NEURAL NETWORK MODEL IDENTIFICATION BASED ON THE SUBTRACTIVE CLUSTERING METHOD

Haralambos Sarimveis, Alex Alexandridis and George Bafas

Department of Chemical Engineering, 9, Heroon Polytechniou str., Zografou Campus, Athens 15780, Greece, Tel.: +30-1-7723236, Fax: +30-1-7723155

Abstract: A new algorithm for training radial basis function neural networks is presented in this paper. The algorithm, which is based on the subtractive clustering technique, has a number of advantages compared to the traditional learning algorithms, including faster training times and more accurate predictions. Due to these advantages the method proves suitable for developing discrete-time models for complex dynamical systems. *Copyright © 2002 IFAC*

Keywords: Radial base function networks, Identification algorithms, Training, Dynamic Modeling, Discrete-time systems.

1. INTRODUCTION

The development of an efficient control scheme is often based on the existence of an appropriate dynamic model of the process. However, the dynamic behavior of most processes cannot be easily modeled by fundamental equations, due to the complexity, nonlinearity and/or uncertainty of the system. On the other hand, due to the continuously reducing cost of informational systems and data storage elements, a plethora of data is electronically stored in an every day basis for a number of processes, which in many cases remain unexploited. These data are often a valuable source of information that can be used for developing dynamic black-box models. Therefore, mathematical tools that can extract dynamic information about a process from a pool of data are necessary today more than ever before. Such a tool that has become very popular during the last decade is the family of Artificial Neural Networks (ANN) architectures.

Radial Basis Function (RBF) neural networks form a class of ANNs, which has certain advantages over other types of ANNs, such as better approximation capabilities, simpler network structures and faster learning algorithms. RBF neural networks have been applied in many different scientific areas, including dynamic system identification and control (Shin and Goel, 2000; Bhartiya and Whiteley, 2001; Li *et al.*, 2001). Given the availability of an information data base, a training algorithm for developing RBF neural network models, consists of two stages: In the first stage the structure of the network, i.e. the number of hidden nodes is selected. In the second, the network parameters associated with the neurons and/or the interconnection links are determined using an optimization algorithm, which minimizes the errors between the true outputs and the network predictions over a set of training examples. This is the training procedure, during which the network learns the relationships between the input and output variables.

Due to the popularity of RBF neural networks, several researchers have been working during the last decade to develop more efficient training algorithms, compared to the standard techniques (Moody and Darken, 1988,1989; Leonard and Kramer, 1991). Some of the algorithms that have been proposed use individual training of each hidden unit based on functional analysis (Houlomb and Morari, 1991), initial selection of a large number of hidden units which is reduced, as the algorithm proceeds (Musavi *et al.*, 1992) or utilization of genetic algorithms (Billings and Zheng, 1995). Most of these methods try to determine both the optimum network structure and the values of the unknown parameters. However, a common drawback in the above algorithms is the long computational times. One additional problem is that since most of the above methods try to solve a nonlinear optimization problem, they may be trapped in local minima.

Although most of the RBF training algorithms are much faster than the techniques used for other types of neural networks, they are still time consuming especially if a large informational database is available. Obviously new techniques, which can further reduce the necessary computations, will enhance the applicability and the effectiveness of the RBF neural network architecture. In this article, a new fast algorithm for training RBF networks is proposed, which selects the hidden node centers using the subtractive clustering (SC) method (Chiu, 1994). The speed of the proposed method is due to the fact that only one pass of the training data is required, so that the solution of a nonlinear optimization problem is avoided. After the selection of the hidden node centers, the rest of the network parameters are obtained using standard techniques: The widths of the nodes are determined by the *P*-nearest neighbor heuristic (Leonard and Kramer, 1991) and the weights between the hidden layer and the output layer are calculated by linear regression.

The methodology is illustrated through the application to a simulated Continuous Stirred Tank Reactor (CSTR). The advantages of the proposed learning strategy are identified and the results are compared with those obtained using the standard MaqQueen *k*-means (Darken and Moody, 1990) training algorithm.

