Particle swarm optimization aided orthogonal forward regression for unified data modeling

Article

Published Version

Particle Swarm Optimization Aided Orthogonal Forward Regression for Unified Data Modeling

Sheng Chen, *Fellow, IEEE,* Xia Hong, *Senior Member, IEEE,* and Chris J. Harris

***Abstract*—We propose a unified data modeling approach that is equally applicable to supervised regression and classification applications, as well as to unsupervised probability density function estimation. A particle swarm optimization (PSO) aided orthogonal forward regression (OFR) algorithm based on leave-one-out (LOO) criteria is developed to construct parsimonious radial basis function (RBF) networks with tunable nodes. Each stage of the construction process determines the center vector and diagonal covariance matrix of one RBF node by minimizing the LOO statistics. For regression applications, the LOO criterion is chosen to be the LOO mean square error, while the LOO misclassification rate is adopted in two-class classification applications. By adopting the Parzen window estimate as the desired response, the unsupervised density estimation problem is transformed into a constrained regression problem. This PSO aided OFR algorithm for tunable-node RBF networks is capable of constructing very parsimonious RBF models that generalize well, and our analysis and experimental results demonstrate that the algorithm is computationally even simpler than the efficient regularization assisted orthogonal least square algorithm based on LOO criteria for selecting fixed-node RBF models. Another significant advantage of the proposed learning procedure is that it does not have learning hyperparameters that have to be tuned using costly cross validation. The effectiveness of the proposed PSO aided OFR construction procedure is illustrated using several examples taken from regression and classification, as well as density estimation applications.**

***Index Terms*—Classification, density estimation, evolutionary computation, leave-one-out cross validation, orthogonal forward regression, particle swarm optimization, radial basis function network, regression.**

I. Introduction

MODELING from data is of fundamental importance in all walks of engineering. Various data modeling applications can be classified into three categories, namely, regression [1]–[3], classification [4]–[6], and probability density function (PDF) estimation [7]–[9]. In regression, the task is to establish a model that links the observation data to their target function or desired output values. The goodness of a regression model is judged by its generalization performance, which can be conveniently determined by the test mean square error (MSE) on the data not used in training the model. Like regression, classification is also a supervised learning problem. However, the desired output is discrete valued, e.g., binary in two-class classification problems, and the goodness of a classifier is determined by its test error probability or misclassification rate. Despite these differences, classifier construction can be expressed in the same framework of regression modeling. The third class of data modeling, namely, PDF estimation, is very different in nature from regression and classification. The task of PDF estimation is to infer the underlying probability distribution that generates the observations. Because the true target function, the underlying PDF, is not available, this is an unsupervised learning problem and can only be carried out based on, often noisy, observation data. Nevertheless, this unsupervised task can be "transformed" into a supervised one, for example, by computing the empirical distribution function (EDF) from the observation data and using it as the target function for the cumulative distribution function (CDF) of the PDF estimation. Thus, a unified regression framework can be adopted for all three classes of data modeling problems.

The radial basis function (RBF) network has found wide-ranging data modeling applications in diverse engineering fields [10]–[26]. The parameters of the RBF network, which include the center vectors and variances or covariance matrices of its hidden nodes, as well as the weights that connect the RBF nodes to the network output, can be trained together via nonlinear optimization using gradient based algorithms [27]–[31], the expectation-maximization (EM) algorithm [32], [33], or various evolutionary algorithms [34]–[38]. Generally speaking, learning based on such a nonlinear approach is computationally expensive and may encounter the problem of local minima. Additionally, the network structure or the number of RBF nodes has to be determined via other means, typically based on cross validation. Alternatively, clustering algorithms can be applied to find the RBF center vectors, as well as the associated basis function variances [39]–[42]. This leaves the RBF weights to be determined by the usual linear least squares solution. Again, the number of clusters has to be determined via cross validation. An alternative RBF network selection criterion is based on sensitivity analysis [43]. However, one of the most popular approaches for constructing RBF models for regression is to formulate the problem as a linear learning problem by considering the training input data points as candidate RBF centers and employing a common variance

Manuscript received January 2, 2009; revised October 16, 2009. Date of publication April 19, 2010; date of current version July 30, 2010. The work of S. Chen was supported by the U.K. Royal Society and Royal Academy of Engineering.

S. Chen and C. J. Harris are with the School of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, U.K. (e-mail: sqc@ecs.soton.ac.uk; cjh@ecs.soton.ac.uk).

