Neurocomputing 74 (2011) 2422–2429

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Fast automatic two-stage nonlinear model identification based on the extreme learning machine

Jing Deng, Kang Li *, George W. Irwin

Intelligent Systems and Control Group, School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, Belfast BT9 5AH, UK

ARTICLE INFO

Available online 12 May 2011

Keywords:
Extreme learning machine
Two-stage stepwise selection
Leave-one-out cross validation
RBF networks

ABSTRACT

It is convenient and effective to solve nonlinear problems with a model that has a linear-in-the-parameters (LITP) structure. However, the nonlinear parameters (e.g. the width of Gaussian function) of each model term needs to be pre-determined either from expert experience or through exhaustive search. An alternative approach is to optimize them by a gradient-based technique (e.g. Newton's method). Unfortunately, all of these methods still need a lot of computations. Recently, the extreme learning machine (ELM) has shown its advantages in terms of fast learning from data, but the sparsity of the constructed model cannot be guaranteed. This paper proposes a novel algorithm for automatic construction of a nonlinear system model based on the extreme learning machine. This is achieved by effectively integrating the ELM and leave-one-out (LOO) cross validation with our two-stage stepwise construction procedure [1]. The main objective is to improve the compactness and generalization capability of the model constructed by the ELM method. Numerical analysis shows that the proposed algorithm only involves about half of the computation of orthogonal least squares (OLS) based method. Simulation examples are included to confirm the efficacy and superiority of the proposed technique.

1. Introduction

In nonlinear system modelling and identification, two main issues need to be resolved. The first is to find a proper nonlinear model structure and the second is to optimize that model structure and the parameters. To address the first issue, various model types have been proposed, such as the nonlinear autoregressive with exogenous input (NARX), nonlinear output error (NOE), artificial neural networks, and fuzzy systems [2]. Generally, most of these models have, or can be transformed into, a linear-in-the-parameters (LITP) structure which is easier to optimize.

The main problem in a LITP model is the determination of the nonlinear parameters in each term, e.g. the width of the Gaussian function and the power of a polynomial term. These parameters are normally found by exhaustive search or optimized by a gradient-based technique which significantly increases the computational complexity. Unlike conventional schemes, a more general model construction concept, namely the extreme learning machine (ELM), has been proposed for LITP models [3]. In an ELM, all the nonlinear parameters are randomly chosen independent of the training data. Training is thus transformed into a standard least-squares problem, leading to a significant improvement in the learning speed. It has been proven that the LITP model with such randomly generated nonlinear parameters can approximate any continuous target function [4]. Further, the ELM has been extended to a much wider class of model terms, including fuzzy rules as well as additive nodes [5].

Though the ELM is efficient in terms of nonlinear model construction, the resultant model is not compact. Some of the terms can be unimportant to the model's interpretability due to the stochastic process. Thus, a sparse model with a satisfactory accuracy needs to be further optimized from these candidate terms based on the parsimonious principle. The most popular approach for this is the subset construction method, including forward stepwise selection, backward stepwise selection and the so-called best subset selection [6]. Amongst these techniques, forward stepwise selection is perhaps the most popular approach in the literature for model construction where a very large term pool has to be considered [7]. The best-known algorithm in forward selection is orthogonal least squares (OLS) [8,9] which is derived from an orthogonal (or QR) decomposition of the regression matrix. An alternative is the recently proposed fast recursive algorithm (FRA) [10] which has been shown to be more efficient and stable than OLS.

The main issue in forward stepwise construction is to terminate the selection procedure to prevent over-fitting. Commonly used methods include use of an information criterion, such as the

* Corresponding author. Tel.: +44 2890974663.
E-mail address: k.li@qub.ac.uk (K. Li).

0925-2312/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2010.11.035

final prediction error (FPE) or Akaike's information criterion (AIC) [2] which provide a trade-off between the training error and model complexity. However, the tuning parameters that penalize the model complexity in these criteria are sensitive to both the specific application and the data sample size. It is still easy to miss an optimal model. To overcome this problem, the regularized forward model construction variant has also been described [11,12]. The main idea here is that each model weight is assigned a prior hyperparameter, and the most probable values of these hyperparameters are estimated iteratively from the data. As a result, those terms that are mainly determined by the noise have large hyperparameter value, and their corresponding model weights are forced near to zero. Sparsity is then achieved by removing irrelevant terms from the trained model.

To assess the generalization ability, the resultant model is usually applied to a set of test data. This involves splitting all the available measured data into different sets. However, the amount of data available is often limited, so it is desirable to use all the data available to train the model without sacrificing any generalization performance. A typical way of doing this is to employ cross validation. The limited data set is split into s parts in which $s-1$ parts are used for model training, and the single remaining part is used for model testing. This procedure is continued until all s different possible combinations has been implemented. The extreme case of cross validation is known as the leave-one-out (LOO) method [2,13], where only one sample is used for testing and the rest left for training. The overall model error or LOO error is then the average test error of each data sample. By adopting this method in a forward construction context, model terms are selected based on their reduction in the LOO error, and the selection procedure automatically terminates at the point where this starts to increase. Though this approach can achieve improved model generalization, its computational complexity is extremely high. Generally, it is therefore feasible only for very small data set, because the computational complexity is proportional to the total number of data samples. However, if the model has a linear-in-the-parameters structure, it has been shown that LOO error can be calculated without splitting the data set explicitly [13].

Though a forward construction scheme can efficiently build a sparse model from a large candidate term pool, the final model is not optimal [14], since the calculation of the contribution of any new term depends on previously selected ones [1]. To reduce this constraint, genetic search has been suggested to refine the model structure [15], but at the expense of a very high computational complexity. In [1], a two-stage stepwise construction method was recently proposed which combined both forward and backward construction procedure. This retains the computational efficiency of the FRA while improving the model compactness.

In the above two-stage construction algorithm, an initial model is first constructed by the FRA where the contribution of a particular term of interest is measured by its reduction in an appropriate cost function. The significance of each selected term is then reviewed at a second stage of model refinement, and insignificant terms are replaced. Specifically, if the contribution of a previously selected term is less than any from the candidate pool, it will be replaced with this. Thus, the cost function can be further reduced without increasing the model size. This checking cycle is iterated until no insignificant model term exists in the trained model, resulting in an optimized model structure with improved performance. The extension of two-stage algorithm to ELM was first proposed in [16], and then further improved in [17]. However, the computation complexity in [17] is still very high which may discourage its application to large data sets.

