Two-stage RBF network construction based on particle swarm optimization

Deng, J., Li, K., Irwin, G., & Fei, M. (2011). Two-stage RBF network construction based on particle swarm optimization. Transactions of the Institute of Measurement and Control, 33, 0-0. DOI: 10.1177/0142331211403795

Published in:
Transactions of the Institute of Measurement and Control

Document Version:
Early version, also known as pre-print

Article

TRANSACTIONS OF THE INSTITUTE OF MEASUREMENT & CONTROL

Transactions of the Institute of Measurement and Control
0(0) 1–9

DOI: 10.1177/0142331211403795
tim.sagepub.com

SAGE

Two-stage RBF network construction based on particle swarm optimization

Jing Deng[1], Kang Li[1], George W Irwin[1] and Minrui Fei[2]

Abstract

The conventional radial basis function (RBF) network optimization methods, such as orthogonal least squares or the two-stage selection, can produce a sparse network with satisfactory generalization capability. However, the RBF width, as a nonlinear parameter in the network, is not easy to determine. In the aforementioned methods, the width is always pre-determined, either by trial-and-error, or generated randomly. Furthermore, all hidden nodes share the same RBF width. This will inevitably reduce the network performance, and more RBF centres may then be needed to meet a desired modelling specification. In this paper we investigate a new two-stage construction algorithm for RBF networks. It utilizes the particle swarm optimization method to search for the optimal RBF centres and their associated widths. Although the new method needs more computation than conventional approaches, it can greatly reduce the model size and improve model generalization performance. The effectiveness of the proposed technique is confirmed by two numerical simulation examples.

Keywords

Nonlinear modelling, particle swarm optimization, radial basis function, two-stage selection

Introduction

Owing to the simple topological structure and universal approximation ability, the radial basis function (RBF) network has been widely used in data mining, pattern recognition, signal processing, system modelling and control (Chen and Billings, 1992). The main issues involved in constructing RBF networks are the optimization of basis functions and the estimation of output layer weights.

The conventional strategy is to handle these issues separately by selecting the RBF centres using unsupervised clustering (Sutanto et al., 1997), optimizing the basis function parameters by gradient-based searches, and estimating the output weights using a least-squares method (Elanayar and Shin, 1994). However, the network parameters are interdependent, and a better strategy is to optimize them simultaneously (McLoone et al., 1998; Li et al., 2006).

For a single hidden layer RBF neural network, it is possible to formulate its weight estimation as linear in the parameters (LITP). A forward selection technique, such as orthogonal least squares (OLS) (Chen et al., 1991; Hong et al., 2008), and the fast recursive algorithm (FRA) (Li et al., 2005), can then be applied to produce a sparse network with satisfactory accuracy. To further improve the generalization capability and prevent over-fitting with noisy data, leave-one-out (LOO) cross-validation and a Bayesian learning framework have also been introduced to these algorithms (Chen et al., 2004; Chen, 2006; Deng et al., 2010b).

Although a forward construction scheme can efficiently build a sparse model from a large candidate term pool, the final model is not optimal (Sherstinsky and Picard, 1996), since the calculation of the contribution of any RBF centre depends on previously selected ones (Li et al., 2006). To reduce this constraint, genetic search has been suggested to refine the model structure (Mao and Billings, 1997), but at the expense of a very high computational complexity. Li et al. (2006) recently proposed a two-stage stepwise construction method which combined both forward and backward construction. This retains the computational efficiency of the FRA while improving the model compactness.

In the two-stage construction algorithm of Li et al. (2006), an initial network is first constructed by the FRA where the contribution of a particular centre of interest is measured by its reduction in an appropriate cost function. The significance of each selected centre is then reviewed at a second network refinement stage, and insignificant centres are replaced. Specifically, if the contribution of a previously selected centre is less than any from the candidate pool, it will be replaced with this one. In this way, the cost function can be further reduced without increasing the network size. This check process is repeated until no insignificant hidden nodes

[1]School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, Belfast, UK
[2]Shanghai Key Laboratory of Power Station Automation Technology, School of Mechatronical Engineering and Automation, Shanghai University, China

Corresponding author:
Kang Li, School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, Belfast BT9 5AH, UK
Email: k.li@qub.ac.uk

remain in the trained network, resulting in an optimized network structure with improved performance.

The main outstanding issues with both the forward and two-stage methods is that the width of the RBF needs to be pre-determined, and the placement of the RBF centres is also limited to the data samples. Further, the distance between input data and the hidden nodes is normalized by the same RBF width. As a result, the network model obtained is not optimal, and more RBF nodes are often needed to achieve a satisfactory accuracy. To overcome these problems, gradient-based methods have been introduced into sub-set selection (Peng et al., 2006; Li et al., 2009). These hybrid methods can reduce the network size and improve the network performance, but the computation complexity is inevitably increased simultaneously.

Recently swarm intelligence has been proposed as a robust and efficient technique for solving nonlinear optimization problems (Blum and Merkle, 2008). Unlike conventional calculus-based methods, swarm intelligence introduces a large number of unsophisticated entities that cooperate to exhibit a global behaviour. The inspiration for this comes from the observations of social insects such as ants, bees and birds. This shows that although a single member of these societies may be an unsophisticated individual, collectively they are able to achieve complex tasks by working in cooperation. Particle swarm optimization (PSO) is a popular version of swarm intelligence that was originally proposed in 1995 (Kennedy and Eberhart, 1995; Eberhart and Shi, 2000). It has been widely applied to optimization problems ranging from scheduling, neural network training and task assignment, to highly specialized applications (Blum and Merkle, 2008). The popular OLS technique has also been revised to utilize PSO to optimize the nonlinear parameters (Chen et al., 2009, 2010).

In this paper, the PSO is effectively integrated with our two-stage selection (TSS) algorithm, leading to a new construction scheme for RBF networks. The method involves continuous optimization of both RBF centres and widths, and a discrete optimization of the network structure. Unlike the original TSS technique which selects the centres from a candidate pool, the new algorithm randomly generates some initial points (particles in the swarm) from the training data as the starting point. PSO is then adopted to optimize these parameters according to their contributions to the cost function. The best global solution found by the entire swarm becomes the new RBF centre and is added to the network. This procedure continues until a satisfactory network has been constructed. Then a second refinement stage is implemented to remove any constraint due to the order in which the centres were selected at the previous stage. The introduction of a residual matrix through the FRA helps to greatly reduce the calculation involved. The results from two simulation experiments are included to confirm the superiority of the approach.