X. Hong is with the School of Systems Engineering, University of Reading, Reading RG6 6AY, U.K. (e-mail: x.hong@reading.ac.uk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TEVC.2009.2035921

for every RBF node. A parsimonious RBF network is then identified using the orthogonal least squares (OLS) algorithm [44]–[48]. A similar linear learning approach is adopted in the support vector machine (SVM) and other sparse kernel modeling methods [49]–[55], which fix the kernel centers to the training input data points and adopt a common kernel variance for every kernel. A sparse kernel model is then sought by making as many kernel weights to near zero as possible based on appropriate optimization criteria. Hybrid learning for RBF networks has also been proposed. For example, an initial RBF network can be constructed using a linear learning method and the resulting RBF model is then further optimized using nonlinear optimization [56].

The SVM and related sparse kernel methods are equally applicable to regression, classification, and density estimation [57]–[62]. The OLS approach has also been extended to all three types of data modeling. In particular, the regularization assisted OLS (ROLS) algorithm based on minimizing the leave-one-out (LOO) MSE [48] offers a state-of-the-art for regression application. The work [63] has developed a ROLS algorithm based on minimizing the LOO misclassification rate for two-class classification applications. Owing to orthogonal decomposition, the LOO misclassification rate can be computed efficiently, just as in the case of the LOO MSE for regression, and this ensures a fast RBF classifier construction. A sparse density estimation technique has been developed in [64], which uses the ROLS algorithm based on the LOO MSE to select a parsimonious density estimate and computes the associated kernel weights using the multiplicative nonnegative quadratic programming (MNQP) algorithm of [65]. Our experimental results [48], [63], [64] have demonstrated that the ROLS-LOO algorithm compares favorably with many other existing fixed-node RBF or kernel modeling methods for data modeling, in terms of model sparsity and generalization performance. One aspect of the linear learning approach for RBF models, which deserves consideration, is the determination of the common RBF variance. Since this variance is not provided by the learning algorithms, it must be treated as a hyperparameter and determined via cross validation. For kernel modeling methods, the learning algorithm's hyperparameters also have to be determined by cross validation. For example, for the SVM algorithm with the ε insensitive cost function [50], the kernel variance as well as the regularization and error-band parameters must be specified.

To avoid using costly cross validation for determining the RBF variance as is required by the above-mentioned linear learning approach for fixed-node RBF or kernel modeling methods, Chen *et al* [66] adopt a strategy of fitting a diagonal covariance matrix to each candidate RBF node, which as usual is centered at a training input point, by optimizing the correlation criterion between the training data and the candidate RBF regressor. Because fitting the covariance matrix of a RBF node by maximizing the correlation criterion is a nonlinear and nonconvex optimization task, a global search algorithm known as the repeated weighted boosting search (RWBS) [67] is employed to perform this optimization. This RBF regression does not need to learn any hyperparameter. However, it is required to fit a diagonal RBF covariance matrix to every training data point, which can be computationally costly, particularly for a large training data set. More effective construction algorithms for the RBF network with tunable nodes for regression and classification are proposed in [68], [69], where each RBF unit has a tunable center vector as well as an adjustable diagonal covariance matrix. An orthogonal forward regression (OFR) procedure is employed to optimize the RBF units one by one by minimizing the LOO statistics. Specifically, each stage of the model construction procedure determines one RBF unit's center vector and diagonal covariance matrix using the RWBS [67]. Because the RBF centers are not restricted to the training input points and each RBF node has an individually adjusted covariance matrix, this OFR-LOO algorithm can produce sparser representations with excellent generalization capability, in comparison with the existing fixed-node RBF modeling methods. This tunable-node approach is also very different from those RBF learning methods based on nonlinear optimization, as it does not attempt to optimize all the RBF units together, which could be a too large and complicated nonlinear optimization task. A drawback of the algorithms [68], [69] for constructing tunable-node RBF models is that they may require more computation in model construction than the algorithms [48], [63] for selecting fixed-node RBF models.