In this paper, the Leave-One-Out cross validation technique is effectively integrated into our recently proposed two-stage stepwise construction method to enhance the sparsity of models constructed from ELM. In this new algorithm, the contribution of each candidate model term is measured by its reduction in the LOO error, and the construction procedure is automatically stopped when this starts to increase. The second stage of model refinement can further improve the trained model by replacing any insignificant terms with ones from the candidate pool. Due to the introduction of a residual matrix which can be updated recursively, the calculation of LOO error is significantly reduced. Results from two simulation examples are included to confirm the efficacy of the proposed method.

The paper is organized as follows. Section 2 gives the preliminaries. The automatic two-stage algorithm is then introduced in Section 3. Section 4 describes the results from two simulation examples to illustrate the effectiveness of the proposed technique, and Section 5 contains a concluding summary.

2. Preliminaries

2.1. General nonlinear system model

Consider a general nonlinear system approximated by a model that has a linear-in-the-parameters structure

$$y(t)=\sum_{k=1}^{n}\theta_k\phi_k(\mathbf{x}(t))+\varepsilon(t) \tag{1}$$

where $y(t)$ is the system output at sample time t, $\mathbf{x}(t)$ is the input vector, $\phi_k(\mathbf{x}(t))$ denotes the nonlinear function (e.g. polynomial or Gaussian), θ_k is a model parameter, and $\varepsilon(t)$ is the model prediction error. In extreme learning machine, the nonlinear parameters in $\phi_k(\mathbf{x}(t))$, $k=1,\ldots,n$ are randomly assigned, and the model parameters $\boldsymbol{\theta}$ are estimated through least squares. The generalization performance mainly depends on the number of nonlinear functions n. Too small of n may result in an under-fitted model, while too large of n may lead to an over-fitted one.

Suppose a set of N data samples $\{\mathbf{x}(t),y(t)\}_{t=1}^{N}$ are used for model training, then (1) can be written in the matrix form

$$\mathbf{y}=\boldsymbol{\Phi}\boldsymbol{\theta}+\mathbf{e} \tag{2}$$

where $\boldsymbol{\Phi}=[\boldsymbol{\phi}_1,\ldots,\boldsymbol{\phi}_n]\in\Re^{N\times n}$ is the regression matrix with column vectors $\boldsymbol{\phi}_i=[\phi_i(\mathbf{x}(1)),\ldots,\boldsymbol{\phi}_i(\mathbf{x}(N))]^T$, $i=1,\ldots,n$, $\mathbf{y}=[y(1),\ldots,y(N)]^T\in\Re^N$, $\boldsymbol{\theta}=[\theta_1,\ldots,\theta_n]^T\in\Re^n$ and $\mathbf{e}=[\varepsilon(1),\ldots,\varepsilon(N)]^T\in\Re^N$.

The model selection procedure aims to build a parsimonious representation. Suppose that the initial model has M candidate terms, and $n(n\ll M)$ regressors $\mathbf{p}_1,\ldots,\mathbf{p}_n$ have been selected from the candidate pool with those remaining in the pool denoted as $\boldsymbol{\phi}_i(\cdot)$, $i=n+1,\ldots,M$. The resulting model is represented by

$$\mathbf{y}=\mathbf{P}_n\boldsymbol{\theta}_n+\mathbf{e} \tag{3}$$

where $\mathbf{P}_n=[\mathbf{p}_1,\ldots,\mathbf{p}_n]$, $\boldsymbol{\theta}_n=[\theta_1,\ldots,\theta_n]^T$, $\mathbf{e}=[\varepsilon(1),\ldots,\varepsilon(N)]^T$, and the least-squares estimation of $\boldsymbol{\theta}$ can be given by

$$\hat{\boldsymbol{\theta}}_n=(\mathbf{P}_n^T\mathbf{P}_n)^{-1}\mathbf{P}_n^T\mathbf{y} \tag{4}$$

The above introduction reveals the three issues in the ELM that have to be dealt with in applications. These are: (i) A larger number of nonlinear functions are usually required; (ii) the singularity problem in $\mathbf{P}$ becomes serious as the model size increased; and (iii) large model generates high computational overhead in deriving the linear parameters. In fact, these three problems are also closely coupled. If the performance of the nonlinear model can be improved, the number of required nonlinear functions and the corresponding number of linear parameters can be significantly reduced, leading to the overall reduction of the computational complexity. Once $\mathbf{P}^T\mathbf{P}$ becomes nonsingular, a number of efficient algorithms can be used for fast

computation of the linear parameters. The basic idea in this paper is to introduce an integrated framework to select the nonlinear terms from the randomly generated candidate pool in ELM to further improve the model performance and to efficiently calculate the linear parameters as well.

2.2. Computation of LOO error

Leave-one-out is a commonly used cross validation method to improve the model generalization capability [13]. It is achieved by using only one sample for model testing in each run, with the remaining $(N-1)$ samples reserved for training. Thus, for $t=1,\ldots,N$, the jth model is estimated from the data set with the tth point being removed. The prediction error can then be calculated by

$$\varepsilon_j^{(-t)} = y(t) - \hat{y}_j^{(-t)}(t) \tag{5}$$

where $y_j^{(-t)}(t)$ is the jth model output estimated by using the remaining $(N-1)$ data samples. The LOO error is obtained by averaging all these prediction errors as

$$J_j = \frac{1}{N}\sum_{t=1}^{N}(\varepsilon_j^{(-t)})^2 \tag{6}$$

The model term with the minimum LOO error will be selected at each step. It is obvious that this procedure is computationally expensive since the amount of calculation involved in using each data sample individually, as described above, is N times that in using the complete data set once. However, the following derivation shows that the LOO error can be obtained without explicitly having to split the training data set sequentially as above [13].