2. RBF NEURAL NETWORKS: AN OVERVIEW

2.1 RBF Network Topology

An RBF neural network can be considered as a special three-layer network which is linear with respect to the output parameters after fixing all the radial basis function centers and nonlinearities $f(\cdot)$ in the hidden layer. Thus, the hidden layer performs a nonlinear transformation and maps the input space onto a new space. The output layer then implements a linear combiner on this space, where the only adjustable parameters are the weights of the linear combiner.

The basic structure of a typical RBF network with *N* inputs and *M* outputs is shown in Fig. 1. The input nodes pass the input values to the connection arcs and the first layer connections are not weighted. Thus, each internal unit receives all the input values, unaltered. The internal units form a single layer of *L* receptive fields, which have localized response functions in the input space. These internal (hidden) nodes are the radial basis function units. The hidden node responses are weighted and the output nodes are simple summations of the weighted responses:

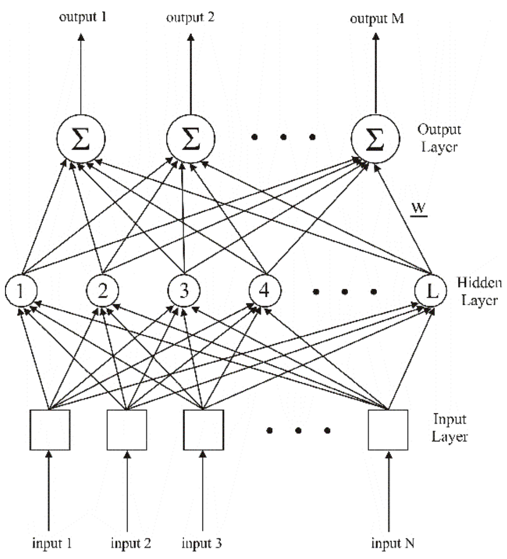


Fig. 1. The typical RBF Neural Network Architecture.

$$\hat{\mathbf{y}}^T = [\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_M] = [g_1(\mathbf{x}), g_2(\mathbf{x}), \ldots, g_M(\mathbf{x})] \quad (1)$$

with:

$$\hat{y}_m = g_m(\mathbf{x}) = \sum_{l=1}^{L} w_{m,l} z_l$$
$$= \sum_{l=1}^{L} w_{m,l} f\left(\left\|\mathbf{x} - \mathbf{x}^*(l)\right\|_2\right), \quad m = 1, \ldots, M \quad (2)$$

where z_l is the activation of the *l*th unit in the hidden layer given the *N*-dimensional input vector **x**, *f* is the radial basis function, $\mathbf{x}^*(l)$ is the *N*-dimensional center of the *l*th unit and $\mathbf{w}_m^T = [w_{m,1}, w_{m,2}, \ldots, w_{m,L}]$ is the vector of weights, which multiply the hidden node responses in order to calculate the *m*th output of the network.

The output function of the hidden nodes is a radially symmetric function. A typical choice, which was also used in this work, is the Gaussian function:

$$f(v) = \exp\left(-\frac{v^2}{\sigma^2}\right) \quad (3)$$

where σ is the width of the node.

2.2 RBF Models of Dynamical Systems

In a dynamic RBF neural network model, at time point *i*, past values of the process input and output variables constitute the input layer to the network. Therefore, an RBF network can be considered as a special type of Nonlinear Auto Regressive with eXogenous inputs (NARX) models. Given a process with *R* process inputs and *M* process outputs, at time point *i* the input vector to the network can be written as follows:

$$\begin{aligned}\mathbf{x}^T(i) &= [x_1(i),\, x_2(i), \ldots, x_N(i)] \\ &= [u_1(i-1), u_1(i-2), \ldots, u_R(i-p_R+1), u_R(i-p_R), \ldots, \\ &\quad y_1(i-1), y_1(i-2), \ldots, y_M(i-q_M+1), y_M(i-q_M)]\end{aligned} \tag{4}$$

