Performance Evaluation of a Sequential Minimal Radial Basis Function (RBF) Neural Network Learning Algorithm

Lu Yingwei, Narashiman Sundararajan, *Fellow, IEEE*, and P. Saratchandran, *Senior Member, IEEE*

***Abstract*—This paper presents a detailed performance analysis of the recently developed minimal resource allocation network (M-RAN) learning algorithm. M-RAN is a sequential learning radial basis function neural network which combines the growth criterion of the resource allocating network (RAN) of Platt with a pruning strategy based on the relative contribution of each hidden unit to the overall network output. The resulting network leads toward a minimal topology for the RAN. The performance of this algorithm is compared with the multilayer feedforward networks (MFN's) trained with 1) a variant of the standard back-propagation algorithm, known as RPROP and 2) the dependence identification (DI) algorithm of Moody and Antsaklis on several benchmark problems in the function approximation and pattern classification areas. For all these problems, the M-RAN algorithm is shown to realize networks with far fewer hidden neurons with better or same approximation/classification accuracy. Further, the time taken for learning (training) is also considerably shorter as M-RAN does not require repeated presentation of the training data.**

***Index Terms*—Growth criterion, pruning strategy, RBF neural network, sequential learning.**

I. Introduction

IT is well known that multilayer feedforward network (MFN) is the most widely used neural network model for pattern classification applications [1], [2]. This is because the topology of the MFN allows it to generate internal representations tailored to classify the input regions that may be either disjointed or intersecting [3]. The hidden layer nodes in the MFN can form hyperplanes to partition the input space into various regions and the output nodes can select and combine the regions that belong to the same class. Backpropagation (BP) [4] and its variants such as Quickprop [5] and RPROP [6], are the most widely used training algorithm for the MFN networks.

Recently researchers have begun to examine the use of radial basis functions (RBF's) [7] for solving function approximation and pattern classification problems. RBF's are well suited for these problems due to their simple topological structure and their ability to reveal how learning proceeds in an explicit manner [8]. In the classical approach to RBF network implementation, the basis functions are usually chosen as Gaussian and the number of hidden units is fixed *a priori* based on some properties of the input data. The weights connecting the hidden and output units are estimated by linear least squares methods, e.g., *least mean square* (LMS) [9], [10], *recursive least squares* (RLS) [11]. The disadvantage with the classical approach is that it is not suitable for sequential learning and it also results, usually, in too many hidden units [12].

A significant contribution to overcome these drawbacks was made by Platt [13] through the development of an algorithm that adds hidden units to the network based on the "novelty" of the input data. The algorithm is suitable for sequential learning and is based on the idea that the number of hidden units should correspond to the complexity of the underlying function as reflected in the observed data. The resulting network, called a resource-allocating network, starts with no hidden units and grows by allocating new hidden units based on the novelty in the observations which arrive sequentially. If an observation has no novelty then the existing parameters of the network are adjusted by an LMS algorithm to fit that observation. Kadirkamanathan *et al*. [14] interpreted Platt's RAN from a function space approach and improved RAN by using an extended Kalaman filter (EKF) instead of the LMS to estimate the network parameters. Their network called RANEKF is more compact and has better accuracy.

A drawback of RAN and RANEKF is that once a hidden unit is created it can never be removed. Because of this RAN could produce networks in which some hidden units, although active initially, may subsequently end up contributing little to the network output. If such inactive hidden units can be detected and removed as learning progresses then a more parsimonious network topology can be realized. This has been achieved in the minimal RAN (M-RAN) algorithm which has been recently developed by the authors [15]. The M-RAN uses a pruning strategy to remove those hidden units which consistently make little contribution to the network output and together with an additional growth criterion ensures that the transition in the number of hidden neurons is smooth. The M-RAN algorithm is described in detail and its superiority in achieving a more compact network compared to both RAN and the enhanced RANEKF is given in [15]–[17].

The main objective of this paper is to present a comprehensive comparison of the performance of M-RAN with MFN's on established benchmark problems in the function approximation and pattern classification areas. The study is intended to compare the complexity of the resulting networks,

Manuscript received September 16, 1996; revised November 25, 1997.

The authors are with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798.

Publisher Item Identifier S 1045-9227(98)01811-6.

accuracy of approximation and classification, and the number of epochs required for training. For this study, the training algorithms considered for the MFN's are the RPROP algorithm [6] and the recently proposed dependence identification (DI) algorithm of Moody and Antsaklis [18], [19]. RPROP is a fast BP variant similar in spirit to Quickprop [5]. It is shown in [6] to be as fast as Quickprop and it requires less adjustment of the parameters to be stable.

The DI algorithm constructs an appropriate MFN to meet the training specifications of a problem by transforming the problem into a set of quadratic optimization problems which are then solved by a succession of linear equations. The main objective of DI is to come up with an algorithmic technique which can quickly determine the MFN structure for the given problem. The overall speed of the DI algorithm is shown in [18] to be orders of magnitude greater than standard BP although the resulting network is usually large. The DI algorithm is a batch learning process unlike M-RAN algorithm which is a sequential (on-line) learning process.

For comparison of M-RAN with RPROP, the benchmark problems used are **hearta**, **flare**, and **cancer** from the PROBEN1 database, which has been recently set up for neural network benchmarking [20]. PROBEN1 is a collection of benchmark problems for neural network learning in the realm of pattern classification and function approximation.[1] The rules and conventions for carrying out tests with the benchmark problems are also provided in PROBEN1. It also gives basic performance measures and results for MFN's which can be used for comparisons.

For comparison of M-RAN with DI, we have used the problems presented in [18] and also a test case from PROBEN1—**hearta1**. The latter benchmark problem has been selected to assess the performance of the two algorithms for a higher dimensional problem.

The paper is organized as follows. Section II gives a brief description of the M-RAN algorithm. Section III gives details of comparison between M-RAN and MFN's trained with the RPROP. Section IV presents the results of performance comparison of M-RAN with DI algorithm. Section V summarizes the conclusions from this study.

II. Sequential Learning Using the M-RAN Algorithm

M-RAN uses the basic idea of Platt's RAN which is a sequential learning Gaussian RBF network. Platt's motivation for RAN stemmed from the fact that learning with a fixed-size network is a NP-complete problem and by allocating new resources, learning could be achieved in polynomial time. The structure of RAN is the same as that of RBF networks (see Fig. 1). Each hidden unit in the network has two parameters called a center (μ), and a width (σ) associated with it. The activation function of the hidden units is radially symmetric in the input space and the output of each hidden unit depends only on the radial distance between the input vector $\mathbf{x}$ and the center parameter μ for that hidden unit. The response of each hidden unit is scaled by its connecting weights $(\alpha$'s$)$ to the output units and then summed to produce the overall network output. The overall network output is therefore

$$f(\mathbf{x}) = \alpha_0 + \sum_{k=1}^{K} \alpha_k \phi_k(\mathbf{x}) \tag{1}$$

and

$$\phi_k(\mathbf{x}) = \exp\left(-\frac{1}{\sigma_k^2}||\mathbf{x} - \mu_\mathbf{k}||^2\right) \tag{2}$$

where $\phi_k(\mathbf{x})$ is the response of the kth hidden unit and $\alpha_\mathbf{k}$ is its connecting weight to the output unit. $\mu_\mathbf{k}$ and σ_k are, respectively, the center and width of the kth hidden neuron and α_0 is the bias term.

