Identification of time-varying nonlinear systems using minimal radial basis function neural networks

L. Yingwei
N. Sundararajan
P. Saratchandran

Indexing terms: Adaptive identification of nonlinear systems, Nonlinear function approximation, RBF neural network

Abstract: An identification algorithm for time-varying nonlinear systems using a sequential learning scheme with a minimal radial basis function neural network (RBFNN) is presented. The learning algorithm combines the growth criterion of the resource allocating network of Platt with a pruning strategy based on the relative contribution of each hidden unit of the RBFNN to the overall network output. The performance of the algorithm is evaluated on the identification of nonlinear systems with both fixed and time-varying dynamics and also on a static function approximation problem. The nonlinear system with the fixed dynamics have been studied extensively earlier by Chen and Billings and the study with the time-varying dynamics reported is new. For the identification of fixed dynamics case, the resulting RBFNN is shown to be more compact and produces smaller output errors than the hybrid learning algorithm of Chen and Billings. In the case of time-varying dynamics, the algorithm is shown to adjust (add/drop) the hidden neurons of the RBFNN to 'adaptively track' the dynamics of the nonlinear system with a minimal RBF network.

1 Introduction

Radial basis function (RBF) neural networks have been widely used for nonlinear function approximation. The original RBF method has been traditionally used for strict multivariable function interpolation [1] and for this the RBF model requires as many RBF neurons (referred to here as computational units) as data points. This is rarely practical because of the large size of the data points. Broomhead and Lowe [2] removed this strict interpolation restriction and provided a neural network architecture where the number of RBF neurons can be far less than the data points.

The RBF neural network mainly consists of two layers, one hidden-layer and one output layer. The number of hidden-layer neurons is a variable to be determined by the user. Different basis functions like thin-plate spline function, multiquadratic function, inverse multiquadratic function and gaussian functions have been studied for the hidden-layer neurons but normally they are selected as gaussian. Compared with other types of neural networks like backpropagation feedforward networks, the RBF network requires less computation time for learning [3] and also has a more compact topology [4]. Apart from pattern recognition, RBF networks have been used successfully in a number of wide-ranging applications such as time-series prediction [5], in speech recognition [6] in adaptive control [7], etc.

In [3], Moody and Darken gave a solution for using a smaller number of hidden neurons than that of the data using a local representation for hidden units. For this method the number of hidden units is chosen *a priori* and no special assumptions are made about the basis functions such as orthogonality or being uniformly distributed over the input space. The basis functions used are radially symmetric where the widths can be computed using various 'P nearest neighbour' methods. They developed a hybrid learning method which combined the linear self-organised learning scheme (the well-known k-means clustering) for estimating the centres of the basis functions and a linear supervised learning method (the standard LMS algorithm) for estimating the output weights. They clearly showed that the RBF learning is faster compared to BP because of the combination of locality of representation and linearity of learning schemes.

A hierarchically self-organising learning (HSOL) algorithm was developed to find the minimal number of hidden units for RBF in [4]. Different types of basis function were analysed e.g. sigmoidal, sinusoidal and gaussian. The gaussian RBF was found suitable not only in generalising a global mapping but also in refining local features without much altering the already learned mapping. The learning algorithm recruits new hidden-layer neurons whenever necessary for improving network performance. The network starts with no hidden units and adds a new unit with a larger width in the beginning of the learning which is reduced for the following units until a minimal radius is obtained. In this algorithm, after adding a new node all the weights are adjusted using backpropagation. In summary, this paper introduced the concept of building up of the hidden neurons from zero to the required number using the inputs with the update of the RBF parameters being done by backpropagation or actually a gradient

IEE Proceedings online no. 19970891

Paper first received 15th February 1996 and in revised form 2nd September 1996

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

descent algorithm with its attendant problems.

The HSOL algorithm starts from no hidden units and then builds up the number of hidden units from the input data. An alternate approach is to start with as many hidden units as the number of inputs and then reduce them using a clustering algorithm which essentially puts close patterns in the input space into a cluster to remove the unnecessary hidden units. This approach has been developed by Musavi *et al.* [8] and it essentially consists of an iterative clustering algorithm that takes class membership into consideration and provides an approach to remove the unnecessary hidden units. Their approach also provides for the width estimation of the basis functions.

In a series of papers, Billings and his coworkers [9–11] have developed RBF neural networks for nonlinear dynamic system identification and they have compared the performance of RBF networks with the standard backpropagation feedforward networks. The results clearly show that the RBF networks are giving better performance with less network complexity. In their use of RBF schemes they have worked mainly on the batch learning scheme where the entire training set is available. Their scheme is also referred to as a hybrid scheme because it consists of an unsupervised learning scheme which estimates the centres of the basis functions and supervised learning scheme which uses the well-known least squares algorithm for determining the output weights. Also, in their original scheme, the number of centres is fixed *a priori* and in an improved version [11] they have indicated a recursive scheme for centre estimation. In all their studies they have chosen radial basis functions of the thin-plate type which does not have a width parameter or even when they have selected gaussian functions, the widths of these functions have been fixed. The unsupervised learning scheme for estimating the centres is the well-known k-means clustering algorithm. They have presented the results of their scheme on a nonlinear dynamic system identification problem which leads to a nonlinear autoregressive moving average with exogenous inputs (NARMAX) time series prediction problem.