where p_r is the number of past values of process input u_r (r=1,….,R) and q_m is the number of past values of process output y_m (m=1,…,M). The output of the network contains the estimated values of the current process outputs and is given by:

$$\begin{aligned}\hat{\mathbf{y}}^T(i) &= [\hat{y}_1(i), \hat{y}_2(i), \ldots, \hat{y}_M(i)] \\ &= [g_1(\mathbf{x}(i)), g_2(\mathbf{x}(i)), \ldots, g_M(\mathbf{x}(i))]\end{aligned} \tag{5}$$

3. AN RBF IDENTIFICATION METHOD BASED ON THE SUBTRACTIVE CLUSTERING TECHNIQUE

The formulation of the training algorithm involves a set of input-output pairs ($\mathbf{x}(i)$, $\mathbf{y}(i)$), i=1, ..., K, where $\mathbf{x}(i)$ is the input vector, $\mathbf{y}(i)$ is the corresponding target or desired output vector and K is the number of training examples. The set of input-output examples is the information base, which is used to determine the values of the unknown parameters, i.e the hidden node centers and the connection weights between the hidden and the output layer. Determination of the hidden node centers is the most crucial step in the development of a successful RBF neural network model. The innovation in this work is the new algorithm which is proposed for selecting those centers, based on the subtractive clustering method. The rest of the network parameters are calculated using standard methods, which will be described briefly in the sequel.

3.1 Using the subtractive clustering method to determine the hidden node centers

The SC method considers each data point as a potential hidden node center. A measure of the potential of each data point is defined as a function of the Euclidean distances to all other input data points:

$$P(i) = \sum_{j=1}^{K} \exp\left(-a\left\|\mathbf{x}(i) - \mathbf{x}(j)\right\|_2^2\right) \quad i=1,\ldots,K \tag{6}$$

where α is a design parameter. Obviously, the potential of a data point is high when it is surrounded by many neighboring data. Given the above definition and provided that proper values have been assigned to the design parameters α, β and ε, the SC algorithm is defined as follows:

Step 1) For i=1,…,K calculate the potential values $P(i)$.

Step 2) Set L=1 and select the data point with the highest potential value as the first hidden node center. Let $\mathbf{x}^*(1)$ be the location of that point and $P^*(1)$ its potential value.

Step 3) Revise the potential of each data point i=1…K by the formula:

$$P(i) = P(i) - P^*(1)\exp\left(-\beta\left\|\mathbf{x}(i) - \mathbf{x}^*(1)\right\|_2^2\right) \tag{7}$$

Step 4) Set L=L+1 and select the data point with the highest potential value as the next center. Let $\mathbf{x}^*(L)$ be the location of the new center and $P^*(L)$ its potential value.

Step 5) Revise the potential of each data point i=1…K by the formula:

$$P(i) = P(i) - P^*(L)\exp\left(-\beta\left\|\mathbf{x}(i) - \mathbf{x}^*(L)\right\|_2^2\right) \tag{8}$$

Step 6) If the inequality

$$P^*(L) < \varepsilon P^*(1) \tag{9}$$

is true, stop the algorithm, else return to step 4.

Remark 1: The parameters α and β can be considered as radiuses, using the following equalities:

$$\alpha = \frac{4}{r_\alpha^2}, \beta = \frac{4}{r_\beta^2} \tag{10}$$

Each of the above radiuses defines a neighbourhood in the N-dimensional space. For example in Eq. (6) the neighborhood is an N-dimensional sphere, where $\mathbf{x}(i)$ is the center and r_α is the radius, i.e. it contains all the points in the N-dimensional space whose distances from $\mathbf{x}(i)$ are less than or equal to r_α. The data points that are inside this neighbourhood have a considerable effect on the potential of $\mathbf{x}(i)$. On the contrary, the data which are outside of the N-dimensional sphere, have little influence on the potential of $\mathbf{x}(i)$.

