Available online at www.sciencedirect.com
ScienceDirect
Neurocomputing 71 (2008) 2831–2847
NEUROCOMPUTING
www.elsevier.com/locate/neucom

An integrated growing-pruning method for feedforward network training

Pramod L. Narasimha[a,*], Walter H. Delashmit[b], Michael T. Manry[a], Jiang Li[c], Francisco Maldonado[d]

[a]*Department of Electrical Engineering, University of Texas at Arlington, Arlington, TX 76013, USA*
[b]*Lockheed Martin, Missiles and Fire Control Dallas, Grand Prairie, TX 75051, USA*
[c]*Electrical and Computer Engineering, Old Dominion University, Norfolk, VA 23529, USA*
[d]*Williams Pyro, Inc., 200 Greenleaf Street, Fort Worth, TX 76107, USA*

Received 12 March 2007; received in revised form 8 August 2007; accepted 20 August 2007
Communicated by Dr. J. Zhang
Available online 12 October 2007

Abstract

In order to facilitate complexity optimization in feedforward networks, several algorithms are developed that combine growing and pruning. First, a growing scheme is presented which iteratively adds new hidden units to full-trained networks. Then, a non-heuristic one-pass pruning technique is presented, which utilizes orthogonal least squares. Based upon pruning, a one-pass approach is developed for generating the validation error versus network size curve. A combined approach is described in which networks are continually pruned during the growing process. As a result, the hidden units are ordered according to their usefulness, and the least useful units are eliminated. Examples show that networks designed using the combined method have less training and validation error than growing or pruning alone. The combined method exhibits reduced sensitivity to the initial weights and generates an almost monotonic error versus network size curve. It is shown to perform better than two well-known growing methods—constructive backpropagation and cascade correlation.

Keywords: Growing; Pruning; Cascade correlation; Back propagation; Output weight optimization–Hidden weight optimization

1. Introduction

When optimizing the structural complexity of feedforward neural networks, different size networks are designed which minimize the mean-square training error. Typically, the machine with the smallest validation error is saved. When different size networks are independently initialized and trained, however, the training error versus network size curve is not monotonic. Similarly, the resulting validation error versus size curve is not smooth, making it difficult to find a proper minimum. Monotonic training curves and smooth validation curves can be produced by growing methods or by pruning methods.

In growing methods [11], new hidden units are continually added to a trained network, and further training is performed [7]. In [31], the addition of new units continues as long as the sum of the accuracies for training and crossvalidation samples improves. This eliminates the requirement of specifying minimum accuracy for the growing network. Guan and Li [15] have used output parallelism to decompose the original problem into several subproblems, each of which is composed of the whole input vector and a fraction of the desired output vector. They use constructive backpropagation (CBP) to train modules of the appropriate size. Modules are merged to get a final network. A drawback of growing methods is that some of the intermediate networks can get trapped in local minima. Also, growing methods are sensitive to initial conditions. Unfortunately, when a sequence of networks is grown, there is no guarantee that

*Corresponding author.
E-mail addresses: pramod.narasimha@uta.edu, l_pramod79@yahoo.com (P.L. Narasimha).

0925-2312/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2007.08.026

all of the added hidden units are properly trained. Some can be linearly dependent.

In pruning methods, a large network is trained and then the least useful nodes or weights are removed [1,16,24,25,27]. Orthogonal least squares or Gram–Schmidt [32] based pruning has been described by Chen et al. [6] and Kaminsky and Strumillo [18] for radial basis functions (RBF) and by Maldonado et al. [21] for the multilayer perceptron (MLP). In crosswise propagation [20], linearly dependent hidden units are detected using an orthogonal projection method, and then removed. The contribution of the removed neuron is approximated by the remaining hidden units. In [4], the hidden units are iteratively removed and the remaining weights are adjusted to maintain the original input–output behavior. A weight adjustment is carried out by solving the system of linear equations using the conjugate gradient algorithm in the least squares sense. Unfortunately, if one large network is trained and pruned, the resulting error versus number of hidden units (N_h) curve is not minimal for smaller networks. In other words, it is possible, though unlikely, for each hidden unit to be equally useful after training.

A promising alternative to growing or pruning alone is to combine them. Huang et al. [17] have developed an online training algorithm for RBF-based minimal resource allocation networks (MRAN), which uses both growing and pruning. However, as MRAN is an online procedure, it does not optimize the network over all past training data. Also, network initialization requires prior knowledge on the data which may not be available.

In this paper, we describe batch mode methods for training and validating sequences of different size neural nets. Each method's training error versus the network size curve is monotonic or approximately monotonic. In Section 2, we give basic notation and briefly describe the output weight optimization-hidden Weight optimization (OWO-HWO) training algorithm. Section 3 gives brief analyses of weight initialization methods. In Section 4 a growing method is described. 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, two methods that combine growing and pruning are presented. It is shown that the combination approaches overcome the shortcomings of growing or pruning alone. Our proposed method is benchmarked against well-known growing algorithms in Section 5. The conclusions are given in Section 6.

2. Multilayer perceptron

In this section we define some notation and review an effective, convergent MLP training algorithm.

2.1. Notation

Fig. 1 depicts a feedforward 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

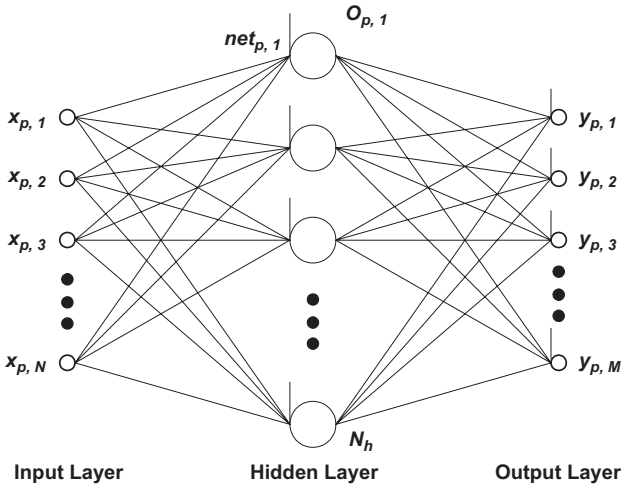

Fig. 1. MLP with single hidden layer.

function and activation are

$$net_{pj} = \sum_{i=1}^{N+1} w_h(j,i) \cdot x_{pi}, \quad 1 \leqslant p \leqslant N_v, \quad 1 \leqslant j \leqslant N_h \tag{1}$$

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

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

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

where $1 \leqslant k \leqslant M$. Here, $w_{oi}(k,i)$ is the weight connecting ith input to kth output and $w_{oh}(k,j)$ is the weight connecting jth hidden unit to kth output. 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 $\mathbf{x}_p$ and a desired output vector $\mathbf{t}_p$. For the pth pattern, the N input values are x_{pi}, $(1 \leqslant i \leqslant N)$ and the M desired output values are t_{pk} $(1 \leqslant k \leqslant M)$.

Example training algorithms are backpropagation [26,19], Levenberg Marquardt [13], Genetic Algorithms [2,8,22], cascade correlation (CC) [12] and OWO-HWO [5] which is described in the next subsection. The squared error for the pth pattern is

$$E_p = \sum_{k=1}^{M} [t_{pk} - y_{pk}]^2. \tag{4}$$

In order to train a neural network for one epoch the mean-squared error (MSE) for the ith output unit is defined as

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

The overall performance of an MLP network, measured as MSE, can be written as

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

2.2. OWO-HWO algorithm

In OWO-HWO training [5,34], we alternately update hidden weights [30] and solve linear equations for output weights [29]. It is well known that optimizing in the gradient direction, as in steepest descent, is very slow. So, an alternative approach for improving hidden weights (HWO) has been developed.

Let the desired net function net_{pjd} for the pth training pattern and jth hidden unit be

$$net_{pjd} = net_{pj} + Z \cdot \Delta net^*_{pj}, \tag{7}$$

where Δnet^*_{pj} written as

$$\Delta net^*_{pj} = net^*_{pj} - net_{pj} \tag{8}$$

is the difference between current net function net_{pj} and the optimal value net^*_{pj}. Now the net function approaches the optimal value net^*_{pj}, instead of moving in the negative gradient direction. The current task for constructing the HWO algorithm is to find Δnet^*_{pj}.

Using Taylor series expansion in Eq. (2) about net_{pj}, we get

$$O^*_{pj} = f(net^*_{pj}) = O_{pj} + f'_{pj} \cdot Z \cdot \Delta net^*_{pj}, \tag{9}$$

where f'_{pj} is the derivative of the activation function $f(net_{pj})$ with respect to net_{pj}. Now Δnet^*_{pj} can be derived based on

$$\left.\frac{\partial E}{\partial net_{pj}}\right|_{net_{pj}=net^*_{pj}} = 0. \tag{10}$$

The desired change in net function can be derived as

$$\Delta net^*_{pj} = \frac{\delta_{pj}}{(f'_{pj})^2 \sum_{i=1}^{M} w^2_{oh}(i,j)}. \tag{11}$$

It can be shown [34] that minimizing E is equivalent to minimizing the weighted hidden error function

$$E_\delta(k) = \frac{1}{N_v} \sum_{p=1}^{N_v} (f'_{pk})^2 \left[\Delta net^*_{pk} - \sum_{n=1}^{N+1} e(k,n) x_{pn} \right]^2, \tag{12}$$

where $e(k,n)$ is the change in hidden weight. The auto and crosscorrelation matrices are found as

$$R(n,m) = \frac{1}{N_v} \sum_{p=1}^{N_v} (x_{pn} \cdot x_{pm}) \cdot (f'_{pk})^2 \tag{13}$$

and

$$C_\delta(k,m) = \frac{1}{N_v} \sum_{p=1}^{N_v} (\Delta net^*_{pk} \cdot x_{pm}) \cdot (f'_{pk})^2. \tag{14}$$

We have $(N+1)$ equations in $(N+1)$ unknowns for the kth hidden unit. After finding the learning factor Z, the hidden weights are updated as

$$w_{hi}(k,n) \leftarrow w_{hi}(k,n) + Z \cdot e(k,n). \tag{15}$$

As discussed by Yu et al. [34], this HWO algorithm converges as the change in the error function E is non-increasing and the algorithm uses backtracking in order to save the best network without diverging from the solution. It is also shown that the convergence speed of the new algorithm is higher than those of other competing techniques.