This paper proposes a particle swarm optimization (PSO) aided OFR algorithm to construct tunable-node RBF models for unified data modeling that includes regression, classification, and density estimation. PSO [70], [71] constitutes a population based stochastic optimization technique, which was inspired by the social behavior of bird flocks or fish schools. The algorithm commences with random initialization of a swarm of individuals, referred to as particles, within the specific problem's search space. It then endeavors to find a globally optimum solution by gradually adjusting the trajectory of each particle toward its own best location and toward the best position of the entire swarm at each optimization step. The PSO method is popular owing to its simplicity in implementation, ability to rapidly converge to a "reasonably good" solution and to "steer clear" of local minima. It has been successfully applied to wide-ranging optimization problems [37], [38], [72]–[117]. Because of the simplicity and efficiency of the PSO method, the proposed PSO aided OFR algorithm based on LOO statistics for constructing tunable-node RBF models not only produces smaller RBF models with better generalization capability but also requires less computation in model construction, in comparison with the efficient ROLS-LOO algorithm for selecting fixed-node RBF models [48], [63], [64]. Regression, classification and density estimation examples are used in our experimental study to demonstrate that the proposed PSO-aided OFR-LOO algorithm offers a state-of-the-art for unified data modeling practice. Hence, the novel contribution of this paper is that we develop a PSO-aided OFR-LOO algorithm for constructing tunable-node RBF models. This PSO-aided tunable RBF modeling approach offers significant advantages over the best existing algorithms for selecting fixed-node RBF models, in terms of achieving smaller model size and better generalization performance as well as imposing lower computational complexity.

II. Unified Data Modeling Using the Tunable RBF Network

Regression, classification, and PDF estimation can be unified under a regression framework of data modeling based on appropriate modeling criteria, where the model is obtained as a linear combination of a set of tunable RBF nodes or kernels. For density estimation, additionally, each RBF kernel is nonnegative and the area under the RBF kernel is unity.

A. Regression

Consider the generic regression modeling problem based on the set of N pairs of training data, $D_N = \{(\mathbf{x}_k, y_k)\}_{k=1}^N$, with the RBF network defined in

$$y_k = \hat{y}_k + e_k = \sum_{i=1}^{M} w_i g_i(\mathbf{x}_k) + e_k = \mathbf{g}^T(k)\mathbf{w} + e_k \tag{1}$$

where the input $\mathbf{x}_k \in \mathcal{R}^m$, the desired output $y_k \in \mathcal{R}$, $\hat{y}_k$ denotes the RBF model output, $e_k = y_k - \hat{y}_k$ is the modeling error, M is the number of RBF units, $\mathbf{w} = [w_1 \ w_2 \cdots w_M]^T$ is the RBF weight vector, $g_i(\bullet)$ for $1 \le i \le M$ denote the RBF regressors, and $\mathbf{g}(k) = [g_1(\mathbf{x}_k) \ g_2(\mathbf{x}_k) \cdots g_M(\mathbf{x}_k)]^T$. We will consider the generic RBF regressor of the form

$$\begin{aligned} g_i(\mathbf{x}) &= K(\mathbf{x}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \\ &= K\left(\sqrt{\left(\mathbf{x}-\boldsymbol{\mu}_i\right)^T \boldsymbol{\Sigma}_i^{-1} \left(\mathbf{x}-\boldsymbol{\mu}_i\right)}\right) \end{aligned} \tag{2}$$

where $\boldsymbol{\mu}_i$ is the center vector of the ith RBF unit, the ith RBF covariance matrix takes a diagonal form $\boldsymbol{\Sigma}_i = \text{diag}\{\sigma_{i,1}^2, \cdots, \sigma_{i,m}^2\}$, and $K(\bullet)$ is the chosen basis or kernel function. Many types of basis function can be used and a commonly adopted one is the Gaussian function of the form

$$K(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{m/2} \det^{1/2} |\boldsymbol{\Sigma}|} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})}. \tag{3}$$

For regression and classification, the factor $1/\big((2\pi)^{m/2} \det^{1/2} |\boldsymbol{\Sigma}|\big)$ can be combined into the RBF weight w.

By defining $\mathbf{y} = [y_1 \ y_2 \cdots y_N]^T$, $\mathbf{e} = [e_1 \ e_2 \cdots e_N]^T$, and $\mathbf{G} = [\mathbf{g}_1 \ \mathbf{g}_2 \cdots \mathbf{g}_M]$ with

$$\mathbf{g}_l = [g_l(\mathbf{x}_1) \ g_l(\mathbf{x}_2) \cdots g_l(\mathbf{x}_N)]^T, \ 1 \le l \le M \tag{4}$$

the regression model in (1) over the training data set D_N can be written in matrix form as

$$\mathbf{y} = \mathbf{G}\mathbf{w} + \mathbf{e}. \tag{5}$$

Note that $\mathbf{g}_k$ denotes the kth column of $\mathbf{G}$ while $\mathbf{g}^T(k)$ is the kth row of $\mathbf{G}$. Let an orthogonal decomposition of the regression matrix $\mathbf{G}$ be $\mathbf{G} = \boldsymbol{\Phi}\mathbf{A}$, where