Remark 2: In order to avoid the selection of hidden node centers that are close the one to the other, β is chosen to be less than α. In this way after selecting a training point as a hidden node center, the potentials of the close training examples are greatly reduced and so are the possibilities for selecting one of these points as a new hidden node center.

Remark 3: The choice of ε is very important for the produced network structure. If ε is selected to be very small, a large number of hidden node centers will be generated. On the contrary, a large value of ε will lead to a small network structure.

Remark 4: The proposed methodology is an alternative to the standard RBF training techniques. One of these techniques is the MaqQueen k-means algorithm, which is briefly described next:

Initially, the centers of the hidden nodes are assigned to different, randomly chosen, data points. During an iterative procedure, for each exemplar (training example) $\mathbf{x}(k)$ the algorithm determines the closest cluster vector $\mathbf{x}^*_{closest}$ in a Euclidean sense, and modifies it according to:

$$\Delta\mathbf{x}^*_{closest} = 1/N_{closest} \cdot (\mathbf{x}(k) - \mathbf{x}^*_{closest}) \qquad (11)$$

where $N_{closest}$ is the total number of data points that have been assigned to $\mathbf{x}^*_{closest}$. None of the other centers is modified by this exemplar. Several passes of the training examples are needed until the algorithm converges (Darken and Moody, 1990).

Remark 5: The two methods will be compared in a later section through the application to dynamic data from a chemical reactor. However, an important observation can be made at this point and concerns the speed of convergence of the two algorithms. Since the most time consuming part in both methods is the calculation of Euclidean distances, the two algorithms can be compared as far as speed is concerned, by counting the required distance calculations. It can easily be verified that the after initialization (Step 1), the proposed method requires only *KL* distance calculations, in contrast to the *k*-means method, which needs *KLI* calculations. In the two expressions, *K* is the number of training examples, *L* is the number of hidden nodes and *I* is the number of iterations in the *k*-means algorithm.

3.2 Selection of the unit widths and the interconnection links

After the receptive field centers have been determined, the receptive "widths" of each hidden node are calculated using the *P*-nearest neighbor heuristic:

$$\sigma(l) = \left(\frac{1}{P} \sum_{j=1}^{P} \left\| \mathbf{x}^*(l) - \mathbf{x}^*(j) \right\|^2 \right)^{1/2} \qquad (12)$$

where $\mathbf{x}^*(j)$ are the *P*-nearest centers to $\mathbf{x}^*(l)$. The goal in the selection of the unit widths is to activate more than one units for any training example. This is achieved by choosing the value of the scaling factor $\sigma(l)$ for each hidden unit to be greater than the distance to the nearest unit center.

The output weights are then calculated by linear least squares regression, since the output nodes are simple summation units. Regression correlates the desired network outputs with the hidden nodes responses, so each training example is passed through the hidden layer of the network to produce a corresponding hidden node activation vector:

$$\mathbf{z}^T(i) = [z_1(i), z_2(i), \ldots, z_L(i)] \quad i = 1, \ldots, K \qquad (13)$$

The objective is to find the set of weights that minimizes the squared norm of the residuals:

$$\begin{aligned} E(\mathbf{W}) &= \frac{1}{2} \sum_{i=1}^{K} \left\| \mathbf{y}(i) - \hat{\mathbf{y}}(i) \right\|^2 \\ &= \frac{1}{2} \sum_{i=1}^{K} \left\| \mathbf{y}(i) - \mathbf{W} \cdot \mathbf{z}(i) \right\|^2 \end{aligned} \qquad (14)$$

where $\mathbf{W}$ is the matrix of output weights. If the following notation is introduced:

$$\mathbf{Z} = [\mathbf{z}(1) \quad \mathbf{z}(2) \quad \ldots \quad \mathbf{z}(K)] \qquad (15)$$

and

$$\mathbf{Y} = [\mathbf{y}(1) \quad \mathbf{y}(2) \quad \ldots \quad \mathbf{y}(K)] \qquad (16)$$

the solution can be found by linear regression using the pseudo-inverse of $\mathbf{Z}$ and is given by:

$$\mathbf{W} = \mathbf{Y} \cdot \mathbf{Z}^T (\mathbf{Z} \cdot \mathbf{Z}^T)^{-1} \qquad (17)$$

