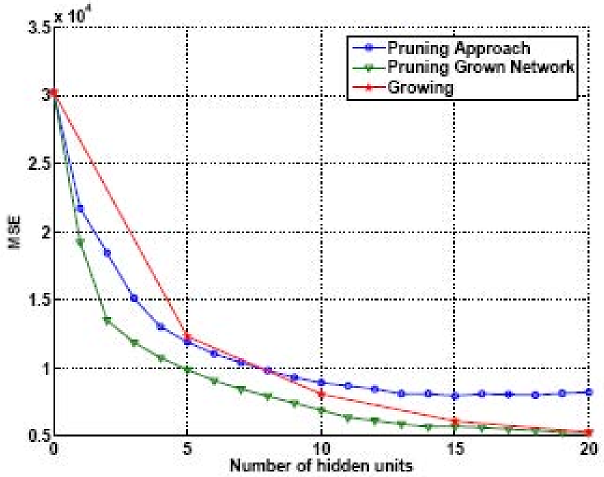

Figure 9: Validation error for Speech data.

Conclusions

In this paper, we explore three approaches for producing sequences of trained and validated feedforward networks. In the growing approach, dependently initialized networks result in monotonically decreasing E_f (N_h) curves. A pruning method is shown that requires one pass through the data. A method is also described for simultaneously validating many different size networks simultaneously, using a single data pass. In the third, combined approach, ordered pruning is applied to grown networks. As seen in the simulations, this approach usually produces smaller training error values than either growing or pruning alone.

References

Arena, P.; Caponetto, R.; Fortuna, L.; and Xibilia, M. G. 1992. Genetic algorithms to select optimal neural network topology. In *Proceedings of the 35th Midwest Symposium on Circuits and Systems*, volume 2, 1381–1383.

Changhua Yu, C., Manry, M.T., and Li, J. 2005. "Effects of nonsingular pre-processing on feed-forward network training ". *International Journal of Pattern Recognition and Artificial Intelligence* , Vol. 19, No. 2, pp. 217-247.

Chung, F. L., and Lee, T. 1995. Network-growth approach to design of feedforward neural networks. *IEE Proceedings Control Theory* 142(5):486–492.

Delashmit, W. H. 2003. Multilayer Perceptron Structured Initialization and Separating Mean Processing. Dissertation, University of Texas at Arlington.

Dettman, J. W. 1962. *Mathematical Methods in Physics and Engineering*, 2nd edit., McGraw-Hill, New York.

Fahlman, Scott E. and Lebierre, C. 1990. “The Cascade Correlation learning architecture,” In David S. Touretzky, editor, *Advances in Neural Information Processing Systems*, vol. 2, pp. 524-532.

Fun, M. H., and Hagan, M. T. 1996. Levenberg-marquardt training for modular networks. In *IEEE Int. Conf. on Neural Networks*, volume 1, 468–473.

Hassibi, B., Stork, D. G. and Wolff, G. J. 1993. “Optimal brain surgeon and general network pruning,” *Proc. Of 1993 IEEE In. Conf. on Neural Networks*, pp. 293-299.

Kaminski, W, and Strumillo, P, 1997. “Kernell orthonormalization in radial basis function neural networks,” *IEEE Trans. Neural Nes*, 8(5), 1177–1183.

Huang, G-B, Saratchandran, P., and Sundararajan, N. 2005. "A generalized growing and pruning RBF Neural Network for function approximation", *IEEE Trans. on Neural Networks*, vol. 16, no. 1, pp. 57-67.

LeCun, Y., Denker, J. S., Solla, S., Howard, R. E., and Jackel, L. D. 1990. "Optimal Brain Damage," in *Advances in Neural Information Processing Systems 2*, (David Touretzky, ed.), Denver, CO.

Maldonado, F., Manry, M.T., and Kim, T., 2003. "Finding Optimal Neural Network Basis Function Subsets Using the Schmidt Procedure," *Proc. of IJCNN'03*.