This paper is organized as follows. Section 2 contains the problem formulation for both the RBF network construction and the PSO updating strategy. The new two-stage construction algorithm is introduced in Section 3. Section 4 provides two examples to illustrate the effectiveness of the proposed method, while Section 5 concludes this paper with some discussions on the limitations.

Preliminaries

Radial basis function neural networks

Consider a general RBF neural network with m inputs, n hidden nodes and a scalar output that can be expressed by the LITP model:

$$y(t) = \sum_{k=1}^{n} \theta_k \varphi_k(\mathbf{x}(t); \mathbf{c}_k; \mathbf{\Sigma}_k) + \varepsilon(t), \tag{1}$$

where $y(t)$ is the actual output at sample time t, $\mathbf{x}(t) \in \Re^m$ is the input vector, $\varphi_k(\mathbf{x}(t); \mathbf{c}_k; \mathbf{\Sigma}_k)$ denotes the RBF (e.g. a Gaussian basis function), $\mathbf{c}_k \in \Re^m$ is the centre vector and $\mathbf{\Sigma}_k$ is a diagonal matrix known as the norm matrix, which has the form

$$\mathbf{\Sigma}_k = \begin{bmatrix} 1/\sigma_{k1}^2 & 0 & 0 & 0 \\ 0 & 1/\sigma_{k2}^2 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 1/\sigma_{km}^2 \end{bmatrix}. \tag{2}$$

Finally, θ_k represents the output layer weight for each RBF node, and $\varepsilon(t)$ is the network error at sample time t.

If a set of N data samples $\{\mathbf{x}(t), y(t)\}_{t=1}^{N}$ is used for network training, Equation (1) can be written in matrix form as

$$\mathbf{y} = \mathbf{\Phi}\boldsymbol{\theta} + \mathbf{e}, \tag{3}$$

where $\mathbf{\Phi} = [\phi_1, \ldots, \phi_n] \in \Re^{N \times n}$ is known as the regression matrix with column vectors $\phi_i = [\varphi_i(\mathbf{x}(1)), \ldots, \varphi_i(\mathbf{x}(N))]^{\mathrm{T}}$, $i = 1, \ldots, n$, $\mathbf{y} = [y(1), \ldots, y(N)]^{\mathrm{T}} \in \Re^N$ is the actual output vector, $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_n]^{\mathrm{T}} \in \Re^n$ and $\mathbf{e} = [\varepsilon(1), \ldots, \varepsilon(N)]^{\mathrm{T}} \in \Re^N$ denotes the network residual vector.

The network training aims to build a parsimonious representation based on optimized RBF centres, width parameters and output layer weights $\boldsymbol{\theta}$ with respect to some appropriate cost function, e.g. sum-squared error (SSE). In this paper the RBF network is built using stepwise construction such that one RBF centre is optimized and added to the network each time. Suppose that k ($k \le n$, $n \ll N$) centres have been selected for inclusion in the network with their corresponding columns in the regression matrix denoted by $\mathbf{p}_1, \ldots, \mathbf{p}_k$. The resultant network can then be represented as

$$\mathbf{y} = \mathbf{P}_k \boldsymbol{\theta}_k + \mathbf{e} \tag{4}$$

where $\mathbf{P}_k = [\mathbf{p}_1, \ldots, \mathbf{p}_k]$.

Outline of particle swarm optimization

In PSO each particle in the swarm represents a possible solution which moves through the problem search space seeking an optimal or satisfactory point. The position of each particle is adjusted according to its velocity and the difference between its current position, the best one it has found so far, and the best position to date found by its neighbours (Blum and Merkle, 2008).

Suppose that $\mathbf{x}_i$ denotes the ith particle in the swarm, $\mathbf{v}_i$ represents its velocity, $\mathbf{u}_i$ is its best position to date, while $\mathbf{u}_g$ denotes the best position from the entire swarm. In inertia-weighted PSO (Eberhart and Shi, 2000; Clerc and Kennedy, 2002), $\mathbf{v}_i$ and $\mathbf{x}_i$ are updated as

$$\mathbf{v}_i \leftarrow w\mathbf{v}_i + c_1\mathbf{r}_1(\mathbf{u}_i - \mathbf{x}_i) + c_2\mathbf{r}_2(\mathbf{u}_g - \mathbf{x}_i), \tag{5}$$

$$\mathbf{x}_i \leftarrow \mathbf{x}_i + \mathbf{v}_i, \tag{6}$$

where w is the inertia weight used to scale the previous velocity term, c_1 and c_2 are acceleration coefficients, and $\mathbf{r}_1$ and $\mathbf{r}_2$ are two vectors comprising random values uniformly generated between 0 and 1. As shown in Equation (5), the velocity of each particle is determined by three parts, the momentum, the cognitive information and the social information. The momentum term $w\mathbf{v}_i$ carries the particle in the direction it has travelled so far with the inertia weight w being used to control the influence of the previous velocity value on the new one. For $w > 1$, the particles diverge eventually beyond the boundaries of the search space. For $w < 0$, the velocity decreases continuously causing the particles to converge. The cognitive part $c_1\mathbf{r}_1(\mathbf{u}_i - \mathbf{x}_i)$ describes the tendency of the particle to return to the best position it has visited so far, while the social part $c_2\mathbf{r}_2(\mathbf{u}_g - \mathbf{x}_i)$ denotes its tendency to move towards the best position from amongst the entire swarm. The acceleration coefficients c_1 and c_2 can be fixed or varied from 0.5 to 2.5 during the iterative procedure (Rajakarunakaran et al., 2008):

$$c_1 = (0.5 - 2.5)l/l_{\max} + 2.5, \tag{7}$$

$$c_2 = (2.5 - 0.5)l/l_{\max} + 0.5, \tag{8}$$

where l is the iteration index. This improves the updating quality as a wider search range is set at the beginning to avoid a local minimum, and quick convergence is guaranteed towards the end.