Chen *et al.* [12] has improved Billings approach by including the output variables also in the centre estimation (trying to adjust the unsupervised learning to a supervised learning) and a critical angle technique to fine tune the shape factors of the basis functions. They have illustrated the improved performance of their algorithm on a number of problems.

In all these studies, the main learning scheme is of batch type and not online learning schemes. In other words, using the terminology of identification literature, these schemes are all batch type and not online/recursive schemes.

In [13] Platt proposed a sequential (online) learning algorithm for resource allocating network (RAN) to remedy this problem. In Platt's algorithm, hidden neurons are added based on the 'novelty' (referred to as 'innovations' in estimations literature) of the new data and also the weights are estimated using the well-known least mean square (LMS) algorithm. A new pattern is considered novel if the input is far away from the existing centres and if the error between the the network output and desired output is large. If no additional hidden neuron is added, the parameters of the existing hidden neurons, such as the centres, widths and weights, are updated.

Kadirkamanathan *et al.* [14] interpreted Platt's RAN from a functional approach and improved RAN by using an extended Kalman filter (EKF) instead of the LMS to estimate the network parameters. Their network, called RANEKF, is more compact and has faster convergence than RAN.

However, one drawback of both RAN and RANEKF is that once a hidden unit is created it can never be removed. Because of this both RAN and RANEKF could produce networks in which some hidden units, although active initially, may subsequently end up contributing little to the network output. Further, pruning becomes imperative for the identification of nonlinear systems with changing dynamics, because failing to prune the network in such cases will result in numerous inactive hidden neurons being present as the dynamics which caused their creation initially becomes nonexistent. If inactive hidden units can be detected and removed as learning progresses a more parsimonious network topology can then be realised. Also, when the neural networks are employed for control, the problems of overparameterisation that have been clearly brought out in [15] should be avoided. These are the motivations for us to combine the pruning strategy to the RANEKF scheme in this paper.

We propose an adaptive identification algorithm for nonlinear systems which adopts the basic idea of RANEKF and augments it with a pruning strategy to obtain 'a minimal' RBFNN. The pruning strategy removes those hidden neurons which consistently make little contribution to the network output. Although theoretical proof showing that the resulting RBFNN topology is in some sense minimal is still under investigation, we have referred to it as 'a minimal' RBFNN based on detailed simulation studies carried out on a number of benchmark problems [16]. For all these problems, the proposed approach produced a network with far fewer hidden neurons than both the RAN and the RANEKF with same or smaller errors.

2 Proposed identification algorithm

In this Section, first a brief description of how an identification problem of time-varying nonlinear system can be put into a function approximation framework using delayed inputs and outputs is shown. Then the identification of this nonlinear system (which is essentially function approximation problem) using a minimal RBF network is presented. As indicated in [11], a single-input and single-output time-varying nonlinear system can be expressed in terms of nonlinear functional expansion of lagged inputs and outputs as follows:

$$y(n) = f_s(y(n-1), \ldots, y(n-m_y), u(n-1), \ldots, u(n-m_u), n) + \text{noise}(n) \quad (1)$$

where $y(n)$, $u(n)$, noise(n) are the system output, input and white noise, respectively, and n represents the time index and the explicit appearance of n in the argument of the functional form indicating its time-varying nature. m_y and m_u are the lags of the output and input, respectively. Essentially the problem for the identification of a nonlinear dynamic system has been converted to a nonlinear time series problem with one step ahead prediction. Eqn. 1 is the general form of representation for the time-varying nonlinear system considered.

Fig. 1 shows the structure of a basic RBF network and it has two layers. The first layer is the hidden-layer which consists of an array of neurons, also referred to as the computing units. Each unit possesses a unit centre. A unit passes the euclidean distance between its centre and the network input vector through a nonlinear function. For general cases, and in particular in this paper, this nonlinear function is selected as a gaussian function. Thus the response of one hidden unit to the network input $\mathbf{x}$ can be expressed as follows:

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

where μ_k is the centre vector for the kth hidden unit and σ_k is the width for the gaussian function. $\|\ \|$ denotes the Euclidean norm and K indicates the total number of hidden neurons in the network. The second layer for the RBF network is essentially a linear combiner. Here we consider only a single output (with multiple inputs) for the simplicity of development and illustration of the algorithm. For networks with multiple outputs the same algorithm can be extended in a straightforward manner. Hence, the overall network response is a mapping f: $\mathbf{R}^m \rightarrow \mathbf{R}$, which is

$$f(\mathbf{x}, n) = \alpha_0 + \sum_{k=1}^{K} \alpha_k \phi_k(\mathbf{x}) \tag{3}$$

where $\mathbf{x} \in \mathbf{R}^m$ and K is the number of hidden units. The coefficient α_k is the connecting weight of the kth hidden unit to output layer. α_0 is the bias term.