After improving hidden weights using HWO, we optimize output weights using OWO [23] by solving the following linear equations

$$\mathbf{W}_0 \cdot \mathbf{R} = \mathbf{C}, \tag{16}$$

where $\mathbf{R} \in \Re^{(N+N_h+1)\times(N+N_h+1)}$ and $\mathbf{C} \in \Re^{M\times(N+N_h+1)}$ are auto and crosscorrelation matrices, respectively, as described in the Appendix. $\mathbf{W}_0 \in \Re^{M\times(N+N_h+1)}$ is the matrix of weights connecting the inputs and hidden units to the output units.

3. Structured initialization

MLP training is inherently very dependent on the initial weights, so proper initialization is critical. In this section, several basic approaches for feedforward network initialization are discussed. To define the problem being considered, assume that a set of MLPs of different sizes (i.e., different numbers of hidden units, N_h) are to be designed for a given training data set.

Let $E_f(N_h)$ denote the final training MSE, E, of an MLP with N_h hidden units.

Axiom 1. If $E_f(N_h) \geqslant E_f(N_h - 1)$, 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.

Three distinct types of network initialization are considered. We give theorems related to these three methods, which give a basis for this paper. Delashmit [11] has, in his work, identified problems in MLP training and has stated them in the form of these theorems. Detailed discussions can be found in his work.

3.1. Randomly initialized networks

When a set of MLPs are randomly initialized (RI), no members of the set are constrained to have any initial weights or thresholds in common. Practically, this means that the initial random number seeds (IRNS) of the networks are different. 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.

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*.

This means that the commonly used RI networks vary in performance, even when they are the same size. A problem with RI networks is that $E_f(N_h)$ is non-monotonic. 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 ith random number seed out of a total of N_s seeds. Let $E_{av}(N_h)$ denote the average $E_i(N_h)$, that is

$$E_{av}(N_h) = \frac{1}{N_s}\sum_{i=1}^{N_s} E_i(N_h). \tag{17}$$

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 [11] and that the network with $N_h + 1$ hidden units is useless.

$$P(E_{av}(N_h+1) > E_{av}(N_h)) \leqslant \frac{var(E_i(N_h+1)) + var(E_i(N_h))}{2\cdot N_s\cdot (m_{av}(N_h) - m_{av}(N_h+1))^2}, \tag{18}$$

where $var()$ represents the variance and $m_{av}(N_h)$ represent the average MSE for the network with N_h hidden units. This result quantifies the probability that RI networks have non-monotonic error versus N_h curves, for given values of N_h.

3.2. Common starting point initialized networks

One way to increase the likelihood of monotonic $E(N_h)$ curve is to force the corresponding initial basis functions to be nested. When a set of MLPs are common starting point initialized with structured weight initialization (CSPI-SWI), each one starts with the same IRNS. Also, 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, if present, are also identical. These networks are useful when it is desired to make performance comparisons of networks that have the same IRNS for the starting point.

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*.

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-SWI networks.

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

The above theorems have been proved by Delashmit [11]. Unfortunately, growing with CSPI-SWI networks although better than RI, does not guarantee a monotonic $E_f(N_h)$ curve. This is true because after training, the resulting networks are not guaranteed to have any hidden basis functions in common.

3.3. 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 networks are initialized using the final weights and thresholds that resulted from training a smaller network. Such dependently initialized (DI) networks take advantage of previous training on smaller networks in a systematic manner.

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. That is, $N_\varepsilon = N_h - N_{hp}$ new hidden units are added to a well-trained smaller network, to initialize the larger one. Then:

(i) $E_{int}(N_h) < E_{int}(N_{hp})$
(ii) $E_f(N_h) \leqslant E_f(N_{hp})$
(iii) $E_{int}(N_h) = E_f(N_{hp})$.

As seen in property (ii), DI networks have a monotonically non-increasing $E_f(N_h)$ versus N_h curve. This occurs because all basis functions in a small network are also present in larger networks. Unfortunately, there is no guarantee that a $E_f(N_h)$ sample represents a global minimum.

3.4. Pruned networks

One of the most popular regularization procedures in neural nets is done by limiting the number of hidden units [3]. Pruning generates series of different size networks starting from a large network and each subsequent network having one less hidden unit than the previous network. In other words, the hidden units will be ordered according to their importance and the pruning will sequentially delete them one at a time starting with the least important hidden unit. Hidden units of bigger networks will always be supersets of those of smaller networks.

Axiom 2. Assume that a large network with N_h hidden units is pruned, using the approach of [6,8,21]. Then the final MSE values satisfy $E_f(k) \leqslant E_f(k-1)$ for $1 \leqslant k \leqslant N_h$.

This axiom indicates that pruning produces monotonically non-increasing $E_f(N_h)$ versus N_h curves as do DI-based growing approaches.

4. Methods of generating networks of different sizes

According to the structural risk minimization (SRM) principle, the best network in an ensemble of MLPs is the one that has minimum guaranteed risk. However, the guaranteed risk is an upper bound for the generalization error $E_{\text{val}}(N_{\text{h}})$. Hence the best network is the one for which $E_{\text{val}}(N_{\text{h}})$ is the smallest. In this section we present four general approaches or design methodologies for generating sequences of networks having monotonic $E_f(N_{\text{h}})$ curves.

4.1. Growing method or Design Methodology 1

In growing methods, we start by training a small network (with small N_{h}) and then successively train larger networks by adding more hidden units, producing a final training MSE versus N_{h} curve, $E_f(N_{\text{h}})$. After training a sequence of these different size DI networks, we calculate validation MSE versus N_{h} curve $E_{\text{val}}(N_{\text{h}})$. The validation error will be the key to decide the network size needed. Growing methods, which are motivated by the properties of DI networks, are collectively denoted as Design Methodology 1 (DM-1).

Given training data $\{\mathbf{x}_p, \mathbf{t}_p\}_{p=1}^{N_v}$, our growing algorithm, which is called DM-1 is described as follows. First, we train a linear network (no hidden units, $N_{\text{h}} = 0$) using the OWO algorithm and the network obtained is represented by $\mathscr{W}_{\text{DM1}}^{0}$. Then we successively add a few hidden units ($N_\varepsilon = 5$ in our case) and re-train the network. During re-training, for the first few iterations (say 20% of the total iterations), we train only the newly added hidden units using OWO-HWO [5] as described in Section 2.2. Then during the remaining iterations, we train all the weights and thresholds. The network obtained at the end of training is represented by $\mathscr{W}_{\text{DM1}}^{N_{\text{h}}}$. Also at each step of growing, we validate the network obtained at that point with validation data set $\{\mathbf{x}_p^{\text{val}}, \mathbf{t}_p^{\text{val}}\}_{p=1}^{N_v^{\text{val}}}$ to help decide upon the final network size. Let $E_{\text{val}}(\mathscr{W}_{\text{DM1}}^{N_{\text{h}}})$ be the validation error for a DM-1 network with N_{h} hidden units. The best network among the sequence of different size networks can be chosen as

$$\mathscr{W}_{\text{DM1}}^{N_{\text{h}}^{\text{opt}}} = \text{argmin } E_{\text{val}}(\mathscr{W}_{\text{DM1}}^{k}), \tag{19}$$

where $\mathscr{W}_{\text{DM1}}^{N_{\text{h}}^{\text{opt}}}$ represents the DM-1 network with optimal number of hidden units $N_{\text{h}}^{\text{opt}}$ and N_{hFinal} represents the maximum number of hidden units at the end of growing. This number should be conveniently large so that the best network falls within the sequence of grown DM-1 networks. The validation error is the MSE on the validation set, given by

$$E_{\text{val}} = \frac{1}{N_v^{\text{val}}} \sum_{p=1}^{N_v^{\text{val}}} \sum_{k=1}^{M} (t_{pk}^{\text{val}} - y_{pk}^{\text{val}})^2. \tag{20}$$

In order to validate the performance of the method, we repeat the growing procedure several times with different random numbers initializing the network. Then the average MSEs are calculated, which gives the expected values of the MSEs for each size network. Also, to obtain a measure for the confidence interval of the method, we calculate the standard deviations of the MSEs. The average MSEs are given by

$$\mathscr{M}(E_{\text{val}}) = \frac{1}{N_{\text{r}}} \sum_{i=1}^{N_{\text{r}}} E_{\text{val}}^i, \tag{21}$$

where E_{val}^i are the validation MSEs for the ith random initial network and N_{r} is the number of random initial networks. The standard deviations of the MSEs are given by

$$\mathscr{SD}(E_{\text{val}}) = \left[\frac{1}{N_{\text{r}}} \sum_{i=1}^{N_{\text{r}}} (E_{\text{val}}^i - \mathscr{M}(E_{\text{val}}))^2 \right]^{1/2}. \tag{22}$$

Fig. 2 shows the sample plot of the training and validation MSEs as a function of number of hidden units (N_{h}) for $N_{\text{r}} = 10$. Note that in this method, there will not be any intermediate size networks as the step size used to grow the network is equal to five ($N_\varepsilon = 5$). A more detailed analysis on the plots of means and standard deviations of the design methodologies will be made at the end of this section and hence at this point we have not shown these plots.

4.2. Ordered pruning or Design Methodology 2

In Design Methodology 2 (DM-2), we prune a large network in order to produce a monotonic $E_f(N_{\text{h}})$ curve. The pruning procedure is explained in appendix. During pruning, the orthonormalization matrix $\mathbf{A} = \{a_{mk}\}$ for $1 \leqslant m \leqslant N_u$ and $1 \leqslant k \leqslant m$ is saved, where $N_u = N + N_{\text{h}} + 1$ is the total number of basis functions.