In order to ensure that each updated particle is still inside the search space, it is essential to check both its position and the velocity before calculating the related cost function. Suppose that the search space of a particle position is $[\mathbf{x}_{\min}, \mathbf{x}_{\max}]$, the appropriate rule is given by

$$\textbf{if } \mathbf{x}_i(j) > \mathbf{x}_{\max}(j), \textbf{ then } \mathbf{x}_i(j) = \mathbf{x}_{\max}(j), \tag{9}$$

$$\textbf{if } \mathbf{x}_i(j) < \mathbf{x}_{\min}(j), \textbf{ then } \mathbf{x}_i(j) = \mathbf{x}_{\min}(j), \quad \text{for } i = 1,\ldots, s; \text{ and } j = 1,\ldots, m, \tag{10}$$

where i is the particle index and j is the index of an element in the input vector $\mathbf{x}_i$. For velocity, the maximum value is normally obtained from the solution search space and is given by

$$\mathbf{v}_{\max} = \frac{1}{2}(\mathbf{x}_{\max} - \mathbf{x}_{\min}), \tag{11}$$

where the search space is defined as $[-\mathbf{v}_{\max}, \mathbf{v}_{\max}]$. Similarly, the rule for the velocity is

$$\textbf{if } \mathbf{v}_i(j) > \mathbf{v}_{\max}(j), \textbf{ then } \mathbf{v}_i(j) = \mathbf{v}_{\max}(j) \tag{12}$$

$$\textbf{if } \mathbf{v}_i(j) < -\mathbf{v}_{\max}(j), \textbf{ then } \mathbf{v}_i(j) = -\mathbf{v}_{\max}(j) \tag{13}$$

$$\textbf{if } \mathbf{v}_i(j) \to \pm 0, \textbf{ then } \mathbf{v}_i(j) = \pm c_v r_v \mathbf{v}_{\max}(j) \quad \text{for } i = 1,\ldots, s; \text{ and } j = 1,\ldots, m. \tag{14}$$

where c_v is a small weight normally set to 0.1 and r_v is a random vector uniformly generated from [0, 1]. Although PSO has been widely used, the analysis of the convergence behaviour of a swarm of multiple interactive particles is still problematical (Clerc and Kennedy, 2002; Trelea, 2003).

New two-stage RBF network construction approach

The two-stage network construction scheme includes a forward centre selection stage and a second network refinement stage. The significance of a RBF centre is measured based on its contribution to the cost function given by

$$J(\mathbf{c},\sigma,\boldsymbol{\theta}_k) = (\mathbf{y} - \mathbf{P}_k\hat{\boldsymbol{\theta}}_k)^{\mathrm{T}}(\mathbf{y} - \mathbf{P}_k\hat{\boldsymbol{\theta}}_k). \tag{15}$$

Here it is proposed that the nonlinear parameters $\mathbf{c}$ and σ are optimized by PSO, while the output layer weights $\boldsymbol{\theta}_k$ are estimated using least squares as

$$\hat{\boldsymbol{\theta}}_k = (\mathbf{P}_k^{\mathrm{T}}\mathbf{P}_k)^{-1}\mathbf{P}_k^{\mathrm{T}}\mathbf{y}, \tag{16}$$

where $\hat{\boldsymbol{\theta}}_k$ are the estimated output layer weights. Equation (16) is not normally used in practice because the noise on the data usually causes the matrix $\mathbf{P}_k$ to be ill-conditioned, and the estimated $\hat{\boldsymbol{\theta}}_k$ from (16) can be inaccurate. The TSS method has proved to be an effective and efficient method for overcoming this problem.

First stage: PSO-assisted forward selection

This stage is similar to the FRA. A recursive matrix $\mathbf{M}_k$ and a residual matrix $\mathbf{R}_k$ are defined to simplify the computation. They are given as

$$\mathbf{M}_k \triangleq \mathbf{P}_k^{\mathrm{T}}\mathbf{P}_k, \quad k = 1,\ldots,n, \tag{17}$$

$$\mathbf{R}_k \triangleq I - \mathbf{P}_k\mathbf{M}_k^{-1}\mathbf{P}_k^{\mathrm{T}}, \quad \mathbf{R}_0 \triangleq I, \tag{18}$$

where $\mathbf{P}_k \in \Re^{N\times k}$ contains the first k columns of the regression matrix $\mathbf{P}$ in (4). According to Li et al. (2005, 2006), the matrix terms $\mathbf{R}_k$, $k = 0,\ldots, n-1$ have the following attractive properties:

$$\mathbf{R}_{k+1} = \mathbf{R}_k - \frac{\mathbf{R}_k\mathbf{p}_{k+1}\mathbf{p}_{k+1}^{T}\mathbf{R}_k^{\mathrm{T}}}{\mathbf{p}_{k+1}^{\mathrm{T}}\mathbf{R}_k\mathbf{p}_{k+1}}, \quad k = 0,1,\ldots,n-1, \tag{19}$$

$$\mathbf{R}_k^{\mathrm{T}} = \mathbf{R}_k; \quad (\mathbf{R}_k)^2 = \mathbf{R}_k, \quad k = 0,1,\ldots,n, \tag{20}$$

$$\mathbf{R}_i\mathbf{R}_j = \mathbf{R}_j\mathbf{R}_i = \mathbf{R}_i, \quad i \geq j; \ i,j = 0,1,\ldots,n, \tag{21}$$

$$\mathbf{R}_k\phi = \begin{cases} \mathbf{0}, & \text{rank}([\mathbf{P}_k,\phi_j]) = k \\ \phi_j^{(k)} \neq \mathbf{0}, & \text{rank}([\mathbf{P}_k,\phi_j]) = k+1 \end{cases}, \quad j = 0,1,\ldots,n, \tag{22}$$

$$\mathbf{R}_{1,\ldots,p,\ldots,q,\ldots,k} = \mathbf{R}_{1,\ldots,q,\ldots,p,\ldots,k}, \quad p,q \leq k. \tag{23}$$