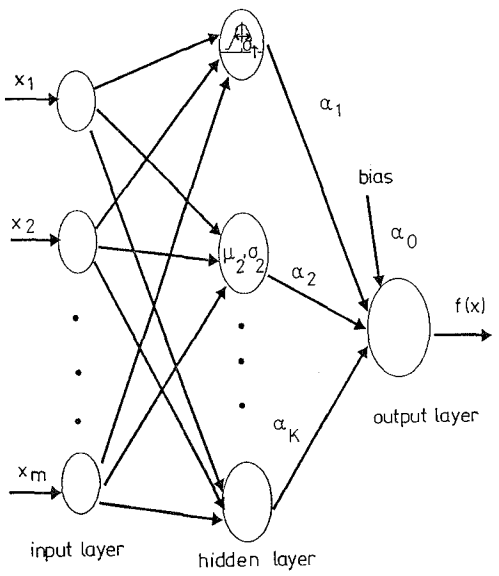

Fig. 1 *Structure of RBF neural network*

The aim of the identification algorithm is to use RBF network response $f()$ to capture or to approximate the underlying dynamics $f_s()$ in eqn. 1. Defining $m = m_y + m_u$, and letting

$$\mathbf{x}_n = [y(n-1), \ldots, y(n-m_y), u(n-1), \ldots, u(n-m_u)]^T \tag{4}$$

the RBF network identification problem becomes a function approximation problem of the type

$$\hat{y}(n) = f(\mathbf{x_n}, n) \tag{5}$$

2.1 Minimal RBF identification algorithm

For online identification applications using RBF network, a recursive algorithm is required for the learning process of the RBF network. For an effective adaptive identification scheme, the learning process requires allocation of new hidden units depending on the inputs as well as network parameter adaptation. The algorithm is explained as follows.

The network begins with no hidden units. As observations are received some of the input data will be taken as the new hidden units. The following three criteria decide whether an input $\mathbf{x}_n$ should be added to the hidden-layer of the network:

$$\|\mathbf{x_n} - \mu_\mathbf{nr}\| > \epsilon_n \tag{6}$$

$$e_n = y(n) - \hat{y}(n) > e_{min} \tag{7}$$

$$e_{rmsn} = \sqrt{\frac{\sum_{i=n-(M-1)}^{n}[y(i) - \hat{y}(i)]^2}{M}} > e'_{min} \tag{8}$$

where μ_{nr} is a centre of a hidden unit whose distance from x_n is the nearest among those of all the other hidden unit centres. ϵ_n, e_{min} and e'_{min} are thresholds to be selected appropriately. M represents both the size of a sliding data window which covers a number of latest observations for calculating the RMS output error e_{rmsn} and the maximal consecutive observations for which the normalized output value r_k^n, (k = 1, ..., K) is less than a previously selected threshold δ. The normalised output r_k^n can be expressed by the following equation:

$$r_k^n = \left\| \frac{o_k^n}{o_{max}^n} \right\|, (k = 1, \ldots, K) \tag{9}$$

$$o_k^n = \alpha_k \exp\left(-\frac{\|\mathbf{x_n} - \mu_\mathbf{k}\|^2}{\sigma_k^2}\right) \tag{10}$$

where o_k^n is the output for the kth hidden unit and $\|o_{max}{}^n\|$ is the largest absolute hidden unit output value for the nth input.

When a new hidden unit is added to the network the parameters associated with the unit are assigned as follows:

$$\alpha_{K+1} = e_n \tag{11}$$

$$\mu_{\mathbf{K}+1} = \mathbf{x}_n \tag{12}$$

$$\sigma_{K+1} = \kappa\|\mathbf{x}_n - \mu_\mathbf{nr}\| \tag{13}$$

where κ is an overlap factor which determines the overlap of the response of a hidden unit in the input spaces. When the observation ($\mathbf{x}_n$, $y(n)$) does not meet the criteria for adding a new hidden unit, extended Kalman filter (EKF) will be used to adjust the parameters of the network, $\mathbf{w} = [\alpha_0, \alpha_1, \mu_1^T, \sigma_1, \ldots, \alpha_K, \mu_K^T, \sigma_K]^T$, given by

$$\mathbf{w}_{(n)} = \mathbf{w}_{(n-1)} + \mathbf{k}_n e_n \tag{14}$$

where k_n is called the Kalman gain vector given by

$$\mathbf{k}_n = \mathbf{P}_{n-1}\mathbf{a}_n[\mathbf{R}_n + \mathbf{a}_n^T\mathbf{P}_{n-1}\mathbf{a}_n]^{-1} \tag{15}$$

where a_n is the gradient vector and has the following form,

$$\mathbf{a}_n = \nabla_\mathbf{w} f(\mathbf{x}_n) \tag{16}$$

$$= [1, \phi_1(\mathbf{x}_n), \phi_1(\mathbf{x}_n)\frac{2\alpha_1}{\sigma_1^2}(\mathbf{x}_n - \mu_1)^T,$$

$$\phi_1(\mathbf{x}_n)\frac{2\alpha_1}{\sigma_1^3}\|\mathbf{x}_n - \mu_1\|^2, \ldots,$$

$$\phi_K(\mathbf{x}_n), \phi_K(\mathbf{x}_n)\frac{2\alpha_K}{\sigma_K^2}(\mathbf{x}_n - \mu_K)^T,$$

$$\phi_K(\mathbf{x}_n)\frac{2\alpha_K}{\sigma_K^3}\|\mathbf{x}_n - \mu_K\|^2]^T \quad (17)$$