The learning process of RAN involves allocation of new hidden units as well as adaptation of network parameters. The network begins with no hidden units. As input–output data are received during training, some of them are used for generating new hidden units. The decision as to whether an input–output $(\mathbf{x_n}, y_n)$ data should give rise to a new hidden unit depends on the novelty in the data which is decided using the following two conditions:

$$||\mathbf{x_n} - \mu_\mathbf{nr}|| > \epsilon_n \tag{3}$$

$$|e_n| = |y_n - f(\mathbf{x_n})| > e_{\min} \tag{4}$$

where $\mu_\mathbf{nr}$ is the center which is nearest to $\mathbf{x_n}$. ϵ_n and $e_{\min}$ are thresholds to be selected appropriately. If the above two conditions are satisfied then the data is deemed to have novelty and a new hidden unit is added. The first condition says that the input must be far away from all the centers and the second condition says that the error between the network output and the target output must be significant. The value of $e_{\min}$ represents the desired approximation accuracy of the network output and the distance ϵ_n represents the scale of resolution in the input space. The algorithm begins with $\epsilon_n = \epsilon_{\max}$. $\epsilon_{\max}$ is chosen as the largest scale of interest in the input space,typically the entire input space of nonzero probability. The distance ϵ_n is decayed exponentially as $\epsilon_n = \max\{\epsilon_{\max}\gamma^n, \epsilon_{\min}\}$, where $0 < \gamma < 1$ is a decay constant. The value for ϵ_n is decayed until it reaches $\epsilon_{\min}$. The exponential decaying of the distance criterion allows fewer basis functions with large widths (smoother basis functions) initially and with increasing number of observations more basis functions with smaller widths are allocated to fine tune the approximation. The parameters associated with the new hidden unit are as follows:

$$\alpha_{K+1} = e_n \tag{5}$$

$$\mu_\mathbf{K+1} = \mathbf{x_n} \tag{6}$$

$$\sigma_{K+1} = \kappa||\mathbf{x_n} - \mu_\mathbf{nr}|| \tag{7}$$

where κ is an overlap factor that determines the amount of overlap of the responses of the hidden units in the input space.

When an observation $(\mathbf{x_n}, y_n)$ does not pass the novelty criteria, a hidden unit is not added but the network parameters μ's, and α's are adapted to fit that observation. Platt used the LMS algorithm for adapting the μ's and α's and did not include the width parameters in the adaptation. An enhancement

[1]The benchmark datasets are available from http://wwwipd.ira.uka.de/~prechelt/NIPS_bench.html.

to RAN was proposed by Kadirkamanathan and Niranjan [14] who used an EKF instead of the LMS method for adjusting the network parameters which in their case included the width parameters as well. The enhanced RAN, called RANEKF in the literature [14], [21] has been shown [14] to produce more parsimonious networks than the basic RAN. However, like RAN, RANEKF also suffers from its inability to remove a hidden unit once it is created and consequently could produce networks in which some hidden units end up contributing little to the network output. An example that clearly illustrates this is given in [15]. If the inactive hidden units can be detected and removed as learning progresses then a more compact network topology can be realized. This is achieved in the M-RAN by incorporating a pruning strategy and augmenting the basic growth criterion of RAN.

The pruning strategy removes those hidden units which make insignificant contribution to the overall network output consecutively over a number of training observations. To carry out this, at each observation n, the outputs of the hidden units are first normalized with respect to the maximum output value $(|o^n_{\max}|)$ among all the hidden neurons. These normalized values are then compared with a threshold δ and if any of them falls below this threshold for M consecutive observations then this particular hidden neuron, for example the kth hidden neuron, is removed from the network. This pruning strategy is indicated as follows:

For every observation $(\mathbf{x_n}, y_n)$,
  compute the hidden unit outputs $o^n_k, (k = 1, \cdots, K)$

$$o^n_k = \alpha_k \exp\left(-\frac{1}{\sigma_k^2}\|\mathbf{x_n} - \mu_\mathbf{k}\|^2\right) \tag{8}$$

  find the largest absolute hidden unit output value $|o^n_{\max}|$
  compute the normalized output values $r^n_k, (k = 1, \cdots, K)$

$$r^n_k = \left|\frac{o^n_k}{o^n_{\max}}\right| \tag{9}$$

  if $r^n_k < \delta$ for M consecutive observations, then
    remove the kth hidden unit
    reduce the dimensionality of EKF
  end if.

The growth criterion in M-RAN augments the basic novelty criteria [(3) and (4)] of RAN with an additional condition based on the RMS value of the output error over a sliding data window. This condition checks if the RMS value of the output error (e_{rms}) over a sliding window (M) is greater than a threshold. It is given by

$$e_{\text{rms}} = \sqrt{\frac{\sum_{i=n-(M-1)}^{n} e_i^2}{M}} > e'_{\min} \tag{10}$$

where e_i is the network output error at ith instant and it is expressed as $e_i = y_i - f(\mathbf{x_i})$ and $e'_{\min}$ is the threshold selected for this criterion. This additional condition was introduced to ensure the transition in the number of hidden units due to growing and pruning is smooth. Examples that illustrate the need for the additional growth condition can be found in [15]. The final M-RAN algorithm is summarized below.

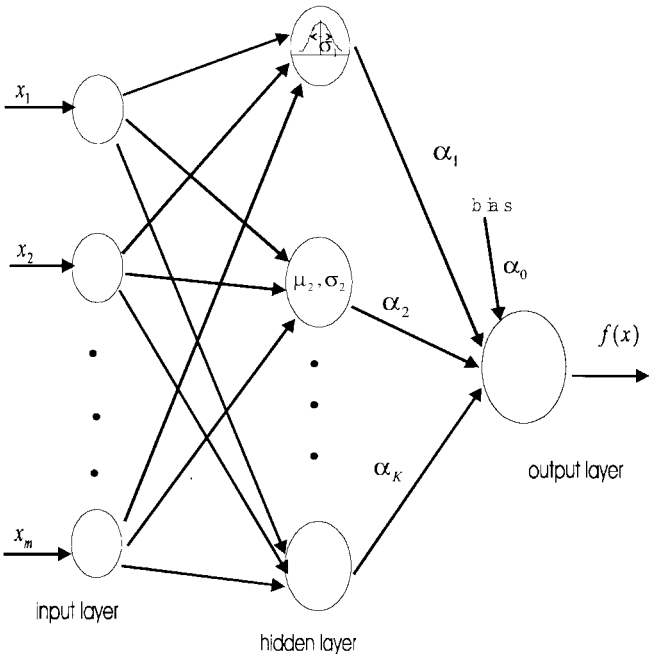


Fig. 1. The structure of RBF neural network.