Equation (23) means that any change in the selection order of the regressor terms $\mathbf{p}_1,\ldots,\mathbf{p}_k$ does not change the residual matrices $\mathbf{R}_k$. This property will help to reduce the computation effort in the second stage. The cost function in (15) can now be rewritten as

$$J(\mathbf{P}_k) = \mathbf{y}^{\mathrm{T}}\mathbf{R}_k\mathbf{y}. \tag{24}$$

In this forward stage, the RBF centres are optimized one at a time, and given by the global best position obtained from entire swarm. Suppose at the kth step, one more centre $\mathbf{p}_{k+1}$ is to be added. The net contribution of $\mathbf{p}_{k+1}$ to the cost function can then be calculated as:

$$\begin{aligned} \Delta J_{k+1}(\mathbf{P}_k,\mathbf{p}_{k+1}) &= \mathbf{y}^{\mathrm{T}}(\mathbf{R}_k - \mathbf{R}_{k+1})\mathbf{y} \\ &= \frac{\mathbf{y}^{\mathrm{T}}\mathbf{R}_k\mathbf{p}_{k+1}\mathbf{p}_{k+1}^{\mathrm{T}}\mathbf{R}_k\mathbf{y}}{\mathbf{p}_{k+1}^{\mathrm{T}}\mathbf{R}_k\mathbf{p}_{k+1}} \\ &= \frac{(\mathbf{y}^{\mathrm{T}}\mathbf{p}_{k+1}^{(k)})^2}{\mathbf{p}_{k+1}^{\mathrm{T}}\mathbf{p}_{k+1}^{(k)}}, \end{aligned} \tag{25}$$

where $\mathbf{p}_{k+1}^{(k)} \triangleq \mathbf{R}_k\mathbf{p}_{k+1}$. According to Equation (19), this net contribution can be further simplified by defining an auxiliary matrix $\mathbf{A} \in \Re^{n\times n}$ and a vector $\mathbf{b} \in \Re^{n\times 1}$ with elements given by

$$a_{i,j} \triangleq (\mathbf{p}_i^{(i-1)})^{\mathrm{T}}\mathbf{p}_j, \quad 1 \leq i \leq j, \tag{26}$$

$$b_j \triangleq (\mathbf{p}_j^{(j-1)})^{\mathrm{T}}\mathbf{y}, \quad 1 \leq j \leq n, \tag{27}$$

where $(\mathbf{p}_j^{(0)} = \mathbf{p}_j)$. The efficiency of the FRA then follows from updating these terms recursively as

$$a_{i,j} = \mathbf{p}_i^{\mathrm{T}}\mathbf{p}_j - \sum_{l=1}^{i-1} a_{l,i}a_{l,j}/a_{l,l}, \tag{28}$$

$$b_j = \mathbf{p}_j^{\mathrm{T}}\mathbf{y} - \sum_{l=1}^{j-1}(a_{l,j}b_l)/a_{l,l}. \tag{29}$$

Now, substituting (26) and (27) into (25), the net contribution of a new RBF centre $\mathbf{p}_{k+1}$ to the cost function can then be expressed as

$$\Delta J_{k+1}(\mathbf{p}_{k+1}) = \frac{b_{k+1}^2}{a_{k+1,k+1}}. \tag{30}$$

This provides a formula for selecting the best particle in the swarm at each iteration. When a pre-set number of updating cycles is reached, the best solution from the entire swarm will be added to the network. This continues until some termination criterion is met (e.g. Akaike's information criterion [AIC] (Nelles, 2001)) or until a maximum number of centres have been added. The initial particle values at each stage can be chosen from the data points rather than randomly generated in an attempt to improve convergence.

Second stage: backward network refinement

This stage involves the elimination of insignificant centres due to the constraint introduced in forward construction. Noting that the last selected centre in forward construction has always been maximally optimized for the whole network through PSO, the backward model refinement can be divided into two main parts: first, a selected centre $\mathbf{p}_k$, $k = 1,\ldots,n-1$ is shifted to the nth position as it was the last optimized one; then a new swarm is generated and its local best and global best positions are updated based on the re-ordered $n-1$ centres. When the maximum number of iterations is reached, the contribution of the best global position in the swarm is compared with the centre at nth position. If the shifted centre is less significant than the new generated position from the swarm it will be replaced, leading to the required improvement in model generalization capability. This review is repeated until a pre-defined number of check loops is reached. This differs from the original TSS method where the second stage is terminated as all of the selected centres are more significant than those remaining in the candidate pool. By contrast, the PSO-assisted refinement process proposed here generates a new solution that is compared with the existing centres, and a better solution can always be found using a check loop due to the stochastic nature of PSO.

Re-ordering of selected centres. Suppose that a selected centre $\mathbf{p}_k$ is to be moved to the nth position in the regression matrix $\mathbf{P}_n$. This can be achieved by repeatedly interchanging two adjacent centres so that

$$\mathbf{p}_q^* = \mathbf{p}_{q+1}, \quad \mathbf{p}_{q+1}^* = \mathbf{p}_q, \quad q = k,\ldots,n-1, \tag{31}$$

where the * is used to indicate the updated value. By noting the property in (23), it is clear that only $\mathbf{R}_q$ in the residual matrix series is changed at each step. This is updated using

$$\mathbf{R}_q^* = \mathbf{R}_{q-1} - \frac{\mathbf{R}_{q-1}\mathbf{p}_q^*(\mathbf{p}_q^*)^{\mathrm{T}}\mathbf{R}_{q-1}^{\mathrm{T}}}{(\mathbf{p}_q^*)^{\mathrm{T}}\mathbf{R}_{q-1}\mathbf{p}_q^*}. \tag{32}$$

Meanwhile, the following terms also need to be updated:

- In matrix $\mathbf{A}$, only the upper triangular elements $a_{i,j}$, $i \leq j$ are used for hidden node selection. The qth and the $(q+1)$th columns with elements from row 1 to $q-1$ need to be modified according to

$$\begin{cases} a_{i,q}^* & = (\mathbf{p}_i^{(i-1)})^{\mathrm{T}}\mathbf{p}_{q+1} & = a_{i,q+1} \\ a_{i,q+1}^* & = (\mathbf{p}_i^{(i-1)})^{\mathrm{T}}\mathbf{p}_q & = a_{i,q} \end{cases}, \quad i = 1,\ldots,q-1. \tag{33}$$

The qth row with elements from column q to column n are also changed using

$$a^*_{q,j} = \begin{cases} a_{q+1,q+1} + a^2_{q,q+1}/a_{q,q} & j = q \\ a_{q,q+1} & j = q+1 \\ a_{q+1,j} + a_{q,q+1}a_{q,j}/a_{q,q} & j \geq q+2 \end{cases} \quad (34)$$

and the $(q+1)$th row $a_{q+1,j}$ for $j = q+1,\ldots, n$ is also changed to

$$a^*_{q+1,j} = \begin{cases} a_{q,q} - a^2_{q,q+1}/a^*_{q,q} & j = q+1, \\ a_{q,j} - a_{q,q+1}a^*_{q,j}/a^*_{q,q} & j \geq q+2. \end{cases} \quad (35)$$