R_n is the variance of the measurement noise; P_n is the error covariance matrix which is updated by

$$\mathbf{P}_n = [\mathbf{I} - \mathbf{k}_n\mathbf{a}_n^T]\mathbf{P}_{n-1} + Q\mathbf{I} \quad (18)$$

where Q is a scalar that determines the allowed random step in the direction of gradient vector [Note 1]. Assume that the number of parameters to be adjusted is N, P_n is therefore a $N \times N$ positive definite symmetric matrix. When a new hidden unit is allocated the dimensionality of P_n increases by

$$\mathbf{P}_n = \begin{pmatrix} \mathbf{P}_{n-1} & 0 \\ 0 & P_0\mathbf{I} \end{pmatrix} \quad (19)$$

and the new rows and columns are initialised by P_0. P_0 is an estimate of the uncertainty in the initial values assigned to the parameters, which in this algorithm is also the variance of the observations $\mathbf{x}_n$ and $y(n)$. The dimension of identity matrix I is equal to the number of new parameters introduced by adding a new hidden unit.

In addition, to keep the RBF network in a minimal size and make sure that there is no superfluous hidden units exiting in the network, for every observation, each normalised hidden unit output value r_k is examined to decide whether or not a hidden neuron should be removed. If r_k is less than a threshold δ for M consecutive observations, then the kth hidden unit should be removed. The following is the summary for the whole algorithm:

For each input $\mathbf{x}_n$, compute:

$$\phi_k(\mathbf{x}_n) = \exp\left(-\frac{1}{\sigma_k^2}\|\mathbf{x}_n - \mu_\mathbf{k}\|^2\right) \quad (20)$$

$$\hat{y}(n) = f(\mathbf{x}_n, n) = \alpha_0 + \sum_{k=1}^{K}\alpha_k\phi_k(\mathbf{x}_n) \quad (21)$$

$$\epsilon_n = \max\{\epsilon_{max}\gamma^n, \epsilon_{min}\}, (0 < \gamma < 1) \quad (22)$$

$$e_n = y(n) - \hat{y}(n) \quad (23)$$

$$e_{rmsn} = \sqrt{\frac{\sum_{i=n-(M-1)}^{n}[y(i) - \hat{y}(i)]^2}{M}} \quad (24)$$

if $e_n > e_{min}$ and $\|x_n - \mu_{nr}\| > \epsilon_n$, and $e_{rmsn} > e'_{min}$, then allocate a new hidden unit with

$$\alpha_{k+1} = e_n \quad (25)$$

$$\mu_{\mathbf{k}+1} = \mathbf{x}_n \quad (26)$$

$$\sigma_{k+1} = \kappa\|\mathbf{x}_n - \mu_{\mathbf{nr}}\| \quad (27)$$

else

$$\mathbf{w}_n = \mathbf{w}_{(n-1)} + \mathbf{k}_n e_n \quad (28)$$

$$\mathbf{k}_n = P_{n-1}\mathbf{a}_n[R_n + \mathbf{a}_n^T P_{n-1}\mathbf{a}_n]^{-1} \quad (29)$$

$$P_n = [I - k_n\mathbf{a}_n^T]P_{n-1} + QI \quad (30)$$

Note 1: According to [14], the rapid convergence of EKF algorithm may prevent the model from adapting to future data. Q is introduced to the model called as random walk model to avoid this problem.

Compute the outputs of all hidden units o_k^n, (k = 1, ..., K)

Find the largest absolute hidden-unit output value $\|o_{max}{}^n\|$

Calculate the normalised value for each hidden unit $r_k^n = \|o_k^n/o_{max}{}^n\|$, ($k$ = 1, ... K)

If $r_k^n < \delta$ for M consecutive observations, then

– prune the kth hidden neurons

– reduce the dimensionality of P_n to fit for the requirement of EKF.

3 Performance results of minimal RBF algorithm

In this Section, simulation results using the proposed algorithm for a static problem, namely, the nonlinear function approximation as well as a dynamic nonlinear system identification problem proposed in [12] are presented. The results for identification of the nonlinear system consists of two parts. The first part considers the case of a fixed dynamics nonlinear system as which has been studied in [11, 12], and the second part studies the case of the preceeding nonlinear system whose dynamics is changed online by varying its parameters. Results from the present study are compared with the earlier results and the advantages of the proposed method are highlighted.