Refer to (3), and define a matrix $\mathbf{M}$:

$$\mathbf{M} = \mathbf{P}^T\mathbf{P} \tag{7}$$

Then, the least-squares parameter estimate of $\boldsymbol{\theta}$ is given by

$$\hat{\boldsymbol{\theta}} = [\mathbf{P}^T\mathbf{P}]^{-1}\mathbf{P}^T\mathbf{y} = \mathbf{M}^{-1}\mathbf{P}^T\mathbf{y} \tag{8}$$

The model residual at sample time t becomes

$$\varepsilon(t) = y(t) - \mathbf{p}(t)\hat{\boldsymbol{\theta}} = y(t) - \mathbf{p}(t)\mathbf{M}^{-1}\mathbf{P}^T\mathbf{y} \tag{9}$$

where $\mathbf{p}(t)$ denotes the tth row of the regression matrix $\mathbf{P}$. If the tth data sample was deleted from the estimation data set, the parameter $\hat{\boldsymbol{\theta}}$ is then calculated as

$$\hat{\boldsymbol{\theta}}^{(-t)} = \{\mathbf{M}^{(-t)}\}^{-1}\{\mathbf{P}^{(-t)}\}^T\mathbf{y}^{(-t)} \tag{10}$$

From the definition of $\mathbf{M}$, it can be shown that

$$\mathbf{M}^{(-t)} = \mathbf{M} - \mathbf{p}(t)^T\mathbf{p}(t) \tag{11}$$

and

$$\{\mathbf{P}^{(-t)}\}^T\mathbf{y}^{(-t)} = \mathbf{P}^T\mathbf{y} - \mathbf{p}(t)^T y(t) \tag{12}$$

By using the well-known matrix inversion lemma $[\mathbf{A}+\mathbf{BCD}]^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}[\mathbf{DA}^{-1}\mathbf{B}+\mathbf{C}^{-1}]^{-1}\mathbf{DA}^{-1}$ [18], the inverse of $\mathbf{M}^{(-t)}$ in (11) can be computed as

$$\{\mathbf{M}^{(-t)}\}^{-1} = \mathbf{M}^{-1} + \frac{\mathbf{M}^{-1}\mathbf{p}(t)^T\mathbf{p}(t)\mathbf{M}^{-1}}{1-\mathbf{p}(t)\mathbf{M}^T\mathbf{p}(t)^T} \tag{13}$$

The model error at data point t is now given as [13]

$$\begin{aligned}\varepsilon^{(-t)}(t) &= y(t) - \mathbf{p}(t)\hat{\boldsymbol{\theta}}^{(-t)} \\ &= y(t) - \mathbf{p}(t)\{\mathbf{M}^{(-t)}\}^{-1}\{\mathbf{p}^{(-t)}\}^T\mathbf{y}^{(-t)} \\ &= \frac{y(t) - \mathbf{p}(t)\mathbf{M}^{-1}\mathbf{P}^T\mathbf{y}}{1-\mathbf{p}(t)\mathbf{M}^{-1}\mathbf{p}^T(t)} \\ &= \frac{\varepsilon(t)}{1-\mathbf{p}(t)\mathbf{M}^{-1}\mathbf{p}^T(t)}\end{aligned} \tag{14}$$

The calculation of the LOO error is simplified by the use of Eq. (14) which does not involve splitting the training data sequentially.

3. Automatic two-stage model construction

In forward stepwise construction scheme, model terms are selected one-by-one with the LOO error being maximally reduced each time. Suppose k such terms have been selected, and the resulting regression matrix is denoted as

$$\mathbf{P}_k = [\mathbf{p}_1,\ldots,\mathbf{p}_k], \quad k=1,\ldots,n \tag{15}$$

From (14), the corresponding model residual at sample time t becomes

$$\varepsilon_k^{(-t)}(t) = \frac{y(t) - \mathbf{p}_k(t)\mathbf{M}_k^{-1}\mathbf{P}_k^T\mathbf{y}}{1-\mathbf{p}_k(t)\mathbf{M}_k^{-1}\mathbf{p}_k^T(t)} = \frac{\varepsilon_k(t)}{1-\mathbf{p}_k(t)\mathbf{M}_k^{-1}\mathbf{p}_k^T(t)} \tag{16}$$

and the LOO error is given by

$$J_k = \frac{1}{N}\sum_{t=1}^{N}(\varepsilon_k^{(-t)}(t))^2 \tag{17}$$

If one more regressor term $\mathbf{p}_{k+1}$ is selected, the regression matrix changes to $\mathbf{P}_{k+1} = [\mathbf{P}_k, \mathbf{p}_{k+1}]$. The selected term should maximally reduces the LOO error compared to all the remaining available candidates. However, this choice involves a constrained minimization of J_{k+1}, which can be solved in a second stage of model refinement.

3.1. Forward model selection—first stage

In order to simplify the computation of the LOO error, a residual matrix series is defined first. Thus

$$\mathbf{R}_k \triangleq \begin{cases} \mathbf{I} - \mathbf{P}_k\mathbf{M}_k^{-1}\mathbf{P}_k^T, & 0 < k \le n \\ \mathbf{I}, & k = 0 \end{cases} \tag{18}$$

According to [10,1], the matrix terms $\mathbf{R}_k, k=0,\ldots,n$ have the following properties:

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

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

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

$$\mathbf{R}_k\boldsymbol{\phi} = \begin{cases} \mathbf{0}, & rank([\mathbf{P}_k,\boldsymbol{\phi}]) = k \\ \boldsymbol{\phi}^{(k)} \ne \mathbf{0}, & rank([\mathbf{P}_k,\boldsymbol{\phi}]) = k+1 \end{cases} \tag{22}$$

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

The last property 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 computational effort in the second stage.

Now, the residual vector $\mathbf{e}$ can be given by

$$\mathbf{e}_k = \mathbf{y} - \mathbf{P}_k\mathbf{M}_k^{-1}\mathbf{P}_k^T\mathbf{y} = \mathbf{R}_k\mathbf{y} \tag{24}$$

where $\mathbf{e}_k = [\varepsilon_k(1),\ldots,\varepsilon_k(t),\ldots,\varepsilon_k(N)]^T$. The denominator in (16) is also the tth diagonal element of the residual matrix $\mathbf{R}_k$. As a result, the LOO error in (17) can be rewritten as

$$J_k = \frac{1}{N}\sum_{t=1}^{N}\frac{\varepsilon_k^2(t)}{d_k^2(t)} \tag{25}$$

where $d_k = diag(\mathbf{R}_k)$.