- For the vector $\mathbf{b}$, only the qth and the $(q+1)$th elements are changed. Thus,

$$b^*_q = b_{q+1} + a_{q,q+1}b_q/a_{q,q}, \quad (36)$$

$$b^*_{q+1} = b_q - a_{q,q+1}b^*_q/a^*_{q,q}. \quad (37)$$

This procedure continues until the kth centre is shifted to the nth position, the new regression matrix and residue matrices series then becoming

$$\mathbf{P}^*_n = [\mathbf{p}_1,\ldots,\mathbf{p}_{k-1},\mathbf{p}_{k+1},\ldots,\mathbf{p}_n,\mathbf{p}_k], \quad (38)$$

$$\mathbf{R}^*_k = [\mathbf{R}_1,\ldots,\mathbf{R}_{k-1},\mathbf{R}^*_k,\ldots,\mathbf{R}^*_n]. \quad (39)$$

Comparison of net contributions. As the centre $\mathbf{p}_k$ of interest has been moved to the nth position in the full regression matrix $\mathbf{P}_n$, its contribution to the cost function now needs to be reviewed. In order to remove the constraint introduced to this centre, a new swarm is generated, in which the contribution of each particle to the cost function is calculated based on the re-ordered centres $\mathbf{p}_j$ ($j = 1,\ldots, n-1$).

According to Equation (30), the calculation of the net contribution to the cost function only involves the associated elements in matrix $\mathbf{A}$ and vector $\mathbf{b}$. Thus, the computation can be greatly reduced for each particle in the swarm due to the recursive updating of $\mathbf{A}$ and $\mathbf{b}$ given in Equations (28) and (29). When the maximum value of the particle updating cycle is reached, the best solution will be given by the global optimal point in the swarm. If this solution is more significant than the shifted centre $\mathbf{p}_n$ in the regression matrix $\mathbf{P}_n$, it will replace $\mathbf{p}_n$, and matrix $\mathbf{A}$ and vector $\mathbf{b}$ will be updated as well.

The moving and comparing procedures described above are repeated until a pre-set number of check loops is reached. Finally, after a satisfactory network has been constructed, the output layer weights are computed recursively using

$$\hat{\theta}_j = \left(b_j - \sum_{i=j+1}^{k} \hat{\theta}_i a_{j,i}\right)/a_{j,j}, \quad j = k,k-1,\ldots,1. \quad (40)$$

Algorithm

The overall new algorithm for RBF network construction can now be summarized as follows:

Step 1. Initialization: Set the network size $k = 0$, and assign initial values for the following terms:
- s: the size of swarm;
- l_s: the maximum number of particle updating;
- $[\mathbf{x}_{\min}, \mathbf{x}_{\max}]$: the search space of the particles;
- $[\mathbf{v}_{\min}, \mathbf{v}_{\max}]$: the speed range of the particles;
- w: the inertia weight in velocity updating.

Step 2. At the kth step:
(a) randomly select s samples from the training data set as the starting points and randomly generate the initial velocity $\mathbf{v}_0$ inside its range;
(b) compute the RBF output for each particle using a Gaussian function;
(c) calculate $a_{i,k}$ ($1 \leq i \leq k$) and b_k for each particle using Equations (28) and (29);
(d) compute the contribution of each particle to the cost function using Equation (30), update the best position that each particle has visited to date and the best position from the entire swarm;
(e) update the velocity and position for each particle using Equations (5) and (6);
(f) check the value of velocity and position for each particle using Equations (9)–(10) and (12)–(14);
(g) if $l < l_s$, let $l = l + 1$ and go to step 2(e); otherwise, go to the next step
(h) update the $a_{i,k}$ in matrix $\mathbf{A}$ for $1 \leq i \leq k$, and b_k with the best solution found in the swarm; if the chosen pre-defined criterion is met, to go step 3; otherwise, let $k = k + 1$ and go to step 2(a);

Step 3. Backward network refinement:
(a) Change the position of $\mathbf{p}_k$ with $\mathbf{p}_{k+1}$($k = n-1,\ldots, 1$), and update the related terms using Equations (33)–(37).
(b) Continue the above step until the regressor $\mathbf{p}_k$ has been moved to the nth position.
(c) Randomly generate a swarm from the data samples, and update $a_{j,j}$, b_j for each particle using Equations (26) and (27).
(d) Update the swarm for l_s cycles using Equations (5) and (6).
(e) Compare the net contribution of the shifted centre and the best one from the swarm. If the new position from the swarm is more significant than the centre $\mathbf{p}_n$, replace it. Let $k = k - 1$ and go to step 3(a) .
(f) If the pre-set number of check loops is reached, go to step 4; otherwise, let $k = n - 1$ again and go to step 3(a) to start a new check loop.

Step 4. Use Equation (40) to calculate the output layer weights.

Computation complexity analysis

The computation in the proposed algorithm is mainly dominated by the calculation of each particle's contribution to the

cost function. Owing to the absence of a candidate pool, this new method needs more computation than our original TSS algorithm. Specifically, in the previous TSS, the algorithm updates one row of the matrix $\mathbf{A}$ in Equation (26) at each step, with the contribution of each candidate term being calculated directly from the above row elements at the same column. (For example, at the kth step, $a_{k,k}$ is calculated from $a_{i,k}$ $(1 \le i < k)$; but in the PSO assisted TSS, all of the elements in the kth column need to be calculated for a new particle.) Calculation of the RBF output also involves more computation. Conventionally, the initial candidate centres are located at data samples and the RBF widths are fixed leading to a single calculation of the RBF output. However, here the potential RBF centres are not pre-determined and have to be optimized. The RBF width parameter also varies as the particles update. Thus, the output of each hidden node needs to be calculated at each step during the construction procedure. Nevertheless, although the new method in this paper is not as efficient as TSS, the total computation is still acceptable compared with other alternatives in the literature.

In order to compare the total computational effort involved here with other methods, a numerical analysis has been carried out. Suppose that the swarm has s particles, and the maximum number of updating cycles is l_s. The number of network inputs is m which is the same as the number of width parameters $\mathbf{\Sigma}$ in each node, so that $2m$ parameters need to be optimized by the particles at each stage. As N data samples are used for training, and n RBF centres are included in the final network, the computational complexity involved in OLS using standard Gram–Schmidt orthogonalization, PSO-assisted OLS, TSS, and TSS based on PSO are reviewed.