3.1 Static function approximation problem

This problem is the one proposed in [12] and it consists of a nonlinear function which consists of six exponential functions in the following way:

$$\begin{aligned} y(\mathbf{x}) = &\exp\left[-\frac{(x_1 - 0.3)^2 + (x_2 - 0.2)^2}{0.01}\right] \\ &+ \exp\left[-\frac{(x_1 - 0.7)^2 + (x_2 - 0.2)^2}{0.01}\right] \\ &+ \exp\left[-\frac{(x_1 - 0.1)^2 + (x_2 - 0.5)^2}{0.02}\right] \\ &+ \exp\left[-\frac{(x_1 - 0.9)^2 + (x_2 - 0.5)^2}{0.02}\right] \\ &+ \exp\left[-\frac{(x_1 - 0.3)^2 + (x_2 - 0.8)^2}{0.01}\right] \\ &+ \exp\left[-\frac{(x_1 - 0.7)^2 + (x_2 - 0.8)^2}{0.01}\right] \end{aligned} \quad (31)$$

The aim here is to evolve a minimal RBF using the method developed in this paper to approximate the function with small error. For this approximation, 121 training patterns ($\mathbf{x}$, y) are provided, where $x_i \in \{0, 0.1, ..., 0.9, 1\}$ and $i \in \{1, 2\}$. The input vector of the network is selected as $\mathbf{x}$. After 121 rounds of training the distributions of network hidden neuron centres achieved by the algorithm developed in this paper is shown in Fig. 2.

This algorithm requires careful selection of the threshold parameters ϵ_n, e_{min} and e'_{min} as defined in eqns. 6–8 which controls the growth characteristics of the network. The parameter δ described in Section 2.1 controls the pruning aspects of the network. The other parameters κ, Q, P_0 relate to the extended Kalman filter algorithm for parameter updates (like the centres μ, widths σ and weights α).

Based on parametric studies, the parameters required for the minimal RBF algorithm are chosen as follows:

$\epsilon_{max} = 0.4$, $\epsilon_{min} = 0.2$, $\gamma = 0.993$, $e_{min} = 0.02$, $\kappa = 0.85$, $P_0 = R_n = 1.0$, $Q = 0.002$, $\delta = 0.0001$; $M = 100$; $e'_{min} = 0.2$.

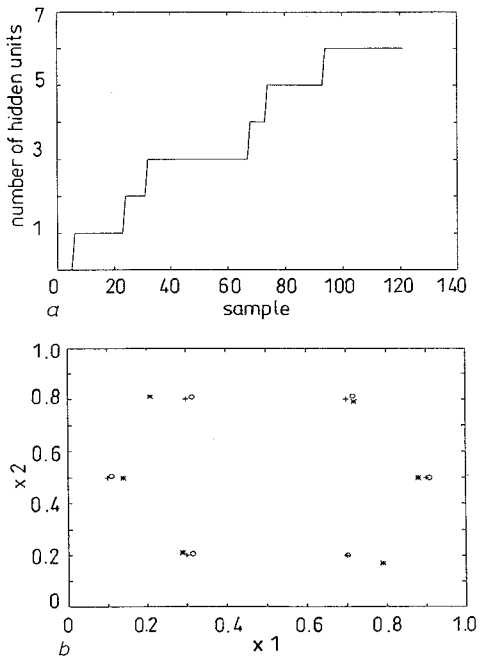

Fig.2 *Number and distribution of hidden neuron centres*
a Growth pattern
b Distribution
∘ estimated centre by minimal RBF
* estimated centre by hybrid method

As the function to approximate is in the form of summation of six gaussian exponential functions, one can say theoretically that the RBF network should have six neurons with gaussian functions in its hidden-layer. Fig. 2*a* displays the evolution of the number of the hidden neurons in the minimal RBFNN. Six hidden units are obtained within about 90 samples. In Fig. 2*b*, '+' indicates the true positions of hidden unit centres (i.e. the centres of the six gaussian functions in eqn. 31) and the circles represent the estimated positions of hidden unit centres obtained from the minimal RBF network. The centres achieved by the hybrid learning algorithm in [12] are indicated with stars. It can be seen from this Figure, that the centres achieved by the minimal RBF network are quite close to the true centres. Table 1 presents the comparison of the width estimation.

From the table it is obvious that both the centres and width values for the gaussian functions in the hidden neurons are close to the true values so that the proposed algorithm can accurately approximate a nonlinear function with a minimal (in this case exact) network size.

The selection of the required threshold values for the algorithm and also the data window size are some of the critical parameters to be chosen properly. For this example, since no measurement noise is assumed data window size is not critical. However, the selection of the threshold e'_{min} is critical. For instance, in the example, simulation results are presented in Fig. 3 for different choices of e'_{min} for one round of training. For the nominal parameter of $e'_{min} = 0.2$, the network selects 6 hidden neurons. If e'_{min} is chosen as 0.25, the network selects 7 hidden units. Also, if e'_{min} is chosen as a smaller value, for example $e'_{min} = 0.15$, 6 hidden units are achieved. Thus, different choices of e'_{min} may result in different network sizes. However, in this example, the increase in the number of hidden neurons is not very significant (seven instead of the true six).