A. The M-RAN Algorithm

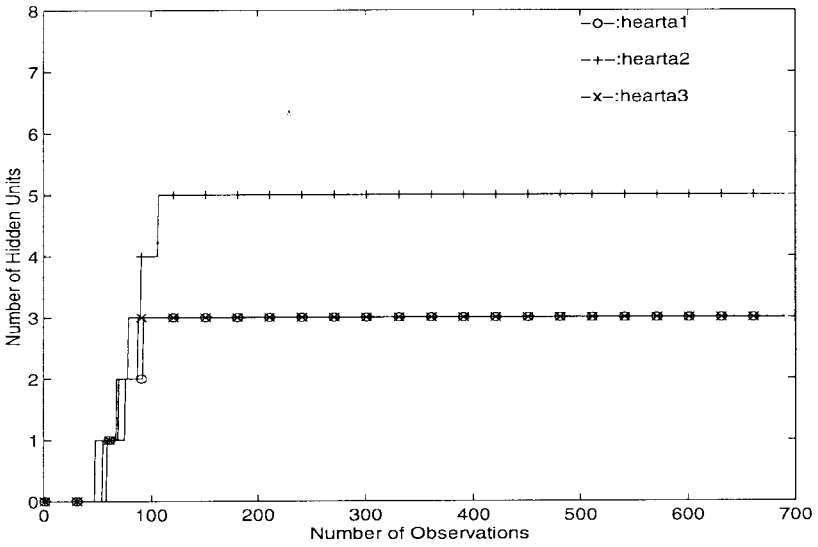
For each observation $(\mathbf{x_n}, y_n)$ do
  compute overall network output:

$$f(\mathbf{x_n}) = \alpha_0 + \sum_{k=1}^{K} \alpha_k \exp\left(-\frac{1}{\sigma_k^2}\|\mathbf{x_n} - \mu_\mathbf{k}\|^2\right)$$

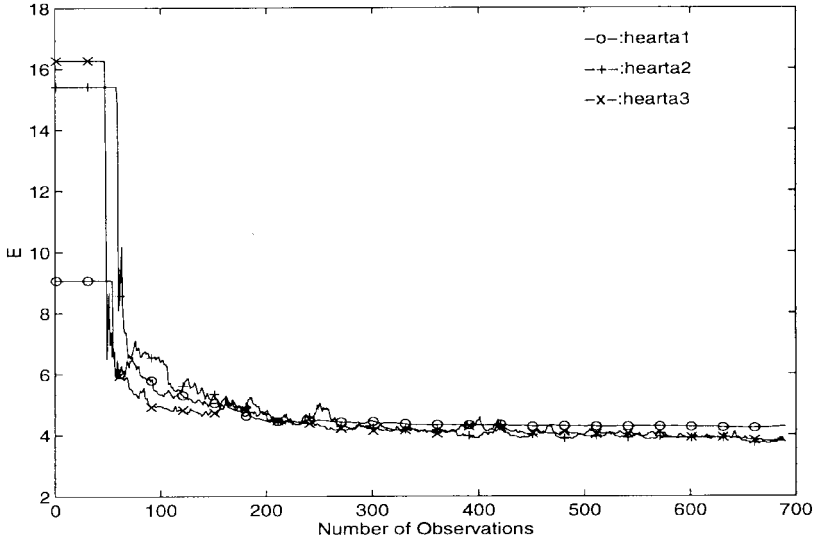
  K is the number of hidden units.
  Calculate the parameters required in the growth criterion:

$$\epsilon_n = \max\{\epsilon_{\max}\gamma^n, \epsilon_{\min}\}, \quad (0 < \gamma < 1)$$
$$|e_n| = |y_n - f(\mathbf{x_n})|$$
$$e_{\text{rms}} = \sqrt{\frac{\sum_{i=n-(M-1)}^{n} e_i^2}{M}}$$

  Apply the criterion for adding hidden units:
    if $(|e_n| > e_{\min})$ and
      $(\|\mathbf{x_n} - \mu_\mathbf{nr}\|) > \epsilon_n)$ and
      $(e_{\text{rms}} > e'_{\min})$ then
      allocate a new hidden unit $K+1$ with:
      $\alpha_{K+1} = e_n$
      $\mu_{\mathbf{K+1}} = \mathbf{x_n}$
      $\sigma_{K+1} = \kappa\|\mathbf{x_n} - \mu_\mathbf{nr}\|$
    else
      adjust the network parameters α_0 and α_k
      $\mu_\mathbf{k}$, σ_k, $(k = 1, \cdots, K)$ using EKF
    end if
  Check the criterion for pruning hidden units:
    compute the hidden unit outputs o^n_k
    $(k = 1, \cdots, K)$ according to (8)
    find the largest absolute hidden unit output value $|o^n_{\max}|$; compute the normalized output values r^n_k, $(k = 1, \cdots, K)$ according to (9)
    if $r^n_k < \delta$ for M consecutive observations, then



(a)



(b)

Fig. 2. Performance of M-RAN for hearta1, hearta2, hearta3. (a) Growth pattern. (b) Approximation error on testset as learning progresses for M-RAN.

remove the kth hidden unit
reduce the dimensionality of EKF
end if
End for.

III. Performance of M-RAN with RPROP

In this section, the performance of the M-RAN is given for two function approximation and one pattern classification problems from PROBEN1 [20].

There are 15 data sets from different domains in PROBEN1. All these datasets represent realistic problems. The datasets are all presented in the same simple format, using an attribute representation that can directly be used for neural network training. Along with the datasets, PROBEN1 defines a set of rules on how to conduct and document neural network benchmarking, thus giving neural network researchers an easy access to data for evaluation of their algorithms and networks and makes direct comparison.

PROBEN1 also provide a standard error measure which is called **squared error percentage** E, where

$$E = 100 * \frac{o_{\max} - o_{\min}}{N * P} \sum_{p=1}^{P} \sum_{i=1}^{N} (o_{pi} - t_{pi})^2. \qquad (11)$$

In the above equation $o_{\max}$ and $o_{\min}$ are the maximum and minimum values of output coefficients, N is the number of output units of the network and P is the number of examples in the test data used. o_{pi} and t_{pi} are the actual and target output values for the ith node for pth example. We have used (11) as the metric for comparing MRAN with MFN's because the results for MFN's are taken from PROBEN1 and we want to compare our results using the same metric.