According to (19), the above LOO error can be further simplified by defining an auxiliary matrix $\mathbf{A} \in \Re^{k\times M}$, and a vector

$\mathbf{b} \in \Re^{M\times 1}$, with elements given by

$$a_{i,j} \triangleq \begin{cases} (\mathbf{p}_i^{(i-1)})^T \mathbf{p}_j, & 1 \le j \le k \\ (\mathbf{p}_i^{(i-1)})^T \boldsymbol{\phi}_j, & k < j \le M \end{cases} \tag{26}$$

$$b_i \triangleq \begin{cases} (\mathbf{p}_i^{(i-1)})^T \mathbf{y}, & 1 \le i \le k \\ (\boldsymbol{\phi}_i^{(k)})^T \mathbf{y}, & k < i \le M \end{cases} \tag{27}$$

Referring to the updating of residual matrix in (19), it shows that $a_{k,j}$, b_k, $\mathbf{d}_k$ and $\mathbf{e}_k$ can be computed recursively as follows:

$$a_{k,j} = \mathbf{p}_k^T \boldsymbol{\phi}_j - \sum_{l=1}^{k-1} a_{l,k} a_{l,j} / a_{l,l}, \quad k = 1,\ldots,n,\ j = 1,\ldots,M. \tag{28}$$

$$b_k = \mathbf{p}_k^T \mathbf{y} - \sum_{l=1}^{k-1} (a_{l,k} b_l)/a_{l,l} \quad k = 1,\ldots,n. \tag{29}$$

$$\mathbf{p}_k^{(k-1)} = \mathbf{p}_k^{(k-2)} - \frac{a_{k-1,k}}{a_{k-1,k-1}} \mathbf{p}_{k-1}^{(k-2)} \tag{30}$$

$$\mathbf{d}_k = \mathbf{d}_{k-1} - (\mathbf{p}_k^{(k-1)})^2 / a_{k,k} \tag{31}$$

$$\mathbf{e}_k = \mathbf{e}_{k-1} - \frac{b_k}{a_{k,k}} \mathbf{p}_k^{(k-1)} \tag{32}$$

The LOO error in (25) can now be calculated recursively using (31) and (32). In this forward construction stage, the significance of each model term is measured based on its reduction in LOO error. Thus, suppose at the kth step, one more term from the candidate pool is to be selected. The new LOO error of the model, which includes the previously selected k terms and the new candidate one, will be computed from (25). The one that gives the minimum LOO error J_{k+1} will be added to the model. Meanwhile, all the regressors in $\boldsymbol{\Phi}$ have been stored in their intermediate forms for the next selection. That is, if the kth term is added to the model, then all previously selected terms will be saved as $\mathbf{p}_1^{(0)},\ldots,\mathbf{p}_k^{(k-1)}$, and all the remaining regressors in the candidate pool will be saved as $\boldsymbol{\phi}_i^{(k)}, i = k+1,\ldots,M$. The diagonal elements $a_{j,j}$ in $\mathbf{A}$, and b_j for $k+1 \le j \le M$ are also pre-calculated for use in the next selection, and are given by

$$a_{j,j}^{(k+1)} = a_{j,j}^{(k)} - a_{k,j}^2 / a_{k,k} \tag{33}$$

$$b_j^{(k+1)} = b_j^{(k)} - a_{k,j} b_k / a_{k,k} \tag{34}$$

This procedure is continued until the LOO error starts to increase, meaning the forward selection stage will be automatically terminated when

$$J_n < J_{n+1} \tag{35}$$

resulting in a compact model with n terms.

3.2. Backward model refinement—second stage

This stage involves the elimination of insignificant terms due to the constraint introduced in forward construction. Noting that the last selected term in forward construction is always more significant than those remaining in the candidate pool, the backward model refinement can be divided into three main procedures: firstly, a selected term $\mathbf{p}_k, k = 1,\ldots,n-1$ is shifted to the nth position as it was the last selected one; secondly, the LOO error of each candidate term is recalculated, and compared with the shifted term. If the LOO error of a selected term is larger than that of a term from the candidate pool, it will be replaced, leading to the required improvement in model generalization capability. This review is repeated until no insignificant term remains in the selected model. Finally, all the selected terms are used to form a new candidate pool, and the forward stepwise construction is implemented again, probably further reducing the model size.

3.2.1. Re-ordering of regressor terms

Suppose a selected model term $\mathbf{p}_k$ is to be moved to the nth position in the regression matrix $\mathbf{P}_n$. This can be achieved by repeatedly interchanging two adjacent regressors so that

$$\mathbf{p}_q^* = \mathbf{p}_{q+1}, \quad \mathbf{p}_{q+1}^* = \mathbf{p}_q, \quad q = k,\ldots,n-1 \tag{36}$$

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

$$\mathbf{R}_q^* = \mathbf{R}_{q-1} - \frac{\mathbf{R}_{q-1}\mathbf{p}_q^* (\mathbf{p}_q^*)^T \mathbf{R}_{q-1}^T}{(\mathbf{p}_q^*)^T \mathbf{R}_{q-1} \mathbf{p}_q^*} \tag{37}$$

Meanwhile, the following terms also need to be updated:

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

$$\begin{cases} a_{i,q}^* = (p_i^{(i-1)})^T p_{q+1} = a_{i,q+1} \\ a_{i,q+1}^* = (p_i^{(i-1)})^T p_q = a_{i,q} \end{cases} \tag{38}$$

  The elements of qth row $a_{q,j}$ from column q to column M $(j = q,\ldots,M)$ are also changed using

$$a_{q,j}^* = \begin{cases} a_{q+1,q+1} + a_{q,q+1}^2 / a_{q,q}, & j = q \\ a_{q,q+1}, & j = q+1 \\ a_{q+1,j} + a_{q,q+1} a_{q,j} / a_{q,q}, & j \ge q+2 \end{cases} \tag{39}$$

  and the elements of the $(q+1)$th row $a_{q+1,j}, (j = q+1,\ldots,M)$ are changed by

$$a_{q+1,j}^* = \begin{cases} a_{q,q} - a_{q,q+1}^2 / a_{q,q}^* & j = q+1 \\ a_{q,j} - a_{q,q+1} a_{q,j}^* / a_{q,q}^* \ \textit{Nonlinear optimization} & j \ge q+2 \end{cases} \tag{40}$$