The basic arithmetic operations involve addition/subtraction and multiplication/division. For convenience, the power operation was treated as one multiplication. The computational complexity is then measured by the total number of these operations. As mentioned above, the calculation of the RBF output is not included in the original OLS and TSS algorithm, so it is analyzed separately. The total computation in OLS is approximately given by (Li et al., 2005)

$$C_{\mathrm{OLS}} \approx 2NM(n^2 + 2n - 1) - \frac{4}{3}Nn(n^2 - 1) + n(n-1)/2, \tag{41}$$

where M is the number of candidate RBF centres. This is normally equal to $N+1$ which is the number of data samples plus an output bias. By contrast, the computation involved in the PSO-assisted OLS comprises two main parts: the algorithm itself and the RBF output calculation. The latter is given by

$$C_{\mathrm{PSO-RBF}} = 4Nnmsl_s. \tag{42}$$

By contrast, conventional OLS or TSS only need $3NMm$ operations for this component. The total computation in PSO-assisted OLS except the RBF output calculation is

$$C_{\mathrm{OLS+PSO}} \approx \left(2Nn(n+1) - 2n - \frac{1}{2}n(n-1)\right)sl_s. \tag{43}$$

For the TSS, the original algorithm is more efficient than conventional OLS, and the computation required is (Li et al., 2006)

$$C_{\mathrm{TSS}} \approx 2NM(n+1) - Nn(n-1) + M(n^2 + 4n - 6) - \frac{n}{3}(2n^2 + 3n - 17) + 8(Mn + 3n^3 + n^2 + 2n)l_{t0}, \tag{44}$$

where l_{t0} is the total number of check loops in the second stage of network refinement. Normally five loops are enough for a satisfactory improvement. It is obvious that the second stage only adds a small amount of computation to the first stage. However, for the new method proposed in this paper, the updating scheme in PSO consumes more computation than the first stage. The operations in computing the RBF output becomes

$$C_{\mathrm{PSO+TSS}} = 4Nnmsl(l_{t1} + 1), \tag{45}$$

where l_{t1} is the total number of check loops involved at the second stage of the new method here. Experiment results show that two or three cycles are enough to reduce most of the constraints in the first stage. The total computation of TSS, apart from the RBF output calculation is given by

$$C_{\mathrm{TSS+PSO}} \approx \left(Nn(2nv + n + 2) + \frac{1}{6}n(n+1)(n+8) + 18mn + 2n\right)sl_{t1}. \tag{46}$$

Normally the final network size is much smaller than the available data, so $n \ll N$ and $n \ll M$. The input size m is usually small. The main computation with and without the RBF output calculation are compared in Table 1. This shows that the Two-Stage Selection (TSS) based on PSO requires about 2-4 times more computation than with conventional alternatives. However, the new method can greatly reduce the network size and enhance the generalization capability. Furthermore, the RBF width needs to be pre-determined in a conventional method, and the search for optimal RBF width needs much more computation than the new method proposed here.

Simulation example

In this section, the new algorithm is compared with several popular alternatives for RBF network construction, including OLS (Chen et al., 1989), OLS with the LOO cross-validation (Hong et al., 2003) and the PSO-assisted FRA (the first stage of the new algorithm).

Example 1. Consider a nonlinear dynamic system defined by (Narendra and Parthasarathy, 1990)

$$y(t) = \frac{y(t-1)y(t-2)y(t-3)u(t-2)[y(t-3)-1] + u(t-1)}{1 + y^2(t-2) + y^2(t-3)}, \tag{47}$$

Table 1 Comparison of the computation complexity of different algorithms (N is the total number of data samples; M is the number of candidate pool size in conventional stepwise construction scheme; n represents the final network size; s is the number of particles in a swarm; l_s denotes the number of particle updating cycle; and l_{t1} is the number of check loops in the second stage of the new method)

Algorithm	Network construction	RBF output calculation	Total
OLS (Chen et al., 1991)	$2NM(n^2+2n)$	$3NMm$	$NM(2n^2+4n+3m)$
TSS (Li et al., 2006)	$2NMn$	$3NMm$	$NM(2n+3m)$
OLS+PSO	$2Nn^2sl_s$	$4Nnmsl_s$	$2Nn(n+2m)sl_s$
TSS+PSO	$2Nn^2l_{t1}sl_s$	$4Nnmsl_s(l_{t1}+1)$	$2Nn((n+4m)l_{t1}+4m)sl_s$

where $u(t)$ is the random system input which is uniformly distributed in the range $[-1, 1]$. A total of 400 data samples were generated, and a Gaussian white noise sequence with zero mean and variance 0.1^2 was added to the first 200. The RBF network employed the Gaussian kernel function $\phi(\mathbf{x},\mathbf{c}_i) = \exp(-\frac{1}{2} \parallel \mathbf{x}-\mathbf{c}_i \parallel^2 \mathbf{\Sigma}_i)$ as the basis function. For the conventional forward selection methods, the width of the Gaussian basis function was fixed at $\sigma^2 = 1.096$ for all centres. The input vector was predetermined as $\mathbf{x}(t) = [y(t-1), y(t-2), y(t-3), u(t-1), u(t-2)]^{\mathrm{T}}$.

With the first 200 data samples used for network training and the remaining 200 noise-free data reserved for testing, the original OLS method selected 28 RBF centres with the assist of the AIC. The OLS with LOO improved the construction procedure, it chose 18 RBF centres before the LOO error started to increase. Finally, for the algorithm described in this paper, the swarm size was set at 15 and the maximum number of iterations was 20. The inertia weight in the velocity updating was 0.8, while c_1, c_2 in Equation (5) were altered as in Equations (7) and (8). As the network size continuously increased, the first stage found that only five RBF centres were sufficient to approximate the nonlinear dynamic system in (47), while the second refinement stage reduced this to three with a smaller training and test error. The root mean-squared error (RMSE) over the training and test data sets are given in Table 2.

Here RMSE is defined as

$$RMSE = \sqrt{\frac{SSE}{N}} = \sqrt{\frac{(\hat{\mathbf{y}}-\mathbf{y})^{\mathrm{T}}(\hat{\mathbf{y}}-\mathbf{y})}{N}}, \tag{48}$$

where SSE is the sum-squared error, $\hat{\mathbf{y}}$ is the RBF neural network prediction, and N is the number of training samples. The one-step-ahead prediction performance of the network constructed by the new algorithm is illustrated in Figure 1. This shows that most of the modelling errors are in the noise range, and that the overall prediction is acceptable.