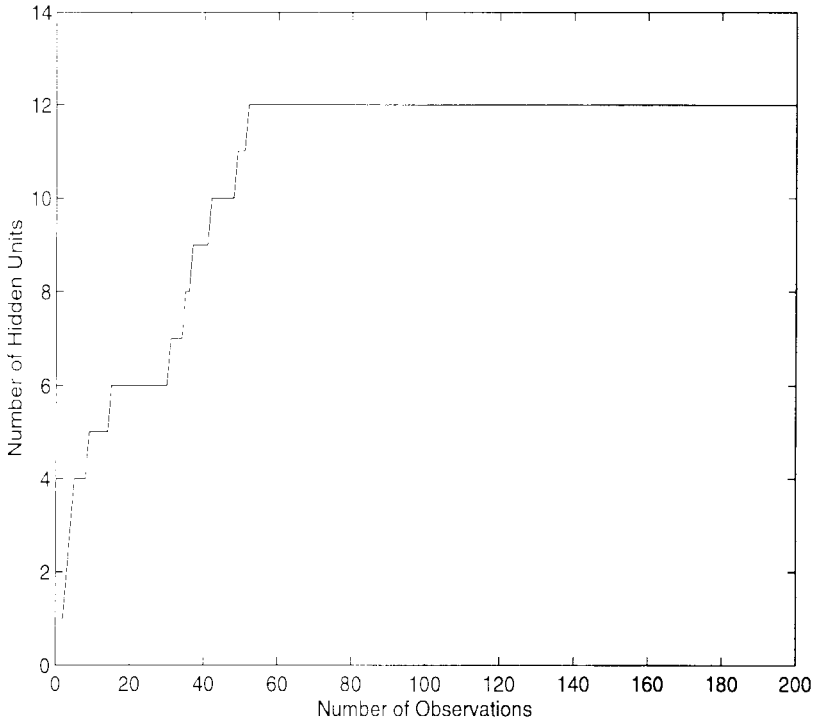


Fig. 3. Growth pattern of two input–two output sinusoidal function for M-RAN neural network.

A. Function Approximation Problems

The function approximation problems selected for this study are the **hearta** and **flare** datasets which are created based on the "heart disease" and "solar flare" problem datasets, respectively, from the UCI repository of machine learning databases. For a detailed description of these problems, refer to [20].

1) Hearta-Problem: The **hearta** dataset consists of 920 examples out of which 690 are to be used for training and the rest 230 to be used for testing. Each example is described by 13 input attributes and one output attribute. Among the training examples only 299 are complete and the rest have one or more missing values. The M-RAN network used 13 input units and a single continuous output unit to represent the five levels of the output attribute. As per PROBEN1 guidelines, missing values were replaced with the mean of the nonmissing values for that attribute. In PROBEN1, the 13 input attributes have been coded into 35 input units for the pivot MFN architectures.[2]

The parameters for the M-RAN algorithm are chosen as: $\epsilon_{\max} = 4.0$, $\epsilon_{\min} = 1.5$, $\gamma = 0.99$, $e_{\min} = 0.15$, $e'_{\min} = 0.3$, $\kappa = 0.4$. $P_0 = 1.0$, $R_n = 1.0$, $Q_0 = 0.000\,01$, $M = 60$, $\delta = 0.000\,001$. The performance of M-RAN was evaluated on the three different permutations of the **hearta** dataset, viz. **hearta1, hearta2, hearta3**, given in PROBEN1. Fig. 2(a) shows the growth pattern for M-RAN as it learns sequentially from the training data for **hearta1, hearta2** and **hearta3**. Fig. 2(b) shows the approximation error (E) as learning progresses in M-RAN. The approximation error is calculated from the test data provided. Table I shows the performance of M-RAN together with the performance of MFN's given in [20]. The values shown for MFN's are for the pivot architectures without shortcut connections for the respective problems and are quoted from [20, Table 11]. It can be seen from the table that M-RAN produces smaller approximation errors with a more compact network. The number of network parameters of the final M-RAN network is one or two order of magnitude less than that of pivot MFN's. The network parameters of M-RAN are the centers (μ), widths (σ) and the weights (α) and those for MFN's are the connection weights. In addition, in this example, M-RAN required only one epoch to learn whereas MFN's took around 50 epochs for learning the function.

2) Flare-Problem: The **flare** dataset consists of 1066 examples out of which 800 are for training and 266 are for testing. Each example is described by ten input attributes and three output attributes. All the training examples are complete. The M-RAN network used ten input units and three continuous output units to represent the output attributes. The performance of M-RAN was evaluated on the three different permutations of the **flare** dataset, viz. **flare1, flare2, flare3**, given in PROBEN1.

Table II shows the performance of M-RAN together with the performance of MFN's. As in the case of **hearta** problem, the values shown for MFN's are for the pivot architectures and are quoted from [20, Table 11]. From the table, it can be seen that M-RAN produced similar approximation errors as the MFN's but with a more compact network. For **flare1** the M-RAN approximation error is marginally higher than that of MFN. Also, M-RAN took only one epoch to learn, whereas MFN's took around 50 epochs for learning the function. As with **hearta**, the number of network parameters used by the

[2] Pivot architectures are the suggested network architectures for all datasets in [20], which produced the lowest validation set error compared with the results of the other network topologies employed in [20].