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

$$b_q^* = b_{q+1} + a_{q,q+1} b_q / a_{q,q} \tag{41}$$

$$b_{q+1}^* = b_q - a_{q,q+1} b_q^* / a_{q,q}^* \tag{42}$$

- Finally, $\mathbf{p}_q^{(q-1)}$ and $\mathbf{p}_{q+1}^{(q)}$ are updated using

$$(\mathbf{p}_q^*)^{(q-1)} = \mathbf{p}_{q+1}^{(q)} + \frac{a_{q,q+1}}{a_{q,q}} \mathbf{p}_q^{(q-1)} \tag{43}$$

$$(\mathbf{p}_{q+1}^*)^{(q)} = \mathbf{p}_q^{(q-1)} - \frac{a_{q,q+1}}{a_{q,q}^*} (\mathbf{p}^*)_q^{(q-1)} \tag{44}$$

This procedure is continued until the kth regressor term is shifted to the nth position, the new regression matrix and residual matrices series becoming

$$\begin{aligned} \mathbf{P}_n^* &= [\mathbf{p}_1^*,\ldots,\mathbf{p}_{k-1}^*,\mathbf{p}_k^*,\ldots,\mathbf{p}_{n-1}^*,\mathbf{p}_n^*] \\ &= [\mathbf{p}_1,\ldots,\mathbf{p}_{k-1},\mathbf{p}_{k+1},\ldots,\mathbf{p}_n,\mathbf{p}_k] \end{aligned} \tag{45}$$

$$\{\mathbf{R}_k^*\} = [\mathbf{R}_1,\ldots,\mathbf{R}_{k-1},\mathbf{R}_k^*,\ldots,\mathbf{R}_n^*] \tag{46}$$

3.2.2. LOO error comparison

When the regressor term p_k has been moved to the nth position in the full regression matrix $\mathbf{P}_n$, its contribution to the reduction of leave-one-out cross validation error needs to be

reviewed. The LOO error of those previously selected n terms remains unchanged, while the new LOO errors of the regressor terms in the candidate pool must be recalculated. The corresponding terms for $n+1 \le j \le M$ to be changed are as follows:

$$a^*_{j,j} = a_{j,j} + (a^*_{n,j})^2 / a^*_{n,n} \tag{47}$$

$$b^*_j = b_j + b^*_n a^*_{n,i} / a^*_{n,n} \tag{48}$$

$$(\boldsymbol{\varphi}^{(n-1)}_j)^* = \boldsymbol{\phi}^{(n)}_j + \frac{a^*_{n,j}}{a^*_{n,n}} (\mathbf{p}^{(n-1)}_n)^* \tag{49}$$

$$\mathbf{d}^*_j = \mathbf{d}_n + \frac{1}{a^*_{n,n}} [(\mathbf{p}^{(n-1)}_n)^*]^2 - \frac{1}{a^*_{j,j}} [(\boldsymbol{\phi}^{(n-1)}_j)^*]^2 \tag{50}$$

$$\mathbf{e}^*_j = \mathbf{e}_n + \frac{b^*_n}{\hat{a}_{n,n}} (\mathbf{p}^{(n-1)}_n)^* - \frac{b^*_j}{a^*_{j,j}} (\boldsymbol{\phi}^{(n-1)}_j)^* \tag{51}$$

Suppose a regressor $\boldsymbol{\phi}_m$ from the candidate term pool has a smaller LOO error than the model term of interest, that is $\mathbf{J}^*_m < \mathbf{J}^*_n$. In this case, $\boldsymbol{\phi}_m$ will replace $\mathbf{p}^*_n$ in the selected regression matrix $\mathbf{P}_n$, and $\mathbf{p}^*_n$ will be put back into the candidate term pool. Meanwhile, the following related terms are updated:

- The vectors $\mathbf{e}_n$ and $\mathbf{d}_n$ are changed to

$$\mathbf{e}^*_n = \mathbf{e}_m, \quad \mathbf{d}^*_n = \mathbf{d}_m \tag{52}$$

- In the matrix A

$$a^*_{i,n} = a_{i,m}, \quad a^*_{i,m} = a_{i,n} \quad (i = 1, \ldots, n-1) \tag{53}$$

$$a^*_{n,j} = \begin{cases} a_{m,m} & j = n \\ a_{n,m} & j = m \\ \boldsymbol{\phi}^T_m \boldsymbol{\phi}_j - \sum_{l=1}^{n-1} a_{l,m} a_{l,j} / a_{l,l} & \text{others} \end{cases} \tag{54}$$

$$(a^*_{j,j})^{(n+1)} = \begin{cases} a_{n,n} - (a^*_{n,m})^2 / a^*_{n,n} & j = m \\ a_{j,j} - (a^*_{n,j})^2 / a^*_{n,n} & j \ne m \end{cases} \tag{55}$$

- In the vector $\mathbf{b}$,

$$b^*_n = b_m \tag{56}$$

$$(b^*_j)^{(n+1)} = \begin{cases} b_n - a^*_{n,m} b^*_n / a^*_{n,n} & j = m \\ b_j - a^*_{n,j} b^*_n / a^*_{n,n} & j \ne m \end{cases} \tag{57}$$

- Finally, $\mathbf{p}^{(n-1)}_n$ and $\boldsymbol{\phi}^{(n)}_j$ for $n < j \le M$ are updated according to

$$(\mathbf{p}^{(n-1)}_n)^* = \boldsymbol{\phi}^{(n-1)}_m \tag{58}$$

$$(\boldsymbol{\phi}^{(n)}_j)^* = \boldsymbol{\phi}^{(n-1)}_j - \frac{a^*_{n,j}}{a^*_{n,n}} (\mathbf{p}^{(n-1)}_n)^* \tag{59}$$