Example 2. Consider a chaotic time series generated by the well-known Mackey–Glass differential delay equation (Mackey and Glass, 1977)

$$\dot{x}(t) = \frac{ax(t-\tau)}{1+x^c(t-\tau)} - bx(t). \tag{49}$$

Table 2 Comparison of network size and performance in experiment 1 (for the new method, the error is from the mean value of 100 runs)

Algorithm	Network size	Training error	Testing error
OLS	28	0.0333	0.0432
OLS with LOO	18	0.0554	0.0564
FRA based on PSO	5	0.0260	0.0218
New	3	0.0141	0.0120

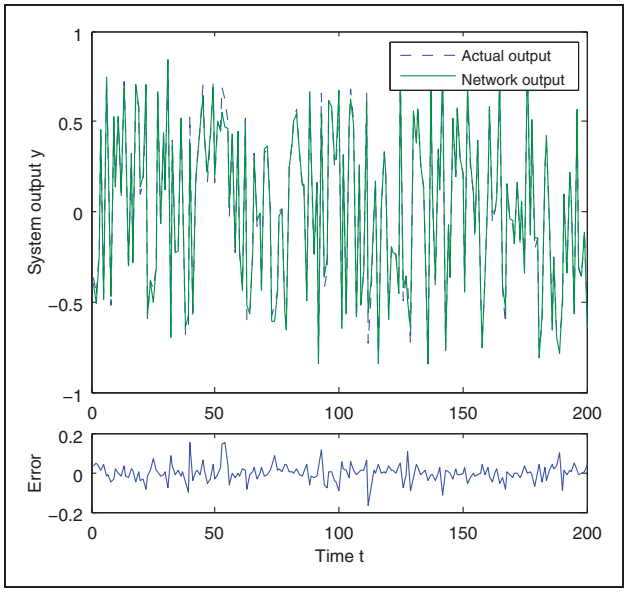

Figure 1 Comparison of network output and the actual output (solid line: one-step-ahead prediction; dashed line: actual system output).

According to Jang (1993) and Chiu (1994), the parameters were set as $a = 0.2$, $b = 0.1$, $c = 10$, and $\tau = 17$. To obtain the time series value at each integer point, the fourth-order Runge–Kutta method was applied to find the numerical solution to Equation (49). The time step was set as 0.1, while the initial condition $x(0) = 1.2$. Then 2000 data

Table 3 Comparison of network size and performance for different algorithms in experiment 2

Algorithm	Network size	Training Error	Test error	Run-time
OLS	20	0.0303	0.0293	1.46 s
FRA + PSO	8	0.0151	0.0148	0.49 s
New	6	0.0131	0.0130	1.91 s

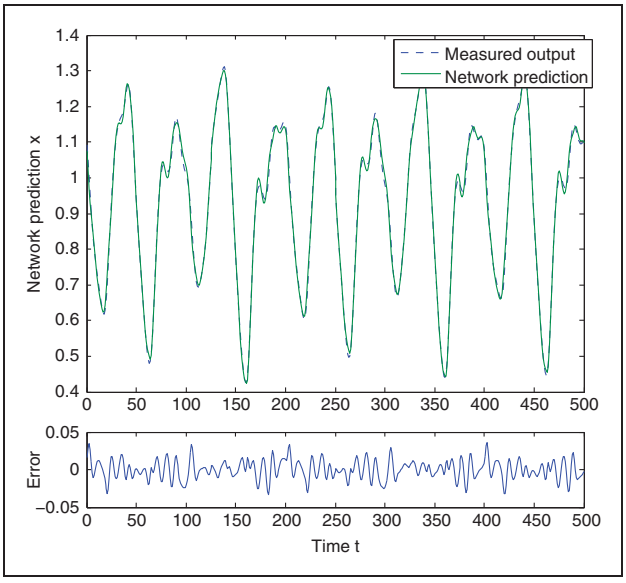

Figure 2 Comparison of measured output and network prediction in experiment 2 (solid line: one-step-ahead prediction; dashed line: measured output).

points were generated, based on which 1000 input–output data pairs were extracted ($t = [118, 1117]$). Similarly, the input vector was chosen as $[x(t-18), x(t-12), x(t-16), x(t)]$, and the model was built to predict the output at $x(t+6)$. The first 500 data pairs were used for network training while the remaining 500 data pairs were reserved for validation.

For conventional OLS method, the width of Gaussian basis function was again fixed at $\sigma = 2$ for all centres. With the assistance of the AIC criterion, OLS selected 22 RBF centres. In PSO-based methods, the swarm size was set as 15; the maximum number of iterations was 20, and c_1, c_2 in Equation (5) were still altered as in Equations (7) and (8). The inertia weight in velocity updating was again fixed at 0.8. The test results shows that FRA with PSO (Deng et al., 2010a) significantly reduced the network size to five with both smaller training and test error. As expected, the new method further improved the network compactness and generalization. Table 3 compares the performance of different methods. It is clear that PSO-assisted algorithms can significantly reduce the network. The new method however requires more computation time as indicated in the tests, but the resultant model is more compact and performs better than other alternatives. The prediction performance of RBF network produced by the new algorithm is further illustrated in Figure 2.

Concluding discussion and future work

In this paper we have proposed a novel two-stage construction approach for RBF networks. It integrates our early proposed TSS method (Li et al., 2006), with PSO, to produce a sparse network with only a few significant RBF centres. Although the new method requires more computation than conventional OLS-based alternatives, a compact model with better generalization capability can be obtained. Simulation results confirmed the effectiveness of the proposed algorithm compared with conventional OLS and its variants.

In addition, there are still some remaining problems that need to be tackled. The new method involves more computational effort. However, for a really large data set, the first stage of the proposed method is sufficient to produce a sparse model with satisfactory performance. This first stage (also known as FRA) has been proven to be more effective and efficient than conventional OLS method.

Further, the use of Sum Squared Error (SSE) as the cost function may cause the network to be easily over-fitted. Basically, lower training error leads to more precise of the model on training data, but over-trained model may fit the noise while using SSE as the cost function. An appropriate criterion is therefore required to terminate the selection process to prevent over-fitting. Cross-validation has proven to be a better alternative where the RBF centres are optimized based on their test error instead of training error. A special case of this technique is the LOO cross-validation (Hong et al., 2003; Deng et al., 2010b). However, the reduction in LOO error may becomes insufficient while more terms are added to an acceptable mode. The Bayesian learning framework (MacKay, 1992; Tipping, 2001) is also useful in preventing over-fitting. Specifically, the output layer weights are assigned with *a priori* known hyperparameters, and the most probable values of these hyperparameters are iteratively estimated from the data. In practice, the posterior distribution of irrelevant weights are sharply peaked around zero (Tipping, 2001). Therefore, centres that are mainly determined by the noise will have large values of hyperparameters, and their corresponding layer weights are forced to be near to zero. Sparsity is then achieved by removing these irrelevant centres from the trained network.