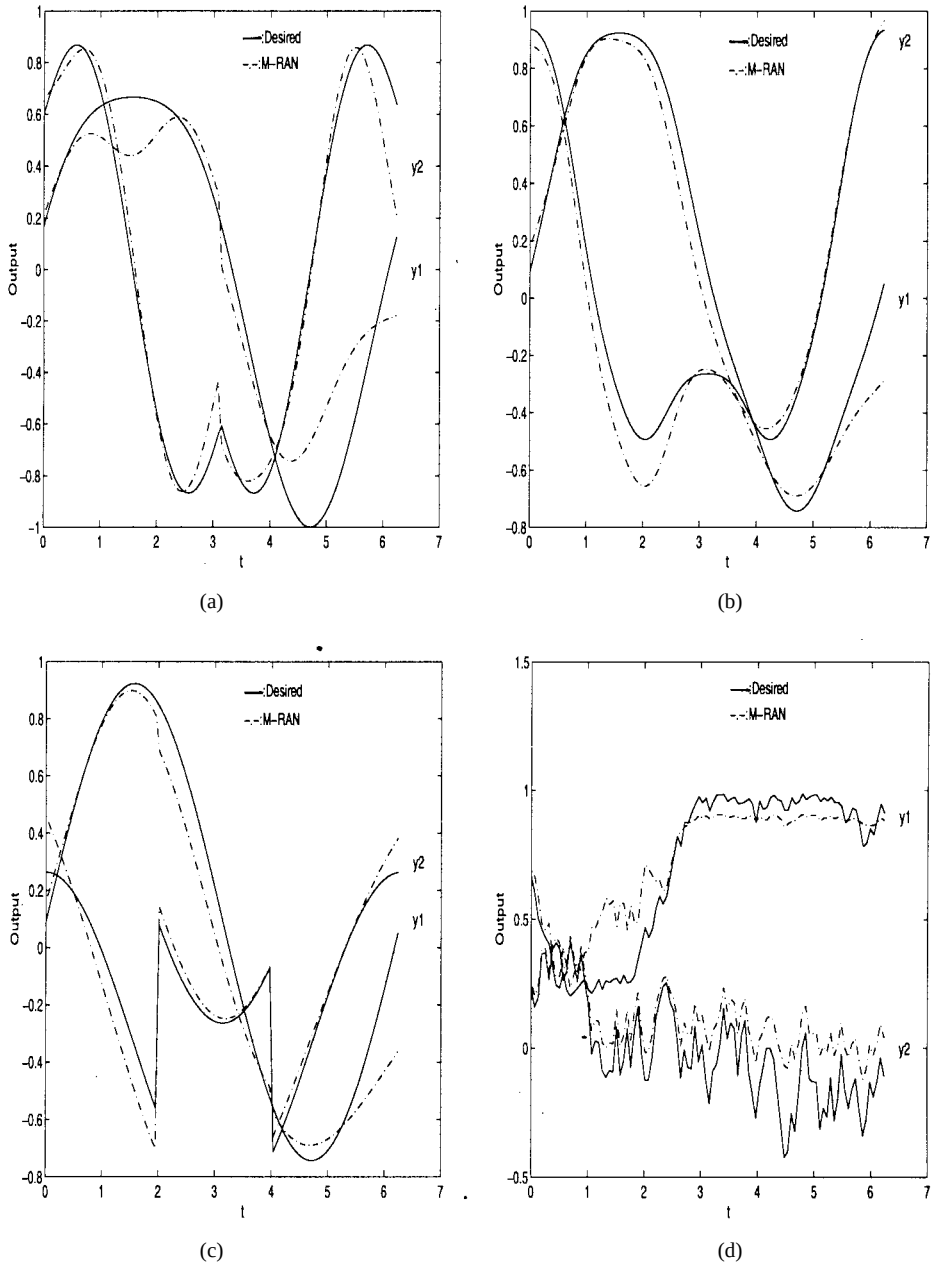


Fig. 4. Testing results of two input–two output sinusoidal function for M-RAN neural network.

pivot MFN's is considerably larger than that of M-RAN. The parameters for the M-RAN used here are: $\epsilon_{\max} = 2.0$, $\epsilon_{\min} = 0.8$, $\gamma = 0.97$, $e_{\min} = 0.05$, $e'_{\min} = 0.1$, $\kappa = 0.8$, $P_0 = 1.0$, $R_n = I_3$, $Q_0 = 0.001$, $M = 200$, $\delta = 0.000\,001$.

B. Classification Problem—Cancer

The pattern classification benchmark problem selected for this study is the **cancer** dataset which is created based on the "breast cancer Wisconsin" problem dataset from the UCI repository of machine learning databases. This dataset consists of 699 examples out of which 525 are to be used for training and the rest 174 to be used for testing. Each example is described by nine input attributes and one output attribute. Sixteen of the training examples have missing values for attribute 6. For this problem, the M-RAN network used nine input units and one output unit. As in the previous cases the performance of M-RAN was evaluated on the three different permutations of the **cancer** dataset, viz. **cancer1, cancer2, cancer3**, given in PROBEN1.

Table III shows the performance of M-RAN together with the performance of pivot MFN's. The values for MFN's quoted are from [20, Table 11]. The classification error indicated in this table is defined in [20] as the percentage of incorrectly classified examples. M-RAN produced similar classification errors with a more compact network than MFN's and took only one epoch to learn whereas MFN's in this cases took more than 50 epochs for learning the function. However, for **cancer1** M-RAN produced slightly higher errors (1.72) compared to that of MFN (1.38). On many cases, the number of network parameters of the final M-RAN is considerably less than that

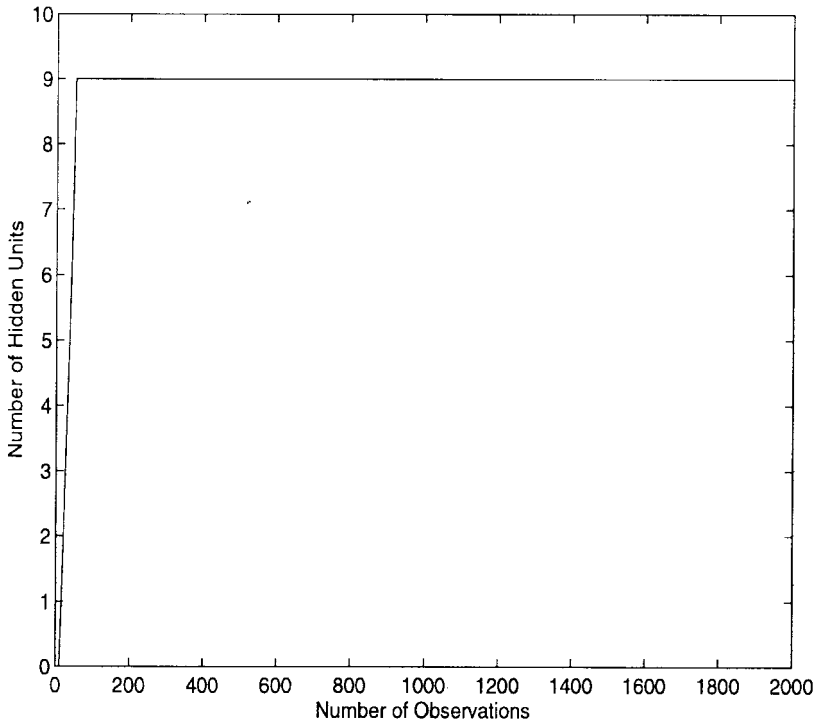


Fig. 5. Growth patterns of three input–one output system for M-RAN neural network.

of pivot MFN's. The final parameters used for the M-RAN algorithm are: $\epsilon_{\max} = 3.0$, $\epsilon_{\min} = 0.5$, $\gamma = 0.97$, $e_{\min} = 0.4$, $e'_{\min} = 0.8$, $\kappa = 0.8$, $P_0 = 1.0$, $R_n = 1.0$, $Q_0 = 0.000\,01$, $M = 50$, $\delta = 0.000\,001$.

IV. Performance of M-RAN with DI Algorithm

In this section, the performance of M-RAN is compared with the DI algorithm on two of the examples used in [18]. The first example is a two input–two output sinusoidal function approximation problem and the second is a three input–one output nonlinear trigonometric function approximation problem. The training and test data used for both of the examples are as per the details given in [18], [19]. For each problem, the same training and test datasets are applied to the two types of networks.

In addition to these two examples,we compared the performance of DI with M-RAN for a higher dimensional function approximation problem, viz. the **hearta1** problem from PROBEN1. To comply with the standard error measure suggested in [20], the **squared error percentage** E defined by (11) is used for this comparison. Besides the errors and network parameters, the comparison also includes the training time taken by the two networks. The results presented are based on the algorithms coded in C and run in Sun SPARC-IPX workstation.

A. Two Input–Two Output Continuous Function Approximation

For this example, M-RAN has to approximate the following two input–two output sinusoidal functions:

$$y_1 = \frac{1}{6}(5\sin(u_1) + \cos(u_2)), \tag{12}$$

$$y_2 = \frac{1}{5}(3\cos(u_1) + 2\sin(u_2)), \tag{13}$$

$$u_1 \in [0, 2\pi], \quad u_2 \in [-2\pi, 2\pi].$$

A training set is created by generating 200 uniformly distributed random values of u_1 and u_2 and calculating the associated values of y_1 and y_2. Fig. 3 displays the growth pattern for M-RAN neural network and Fig. 4 shows the results of testing the system with u_1 and u_2 set to four different parameterized functions of t. In all these cases, t takes on 100 evenly spaced values from 0 to 2π. The solid curve represents the desired output curve of the function and the slash-dot curve indicate the actual output of the proposed RBF network.