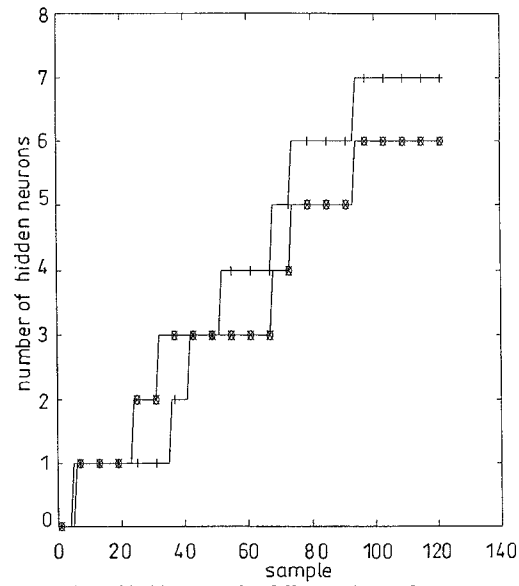

Fig.3 *Number of hidden units for different e'_{min} values*
× $e'_{min} = 0.15$
∘ $e'_{min} = 0.2$
+ $e'_{min} = 0.25$

3.2 Nonlinear system identification

3.2.1 Fixed dynamics case: Here the algorithm is tested on the same nonlinear dynamic system as discussed in Chen and Billings [11] and the results of this algorithm are compared with the results obtained in [11] using the hybrid learning algorithm. The nonlinear system is of the form

$$\begin{aligned} y(n) =& (0.8 - 0.5\exp(-y(n-1)^2))y(n-1) \\ & - (0.3 + 0.9\exp(-y(n-1)^2))y(n-2) \\ & + 0.1\sin(3.1415926y(n-1)) + \text{noise}(n) \end{aligned} \quad (32)$$

In eqn. 32, noise (n) is a gaussian white sequence with means of zero and variance of 0.04. In [11], the hybrid learning algorithm is employed to identify this system. This algorithm applies recursive n-means clustering algorithm for adjusting the RBF centres and least square algorithm for updating the RBF weights. The number of hidden neurons must be decided before the beginning of the identification process. The hidden neurons for algorithm can be neither added nor reduced during the learning process. 30 hidden neurons are employed in that network. The network input vector is $\mathbf{x}_n = [y(n-2), y(n-1)]'$ and output $\hat{y}(n)$ is

Table 1: True and estimated centres and width values

True centre	(0.3, 0.2)	(0.7, 0.2)	(0.1, 0.5)	(0.9, 0.5)	(0.3, 0.8)	(0.7, 0.8)
Estimated centre	(0.32, 0.2)	(0.7, 0.2)	(0.11, 0.5)	(0.9, 0.5)	(0.32, 0.8)	(0.72, 0.82)
True width σ^2	0.01	0.01	0.02	0.02	0.01	0.01
Estimated width σ^2	0.012	0.013	0.024	0.023	0.012	0.012

used to approximate $y(n)$. Thus, the network can be considered as a one-step ahead predictor. 1500 samples of data are generated by the above model and is used for identification purposes.

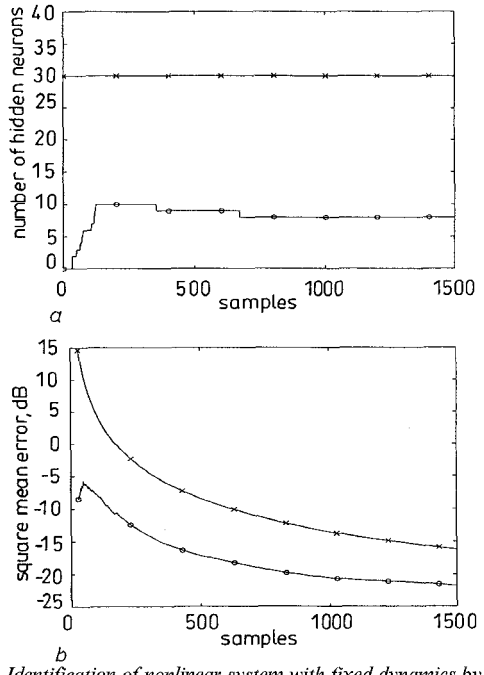

Fig. 4 *Identification of nonlinear system with fixed dynamics by minimal RBF neural networks*
a Growth pattern
b Square mean error
× recursive hybrid algorithm
∘ minimal RBF network

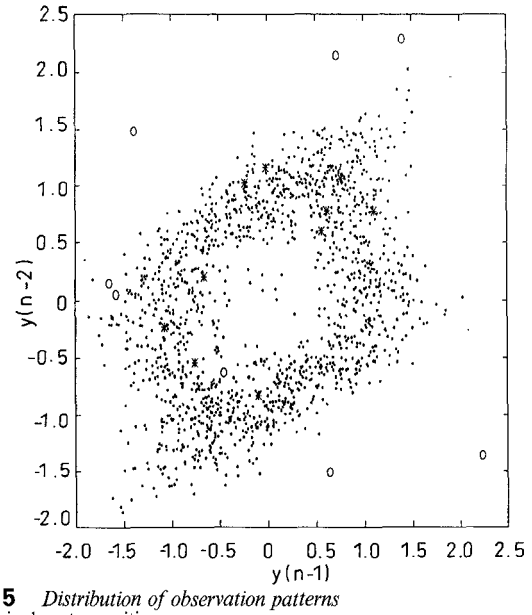

Fig. 5 *Distribution of observation patterns*
* original centre position
∘ final centre position
• observation

The initial points $y(0)$ and $y(-1)$ are set as $y(0) = 0.1$, $y(-1) = 0.01$, respectively. Fig. 4*a* and Fig. 4*b* show the number of hidden neurons used for the hybrid learning algorithm and the corresponding evolution of the mean square errors indicated by curves with 'x', respectively.