$$\mathbf{A} = \begin{bmatrix} 1 & \alpha_{1,2} & \cdots & \alpha_{1,M} \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \alpha_{M-1,M} \\ 0 & \cdots & 0 & 1 \end{bmatrix} \tag{6}$$

and

$$\boldsymbol{\Phi} = [\boldsymbol{\phi}_1 \ \boldsymbol{\phi}_2 \cdots \boldsymbol{\phi}_M] \tag{7}$$

with the orthogonal columns that satisfy $\boldsymbol{\phi}_i^T \boldsymbol{\phi}_j = 0$, if $i \neq j$. The orthogonalization can, for example, be performed by the Gram–Schmidt procedure [44]. The regression model in (5) can alternatively be expressed as

$$\mathbf{y} = \boldsymbol{\Phi}\boldsymbol{\theta} + \mathbf{e} \tag{8}$$

where the weight vector $\boldsymbol{\theta} = [\theta_1 \ \theta_2 \cdots \theta_M]^T$ defined in the orthogonal model space $\boldsymbol{\Phi}$ satisfies the triangular system, $\mathbf{A}\mathbf{w} = \boldsymbol{\theta}$. Since the space spanned by the original model bases $g_i(\bullet)$, $1 \le i \le M$, is identical to the space spanned by the orthogonal model bases, the RBF model output is equivalently expressed by

$$\hat{y}_k = \boldsymbol{\phi}^T(k)\boldsymbol{\theta} \tag{9}$$

where $\boldsymbol{\phi}^T(k) = [\phi_1(k) \ \phi_2(k) \cdots \phi_M(k)]$ is the kth row of $\boldsymbol{\Phi}$.

B. Classification

For notational simplification, we restrict to the two-class classification problem with the given training data set $D_N = \{\mathbf{x}_k, y_k\}_{k=1}^N$, where $\mathbf{x}_k \in \mathcal{R}^m$ is an m-dimensional pattern vector and $y_k \in \{-1, +1\}$ is the class label for $\mathbf{x}_k$. The task is to construct a RBF classifier of the form

$$\tilde{y}_k = \text{sgn}\,(\hat{y}_k) \tag{10}$$

with the classifier

$$\hat{y}_k = \sum_{i=1}^{M} w_i g_i(\mathbf{x}_k) \tag{11}$$

where $\tilde{y}_k$ is the estimated class label for $\mathbf{x}_k$ and

$$\text{sgn}(y) = \begin{cases} -1, & y \le 0 \\ +1, & y > 0. \end{cases} \tag{12}$$

By defining the modeling error as $e_k = y_k - \hat{y}_k$, the classification model over the training data set D_N can be expressed in the regression model of (5), namely, $\mathbf{y} = \mathbf{G}\mathbf{w}+\mathbf{e}$, or equivalently in the orthogonal regression model of (8), i.e., $\mathbf{y} = \boldsymbol{\Phi}\boldsymbol{\theta} + \mathbf{e}$, where all the relevant notations are as defined in Section II-A. Thus, the classifier construction can be expressed in the same regression modeling framework of Section II-A, and the only difference with regression is that the target function y_k in classification applications is discrete valued. In particular, for the two-class classification problem, y_k is binary.

C. Density Estimation

Given the finite data sample set $D_N = \{\mathbf{x}_k\}_{k=1}^N$ drawn from a density $p(\mathbf{x})$, where $\mathbf{x}_k \in \mathcal{R}^m$, the task is to estimate the unknown density $p(\mathbf{x})$ using the density estimate of the form

$$\hat{p}(\mathbf{x}) = \sum_{i=1}^{M} w_i g_i(\mathbf{x}) \tag{13}$$

with the constraints

$$w_i \ge 0, \ 1 \le i \le M \tag{14}$$

and

$$\mathbf{w}^T \mathbf{1}_M = 1 \tag{15}$$

where $\mathbf{1}_M$ denotes the vector of ones with dimension M. The basis function used in this study is chosen to be the Gaussian function as given in (3). However, many other basis functions can also be used in the density estimate of (13). This density estimation is an unsupervised learning problem, as the desired response for the training data points $\mathbf{x}_k$ is unknown. We follow the approach of [64] to transform it into a supervised learning problem.