4. RESULTS

The proposed methodology was used to model a Multi Input – Multi Output (MIMO) non-isothermal Continuous Stirred Tank Reactor (CSTR), where the following exothermal irreversible reaction between sodium thiosulfate and hydrogen peroxide is taking place:

$$2Na_2S_2O_3 + 4H_20 \rightarrow Na_2S_3O_6 + Na_2SO_4 + 4H_2O$$

This process is characterized by the following dynamic equations (Kazantzis and Kravaris, 2000):

$$\begin{aligned} \frac{dC_A}{dt} &= \frac{F}{V}\left(C_{A,in} - C_A\right) - 2k_o \exp\left(-\frac{E}{RT}\right) C_A^2 \\ \frac{dT}{dt} &= \frac{F}{V}\left(T_{in} - T\right) + 2\frac{(-\Delta H)_R}{\rho c_p} k_o \exp\left(-\frac{E}{RT}\right) C_A^2 - \frac{UA}{V\rho c_p}(T - T_j) \end{aligned} \qquad (18)$$

where V is the volume of the CSTR; $(\Delta H)_R$ is the heat of the reaction; and $-E/R$, k_o, c_p, ρ are constants of the reaction and the reactants. The variables F, $C_{A,in}$, T_{in} and T_j are considered as inputs to the system, while C_A and T are the outputs. F is the flow rate into the reactor, $C_{A,in}$ is the inlet concentration of the reactant $Na_2S_2O_3$, C_A is the concentration of $Na_2S_2O_3$ inside the reactor, T_{in} is the inlet temperature, T is the temperature inside the reactor and T_j is the temperature of the coolant. The values of the process parameters are shown in Table 1.

Using a sampling period of 1s, a set of 2000 data examples was created by adding Random Number signals to the steady state values of the input variables, which are shown in Table 2. The first 1000 points were used for training the network and the rest of the data were used for validation. The variables were scaled between –1 and 1 so that all the input and output values were of the same order of magnitude. The input vector to the RBF network consisted of twenty past values of each process input, a total of 80 variables. No values of the process outputs (concentration of $Na_2S_2O_3$ and temperature) were used as additional inputs to the network.

Table 1. Process parameter values in the CSTR example

Process Parameters	**Values**
V	100 l
UA	20000 J/s*K
P	1000 g/l
C_p	4.2 J/g*K
$-(\Delta H)_R$	596619 J/mol
k_o	6.85E+11 l/s*mol
E	76534.704 J/mol
R	8.314 J/mol*K

Table 2. Steady state values of the input variables in the CSTR example

Input variable	**Steady State**
F	20 l/s
T_i	275 K
T_j	250 K
$C_{A,in}$	1 mol/l

The proposed algorithm was applied to develop a number of neural network dynamical models, using different values for the parameter ε. The parameters α and β were kept constant and equal to 0.8 and 0.4 respectively. The rest of the neural network parameters were determined using the standard techniques. The parameter P in the nearest neighbour heuristic was set equal to one fourth of the number of hidden nodes and rounded to the closest integer. Each value of ε produced a different neural network structure, which for comparison purposes was retrained using the standard MaqQueen *k*-means algorithm.

The results are summarized in Tables 3 and 4 (for the subtractive clustering and the MacQueen *k*-means algorithms respectively), where the training times, numbers of hidden nodes and Sum of Squared Errors (SSE) for the set of validation data are shown.