The parameters for the M-RAN algorithm chosen are $\epsilon_{\max} = 4.0$, $\epsilon_{\min} = 0.4$, $\gamma = 0.98$, $e_{\min} = 0.4$, $e'_{\min} = 0.6$, $\kappa = 1.0$, $P_0 = 1.0$, $R_n = I_2$, $Q_0 = 0.19$, $M = 13$, $\delta = 0.0001$.

The inputs for the four different graphs in Fig.4 are the same as in [18] and are given by the following equations. For Fig. 4(a)

$$u_1(t) = t \tag{14}$$

$$u_2(t) = \begin{cases} 2t & 0 \le t < \pi \\ 2\pi - 2t & \pi \le t < 2\pi \end{cases} \tag{15}$$

For Fig. 4(b)

$$u_1(t) = t \tag{16}$$

$$u_2(t) = \sin(2t). \tag{17}$$

For Fig. 4(c)

$$u_1(t) = t \tag{18}$$

$$u_2(t) = \mathit{step}(t) - 2\mathit{step}(t-2) + 2\mathit{step}(t-2) - \cdots \tag{19}$$

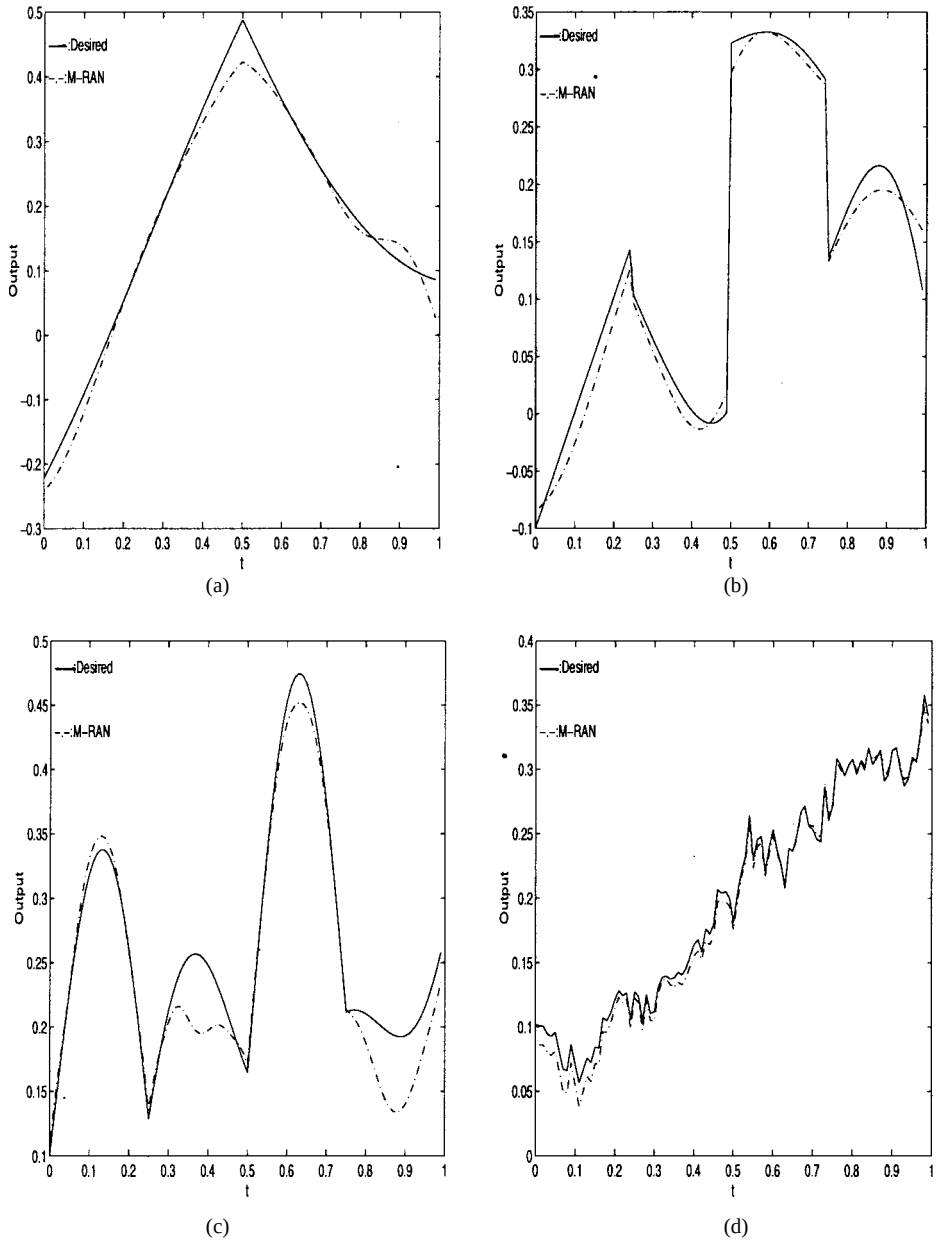


Fig. 6. Testing resutls of three input–one output system for M-RAN neural network.

where $u_2(t)$ is in fact an oscillating "clock" function between one and -1 with period 2. For Fig. 4(d)

$$u_1(t_i) = u_1(t_{i-1}) + \textit{random} \tag{20}$$

$$u_2(t_i) = u_2(t_{i-1}) + \textit{random}. \tag{21}$$

There are total of 400 points on which the function approximations are tested (four different input functions with 100 on each).

Table IV shows that compared to DI,[3] M-RAN produces a more compact network (less than 10% of DI) with similar test errors. The training time for M-RAN is less than half of the training time for DI algorithm.

[3]Number of parameters in a single hidden layer DI network is $m * (k - 1) + 1 + k * n$ where m, k and n are the number of inputs, hidden layer units and outputs, respectively.