4.2.1. One pass validation

Given the matrix $\mathbf{A}$ and the MLP network with ordered hidden units, we wish to generate the validation error versus N_{h} curve $E_{\text{val}}(N_{\text{h}})$ from the validation data set $\{\mathbf{x}_p^{\text{val}}, \mathbf{t}_p^{\text{val}}\}_{p=1}^{N_v^{\text{val}}}$. For each pattern, we augment the input vector as in the previous sections. So the augmented input vector is $\mathbf{x}_p^{\text{val}} \leftarrow [\mathbf{x}_p^{\text{val}^{\text{T}}}, 1, \mathbf{O}_p^{\text{val}^{\text{T}}}]^{\text{T}}$. Then the augmented vector is converted into orthonormal basis functions by the transformation

$$x_p^{\text{val}'}(m) = \sum_{k=1}^{m} a_{mk} \cdot x_p^{\text{val}}(k) \quad \text{for } 1 \leqslant m \leqslant N_u. \tag{23}$$

In order to get the validation error for all size networks in a single pass through the data, we use the following strategy:

Let $y_{pi}^{\text{val}}(m)$ represent the ith output of the network having m hidden units for the pth pattern, let $E_{\text{val}}(m)$ 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

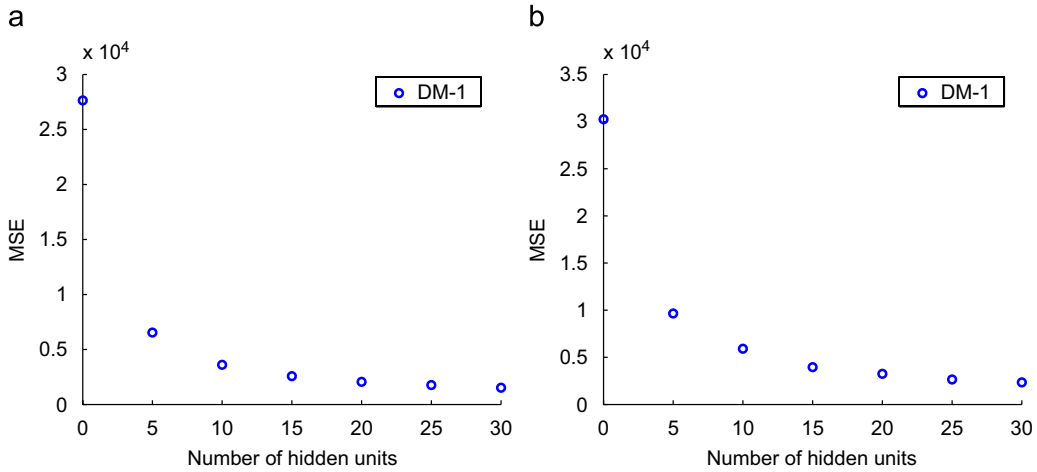

Fig. 2. MSEs of speech data set for different network sizes. (a) Training and (b) validation.

calculated as follows:

$$y_{pi}^{\mathrm{val}}(0) = \sum_{k=1}^{N+1} w_0'(i,k) \cdot x_p^{\mathrm{val}'}(k) \quad \text{for } 1 \leqslant i \leqslant M,$$

$$E_{\mathrm{val}}(0) \leftarrow E_{\mathrm{val}}(0) + \sum_{i=1}^{M} [t_{pi}^{\mathrm{val}} - y_{pi}^{\mathrm{val}}(0)]^2. \tag{24}$$

Then for $1 \leqslant m \leqslant N_{\mathrm{h}}$, the following two steps are performed:

Step 1: For $1 \leqslant i \leqslant M$

$$y_{pi}^{\mathrm{val}}(m) = y_{pi}^{\mathrm{val}}(m-1) + w_{o\mathrm{h}}'(i,m) \cdot O_p^{\mathrm{val}'}(m). \tag{25}$$

Step 2:

$$E_{\mathrm{val}}(m) \leftarrow E_{\mathrm{val}}(m) + \sum_{i=1}^{M} [t_{pi}^{\mathrm{val}} - y_{pi}^{\mathrm{val}}(m)]^2, \tag{26}$$

where $w_{o\mathrm{h}}'(i,m)$ is the orthonormal weight connecting mth hidden unit to ith output. Apply Eqs. (23)–(26) for $1 \leqslant p \leqslant N_v$ and get the total validation error over all the patterns for each size network. Then these error values should be normalized as

$$E_{\mathrm{val}}(m) \leftarrow \frac{E_{\mathrm{val}}(m)}{N_v^{\mathrm{val}}} \quad \text{for } 0 \leqslant m \leqslant N_{\mathrm{h}}. \tag{27}$$

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

4.3. Pruning a grown network or Design Methodology 3

The DI network growing approach (DM-1) generates monotonic $E_f(N_{\mathrm{h}})$ 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 $E_f(N_{\mathrm{h}})$ sample represents a global minimum. There is also no guarantee that the hidden units are ordered properly, since useless hidden units can occur at any point in the growing process. In Design Methodology 3 (DM-3a), we attempt to solve these problems by (1) performing growing as in DI network (DM-1) and (2) performing ordered pruning of the final network (DM-2). This forces the grown hidden units to be stepwise optimally ordered. This method works better than pruning a large trained network because the growing by DI network approach would produce sets of hidden units that are optimally placed with respect to the previously trained hidden units. Pruning the grown network (DM-3a) always produces a monotonic $E_f(N_{\mathrm{h}})$ curve. Let us denote the DM-3a network for N_{h} hidden units by $\mathscr{W}_{\mathrm{DM3a}}^{N_{\mathrm{h}}}$.

An alternative solution (DM-3b) would be to save the intermediate grown networks and to select the best performing network obtained either by DM-1 or DM-3a step. In other words,

$$\mathscr{W}_{\mathrm{DM3b}}^{k} = \underset{dm \in \{\mathrm{DM1,DM3a}\}}{\operatorname{argmin}} E_{\mathrm{val}}\{\mathscr{W}_{dm}^{k}\} \tag{28}$$

for $1 \leqslant k \leqslant N_{\mathrm{hFinal}}$, and the optimal network is obtained by picking the network with minimum validation MSE.

$$\mathscr{W}_{\mathrm{DM3b}}^{N_{\mathrm{h}}^{\mathrm{opt}}} = \operatorname{argmin} E_{\mathrm{val}}\{\mathscr{W}_{\mathrm{DM3b}}^{k}\}. \tag{29}$$

One problem with this method is that the training error curve $E_{\mathrm{trn}}(N_{\mathrm{h}})$ and the validation error curve $E_{\mathrm{val}}(N_{\mathrm{h}})$ are not going to be monotonic, but the MSEs for smaller number of hidden units will improve over the case where there is no minimum operator. Let us denote the former case, without the minimum operator as DM-3a and the later case, with the minimum operator as DM-3b. Fig. 3 shows the training and validation MSEs for the two cases. It is clearly seen that monotonicity has been lost in DM-3b. However, as the scale on the y-axis is very large, the actual value of the MSE is significantly lower for some networks.

Unfortunately, the DM-3 approaches do not guarantee that all the hidden units for every network are useful. Some of the hidden units may be saturated and increase complexity while contributing very little to the generalization of the network. Also these saturated hidden units might affect the learning of other hidden units.

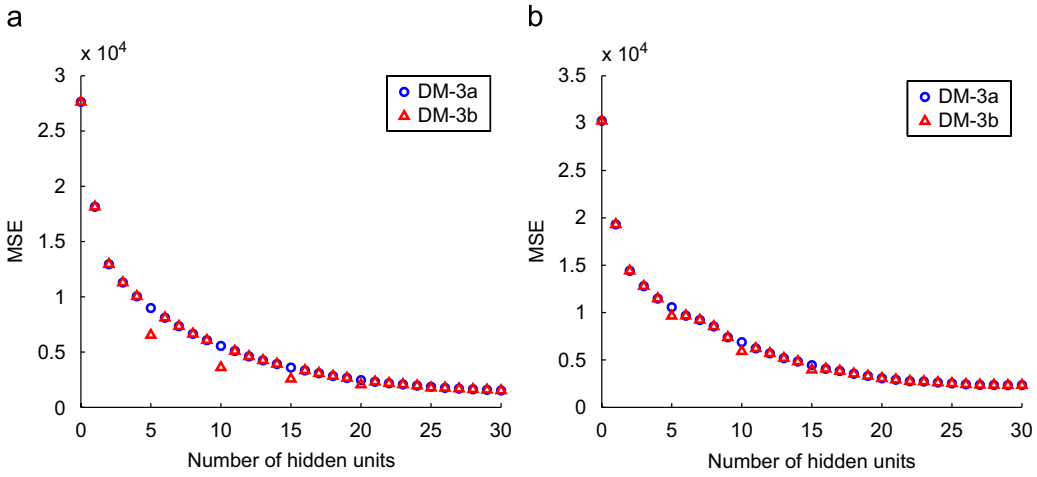

Fig. 3. MSEs of Speech data set for different network sizes (DM-3) for $N_r = 10$. (a) Training and (b) validation.

4.4. Integrated growing, pruning or Design Methodology 4

The problems faced in DM-3 approaches can be solved by our last approach called DM-4, where we simultaneously grow and prune the hidden units of the network. During pruning, we delete the hidden units which contribute less than η% to the network performance (i.e., less than η% change in MSE when removed). This technique eliminates any saturated hidden units and in turn helps in adding new hidden units with different starting points. Hence the technique is termed a Pseudo-Genetic approach.

As in DM-3, we can save all the intermediate networks and select the best performing network obtained either by the DM-1 or DM-2 step. Note that in DM-4, we have lot more networks in the pool and we should choose the one with the minimum validation error. Again, we can denote the case with no minimum operator by DM-4a and the one with the minimum operator by DM-4b.

Algorithm (*DM-4b*). Given the training data and maximum number of hidden units N_{hFinal}

(i) Train a linear network to get $w_{oi}(k,i) \in \mathscr{W}^{0}_{\mathrm{DM4}}$ and $w_{\mathrm{h}i}(j,i) \in \mathscr{W}^{0}_{\mathrm{DM4}}$ (here $N_{\mathrm{h}} = 0$), where $1 \leqslant k \leqslant M$, $1 \leqslant j \leqslant N_{\mathrm{h}}$ and $1 \leqslant i \leqslant N$.

(ii) Add N_{ε} hidden units with random initial input weights and zero output weights ($N_{\mathrm{h}} = N_{\mathrm{h}} + N_{\varepsilon}$).

(iii) Train only the new hidden units for 20% of the iterations and the entire network for the remaining 80% of the iterations. We will get a new set of $w_{oi}(k,i) \in \mathscr{W}^{N_{\mathrm{h}}}_{\mathrm{DM4}}$, $w_{o\mathrm{h}}(k,j) \in \mathscr{W}^{N_{\mathrm{h}}}_{\mathrm{DM4}}$ and $w_{\mathrm{h}i}(j,i) \in \mathscr{W}^{N_{\mathrm{h}}}_{\mathrm{DM4}}$, where $1 \leqslant k \leqslant M$, $1 \leqslant j \leqslant N_{\mathrm{h}}$ and $1 \leqslant i \leqslant N$.