3.2.3. Model refinement

The above two procedures in Sections 3.2.1 and 3.2.2 are repeated until there is no insignificant centres remaining in the full regression matrix $\mathbf{P}_n$. As described in the forward selection stage, the reduction in LOO error involves a constrained minimization, since the selection of additional model terms will depend on those previously chosen. This is also true in determining the stopping point at this step. The total number of RBF centres from the first stage in the network model is not optimal. Fig. 1 illustrates n centres being selected in the first stage, with a further reduction of the LOO error given during the second stage

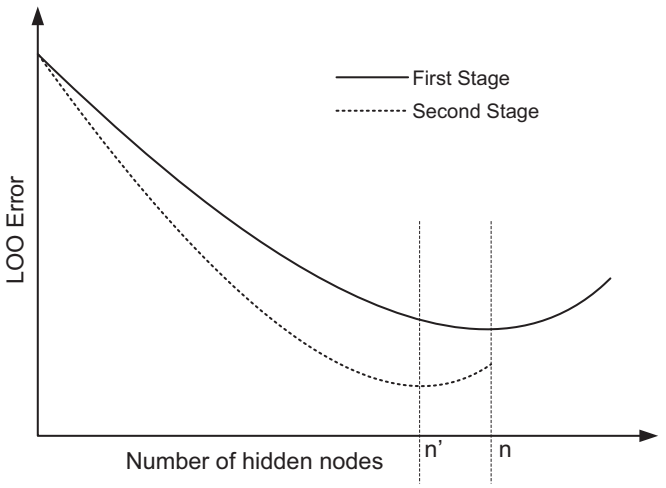

Fig. 1. LOO error at different stages (The forward selection stage is stopped at n, and the second stage is stopped at n', $n' \le n$).

at the value n. However, this number n is not the optimal model size at the second stage since n' is in fact the optimal number of centres. The third procedure involves re-ordering the selected centres by their contributions to the network. This is done by putting all n selected hidden nodes into a smaller term pool, and applying forward selection again. This process will either be automatically terminated at the point n', or when all n terms have been re-selected.

3.3. Algorithm

The proposed algorithm for selecting a sub-model can now be summarized as follows:

Step 1 Initialization: Construct the candidate regression matrix based on ELM, and let the model size $k = 0$. Then assign the initial value for the following terms ($j = 1, \ldots, M$):
- $J_0 = \frac{1}{N} \sum_{t=1}^{N} y(t)^2$;
- $\mathbf{d}_0 = [1, \ldots, 1]^T \in \Re^{N \times 1}$; $\mathbf{e}_0 = \mathbf{y}$;
- $\boldsymbol{\phi}^{(0)}_j = \boldsymbol{\phi}_j$; $a^{(1)}_{j,j} = \boldsymbol{\phi}^T_j \boldsymbol{\phi}_j$; $b^{(1)}_j = \boldsymbol{\phi}^T_j \mathbf{y}$;

Step 2 Forward selection:
- (a) At the kth step ($1 \le k \le M$), use (31), (32), and (35) to calculate $\mathbf{d}_j$, $\mathbf{e}_j$ and their corresponding LOO error J_k for each candidate term.
- (b) Find the candidate regressor that gives the minimal LOO error, and add it to the regression matrix $\mathbf{P}$. Then compute $a_{k,j}$ and pre-calculate $\boldsymbol{\phi}^{(k)}_j$, $a^{(k+1)}_{j,j}$, and $b^{(k+1)}_j$ for $j = k+1, \ldots, M$.
- (c) If the LOO error $J_{k-1} > \mathbf{J}_k$, set $k = k+1$, and go back to step (a). Otherwise, go to step 3.

Step 3 Backward model 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 (38)–(44).
- (b) Continue the above step until the regressor $\mathbf{p}_k$ has been moved to the nth position.
- (c) Update $a_{j,j}, b_j, \boldsymbol{\phi}^{(n-1)}_j, \mathbf{d}_j, \mathbf{e}_j$ for each candidate regressors using (47)–(51), and compute their new LOO error.
- (d) If the LOO error of the candidate term $\mathbf{J}^*_m$ is less than $\mathbf{J}^*_n$, then replace $\mathbf{p}^*_n$ with $\boldsymbol{\phi}_m$, and put $\mathbf{p}^*_n$ back into the candidate term pool. Update the related terms according to (52)–(59).
- (e) If $k > 1$, set $k = k-1$, and go to step 3(a).

(f) If one or more regressor terms were changed in the last review, then set $k=n-1$, and repeat steps 3(a)–(e) to review all the terms again. Otherwise, the procedure is terminated.

Step 4 Final forward selection: Put all n selected regressor terms together to form a small candidate term pool, and apply the forward selection procedure again. This selection process automatically terminates at n_0, $n_0 \le n$.

3.4. Computational complexity

The computation in two-stage selection with leave-one-out cross validation is mainly dominated by the first forward selection stage. It will be shown that the proposed method here is still more efficient than orthogonal least squares (OLS). Specifically, suppose there are initially M candidate regressors in the selection pool, and only n terms are to be included in the final model. As N data samples are used for training, the computational complexity involved in OLS using standard Gram–Schmidt orthogonalisation, Forward recursive algorithm (FRA), the two-stage selection and their combination with cross validation are all now reviewed.

The computational complexity is measured by the total number of basic arithmetic operations involving addition/subtraction and multiplication/division. For OLS with leave-one-out cross validation, the total computation is given by

$$C_{(OLS)} \approx NM(2n^2+11n)-n(n-1)(11N+M-1)/2 - n(n-1)(2n-1)(4N-1)/6+2N-2M \quad (60)$$

For the first stage of the proposed algorithm, the computation is the same as FRA with LOO, and is given by

$$C_{(FRA)} \approx NM(13n+4)-Nn(13n-9) - n(7n+3)+7Mn-2M-2N \quad (61)$$

The computation for the second refinement stage includes the shifting of each selected term to the last position, the new LOO error comparison and the change of a centre of interest with a candidate pool term. In the extreme case, all the terms of interest are insignificant compared to a candidate regressor. The total computation involved in one checking loops is then calculated using

$$C_{(2nd)} \approx 4N(m-n)+m(3n^2-n+6) + 2N(n^2-1)-n^3+5n^2-14n \quad (62)$$

Generally, most of the shifted terms are more significant than the candidate ones, thus the actual computation for the second stage is much less than (62), and the total number of checking loop repeated is normally less than 5. In practice $n \ll N$ and $n \ll M$, so the computational effort mainly comes from the first term in above equations. Table 1 compares these algorithms. It shows that the new one described here needs about half of the computation of OLS with LOO cross validation.