B. Three Input–One Output Continuous Function Approximation

In this example, M-RAN is used to approximate the following function:

$$y = \frac{1}{10}(e^{u_1} + u_2 u_3 \cos(u_1 u_2) + u_1 u_3) \tag{22}$$

$$u_1 \in [0, 1], \quad u_2, u_3 \in [-2, 2].$$

This function is highly nonlinear, involving an exponential, three multiplications and a trigonometric function. The input vector for the M-RAN is $\mathbf{x} = [u_1, u_2, u_3]^T$ and y is approximated by the network output $f(\mathbf{x})$. A training set is created by generating 2000 uniformly distributed random values of u_1, u_2 and u_3 and calculating the associated values of y. The final parameters for the M-RAN are $\epsilon_{\max} = 3.0$, $\epsilon_{\min} = 0.3$,

TABLE I
PERFORMANCE OF DIFFERENT NETWORKS FOR HEARTA1, HEARTA2, AND HEARTA3

Benchmark Data	Network Used	Network Architecture	Squared Error Percentage for Testing Sets (E)	Number of Network Parameters	Number of Training Epochs
hearta1	MFN	35-32-1	4.55	1185	47
	M-RAN	13-3-1	4.27	46	1
hearta2	MFN	35-16-1	4.33	593	54
	M-RAN	13-5-1	3.81	76	1
hearta3	MFN	35-32-1	4.89	1185	46
	M-RAN	13-3-1	3.80	46	1

TABLE II
PERFORMANCE OF DIFFERENT NETWORKS FOR FLARE1, FLARE2, AND FLARE3

Benchmark Data	Network Used	Network Architecture	Squared Error Percentage for Testing Sets (E)	Number of Network Parameters	Number of Training Epochs
flare1	MFN	24-32-3	0.54	899	48
	M-RAN	10-17-3	0.57	241	1
flare2	MFN	24-32-3	0.32	899	47
	M-RAN	10-17-3	0.29	241	1
flare3	MFN	24-24-3	0.36	675	57
	M-RAN	10-16-3	0.36	227	1

TABLE III
PERFORMANCE OF DIFFERENT NETWORKS FOR CANCER1, CANCER2, CANCER3

Benchmark Data	Network Used	Network Architecture	Squared Error Percentage for Testing Sets (E)	Classification Error for Test Sets	Number of Network Parameters	Number of Training Epochs
cancer1	MFN	9-4-2-2	1.32	1.38	56	116
	M-RAN	9-2-1	1.86	1.72	23	1
cancer2	MFN	9-8-4-2	3.47	4.77	126	54
	M-RAN	9-2-1	3.25	4.02	23	1
cancer3	MFN	9-16-8-2	2.60	3.70	314	54
	M-RAN	9-2-1	2.60	3.44	23	1

$\gamma = 0.97$, $e_{\min} = 0.02$, $e'_{\min} = 0.12$, $\kappa = 0.7$, $P_0 = 1.0$, $R_n = 1.0$, $Q_0 = 0.05$, $M = 30$, $\delta = 0.0001$.

Fig. 5 presents the growth pattern for M-RAN. Fig. 6 shows the results of testing the system with u_1, u_2, and u_3 set to four different parameterized functions of t. In all these cases, t takes on 100 evenly spaced values from zero to one. The values of the inputs for the four different graphs in Fig. 6 are:
for Fig. 6(a)

$$u_1(t) = t \tag{23}$$

$$u_2(t) = 1.61(\text{a constant}), \tag{24}$$

$$u_3(t) = \begin{cases} 8t - 2 & 0 \le t < \frac{1}{2} \\ -8t + 6 & \frac{1}{2} \le t < 1 \end{cases} \tag{25}$$

for Fig. 6(b)

$$u_1(t) = t \tag{26}$$

$$u_2(t) = \begin{cases} 8t - 2 & 0 \le t < \frac{1}{2} \\ -8t + 6 & \frac{1}{2} \le t < 1 \end{cases} \tag{27}$$

$$u_3(t) = \mathit{step}(t) - 2\mathit{step}(t - 0.25) + 2\mathit{step}(t - 0.5) - \cdots \tag{28}$$

TABLE IV
COMPARATIVE RESULTS FOR THE TWO INPUT–TWO OUTPUT SINUSOIDAL FUNCTION

Network Used	Network Architecture	Squared Error Percentage for Testing Sets (E)	Number of Network Parameters	Training Time (seconds)
M-RAN	2-12-2	3.015	60	95.2
DI	3-145-2	3.079	723	225.87

TABLE V
COMPARATIVE RESULTS FOR THE THREE INPUT–ONE OUTPUT SYSTEM

Network Used	Network Architecture	Squared Error Percentage for Testing Sets (E)	Number of Network Parameters	Training Time (seconds)
M-RAN	3-9-1	0.0274	45	497.8
DI	4-280-1	0.0295	1397	609.7

TABLE VI
COMPARATIVE RESULTS FOR HEARTAL

Network Used	Network Architecture	Squared Error Percentage for Testing Sets (E)	Number of Network Parameters	Training Time (seconds)
M-RAN	13-3-1	4.27	46	150.6
DI	14-245-1	6.89	3662	1097.1

where u_s is an oscillating "clock" function between $+1$ and -1 with period of 0.25,
for Fig. 6(c)

$$u_1(t) = t \tag{29}$$

$$u_2(t) = \mathit{step}(t) - 2\mathit{step}(t-0.25) + 2\mathit{step}(t-0.5) - \cdots \tag{30}$$

$$u_3(t) = 2\sin(4\pi t) \tag{31}$$

for Fig. 6(d)

$$u_1(t) = t \tag{32}$$

$$u_2(t_i) = u_2(t_{i-1}) + \mathit{random}. \tag{33}$$

$$u_3(t_i) = u_3(t_{i-1}) + \mathit{random}. \tag{34}$$

Results of comparison are shown in Table V. There are a total of 400 points on which the solutions are tested (four different input functions with 100 points on each). The results show that M-RAN produces a more compact network (less than 10% of DI) with similar test errors. The training time for M-RAN is also less than that of DI algorithm.

From the figures of the growth patterns for MRAN for these problems, one can see MRAN does not perform any pruning. This is mainly due to the nature of the problems which are essentially static approximations. In [15]—[17], examples have been shown in which MRAN prunes the network dynamically.

C. Approximation Problem from PROBEN1—hearta1

In this section, we are comparing the performance of M-RAN with DI for the problem of **hearta1** from PROBEN1. The **hearta1** problem is a higher dimensional function approximation problem with 13 inputs and one output. The comparison results are presented in Table VI.

Table VI shows that M-RAN achieves a smaller test error for a higher dimensional problem with a compact network which is two orders of magnitude smaller than that of DI. The training time for M-RAN is one order lower than that of DI algorithm.

V. CONCLUSION

In this paper, the performance of the M-RAN algorithm is compared with MFN's using the BP variant RPROP and DI algorithm for pattern classification and function approximation problems. Results based on benchmark problems from the PROBEN1 database show that M-RAN produces a RBF neural network with smaller complexity than MFN's using RPROP with similar approximation/classification accuracy. Comparison of results with DI algorithm also show M-RAN producing a more compact network with similar accuracy as DI. Specifically, it has been shown that M-RAN produces these compact networks with less training time than that of DI algorithm.

ACKNOWLEDGMENT

The authors wish to thank Dr. J. O. Moody of the Department of Electrical Engineering, University of Nortre Dame, IN, who provided the source code and data for the DI algorithm and also clarified queries readily during this study.