Table 3. Different runs of the proposed training algorithm in the CSTR example

ε	# of Hidden Nodes	P	SSE Validation		CPU Time (sec)
			C_A	T	
0.05	397	99	0.0059	541.0	93
0.1	315	79	0.0064	590.3	77
0.2	204	51	0.0073	680.8	60
0.3	116	29	0.0089	1045.2	48
0.4	69	17	0.0191	2320.4	42

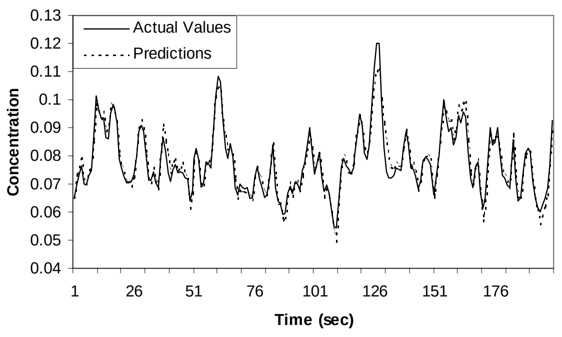
Table 4. Different runs of the MaqQueen *k* - means training algorithm in the CSTR example

# of Hidden Nodes	P	SSE Validation		CPU Time (sec)
		C_A	T	
397	99	0.0067	965.8	1208
315	79	0.0068	936.8	991
204	51	0.012	1176.3	703
116	29	0.0159	2290.0	334
69	17	0.0208	3491.5	204

The networks produced by the proposed algorithm proved to be very successful in predicting the dynamic behavior of the concentration of $Na_2S_2O_3$ and the temperature inside the reactor when the second subset of data was used for validation. The training algorithm not only produced very accurate models, but also completed all the calculations in small computational CPU time using a PC with a 1400 MHz Pentium IV processor. This result verifies the observation made in remark 5.

The predictions of the neural network structure consisting of 204 hidden nodes along with the actual data for the concentration of $Na_2S_2O_3$ and the temperature inside the reactor are shown in Figs. 2 and 3 for a sample of 200 validation data points.

Fig. 2 refers to the network trained with the proposed algorithm, while Fig. 3 shows the predictions of the network trained with the MaqQueen *k*-means method. The results show that the proposed method can produce more accurate predictions, but most importantly, train the network in an order of magnitude less time than the standard technique.



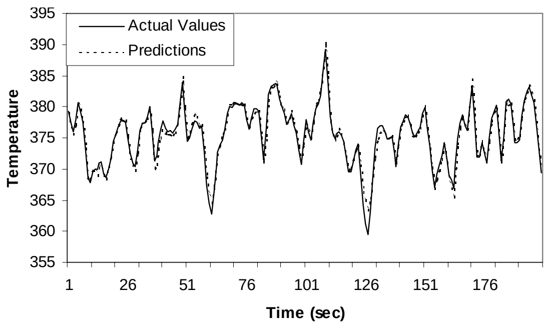
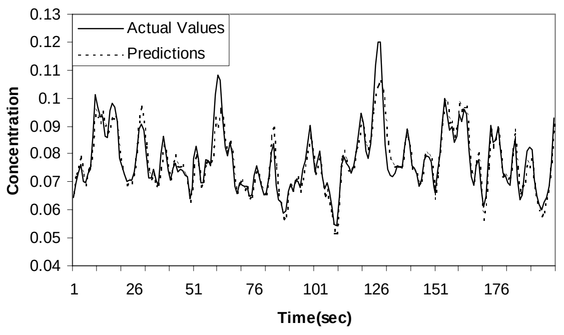


Fig. 2. Actual values and neural network predictions for concentration and temperature using the subtractive clustering training method.