(iv) Prune the network: Order the hidden units using the schmidt procedure explained in Section 4.2. Then compute the errors for each size of the network ($E_f(N_{\mathrm{h}})$) and delete those hidden units which results in less than $\eta = 1$% error change when removed. The sequence of networks obtained from pruning (DM-2) the network $\mathscr{W}^{N_{\mathrm{h}}}_{\mathrm{DM4}}$ will be denoted by $\mathscr{W}^{(k,N_{\mathrm{h}})}_{\mathrm{DM4}}$, for $1 \leqslant k \leqslant N_{\mathrm{h}}$.

(v) If $N_{\mathrm{h}} < N_{\mathrm{hFinal}}$, then goto step 2.

(vi) If $N_{\mathrm{h}} \neq N_{\mathrm{hFinal}}$ add appropriate number of hidden units so that the total number of hidden units equal N_{hFinal} and train the network the same way as in step 3.

(vii) For each size network, we have to select the network that gives minimum validation error. We get an equation similar to Eq. (28).

$$\mathscr{W}^{(k,N_{\mathrm{hFinal}})}_{\mathrm{DM4}} = \underset{n \in \mathbb{N}_h}{\operatorname{argmin}} E_{\mathrm{val}}\{\mathscr{W}^{(k,n)}_{\mathrm{DM4}}\} \tag{30}$$

for $0 \leqslant k \leqslant N_{\mathrm{hFinal}}$. Here $\mathbb{N}_{\mathrm{h}}$ is the population of all the networks obtained during step 4 of the DM-4 procedure. The best final network chosen will be that which gives the minimum validation error over the sequence of all size networks starting from no hidden units to N_{hFinal} hidden units. We get an equation similar to Eq. (29):

$$\mathscr{W}^{(k^{\mathrm{opt}},N_{\mathrm{hFinal}})}_{\mathrm{DM4}} = \operatorname{argmin} E_{\mathrm{val}}\{\mathscr{W}^{(k,N_{\mathrm{hFinal}})}_{\mathrm{DM4}}\}. \tag{31}$$

It can be seen from Fig. 4 that the minimum operator not only decreases the MSE for smaller size networks, but also preserves the monotonicity fairly well. Hence, DM-4b is the best of the proposed methods. In the remainder of this paper, DM-4 refers to DM-4b.

In Fig. 5, we have shown the performances of the four design methodologies discussed in this section. Fig. 5(a) shows the averages of the MSEs obtained over different random initialization of the network during training and Fig. 5(b) shows the standard deviations of the MSEs over

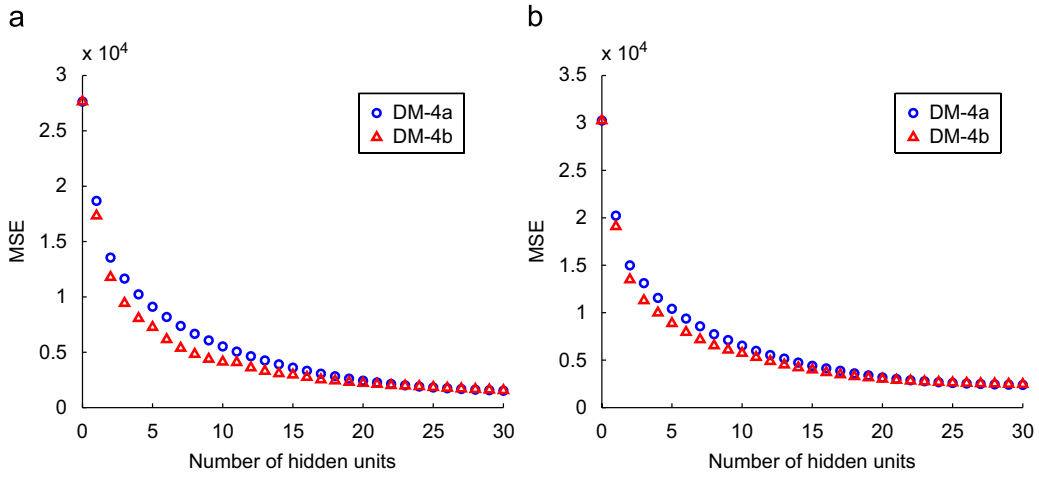

Fig. 4. MSEs of speech data set for different network sizes (DM-4) for $N_r = 10$. (a) Training and (b) validation.

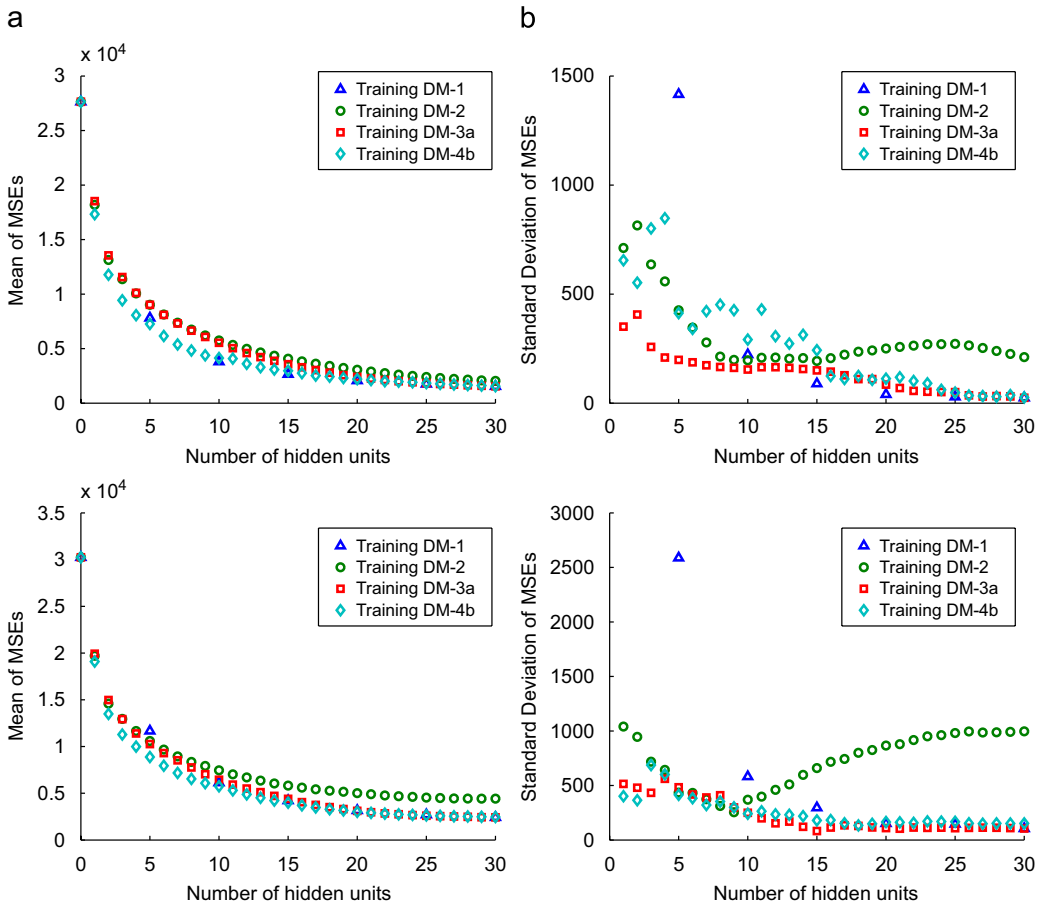

Fig. 5. Comparison of our design methodologies for the speech data for $N_r = 10$. (Top row is training result and bottom row is validation result.) (a) Means and (b) standard deviations of the MSEs over different initial networks (random number seeds).

these initializations. The averages and standard deviations of MSEs are necessary to analyze the robustness of the various techniques.

The following are the comments and observations on these plots. Clearly, we can see the disadvantage of DM-2, in which we train a large network and prune to obtain

different size networks. Both the training and validation errors are high in this case. Also, the standard deviation increases for larger networks (i.e., as number of hidden units N_h increases) in DM-2, which indicates that the larger networks in this technique are more sensitive to initial weights. This also demonstrates that the growing method performs better than training a RI large network. In DM-1, the average MSE for smaller networks is less than that of the other methods, but the standard deviation is very high when compared to other methods. This shows that the smaller networks in DM-1 are very sensitive to initial weights. As our goal is to find a network generation technique that is less sensitive to initial conditions and have monotonic error versus number of hidden units curve, we need to find a technique that has advantages of both DM-1 and DM-2. So, in DM-3, we first grow the MLP and later prune the final large grown network. From the plots, it can be seen that both average MSE and standard deviation of the MSE for DM-3 are less than for the other methods. Further, if there are saturated hidden units or the hidden units that are linearly dependent, those units can be removed. This is achieved using the DM-4 approach, where the hidden units are pruned in every step of growing. Consistent to our analysis, the DM-4 method does give the best result.

Note that in DM-4, **R** and **C** do not need to be calculated for pruning in Eqs. (A.5) and A.11since they are already found during training in the OWO step. This saves an extra pass through the data at each growing step. The number of multiples eliminated for calculating the hidden unit activations, auto and crosscorrelation matrices is

$$N_m = N_v \cdot \left[(M+N)\cdot N_h + \frac{N_h\cdot(N_h+1)}{2}\right]. \tag{32}$$

This savings in multiples is significant, though small.

5. Numerical results

In the previous section, we show that DM-4 has advantages over other training algorithms we have developed. In this section, we briefly discuss two existing well-known DI approaches for growing a sequence of networks and compare them to DM-4.

5.1. Existing growing methods

5.1.1. Constructive backpropagation (CBP)

In the CBP algorithm of Lehtokangas [19], the back-propagation algorithm is used to train the network. Let the initial network trained be linear ($N_h = 0$) and be denoted by $\mathscr{W}_{cbp}^{0}$. Then add a batch of hidden units ($N_h = N_h + N_\varepsilon$) and train them to get another network $\mathscr{W}_{cbp}^{N_h}$. Once the new hidden units are trained, their input and output weights are frozen and further training is continued only on the newly added hidden units. There is only one error function to be minimized, which is the squared error between the previous error of the output unit and the weighted sum of the newly added hidden units:

$$\begin{aligned} E_{cbp} &= \sum_{p=1}^{N_v}\sum_{k=1}^{M}\left\{t_{pk} - y_{pk}^{cbp}\right\}^2 \\ &= \sum_{p=0}^{N_v}\sum_{k=1}^{M}\left\{t_{pk} - \left(\sum_{i=1}^{N+1} w_{oi}^{cbp}(k,i)\cdot x_{pi} + \sum_{j=1}^{N_h-N_\varepsilon} w_{oh}^{cbp}(k,j)\cdot O_{pj}\right) - \sum_{n=N_h-N_\varepsilon+1}^{N_h} w_{oh}^{cbp}(k,n)\cdot O_{pn}\right\}^2 \\ &= \sum_{p=1}^{N_v}\sum_{k=1}^{M}\left\{e_{pk}^{cbp} - \sum_{n=N_h-N_\varepsilon+1}^{N_h} w_{oh}^{cbp}(k,n)\cdot O_{pn}\right\}^2. \end{aligned} \tag{33}$$

Here y_{pk}^{cbp} is the kth output of the pth pattern of the network $\mathscr{W}_{cbp}^{N_h}$, e_{pk} is the error between the desired output and the output of the previous network $\mathscr{W}_{cbp}^{N_h-N_\varepsilon}$. The weights connecting the inputs and hidden units to the output units respectively, are represented by $w_{oi}^{cbp} \in \mathscr{W}_{cbp}^{N_h}$ and $w_{oh}^{cbp} \in \mathscr{W}_{cbp}^{N_h}$.