For the same example, the minimal RBFNN algorithm is used to identify this system. The input and output vectors of the network are chosen to be the same as those in [11, 12]. The parameters required by the algorithm are selected as $\epsilon_{max} = 4.0$, $\epsilon_{min} = 0.4$, $\gamma = 0.96$, $e_{min} = 0.2$, $\kappa = 0.87$, $P_0 = R_n = 1.0$, $Q = 0.02$, $\delta = 0.0001$; $e'_{min} = 0.4$. As noise presents in the identification process, a data window with size of $M = 25$ is employed here to avoid the problem of overfitting for the neural network. The identification results achieved by our algorithm are displayed in Fig. 4 by curves marked with symbol '∘'. In Fig. 4*a*, the number of hidden neurons achieved for the minimal RBFNN is shown to be eight which is much less than 30 required by the networks proposed in [11, 12]. Fig. 4*b* displays the evolution of the mean square errors obtained by our algorithm. Comparing this error history with that obtained by the hybrid learning algorithm, it can be clearly seen that the our algorithm not only reduces the size of the network but also reduces the network approximation errors. In addition, Fig. 5 also shows the distribution time history of the hidden centres achieved in the minimal RBFNN. The stars indicate the centre positions corresponding to the hidden units which were added first to the network and the circles represent the the final positions of the centres in the network after 1500 observations. The dots represent the distributions of the observations (i.e. $y(n-2)$ via $y(n-1)$). It can be seen that the first ten hidden units are added to the network. As more observations are received, some of the hidden units becomes superfluous and are removed from the network. Finally, there are only eight hidden units left in the network. From this Figure, notice that during the beginning stage, the system observations are distributed in the middle of the graph so the initial positions of hidden unit centres in the network are also chosen to be located in the middle of the graph. As the system observations evolve to the edge of the graph, the positions of the hidden centres are adapted to move to the edge of the graph.

3.2.2 Varying dynamics case: The results presented in the previous Section were for the identification of a nonlinear system whose dynamics was fixed. To further study the adaptive capability of the minimal RBFNN algorithm, the dynamics of the system is deliberately changed during the simulation as follows:

$$\begin{aligned} y(n) =& (0.8 - c_1(n)\exp(-y(n-1)^2))y(n-1) \\ & - (0.3 + c_2(n)\exp(-y(n-1)^2))y(n-2) \\ & + c_3(n)\sin(3.1415926y(n-1)) + \text{noise}(n) \end{aligned} \quad (33)$$

where $c_1(n)$, $c_2(n)$, $c_3(n)$ are time-varying coefficients and their expressions are shown as follows:

$$c_1(n) = \begin{cases} 0 & \text{if } 2000 < n < 7000 \\ 0.5 & \text{otherwise} \end{cases} \quad (34)$$

$$c_2(n) = \begin{cases} 0 & \text{if } 3000 < n < 6000 \\ 0.9 & \text{otherwise} \end{cases} \quad (35)$$

$$c_3(n) = \begin{cases} 0 & \text{if } 4000 < n < 5000 \\ 0.1 & \text{otherwise} \end{cases} \quad (36)$$

it can be seen that during the period $n < 1500$ $c_1(n)$, $c_2(n)$, $c_3(n)$ are fixed at 0.5, 0.9, 0.1, respectively, which is the same case as suggested those in [12, 11]. After 2000 observations, the dynamics of this system changes in the way expressed by eqns. 34–36. The minimal RBF algorithm is used for online identification of this time-varying nonlinear system.

Simulation results for a total of 10000 observations are presented in Fig. 6. As $c_1(n)$, $c_2(n)$ and $c_3(n)$ change to 0 and later recover to their original values sequentially, the minimal RBFNN algorithm is able to track

the varying system dynamics and prune/add hidden neurons appropriately to keep the size of RBFNN minimal. From 2000th observation to 4000th observation, $c_1(n)$, $c_2(n)$ and $c_3(n)$ change to 0 in sequence. Some of the hidden neurons generated during the first 1500 samples appear to be superfluous and the minimal RBFNN algorithm prunes these neurons and reduces the size of network to contain only five hidden neurons. After 5000 observations, $c_1(n)$, $c_2(n)$ and $c_3(n)$ begin to recover to their original values and the minimal RBFNN algorithm adds hidden units to the network. Finally, when all the values of $c_1(n)$, $c_2(n)$ and $c_3(n)$ reach the original values as those in [11], the number of the hidden units in the minimal RBF network reaches eight again.