In future work we will try to effectively integrate such techniques into the above proposed construction framework to produce even better RBF network models. Further, this new algorithm can also be easily generalized to a wide range of nonlinear models that have a LITP structure, such as the Nonlinear AutoRegressive with eXogenous input (NARX) model.

Funding

Jing Deng wishes to thank Queens University Belfast for the award of an ORS scholarship to support his doctoral studies. This work is also partially supported by the UK EPSRC under grants EP/G042594/1 and EP/F021070/1, and the Key Project of Science and Technology

Commission of Shanghai Municipality (grant numbers 08160512100 and 08160705900).

References

Blum C and Merkle D (2008) *Swarm Intelligence: Introduction and Applications*. New York: Springer-Verlag.

Chen S (2006) Local regularization assisted orthogonal least squares regression. *Neurocomputing* 69: 559–85.

Chen S and Billings SA (1992) Neural networks for nonlinear dynamic system modelling and identification. *International Journal of Control* 56: 319–46.

Chen S, Billings SA and Luo W (1989) Orthogonal least squares methods and their application to non-linear system identification. *International Journal of Control* 50: 1873–96.

Chen S, Cowan CFN and Grant PM (1991) Orthogonal least squares learning algorithm for radial basis function networks. *IEEE Transactions on Neural Networks* 2: 302–9.

Chen S, Hong X and Harris C (2010) Particle swarm optimization aided orthogonal forward regression for unified data modeling. *IEEE Transactions on Evolutionary Computation* 14: 477–99.

Chen S, Hong X and Harris CJ (2004) Sparse kernel density construction using orthogonal forward regression with leave-one-out test score and local regularization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 34: 1708–17.

Chen S, Hong X, Luk BL and Harris CJ (2009) Non-linear system identification using particle swarm optimisation tuned radial basis function models. *International Journal of Bio-Inspired Computation* 1: 246–58.

Chiu S (1994) Fuzzy model identification based on cluster estimation. *Journal of intelligent and Fuzzy systems* 2: 267–78.

Clerc M and Kennedy J (2002) The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation* 6: 58–73.

Deng J, Li K, Irwin G and Fei M (2010a) Fast forward RBF network construction based on particle swarm optimization. *Life System Modeling and Intelligent Computing 2010 (Lecture Notes in Computer Science*. Vol 6329, Berlin: Springer, 40–8.

Deng J, Li K and Irwin GW (2010b) A two-stage algorithm for automatic construction of RBF neural models. In: Gatt E (ed.) *Proceedings of the 15th IEEE Mediterranean Electrotechnical Conference, MELECON 2010*. Valletta, Malta.

Eberhart RC and Shi Y (2000) Comparing inertia weights and constriction factors in particle swarm optimization. In *Proceedings of the Congress on Evolutionary Computation*, Vol. 1, pp. 84–8.

Elanayar V and Shin Y (1994) Radial basis function neural network for approximation and estimation of nonlinear stochastic dynamic systems. *IEEE Transactions on Neural Networks* 5: 594–603.

Hong X, Mitchell RJ, Chen S, Harris CJ, Li K and Irwin GW (2008) Model selection approaches for non-linear system identification: a review. *International Journal of Systems Science* 39: 925–46.

Hong X, Sharkey PM and Warwick K (2003) Automatic nonlinear predictive model-construction algorithm using forward regression and the press statistic. *IEE Proceedings: Control Theory and Applications* 150: 245–54.

Jang J (1993) ANFIS: Adaptive-network-based fuzzy inference system. *IEEE Transactions on Systems, Man, and Cybernetics* 23: 665–85.

Kennedy J and Eberhart R (1995) Particle swarm optimization. In: *Proceedings of IEEE International Conference on Neural Networks*, Vol. 4, Perth, Australia, pp. 1942–48.

Li K, Peng JX and Bai EW (2006) A two-stage algorithm for identification of nonlinear dynamic systems. *Automatica* 42: 1189–97.

Li K, Peng JX and Bai EW (2009) Two-stage mixed discrete–continuous identification of radial basis function (RBF) neural models for nonlinear systems. *IEEE Transactions on Circuits and Systems—I* 56: 630–43.

Li K, Peng JX and Irwin GW (2005) A fast nonlinear model identification method. *IEEE Transactions on Automatic Control* 50: 1211–6.

MacKay DJC (1992) Bayesian interpolation. *Neural Computation* 4: 415–47.

Mackey M and Glass L (1977) Oscillation and chaos in physiological control systems. *Science* 197: 287–9.

Mao KZ and Billings SA (1997) Algorithms for minimal model structure detection in nonlinear dynamic system identification. *International Journal of Control* 68: 311–30.

McLoone S, Brown MD, Irwin GW and Lightbody G (1998) A hybrid linear/nonlinear training algorithm for feedforward neural networks. *IEEE Transactions on Neural Networks* 9: 669–84.

Narendra KS and Parthasarathy K (1990) Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks* 1: 4–27.

Nelles O (2001) *Nonlinear System Identification*. Berlin: Springer.

Peng JX, Li K and Huang DS (2006) A hybrid forward algorithm for RBF neural network construction. *IEEE Transactions on Neural Networks* 17: 1439–51.

Rajakarunakaran S, Devaraj D and Suryaprakasa Rao K (2008) Fault detection in centrifugal pumping systems using neural networks. *International Journal of Modelling, Identification and Control* 3: 131–9.

Sherstinsky A and Picard RW (1996) On the efficiency of the orthogonal least squares training method for radial basis function networks. *IEEE Transactions on Neural Networks* 7: 195–200.

Sutanto EL, Mason JD and Warwick K (1997) Mean-tracking clustering algorithm for radial basis function centre selection. *International Journal of Control* 67: 961–77.

Tipping ME (2001) Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research* 1: 211–44.

Trelea IC (2003) The particle swarm optimization algorithm: convergence analysis and parameter selection. *Information Processing Letters* 85: 317–25.