Also it is shown in [19] that adding multiple hidden units at a time and training together is much more efficient than adapting many single units trained independently. Hence in this paper, we have fixed the growing step size to be five.

In order to compare the convergence time of the algorithms, we have to compute the number of multiples for CBP training as shown in the following inequality:

$$\begin{aligned} M_{cbp} \leqslant N_{it}N_v\Big\{&0.2N_\varepsilon\left[N + N_h + \frac{1}{2}(1-N_\varepsilon) + M\right] \\ &+ 0.8\left[(N+M)N_h + \frac{N_h(N_h+1)}{2}\right] \\ &+ [N_h(N+2M+3)]\Big\}. \end{aligned} \tag{34}$$

Here N_ε is the number of new hidden units added in every growing step. M_{cbp} is proportional to the time taken for the entire training algorithm.

5.1.2. Cascade correlation (CC)

The CC learning procedure [12] is similar to CBP, in that the initial network is linear (i.e., $N_h = 0$). The network obtained is represented by $\mathscr{W}_{cc}^{0}$. We have slightly modified the original CC algorithm so that we can add multiple hidden units at each growing step, making it easier to compare with the other methods. Also, the CC architecture can be viewed as standard single hidden layer MLP with additional lateral weights in the hidden layer connecting every hidden unit to all its previous hidden units. This set of lateral weights can be represented by $w_{hh}^{cc}(i,j) \in \mathscr{W}_{cc}^{N_h}$ for $2 \leqslant i \leqslant N_h$ and $1 \leqslant j < i$. Therefore, $w_{hh}^{cc}(i,j)$ is the weight going from jth hidden unit to ith hidden unit. The cost

function to be maximized is

$$E_{cc} = \sum_{j=N_h-N_c+1}^{N_h} \sum_{k=1}^{M} \left| \sum_{p=1}^{N_v} (O_{pj} - \hat{O}_j) \cdot (E_{pk} - \hat{E}_k) \right|, \quad (35)$$

where $\hat{O}_j$ is the mean of the activations of the jth hidden unit and $\hat{E}_k$ is the mean of the errors of the kth output unit over all the patterns. Here also, once the hidden unit's input and output weights are trained, they will be frozen and the next step in growing will adapt only the weights connecting the new hidden units.

The number of multiples for CC training is given in the following inequality:

$$M_{cc} \leqslant N_{it}\{N_v[4MN_h + N_c(3N_h + 5M)] + N_c(2M + 5)\}. \quad (36)$$

5.2. Comparison results

Here, we compare DM-4 to CBP and CC. Note that both these latter techniques are examples of DM-1. All the techniques were evaluated on seven different data sets. As explained in the previous section, all techniques are executed $N_r = 10$ times with different random initial weights. The average and standard deviation of MSEs for $N_h = 5, 10, 30$ on all the data sets are shown in Table 1. The italicized values are the ones in which our proposed methodology has suboptimal performance compared to other algorithms. We have justified this behavior in the discussion of each data set that is presented in the following. Also some example plots of the average and standard deviation of the MSEs versus hidden units are given for a couple of data sets. The first five data sets are available for public use [33]. The last two datasets are the well-known benchmark data sets available for public to access at StatLib repository and PROBEN1 collection.

Speech: This 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 are 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. Fig. 6 gives the performance comparison of the proposed method with the CBP and CC algorithms. It can be seen clearly in Fig. 6 and Table 1 that the averages and standard deviations of validation MSEs for all size networks for the proposed method are less than for the CC and CBP algorithms.

California housing: This is a data set obtained from the StatLib repository. The information was collected on the variables using all the block groups in California from the 1990 Census. In this sample a block group on average includes 1425.5 individuals living in a geographically compact area. Naturally, the geographical area included varies inversely with the population density. The distances are computed among the centroids of each block group as measured in latitude and longitude. The final data contained 20,640 observations on nine variables. The dependent variable is ln(median house value). It can be seen clearly in Fig. 7 and Table 1 that the average MSE and standard deviation of MSE for all size networks in the proposed method are smaller than those for CC and CBP.

F17: This set contains prognostics data for onboard flight load synthesis (FLS) in helicopters [5], where we estimate mechanical loads on critical parts, using measurements available in the cockpit. There are 17 inputs and nine outputs. From Table 1, it is clearly seen that our proposed method outperforms the existing techniques in both

Table 1
Performance comparison of various data sets for $N_r = 10$

	$N_h = 5$			$N_h = 10$			$N_h = 30$		
Data sets	CBP	CC	DM-4b	CBP	CC	DM-4b	CBP	CC	DM-4b
Speech	2.83e + 004	2.65e + 004	8.88e + 003	1.86e + 004	1.19e + 004	5.74e + 003	9.60e + 003	5.83e + 003	2.46e + 003
	1.31e + 003	1.46e + 003	4.15e + 002	3.32e + 003	1.29e + 003	2.41e + 002	6.20e + 002	3.15e + 002	1.52e + 002
F17	1.84e + 008	1.83e + 008	3.52e + 007	1.54e + 008	1.41e + 008	2.31e + 007	1.07e + 008	9.46e + 007	1.65e + 007
	1.08e + 007	7.13e + 006	5.20e + 006	1.04e + 007	1.14e + 007	3.33e + 006	6.19e + 006	5.85e + 006	4.75e + 006
Oh7	3.07e + 000	3.01e + 000	1.54e + 000	1.84e + 000	1.77e + 000	1.51e + 000	1.55e + 000	1.52e + 000	1.61*e* + 000
	2.28e − 001	1.93e − 001	1.81e − 002	9.63e − 002	9.62e − 002	1.87e − 002	1.61e − 002	1.58e − 002	4.54*e* − 002
Single2	1.49e + 000	1.49e + 000	6.55e − 002	9.57e − 001	1.12e + 000	4.88e − 002	4.99e − 001	6.12e − 001	6.08e − 002
	3.00e − 002	3.97e − 002	3.59*e* − 002	2.44e − 001	1.68e − 001	3.55e − 002	1.76e − 001	2.06e − 001	4.82e − 002
Twod	3.39e − 001	3.37e − 001	1.71e − 001	2.85e − 001	2.92e − 001	1.37e − 001	2.34e − 001	2.56e − 001	1.16e − 001
	9.80e − 003	8.39e − 003	6.89e − 003	1.10e − 002	2.81e − 002	4.30e − 003	1.09e − 002	2.26e − 002	5.31e − 003
Cal. housing	4.54e + 009	4.58e + 009	3.31e + 009	4.20e + 009	4.13e + 009	3.06e + 009	3.88e + 009	3.72e + 009	2.88e + 009
	8.17e + 007	1.85e + 008	6.65e + 007	1.58e + 008	1.19e + 008	4.35e + 007	2.70e + 008	2.10e + 008	1.08e + 008
Building1	2.44e − 002	2.41e − 002	2.17e − 002	2.31e − 002	2.24e − 002	2.26*e* − 002	2.39e − 002	2.52e − 002	3.07*e* − 002
	5.81e − 004	5.56e − 004	1.76*e* − 003	1.66e − 003	1.55e − 003	1.40e − 003	2.16e − 003	5.36e − 003	8.11*e* − 003

First row is the mean and the second row is the standard deviation of the validation MSEs over different initial random networks.

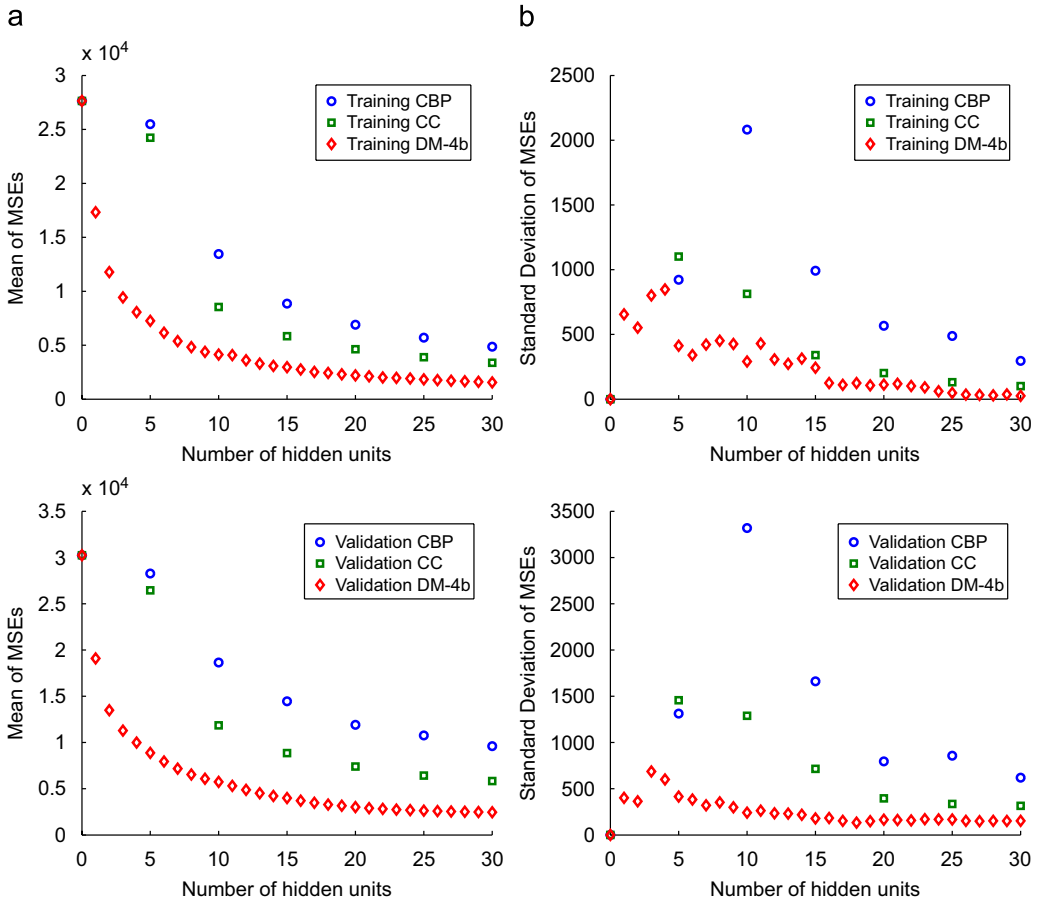

Fig. 6. Comparison of our design methodologies with the other methods for the speech data for $N_r = 10$. (Top row is training result and bottom row is validation result.) (a) Means and (b) standard deviations of the MSEs over different initial networks.

averages and standard deviations of validation MSE for all the network sizes considered. The large MSE values for this data set are due to the large variances of the desired outputs.