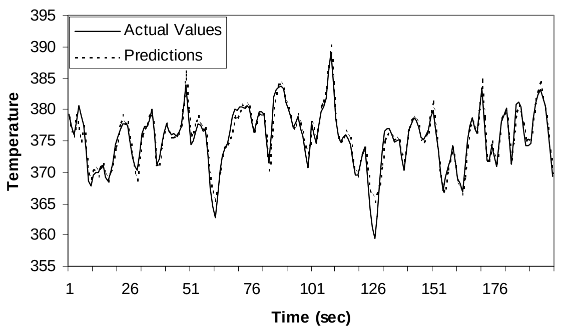


Fig 3. Actual values and neural network predictions for concentration and temperature using the MacQueen k – means algorithm.

5. CONCLUSION

In this work a new algorithm was proposed for training RBF neural networks. The algorithm first determines the centers of the nonlinear internal units in a very fast manner using the subtractive clustering technique. In a second stage the algorithm calculates the widths of the Gaussian functions and the connection weights between the hidden and the output layers. The proposed methodology has a number of advantages compared to other learning techniques:

- It is characterized by very fast computational times, since it requires only one pass of the training examples.
- It does not depend on an initial random selection of the hidden nodes, so for the same neural network structure, the same network parameters are always obtained.
- Due to the high speed of convergence, it gives the opportunity to try a number of design parameters α, β, ε, which will lead to the selection of a proper network structure, i.e. a proper number of hidden layer nodes.

The ability of the proposed algorithm to approximate the dynamics of nonlinear processes was tested, using simulated data generated from a MIMO CSTR model. The results show that the methodology can be very successful when it is applied to system identification problems. The methodology was compared to a standard learning technique, which is based on the MaqQueen k-means clustering. In both examples, the proposed technique surpassed the standard learning algorithm in speed and accuracy of predictions.

REFERENCES

Bhartiya, S. and J. R. Whiteley (2001). Factorized Approach to Nonlinear MPC Using a Radial Basis Function Model**.** *AIChE Journal,* **47**, 358-368.

Billings, S.A. and G. L. Zheng (1995). Radial Basis Function Network Configuration Using Genetic Algorithms. *Neural Networks*, **8**, 877-890.

Chiu, S.L. (1994). Fuzzy Model Identification Based on Cluster Estimation. *Journal of Intelligent and Fuzzy Systems*, **2**, 267-278.

Darken, C. and J. Moody (1990). Fast Adaptive K-Means Clustering: Some Empirical Results. *IEEE INNS Int. J. Conf. On Neural Networks; Proceedings,* **2**, 233-238.

Houlcomb, T. and M. Morari (1991). Local Training for Radial Basis Function Networks: Towards Solving the Hidden Unit Problem. *Amer. Control Conf.; Proceedings,* 2331-2336.

Kazantzis, N. and C. Kravaris (2000). Synthesis of state feedback regulators for nonlinear processes. *Chemical Engineering Science*, **55**, 3437-3449.

Leonard, J.A. and M.A. Kramer (1991). Radial Basis Function Networks for Classifying Process Faults. *IEEE Control Systems*, **11**, 31-38.

Li, Y., N. Sundararajan and P. Saratchandran (2001). Neuro-controller design for nonlinear fighter aircraft maneuver using fully tuned RBF networks. *Automatica,* **37***,* 1293-1301.

Moody, J. and C. Darken (1988). Learning with Localized Receptive Fields. In: *Proceedings of the 1988 Connectionist Models Summer School* (Touretzky, Hinton and Sejnowski (Eds.)), 133-143. Morgan-Kaufmann, San Mateo, CA.

Moody, J. and C. Darken (1989). Fast Learning in Networks of Locally-Tuned Processing Units. *Neural Computation*, **1**, 281-294 .

Musavi, M. T., W. Ahmed, K. H. Chan, K. B. Faris and D. M. Hummels (1992). On the training of Radial Basis Function Classifiers. *Neural Networks*, **5**, 595-603.

Shin, Miyoung and Amrit L. Goel (2000). Empirical data modeling in software engineering using radial basis functions. *IEEE Transactions on Software Engineering,* **26***,* 567-576.