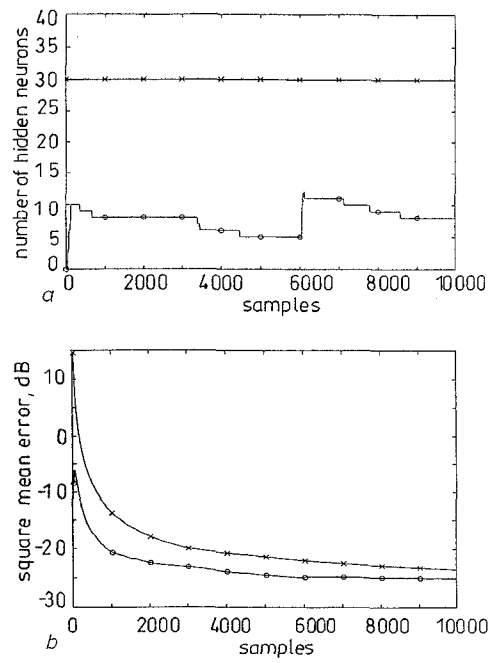

Fig. 6 *Identification of nonlinear system with time-varying dynamics by minimal RBF neural network and recursive hybrid algorithm*
a Growth pattern
b Square mean error
× Recursive hybrid algorithm
◦ minimal RBF network

The hybrid learning algorithm of [11, 12], is also applied to this time-varying nonlinear system and the results are also shown in Fig. 6 for comparison purposes. Since there is no method of reducing/increasing the number of hidden units for the hybrid algorithm proposed in [11, 12], the size of the network will keep unchanged during the process of identification for this time-varying system. Fig. 6*b* shows the mean square error achieved by both the proposed minimal RBFNN algorithm and the hybrid learning algorithm. The number of neurons assumed for hybrid algorithm is 30 which is highly in excess of the eight required in the minimal RBF algorithm. Even then the mean square error for the hybrid algorithm is higher. Also, this clearly shows that assuming the neurons to be 30 in this case results in a highly overparameterised model with all the the attendant problems [15].

4 Conclusion

An algorithm for adaptively identifying nonlinear systems using a minimal RBF network has been developed. This algorithm provides ways to increase and decrease the RBF hidden neurons depending on the input data and thus results in a minimal network topology. Tests of this algorithm for both nonlinear function approximation and nonlinear dynamic system identification have been presented. Specifically, the adaptive capability of the algorithm to track the varying dynamics of a nonlinear system with a minimal RBF neural network has been demonstrated.

5 References

1 POWELL, M.J.D.: 'Radial basis function for multivariate interpolation: A review', *in* MASON, J.C., and COX, M.G. (Eds.): 'Algorithm for approximation' (Clarendon Press, Oxford, 1987), pp. 143–168
2 BROOMHEAD, D.S., and LOWE, D.: 'Multivariable functional interpolation and adaptive networks', *Complex Syst.*, 1988, **2**, pp. 321–355
3 MOODY, J., and DARKEN, C.J.: 'Fast learning in network of locally-tuned processing units', *Neural Comput.*, 1989, **1**, pp. 281–294
4 LEE, S., and KIL, R.M.: 'A gaussian potential function network with hierarchically self-organizing learning', *Neural Netw.*, 1991, **4**, pp. 207–224
5 POGGIO, T., and GIROSI, F.: 'Networks for approximation and learning', *Proc. IEEE*, 1990, **78**, pp. 1481–1497
6 NIRANJAN, M., and FALLSIDE, F.: 'Neural networks and radial basis functions in classifying static speech patterns', *Comput. Speech Lang.*, 1990, **4**, pp. 275–289
7 SANNER, R.M., and SLOTINE, J.-J.E.: 'Gaussian networks for direct adaptive control', *IEEE Trans. Neural Netw.*, 1992, **3**, (6), pp. 837–863
8 MUSAVI, M.T., AHMED, W., CHAN, K.H., FARIS, K.B., and HUMMELS, D.M.: 'On training of radial basis function classifiers', *Neural Netw.*, 1992, **5**, pp. 595–603
9 CHEN, S., BILLINGS, S.A., COWAN, C.F.M., and GRANT, P.M.: 'Practical identification of narmax models using radial basis functions', *Int. J. Control*, 1990, **52**, pp. 1327–1350
10 CHEN, S., and BILLINGS, S.A.: 'Neural networks for non-linear system identification', *Int. J. Control*, 1992, **56**, pp. 319–346
11 CHEN, S., BILLINGS, S.A., and GRANT, P.M.: 'Recursive hybrid algorithm for non-linear system identification using radial basis function networks', *Int. J. Control*, 1992, **55**, pp. 1051–1070
12 CHEN, C.L., CHEN, W.C., and CHENG, F.Y.: 'Hybrid learning algorithm for gaussian potential function network', *IEE Proc. D*, 1993, **140**, pp. 442–448
13 PLATT, J.C.: 'A resource allocating network for function interpolation', *Neural Comput.*, 1991, **3**, pp. 213–225
14 KADIRKAMANATHAN, V., and NIRANJAN, M.: 'A function estimation approach to sequential learning with neural network', *Neural Comput.*, 1993, **5**, pp. 954–975
15 WARWICK, K.: 'A critique of neural networks for discrete-time linear control', *Int. J. Control*, 1995, **61**, (6), pp. 1253–1264
16 YINGWEI, L., SUNDARARAJAN, N., and SARATCHANDRAN, P.: 'A sequential learning scheme for function approximation using minimal radial basis function neural networks'. Tech. report CSP/9505, Center for Signal Processing, School of Electrical and Electronic Engineering, Nanyang Technological University, November 1995