Oh7: This data set is for inversion of radar scattering from bare soil surfaces [28]. It has 20 inputs and three outputs. The training set contains VV and HH polarization at L 30°, 40°, C 10°, 30°, 40°, 50°, 60°, and X 30°, 40°, 50° along with the corresponding unknowns rms surface height, surface correlation length, and volumetric soil moisture content in g/cm^3. In this case, it is observed from Table 1 that the average and standard deviation of validation MSEs for big networks ($N_h = 30$) for the proposed method are larger than the existing methods. However, the averages and standard deviations of validation MSE for smaller network ($N_h = 5, 10$) are lesser. Hence the networks with larger MSEs are discarded.

Single2: This data set is for the inversion of surface permittivity [14]. These data have 16 inputs and three outputs. The inputs represent the simulated back scattering coefficient measured at 10°, 30°, 50°and 70°at both vertical and horizontal polarization. The remaining eight are various combinations of ratios of the original eight values. For this data set, all the averages of validation MSEs for the proposed method are clearly better than for the existing methods as seen in Table 1, but the standard deviation of validation MSE for $N_h = 5$ on the proposed method is a little higher. However, the average MSEs for proposed method are so much better than for the existing methods, that the higher standard deviation does not affect actual performance.

Twod: This training file [9,10] is used in the task 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 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 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

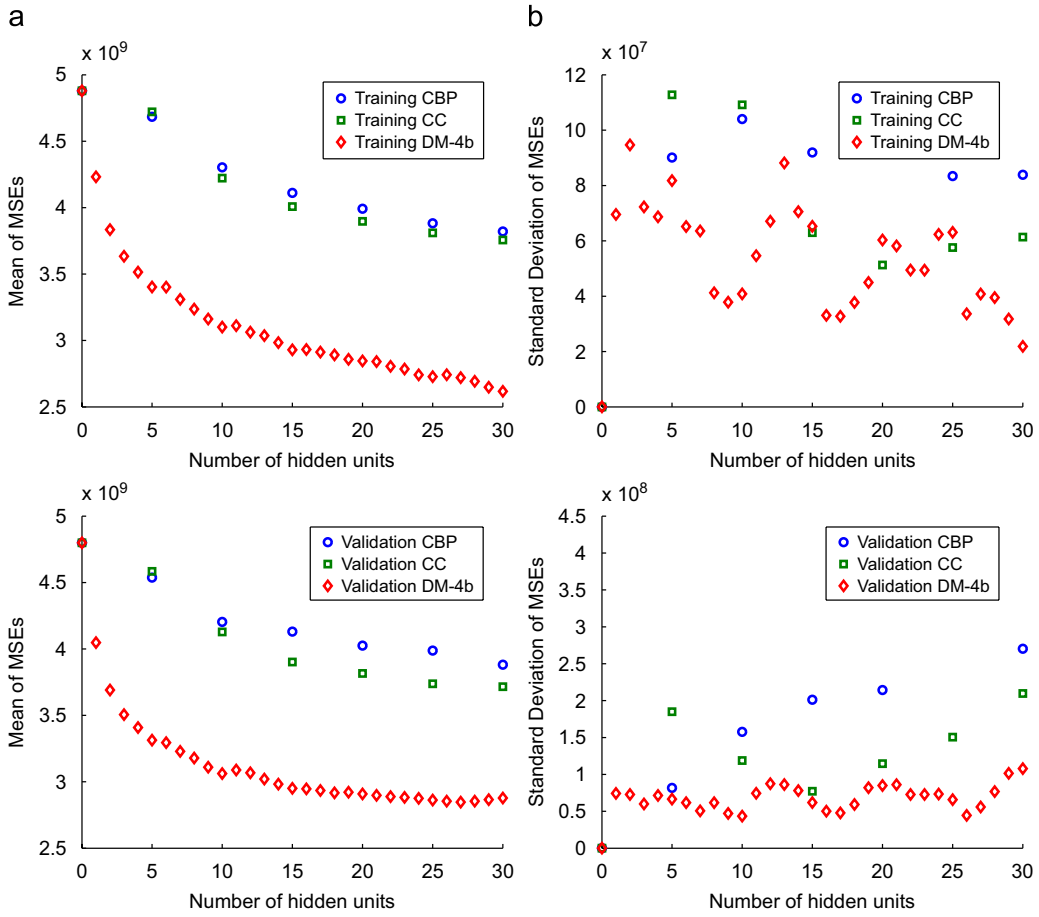

Fig. 7. Comparison of our design methodologies with the other methods for the California housing data for $N_r = 10$. (Top row is training result and bottom row is validation result.) (a) Means and (b) standard deviations of the MSEs over different initial networks.

scattering albedo which had a joint uniform pdf. Again on this data set, it is clearly seen in Table 1 that our proposed method outperforms the existing techniques in both averages and standard deviations of validation MSE for all the network sizes considered.

Building1: The Building1 problem (taken from PROBEN1 benchmark collection) predicts the energy consumption in a building. It tries to predict the hourly consumption of electrical energy, hot water, and cold water, based on the date, time of day, outside temperature, outside air humidity, solar radiation, and wind speed. It has 14 inputs, three outputs, and 4208 patterns. It is seen in Table 1 that our proposed method has shown sub-optimal performance for larger networks ($N_h = 10, 30$) compared to the CBP and/or CC algorithms on this data set. However, for $N_h = 5$, the average validation MSE of our method is lesser than the other methods considered, but the standard deviation is higher than the other methods. For $N_h = 5$, our proposed method with any random initial network performs better than or same as any of the two methods CBP and CC. Hence, we discard all the big networks and save the network which gives the least validation MSE which will be the network with $N_h = 5$.

5.3. Convergence related issues

Several issues related to convergence remain to be discussed. These include (1) determining how well the desired outputs can be approximated, (2) the number of required multiples for the various training algorithms, and (3) the actual training speed of the algorithms.

First, we compute the coefficient of determination (R^2) values that will give information about the goodness of fit of the model. For each data set we have compared R^2 values (for both training and validation) of our proposed algorithm with the two benchmark techniques. For computing this, we use the sum of squared errors (SS_E) of the best network for each data set.

$$R^2 = 1 - \frac{SS_E}{SS_T}, \tag{37}$$

Table 2
Coefficient of determination (R^2) values for all data sets computed using the sum of squared errors of the best network for each technique

Datasets	Training			Validation		
	Constructive backpropagation	Cascade correlation	OWO-HWO (DM-4b)	Constructive backpropagation	Cascade correlation	OWO-HWO (DM-4b)
Speech	0.895355	0.927280	0.967146	0.796655	0.876492	0.949425
F17	0.972791	0.975443	0.997999	0.970566	0.973926	0.996710
Oh7	0.841103	0.843756	0.883139	0.838076	0.841178	*0.832213*
Single2	0.992278	0.989352	0.999960	0.986611	0.983582	0.999400
Twod	0.803962	0.802961	0.923472	0.788988	0.770901	0.902647
Cal. housing	0.714922	0.719687	0.804399	0.705777	0.718300	0.777062
Building1	0.900011	0.892534	0.934435	0.110040	0.132986	0.135555

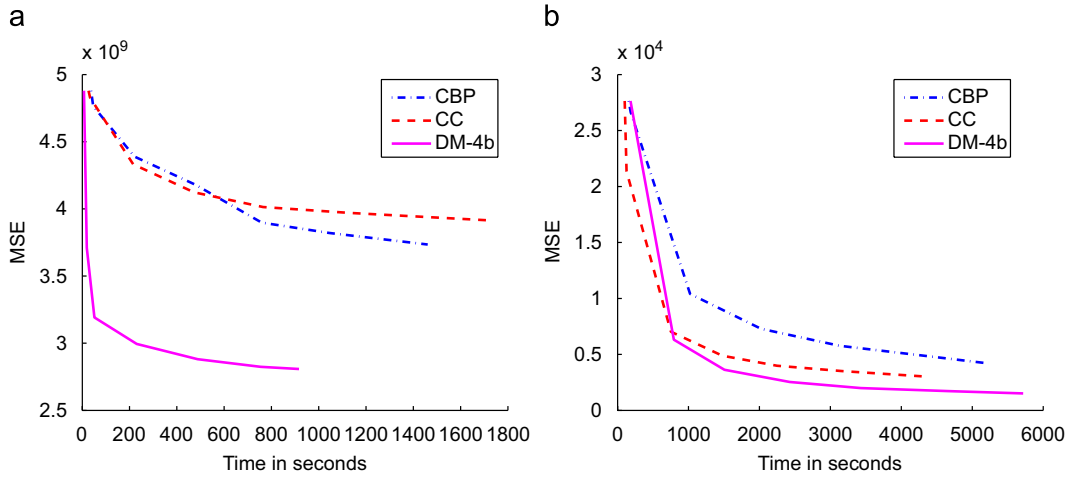

Fig. 8. Comparison of convergence time of our design methodologies with the other methods for the (a) California housing, (b) speech data sets. Note that these plots show MSEs for only one random number seed and hence they are not average MSEs.

where SS_E is the sum of squared error (residual error) and SS_T is the total sum of squares of the desired outputs. Table 2 shows the coefficient of determination values for both training and validation. Note that in terms of model fitting (training R^2), our proposed method outperforms the other methods. In terms of generalization, our proposed method is better in all of the data sets except for Oh7 where the R^2 value is insignificantly lower than that of the other methods.