REFERENCES

[1] J. G. Yang, "Classification of acoustic emission signals via Hebbian feature extraction," in *Int. Joint Conf. Neural Networks*, Seattle, WA, vol. 1, 1991, pp. 113–118.
[2] S. Haykin and C. Deng, "Classification of radar clutter using neural networks," *IEEE Trans. Neural Networks*, vol. 2, pp. 589–600, 1991.
[3] R. P. Lippmann, "An introduction to computing with neural nets," *IEEE Acoust., Speech, Signal Processing Mag.*, vol. 4, pp. 4–22, Apr. 1987.
[4] G. D. E. Rumelhart and R. J. Williams, "Learning internal representations by error propagation," *Nature*, vol. 323, pp. 533–536, 1986.
[5] S. E. Fahlman, "An empirical study of learning speed in backpropagation networks," School of Comput. Sci., Carnegie Mellon Univ., Pittsburgh, PA, Tech. Rep. CMU-CS-88-162, 1988.
[6] M. Riedmiller and H. Brau, "A direct adaptive method for faster backpropagation learning: The rprop algorithm," in *Proc. IEEE Int. Conf. Neural Networks*, San Francisco, CA, Apr. 1993.
[7] D. Broomhead and D. Lowe, "Mutivariable functional interpolation and adaptive networks," *Complex Syst.*, vol. 2, pp. 321–355, 1988.
[8] K. Tao, "A closer look at the radial basis function (RBF) networks," in *Conf. Rec. 27th Asilomar Conf. Signals, Syst., Comput.*, Pacific Grove, CA, 1993, pp. 401–405.
[9] J. Moody and C. J. Darken, "Fast learning in network of locally-tuned processing units," *Neural Computa.*, vol. 1, pp. 281–294, 1989.

[10] M. Musavi, W. Ahmed, K. Chan, K. Faris, and D. Hummels, "On training of radial basis function classifiers," *Neural Networks*, vol. 5, pp. 595–603, 1992.
[11] S. Chen, S. Billings, and P. Grant, "Recursive hybrid algorithm for nonlinear system identification using radial basis function networks," *Int. J. Contr.*, vol. 55, pp. 1051–1070, 1992.
[12] A. G. Bors and M. Gabbouj, "Minimal topology for a radial basis functions neural network for pattern classification," *Digital Signal Processing*, vol. 4, pp. 173–188, 1994.
[13] J. Platt, "A resource allocating network for function interpolation," *Neural Computa.*, vol. 3, pp. 213–225, 1991.
[14] V. Kadirkamanathan and M. Niranjan, "A function estimation approach to sequential learning with neural network," *Neural Computa.*, vol. 5, pp. 954–975, 1993.
[15] L. Yingwei, N. Sundararajan, and P. Saratchandran, "A sequential learning scheme for function approximation by using minimal radial basis function neural networks," *Neural Computa.*, vol. 9, pp. 461–478, 1997.
[16] ———, "Identification of time-varying nonlinear systems using minimal radial basis function neural networks," in *IEE Proc. Contr. Theory Applicat.*, vol. 144, no. 2, 1997, pp. 202–208.
[17] ———, "A sequential learning scheme for function approximation using minimal radial basis function (RBF) neural networks," Center for Signal Processing, School of Electrical and Electronic Engineering, Nanyang Technol. Univ., Singapore, Tech. Rep. CSP/9505, Nov. 1995.
[18] J. O. Moody and P. J. Antsaklis, "The dependence identification neural network construction algorithm," *IEEE Trans. Neural Networks*, vol. 7, pp. 1–8, Jan. 1996.
[19] J. O. Moody, "A new method for constructing and training multilayer neural networks," Master's thesis, Dept. Electr. Eng., Univ. Notre Dame, Notre Dame, IN, Apr. 1993.
[20] L. Prechelt, "Proben1—A set of neural network benchmark problems and benchmarking rules," Fakultät für Informatik, Univ. Karlsruhe, Germany, Tech. Rep. 21/94, Sept. 1994.
[21] M. J. L. Orr., "Regularization on the selection of radial basis function centers," *Neural Computa.*, vol. 7, pp. 606–623, 1995.

Lu Yingwei received the B.Eng. degree in 1993 from Beijing Polytechnic University, Beijing, China, and the Master's degree in electrical and electronic engineering from Nanyang Technological University, Singapore, in 1997.

She is currently with the Institute of Systems Science at the National University of Singapore.

Narasimhan Sundararajan (S'69–M'71–SM'84–F'96) received the B.E. degree in electrical engineering with first class honors from the University of Madras, India, in 1966, the M. Tech. degree from the Indian Institute of Technology, Madras, in 1968, and the Ph.D. degree in electrical engineering from the University of Illinois, Urbana-Champaign, in 1971.

From 1972 to 1991, he was working in the Indian Space Research Organization, Trivandrum, in positions ranging from Control System Designer to Director, Launch Vehicle Design Group contributing to the design and development of the Indian satellite launch vehicles. He has also worked as an NRC Research Associate at NASA—Ames in 1974 and as a Senior Research Associate at NASA Langley from 1981 to 1986. Since February 1991, he has been working as an Associate Professor in the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. He has more than 90 publications including two books. His research interests include aerospace control, neural networks and parallel implementation of neural networks.

Dr. Sundararajan is an Associate Fellow of AIAA. Presently he is an Associate Editor for the IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY, *IFAC Journal on Control Engineering Practice (CEP)*, *IEEE Robotics and Automation Magazine*, and also was an Associate Editor for *Control—Theory and Advanced Technology (C-TAT)*, Japan. He has contributed as a program committee member in a number of international conferences including as the Technical Program Chairman for the Third International Conference on Automation, Robotics and Computer Vision (ICARCV'94), held in Singapore in November 1994 and the IEEE Singapore International Conference on Intelligent Control and Instrumentation—SICICI'95, held in Singapore in July 1995. He is listed in Marquis *Who's Who in Science and Engineering*, and *Men of Achievement*, International Biographical Center, Cambridge, U.K.

P. Saratchandran (M'87–SM'96) received the B.Sc. (Eng.) degree from Regional Engineering College, Calicut, India and the M.Tech. degree in electrical engineering from the Indian Institute of Technology, Kharagpur, India. He received the M.Sc. degree in systems engineering from the City University, London, and the Ph.D. degree in the area of control engineering from Oxford University, U.K.

He worked for two years as a Scientist with the Indian Space Research Organization and spent five years as a Senior Design Engineer with the Hindustan Aeronautics Ltd., India, designing defence related avionics systems. From 1984 to 1990 he worked in Australia for various defense industries as a Systems Consultant and Manager developing real-time software/systems in Ada for the Australian Defence forces. During this period he was also a Visiting Fellow at the Department of Mathematics and Computer Science at the Macquarie University in Sydney, Australia. Since 1990 he has been with the Nanyang Technological University, where he is an Associate Professor. He has several publications in refereed journals and has also authored a book titled *Parallel Implementations of Backpropagation Neural Networks* (Singapore: World Scientifi). His interests include neural networks, parallel computing and control.

Dr. Saratchandran is an Editor for the journal *Neural Parallel and Scientific Computations*. He is listed in the Macquarie *Who's Who in the World* and in the *Leaders in the World* of the International Biographics Centre, Cambridge, U.K.