Table 1
Comparison of computational complexity (five checking loops are used at the second stage; N is the number of samples; M is the size of initial term pool and n represents the final model size).

Algorithm	Computation
OLS	$2NM(n^2+2n)$
OLS+LOO	$NM(2n^2+11n)$
FRA	$2NMn$
FRA+LOO	$13NMn$
Two-stage	$NM(4n+15)/2$
New	$NM(13n+20)$

4. Simulation example

This section presents the simulation results for nonlinear system modelling using RBF neural networks which have the linear-in-the-parameters structure. The following three methods were applied and assessed: (1) OLS method. Here, the width of the activation function is chosen 'a priori', and the data samples are used as the centres of the candidate RBF nodes. Then the OLS is used to select the most significant terms, and AIC (Akaike's information criterion) is adopted to control the model complexity. (2) Two-stage with LOO. The experimental condition is set the same as the above, and two-stage with LOO is then applied to automatically build the model. (3) The proposed method. Here, the width and centres of the RBF nodes are randomly selected instead of setting 'a priori', and the proposed method is then used for automatic model construction.

Example 1: Consider a nonlinear system to be approximated by an RBF neural model [19] defined by

$$y(x) = 0.1x + \frac{\sin(x)}{x} + \sin(0.5x), \quad -10 \le x \le 10 \quad (63)$$

A total of 400 data samples were generated with x uniformly distributed in the range $[-10,10]$. Gaussian white noise with a zero mean and variance 0.1^2 was added to the first 200 data samples. The RBF model employed the Gaussian kernel function $\phi(\mathbf{x},\mathbf{c}_i) = \exp(-\|\mathbf{x}-\mathbf{c}_i\|^2/\sigma^2)$ as the basis function. In OLS, the width of the Gaussian basis function was chosen as $\sigma=4$. For the method proposed here, the support function width was chosen randomly in the interval [0, 2] and the hidden nodes were randomly selected from the data samples. With the first 200 noise samples used for training and the remaining 200 noise-free data reserved for testing, the conventional OLS algorithm had selected 11 centres at the point where the AIC value started to increase. The two-stage with leave-one-out cross validation constructed a more compact network with eight RBF centres under both conventional candidate pool and ELM. The root mean-squared error (RMSE) [19] over the training data and testing data sets are given in Table 2. This RMSE is defined as

$$RMSE = \sqrt{\frac{\mathbf{SSE}}{N}} = \sqrt{\frac{(\hat{y}-\mathbf{y})^T(\hat{y}-\mathbf{y})}{N}} \quad (64)$$

where SSE is the sum-squared error, $\hat{y}$ is the RBF neural network prediction, and N is the number of samples. As shown in Table 2, it is clear that:

- OLS produced an over-fitted network with a smaller training error and larger testing error.
- Two-stage with LOO gave a better RBF network with a slightly increased training error and a smaller testing error.
- By using extreme learning machine, the new method constructed a more compact RBF model than OLS.

Though the proposed method did not produce a better model than the other two methods in terms of testing error, the main

Table 2
Comparison of model performance in experiment 1 (RMSE).

Algorithm	Network size	Training error	Testing error
OLS	11	0.1006	0.0326
Two-stage+LOO	8	0.1011	0.0201
New	8	0.1004	0.0337

Table 3
Evolution of term selection in the model refining procedure in experiment 1.

Loop	Selected terms	Training error (RMSE)	LOO error
0	[24,163,100,55,150,146,175,120]	0.1056	0.0127
1	[24,163,100,55,150,146,120,5]	0.1060	0.0123
2	[24,163,100,55,150,120,5,149]	0.1061	0.0123
3	[24,163,100,55,120,5,149,121]	0.1031	0.0116
4	[24,120,5,149,121,55,100,96]	0.1026	0.0115
5	[24,96,100,55,121,149,5,110]	0.1018	0.0113
6	[24,96,100,55,121,110,5,146]	0.1005	0.0110
7	[24,96,100,55,110,5,146,16]	0.1004	0.0110

Table 4
Comparison of model performance in experiment 2 (RMSE).

Algorithm	Network size	Training error	Testing error
OLS	26	0.0913	0.0735
Two-stage+LOO	13	0.1020	0.0692
New	13	0.0999	0.0572

Table 5
Evolution of term selection in the model refining procedure in experiment 2.

Loop	Number of changed terms	Training error (RMSE)	LOO error
0	0	0.1053	0.0129
1	2	0.1060	0.0127
	⋮		
11	8	0.0999	0.0113

advantage is that the network can be rapidly constructed without need to test different RBF widths. Table 3 shows the refinement procedure at the second stage while using the extreme learning machine, where seven check loops were performed to minimize the LOO error. Here a total of four RBF nodes were replaced at the second network refinement stage, and the RMSE value was reduced without increasing the model size.

Example 2: The proposed algorithm was then applied to approximate a nonlinear dynamic system given by [13]

$$z(t)=\frac{z(t-1)z(t-2)z(t-3)u(t-2)[z(t-3)-1]+u(t-1)}{1+z^2(t-2)+z^2(t-3)} \tag{65}$$

where $u(t)$ is the random system input which was uniformly distributed in the range $[-1,1]$. Again, a total of 400 data samples were generated with the first 200 noise data samples used for training and the remaining 200 noise-free data samples for testing. The noise series had a Gaussian distribution with a zero mean and variance of 0.1^2. The activation function had the same form as described in experiment 1 with $\sigma=1.096$ for conventional network construction scheme. The input vector was $\mathbf{x}(t)=[y(t-1),y(t-2),y(t-3),u(t-1),u(t-2)]^T$. In this case, the OLS method selected 26 RBF nodes while the proposed algorithm chose 13 nodes. In Table 4, the OLS still produced an over-fitted network with a smaller training error and a larger testing error is shown. By contrast, the proposed method gave a compact network model with better generalization performance. Table 5 illustrates the reduction of LOO error in each refining loop of the two-stage algorithm under the ELM scheme.