In order to analyze the complexity of the training algorithms, we compare the total number of multiples for each training algorithm to train a sequence of networks. Then we give comparison plots of MSE versus time in seconds for the three training algorithms on example data sets. The number of multiples for DM-4 is given below:

$$M_{DM4} \leqslant M_{cbp} + N_{it}\left\{\frac{7N^2+3N}{2} + N_h N(N+1)\right\} + \frac{N_h(N_h+1)(7N^2+3N)}{4}. \quad (38)$$

As mentioned before, N_e is the number of hidden units added in each growing step (see Step 2 of Algorithm DM-4b), M_{cbp} is given in Eq. (34). From the above equation, it is clear that the number of multiples for DM-4 algorithm is greater than that of CBP. However, it should be noted that the additional term is very small compared to the M_{cbp} value as it does not involve the number of patterns, N_v, which is usually very large. This small additional computation sometimes helps the algorithm to converge faster as discussed below.

Figs. 8(a) and (b) show the plots of the MSE versus time in seconds for California housing and speech data sets, respectively. Note that the convergence time for DM-4 on California housing data set is shorter as it reaches the local minima before it executes total number of iterations, N_{it}. This is because, if the learning factor is very small and the MSE does not decrease for more than five iterations, then a local minimal is reached and the training is stopped for that network.

6. Conclusions

In this paper, we explore four design methodologies for producing sequences of trained and validated feed-forward networks of different size. In the growing

approach (DM-1), DI networks result in monotonically decreasing $E_f(N_h)$ curves. A pruning method (DM-2) is shown that requires one pass through the data. A method is also described for simultaneously validating many different size networks, using a second data pass. In the third, combined approach (DM-3), ordered pruning is applied to grown networks. Lastly, our final combined approach (DM-4) successively grows the network by a few units, and then prunes. As seen in the simulations, this approach usually produces smaller training and validation errors than the other methodologies. The final MSE versus N_h curves are almost monotonic. The DM-4 approach also exhibits reduced sensitivity to initial weights.

Our final method was compared with the two other well known growing techniques: CBP and CC. On seven different data sets, the results show that our proposed method performs significantly better than these other methods.

Appendix A. Hidden unit pruning using the Schmidt procedure

For the ease of notation, in this section, we use the augmented input vector, i.e., $\mathbf{x}_p \leftarrow [\mathbf{x}_p^T, 1, \mathbf{O}_p^T]^T$, where 1 is for the threshold. The new dimension of $\mathbf{x}_p$ is $N_u = N + N_h + 1$. Also, for simplicity, the subscript p indicating the pattern number will not be used unless it is necessary. The output of the network in Eq. (3), can be rewritten as

$$y_i = \sum_{k=1}^{N_u} w_0(i,k) \cdot x_k, \tag{A.1}$$

where w_0 is the corresponding augmented weight matrix consisting of w_{oi} and w_{oh} including the threshold. In Eq. (A.1), the signals x_k are the raw basis functions for producing y_i.

The normal Gram–Schmidt procedure [32] 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.

A.1. Basic algorithm

The mth orthonormal basis function x'_m, can be expressed as

$$x'_m = \sum_{k=1}^{m} a_{mk} \cdot x_k. \tag{A.2}$$

From Eq. (A.2), for $m = 1$, the first basis function is obtained as

$$x'_1 = \sum_{k=1}^{1} a_{1k} \cdot x_k = a_{11} x_1, \tag{A.3}$$

$$a_{11} = \frac{1}{\|x_1\|} = \frac{1}{r(1,1)^{1/2}}, \tag{A.4}$$

where elements of the autocorrelation matrix $\mathbf{R}$ are defined as

$$r(i,j) = \langle x_i, x_j \rangle = \frac{1}{N_v} \sum_{p=1}^{N_v} x_{pi} \cdot x_{pj}. \tag{A.5}$$

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

$$c_i = \sum_{q=1}^{i} a_{iq} \cdot r(q,m). \tag{A.6}$$

Then obtain m coefficients b_k as

$$b_k = \begin{cases} -\sum_{i=k}^{m-1} c_i \cdot a_{ik}, & 1 \leqslant k \leqslant m-1, \\ 1, & k = m. \end{cases} \tag{A.7}$$

Finally, for the mth basis function the new a_{mk} coefficients (for $1 \leqslant k \leqslant m$) are found as

$$a_{mk} = \frac{b_k}{\left[r(m,m) - \sum_{i=1}^{m-1} c_i^2 \right]^{1/2}}. \tag{A.8}$$

The output equation (A.1) can be written as

$$y_i = \sum_{q=1}^{N_u} w'_0(i,q) \cdot x'_q, \tag{A.9}$$

where the weights in the orthonormal system are

$$w'_0(i,q) = \sum_{k=1}^{q} a_{qk} \cdot \langle x_k, t_i \rangle = \sum_{k=1}^{q} a_{qk} \cdot c(i,k), \tag{A.10}$$

where elements of the crosscorrelation matrix $\mathbf{C}$ are defined as

$$c(i,k) = \frac{1}{N_v} \sum_{p=1}^{N_v} t_{pi} \cdot x_{pk}. \tag{A.11}$$

Using Eq. (A.2), we obtain output weights for the system as

$$w_0(i,k) = \sum_{q=k}^{N_u} w'_0(i,q) \cdot a_{qk}. \tag{A.12}$$

Substituting Eq. (A.9) into Eq. (5), we obtain $E(i)$ in orthonormal system as

$$E(i) = \left\langle \left(t_i - \sum_{k=1}^{N_u} w'_0(i,k) \cdot x'_k \right), \left(t_i - \sum_{q=1}^{N_u} w'_0(i,q) \cdot x'_q \right) \right\rangle. \tag{A.13}$$

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

$$E(i) = \langle t_i, t_i \rangle - \sum_{k=1}^{N+1+N_{hd}} (w'_0(i,k))^2. \tag{A.14}$$

Modifying Eq. (A.12), the output weights would be

$$w_0(i,k) = \sum_{q=k}^{N+1+N_{hd}} w'_o(i,q) \cdot a_{qk}. \quad (A.15)$$

The purpose of pruning is to eliminate useless inputs and hidden units as well as hidden units that are less useful. Useless units are those which (1) have no information relevant for estimating outputs or (2) are linearly dependent on inputs or hidden units that have already been orthonormalized.

We modify the Schmidt procedure so that during pruning, the hidden units are ordered according to their usefulness and 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 $1 \leqslant m \leqslant N_u$ and $1 \leqslant j(m) \leqslant N_u$. 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

$$x'_m = \sum_{k=1}^{m} a_{mk} \cdot x_{j(k)}. \quad (A.16)$$

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

$$a_{11} = \frac{1}{\|x_{j(1)}\|} = \frac{1}{r(j(1),j(1))^{1/2}}. \quad (A.17)$$

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

$$c_i = \sum_{q=1}^{i} a_{iq} \cdot r(j(q),j(m)) \quad \text{for } 1 \leqslant i \leqslant m-1. \quad (A.18)$$

Second, we set $b_m = 1$ and get

$$b_k = -\sum_{i=k}^{m-1} c_i \cdot a_{ik} \quad \text{for } 1 \leqslant k \leqslant m-1. \quad (A.19)$$

Lastly, we get coefficients a_{mk} as

$$a_{mk} = \frac{b_k}{\left[r(j(m),j(m)) - \sum_{i=1}^{m-1} c_i^2\right]^{1/2}} \quad \text{for } 1 \leqslant k \leqslant m. \quad (A.20)$$

Then weights in the orthonormal system are found as

$$w'_o(i,m) = \sum_{k=1}^{m} a_{mk} \cdot c(i,j(k)) \quad \text{for } 1 \leqslant i \leqslant M. \quad (A.21)$$

The goal of 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 Eq. (A.20) 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

$$j(m) = m \quad \text{for } 1 \leqslant m \leqslant N+1. \quad (A.22)$$

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

$$S(m) = \begin{cases} \{\phi\} & \text{for } m = 0, \\ \{j(1), j(2), \ldots, j(m)\} & \text{for } 0 < m \leqslant N_u. \end{cases} \quad (A.23)$$

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 $S^c\{m\} = \{1,2,3,\ldots,N_u\} - S(m)$, which is $\{N+2, N+3, \ldots, N_u\}$. For $N+2 \leqslant m \leqslant N_u$, we obtain $S^c\{m-1\}$. For each trial value of $j(m) \in S^c\{m-1\}$, we perform operations in Eqs. (A.18)–(A.21). Then $P(m)$ is

$$P(m) = \sum_{i=1}^{M} [w'_o(i,m)]^2. \quad (A.24)$$

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

$$S(m) = S(m-1) \cup \{j(m)\}. \quad (A.25)$$

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 Eq. (A.24), after validation all the candidate basis function, $j(m)$ takes its value and $S(m)$ is updated according to Eq. (A.25). 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 Eq. (A.30). 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 Eq. (A.1). After reordering the rows, because only the N_{hd} units are kept then the remaining units (O_{pj} with $N_{hd} < j \leqslant N_h$) are pruned by deleting the last $N_h - N_{hd}$ rows.

A.2. 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 ($1 \leqslant m \leqslant N$) and $j(m)$ has taken on a trial value. Then

$$x_{j(m)} = \sum_{k=1}^{m-1} d_k \cdot x'_k. \quad (A.26)$$

Now the denominator of a_{mk} in (A.20) can be rewritten as

$$g = \langle z_m, z_m \rangle^{1/2}, \tag{A.27}$$

where

$$z_m = x_{j(m)} - \sum_{i=1}^{m-1} \langle x_i', x_{j(m)} \rangle \cdot x_i'. \tag{A.28}$$

Substituting (A.26) into (A.28), however, we get

$$\langle x_i', x_{j(m)} \rangle = d_i \tag{A.29}$$

and z_m and g are both zero.

If $j(m)$ denotes an input and g is infinitesimally small, then we equate $j(k)$ to k for $1 \leqslant k < m$ and $j(k) = k + 1$ for $m \leqslant k \leqslant N$. 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.

Once we get these orthonormal weights, the hidden units and their weights are reordered in the descending order of their energies. Then the output weights of the original system are obtained using the equation

$$w_o^{\mathrm{DM2}}(i,k) = \sum_{q=k}^{N_u} w_o'(i,q) \cdot a_{qk}. \tag{A.30}$$

Here $w_o^{\mathrm{DM2}} \in \mathscr{W}_{\mathrm{DM2}}^{N_h}$ is the output weights obtained using Schmidt procedure.