5. Concluding summary

In this paper, a previously proposed two-stage stepwise identification algorithm [1] is combined with the leave-one-out cross validation to improve the compactness of models constructed by the extreme learning machine. At the first stage, the selection procedure can be automatically terminated based on the LOO error. At the second stage, the contribution of each selected model term is reviewed, and insignificant ones are replaced. This procedure repeats until no insignificant RBF nodes exist in the final model. The forward selection step is then implemented again to re-order the selected terms, producing a potential further reduction in the model size. Results from two simulation examples show that the proposed improved extreme learning machine scheme can efficiently produce a nonlinear model with satisfactory performance. The potential to build a more compact model by integrating other techniques, such as Bayesian regularization, is regarded as the future work.

Acknowledgment

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

References

[1] K. Li, J.X. Peng, E.W. Bai, A two-stage algorithm for identification of nonlinear dynamic systems, Automatica 42 (7) (2006) 1189–1197.
[2] O. Nelles, Nonlinear System Identification, Springer, 2001.
[3] G. Huang, Q. Zhu, C. Siew, Extreme learning machine: theory and applications, Neurocomputing 70 (2006) 489–501.
[4] G. Huang, L. Chen, C. Siew, Universal approximation using incremental constructive feedforward networks with random hidden nodes, IEEE Transactions on Neural Networks 17 (4) (2006) 879–892.
[5] G. Huang, L. Chen, Convex incremental extreme learning machine, Neurocomputing 70 (16–18) (2007) 3056–3062.
[6] A. Miller, Subset Selection in Regression, CRC Press, 2002.
[7] X. Hong, R.J. Mitchell, S. Chen, C.J. Harris, K. Li, G.W. Irwin, Model selection approaches for non-linear system identification: a review, International Journal of Systems Science 39 (10) (2008) 925–946.
[8] S. Chen, S.A. Billings, W. Luo, Orthogonal least squares methods and their application to non-linear system identification, International Journal of Control 50 (5) (1989) 1873–1896.
[9] S. Chen, C.F.N. Cowan, P.M. Grant, Orthogonal least squares learning algorithm for radial basis function networks, IEEE Transactions on Neural Networks 2 (2) (1991) 302–309.
[10] K. Li, J.X. Peng, G.W. Irwin, A fast nonlinear model identification method, IEEE Transactions on Automatic Control 50 (8) (2005) 1211–1216.
[11] S. Chen, E.S. Chng, K. Alkadhimi, Regularized orthogonal least squares algorithm for constructing radial basis function networks, International Journal of Control 64 (5) (1996) 829–837.
[12] D.J. Du, K. Li, M.R. Fei, G.W. Irwin, A new locally regularised recursive method for the selection of RBF neural network centres, in: R. Walter (Ed.), Proceedings of the 15th IFAC Symposium on System Identification, SYSID 2009, Saint-Malo, France, 2009.
[13] X. Hong, P.M. Sharkey, K. Warwick, Automatic nonlinear predictive model construction algorithm using forward regression and the press statistic, IEE Proceedings: Control Theory and Applications 150 (3) (2003) 245–254.
[14] A. Sherstinsky, R.W. Picard, On the efficiency of the orthogonal least squares training method for radial basis function networks, IEEE Transactions on Neural Networks 7 (1) (1996) 195–200.
[15] K.Z. Mao, S.A. Billings, Algorithms for minimal model structure detection in nonlinear dynamic system identification, International Journal of Control 68 (2) (2007) 311–330.
[16] K. Li, G. Huang, S. Ge, A Fast Construction Algorithm for Feedforward Neural Networks, Springer Verlag, 2010.
[17] Y. Lan, Y. Soh, G. Huang, Two-stage extreme learning machine for regression, Neurocomputing 73 (16-18) (2010) 3028–3038.
[18] L. Ljung, System Identification: Theory for the User, Prentice Hall, Cliffs, NJ, 1987.
[19] J.X. Peng, K. Li, D.S. Huang, A hybrid forward algorithm for RBF neural network construction, IEEE Transactions on Neural Networks 17 (6) (2006) 1439–1451.

Jing Deng received his B.Sc. degrees from National University of Defence Technology, China, in 2001 and 2005, and his M.Sc. degrees from the Shanghai University, China, in 2005 and 2007. From March 2008 he started his Ph.D. research at Intelligent Systems and Control (ISAC) Group at Queen's University Belfast, UK. His main research interests include system modelling, pattern recognition and fault detection.

Kang Li is a Reader in Intelligent Systems and Control at Queen's University Belfast. His research interests include advanced algorithms for training and construction of neural networks, fuzzy systems and support vector machines, as well as advanced evolutionary algorithms, with applications to nonlinear system modelling and control, microarray data analysis, systems biology, environmental modelling and monitoring, and polymer extrusion. He has produced over 150 research papers and co-edited seven conference proceedings in his field. He is a Chartered Engineer, a member of the IEEE and the InstMC and the current Secretary of the IEEE UK and Republic of Ireland Section.

George W. Irwin leads the Intelligent Systems and Control Research group and is Director of the University Virtual Engineering Centre at Queen's University Belfast. He has been elected as a Fellow of the Royal Academy of Engineering, a Member of the Royal Irish Academy and is a Chartered Engineer, an IEEE Fellow, a Fellow of the IEE and a Fellow of the Institute of Measurement and Control. His research covers identification, monitoring, and control, including neural networks, fuzzy neural systems and multivariate statistics and has published over 350 research papers and 12 edited books. He is currently working on wireless networked control systems, fault diagnosis of internal combustion engines and novel techniques for fast temperature measurement and was a Technical Director of Anex6 Ltd., a spin-out company from his group specialising in process monitoring. He has been awarded a number of prizes including four IEE Premiums, a Best Paper award from the Czech Academy of Sciences and the 2002 Honeywell International Medal from the Institute of Measurement and Control. International recognitions include Honorary Professor at Harbin Institute of Technology and Shandong University, and Visiting Professor at Shanghai University. He is a former Editor-in-Chief of the IFAC Journal Control Engineering Practice and past chair of the UK Automatic Control Council. He currently chairs the IFAC Publications Committee and serves on the editorial boards of several journals.