References

[1] H. Amin, K.M. Curtis, B.R.H. Gill, Dynamically pruning output weights in an expanding multilayer perceptron neural network, in: 13th International Conference on DSP, vol. 2, 1997.

[2] P. Arena, R. Caponetto, L. Fortuna, M.G. Xibilia, Genetic algorithms to select optimal neural network topology, in: Proceedings of the 35th Midwest Symposium on Circuits and Systems, vol. 2, 1992.

[3] C.M. Bishop, Neural Networks and Machine Learning, vol. 168, Springer, Berlin, 1998.

[4] G. Castellano, A.M. Fanelli, M. Pelillo, An iterative pruning algorithm for feedforward neural networks, IEEE Trans. Neural Networks 8 (3) (1997) 519–531.

[5] H.H. Chen, M.T. Manry, H. Chandrasekaran, A neural network training algorithm utilizing multiple sets of linear equations, Neurocomputing 25 (1–3) (1999) 55–72.

[6] S. Chen, C.F.N. Cowan, P.M. Grant, Orthogonal least squares learning algorithm for radial basis function networks, IEEE Trans. Neural Networks 2 (2) (1991) 302–309.

[7] F.L. Chung, T. Lee, Network-growth approach to design of feedforward neural networks, IEE Proce. Control Theory 142 (5) (1995) 486–492.

[8] J. Davila, Genetic optimization of NN topologies for the task of natural language processing, in: Proceedings of International Joint Conference on Neural Networks, vol. 2, 1999.

[9] M.S. Dawson, A.K. Fung, M.T. Manry, Surface parameter retrival using fast learning neural networks, Remote Sensing Rev 7 (1993).

[10] M.S. Dawson, J. Olvera, A.K. Fung, M.T. Manry, Inversion of surface parameters using fast learning neural networks, in: Proceedings of IGARSS, vol. 2, 1992.

[11] W.H. Delashmit, Multilayer perceptron structured initialization and separating mean processing, Dissertation, University of Texas, Arlington, December 2003.

[12] S.E. Fahlman, C. Lebiere, The cascade-correlation learning architecture, in: Advances in Neural Information Processing Systems 2, Morgan Kaufmann, Los Altos, CA, 1990.

[13] M.H. Fun, M.T. Hagan, Levenberg-marquardt training for modular networks, in: IEEE International Conference on Neural Networks, vol. 1, 1996.

[14] A.K. Fung, Z. Li, K.S. Chen, Back scattering from a randomly rough dielectric surface, IEEE Trans. Geosci. Remote Sensing 30 (2) (1992) 356–369.

[15] S.-U. Guan, S. Li, Parallel growing and training of neural networks using output parallelism, IEEE Trans. Neural Networks 13 (3) (2002) 542–550.

[16] Y. Hirose, K. Yamashita, S. Hijiya, Back-propagation algorithm that varies the number of hidden units, Neural Networks 4 (1991).

[17] G.B. Huang, P. Saratchandran, N. Sundararajan, A generalized growing and pruning RBF neural network for function approximation, IEEE Trans. Neural Networks 16 (1) (2005) 57–67.

[18] W. Kaminski, P. Strumillo, Kernel orthonormalization in radial basis function neural networks, IEEE Trans. Neural Networks 8 (5) (1997) 1177–1183.

[19] M. Lehtokangas, Modelling with constructive backpropagation, Neural Networks 12 (1999) 707–716.

[20] X. Liang, L. Ma, A study of removing hidden neurons in cascade correlation neural networks, in: IEEE International Joint Conference on Neural Networks, vol. 2, 2004.

[21] F.J. Maldonado, T.H.K.M.T. Manry, Finding optimal neural network basis function subsets using the schmidt procedure, in: Proceedings of the International Joint Conference on Neural Networks, vol. 1, 2003, pp. 20–24.

[22] V. Maniezzo, Genetic evolution of the topology and weight distribution of neural networks, IEEE Trans. Neural Networks 5 (1) (1994).

[23] M.T. Manry, S.J. Apollo, L.S. Allen, W.D. Lyle, W. Gong, M.S. Dawson, A.K. Fung, Fast training of neural networks for remote sensing, Remote Sensing Rev. 9 (1994).

[24] P.V.S. Ponnapalli, K.C. Ho, M. Thomson, A formal selection and pruning algorithm for feedforward artificial neural network optimization, IEEE Trans. Neural Networks 10 (4) (1999) 964–968.

[25] R. Reed, Pruning algorithms—a survey, IEEE Trans. Neural Networks 4 (5) (1993) 740–747.

[26] D.E. Rumelhart, J.L. McClelland, Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. 1, MIT Press, Cambridge, MA, 1986.

[27] I.I. Sakhnini, M.T. Manry, H. Chandrasekaran, Iterative improvement of trigonometric networks, in: Proceedings of the International Joint Conference on Neural Networks, vol. 2, 1999.

[28] Y.K. Sarabandi, F.T. Ulaby, An empirical model and an inversion technique for radar scattering from bare soil surfaces, IEEE Trans. Geosci. Remote Sensing 30 (2) (1992) 370–381.

[29] M.A. Sartori, P.J. Antsaklis, A simple method to derive bounds on the size and to train multilayer neural networks, IEEE Trans. Neural Networks 2 (4) (1991) 467–471.

[30] R.S. Scalero, N. Tepedelenlioglu, A fast new algorithm for training feedforward neural networks, IEEE Trans. Signal Process. 40 (1992) 202–210.

[31] R. Sentiono, Feedforward neural network construction using cross validation, Neural Computation 13 (2001) 2865–2877.

[32] G. Strang, Linear Algebra and its Application, Harcourt Brace, New York, 1988.

[33] University of Texas at Arlington, Training datasets, ⟨http://www-ee.uta.edu/eeweb/IP/training_data_files.htm⟩.

[34] C. Yu, M.T. Manry, J. Li, P.L. Narasimha, An efficient hidden layer training method for the multilayer perceptron, Neurocomputing 70 (2006) 525–535.

Pramod Lakshmi Narasimha received his B.E. in Telecommunications Engineering from Bangalore University, India. He joined the Neural Networks and Image Processing Lab in the Department of Electrical Engineering at the University of Texas at Arlington (UTA) as a Research Assistant in 2002. In 2003, he received his M.S. degree in EE at UTA. Currently, he is working on his Ph.D. His research interests focus on machine learning, neural networks, estimation theory, pattern recognition and computer vision. He is a member of Tau Beta Pi and a student member of the IEEE.

Walter H. Delashmit was born in Memphis, Tennessee, in 1944. He received the B.S. degree from Christian Brothers University in Electrical Engineering in 1966 and the M.S. degree from the University of Tennessee in Electrical Engineering with a minor in Mathematics in 1968. He graduated with the Ph.D. degree in Electrical Engineering at the University of Texas at Arlington, Spring 2003. Currently, he is employed at Lockheed Martin Missiles and Fire Control (formerly Loral Vought Systems and LTV Aerospace and Defense) in Dallas, Texas, where he is a Senior Staff Research Engineer. Among his duties, he is Program Manager/Principal Investigator for the Weapon Seeker Improvement Program and Manager of the Signal and Image Processing Group. He is a Senior Member of the IEEE, a member of Tau Beta Pi and a member of Eta Kappa Nu. His research interests are in the areas of neural networks, two- and three-dimensional image processing and signal processing techniques. Dr. Delashmit is also an avid jogger.

Michael T. Manry was born in Houston, Texas, in 1949. He received the B.S., M.S., and Ph.D. in Electrical Engineering in 1971, 1973, and 1976, respectively, from The University of Texas at Austin. After working there for two years as an Assistant Professor, he joined Schlumberger Well Services in Houston where he developed signal processing algorithms for magnetic resonance well logging and sonic well logging. He joined the Department of Electrical Engineering at the University of Texas at Arlington in 1982, and has held the rank of Professor since 1993. In summer 1989, Dr. Manry developed neural networks for the Image Processing Laboratory of Texas Instruments in Dallas. His recent work, sponsored by the Advanced Technology Program of the state of Texas, E-Systems, Mobil Research, and NASA, has involved the development of techniques for the analysis and fast design of neural networks for image processing, parameter estimation, and pattern classification. Dr. Manry has served as a consultant for the Office of Missile Electronic Warfare at White Sands Missile Range, MICOM (Missile Command) at Redstone Arsenal, NSF, Texas Instruments, Geophysics International, Halliburton Logging Services, Mobil Research and Verity Instruments. He is a Senior Member of the IEEE.

Jiang Li received his B.S. degree in Electrical Engineering from Shanghai Jiaotong University, China, in 1992, the M.S. degree in automation from Tsinghua University, China, in 2000, and the Ph.D. degree in Electrical Engineering from the University of Texas at Arlington, TX, in 2004. His research interests include machine learning, computer aided medical diagnosis systems, medical signal/image processing, neural network and modeling and simulation. Dr. Li worked as a postdoctoral fellow at the Department of Radiology, National Institutes of Health, from 2004 to 2006. He joined ECE Department at ODU as an Assistant Professor in 2006. He is also affiliated with Virginia Modeling, Analysis, and Simulation Center (VMASC). Dr. Li is a member of the IEEE and Sigma Xi.

Francisco Javier Maldonado Diaz has received the Masters in Electrical Engineering from the Chihuahua Institute of Technology, and the Ph.D. in Electrical Engineering from the University of Texas at Arlington (UTA). He was the recipient of the Fulbright sponsorship for doctoral studies at UTA. He has worked on Intelligent Systems with Dr. Frank Lewis (Advanced Controls, Sensors and MEMS group) at the Automation and Robotics Research Institute of UTA. Dr. Maldonado has also developed advanced learning algorithms with Dr. Michael Manry in the Image Processing and Neural Networks Lab at UTA. In 2002, Dr. Maldonado joined Williams-Pyro's research and development department and has been involved in several SBIR projects for the US government. Dr. Maldonado's research has centered on using neural networks for (1) failure detection, (2) implementation in embedded systems, and (3) pseudogenetic algorithms for multi-layer perceptron design. He is a reviewer for the Neurocomputing International Journal and has published eight conference papers on neural networks and intelligent systems. He is a member of the IEEE Control and Instrumentation Systems societies and has been named a Marquis Who's Who in the World, 2006 edition. He has been a member of the Society of Hispanic Engineers at UTA, being an official of the society during 2000. He is also a member of HKN and was an International University Scholar and University Scholar at UTA in 2001.