Given the training set D_N, the well-known Parzen window (PW) estimate [7] is obtained as

$$\begin{aligned}\hat{p}_{\text{Par}}(\mathbf{x}) &= \sum_{k=1}^{N} \frac{1}{N} K(\mathbf{x}; \mathbf{x}_k, \rho_{\text{Par}}^2) \\ &= \sum_{k=1}^{N} \frac{1}{N} K\left(\sqrt{(\mathbf{x}-\mathbf{x}_k)^T(\mathbf{x}-\mathbf{x}_k)/\rho_{\text{Par}}^2}\right) \qquad (16)\end{aligned}$$

where ρ_{Par} is the kernel width and $K(\bullet)$ is usually chosen as the Gaussian kernel function. The kernel width ρ_{Par} is typically determined via cross validation. A disadvantage associated with the PW estimate is its high computational cost of the point density estimate for a future data sample, as the PW estimate employs the full training data sample set in defining density estimate for subsequent observation. The PW estimate, however, is simple to construct and remarkably accurate [7]. Moreover, it can be regarded as the "observation" of the true density, namely

$$\hat{p}_{\text{Par}}(\mathbf{x}) = p(\mathbf{x}) + e(\mathbf{x}) \qquad (17)$$

where $e(\mathbf{x})$ can be viewed as the "observational noise" at the point $\mathbf{x}$. Thus, the density estimation problem in (13) can be viewed as a "supervised" regression problem with the PW estimate as the "desired response," subject to the constraints given in (14) and (15).

Specifically, the PW estimate in (16) and the generic density estimate in (13) at the training data point $\mathbf{x}_k$ are, respectively, defined as $y_k = \hat{p}_{\text{Par}}(\mathbf{x}_k)$ and $\hat{y}_k = \hat{p}(\mathbf{x}_k)$. Further, denote the associated modeling error at $\mathbf{x}_k$ as $e_k = y_k - \hat{y}_k$. Then the generic density estimation problem in (13) is expressed in the same regression modeling framework of (5), that is, $\mathbf{y} = \mathbf{G}\mathbf{w} + \mathbf{e}$, subject to the nonnegative constraint in (14) and the unity constraint in (15), where all the relevant notations have been defined in Section II-A. The regression model in (5) can of course be written equivalently in the form of (8), namely, $\mathbf{y} = \mathbf{\Phi}\boldsymbol{\theta} + \mathbf{e}$.

Appendix A explains in detail the two common approaches for "converting" the unsupervised density estimation problem into a supervised regression problem.

III. PSO Aided OFR for the Tunable RBF Network

As mentioned in the previous section, regression, classification, and PDF estimation can all be unified in a common regression modeling framework. We propose a PSO aided OFR algorithm for efficient construction of the tunable-node RBF model in this unified regression framework. First, the PSO method adopted is briefly summarized.

A. PSO

Consider solving the generic optimization problem

$$\mathbf{u}_{\text{opt}} = \arg \min_{\mathbf{u} \in \prod_{j=1}^{m'} \mathsf{P}_j} F(\mathbf{u}) \qquad (18)$$

using the PSO algorithm [70], [71], where $\mathbf{u} = [u_1\ u_2 \cdots u_{m'}]^T$ is the m'-dimensional parameter vector to be optimized, $F(\bullet)$ is the cost function, and

$$\prod_{j=1}^{m'} \mathsf{P}_j = \prod_{j=1}^{m'} [P_{j,\min},\ P_{j,\max}] \qquad (19)$$

defines the search space. A swarm of particles, $\{\mathbf{u}_i^{(l)}\}_{i=1}^{S}$, that represent potential solutions are "flying" in the search space $\prod_{j=1}^{m'} \mathsf{P}_j$, where S is the swarm size and index l denotes the iteration step. The algorithm is summarized as follows.

a) *Swarm Initialization:* Set the iteration index $l = 0$ and randomly generate $\{\mathbf{u}_i^{(l)}\}_{i=1}^{S}$ in the search space $\prod_{j=1}^{m'} \mathsf{P}_j$.

b) *Swarm Evaluation:* Each particle $\mathbf{u}_i^{(l)}$ has a cost $F(\mathbf{u}_i^{(l)})$ associated with it. Each particle $\mathbf{u}_i^{(l)}$ remembers its best position visited so far, denoted as $\mathbf{pb}_i^{(l)}$, which provides the cognitive information. Every particle also knows the best position visited so far among the entire swarm, denoted as $\mathbf{gb}^{(l)}$, which provides the social information. The cognitive information $\{\mathbf{pb}_i^{(l)}\}_{i=1}^{S}$ and the social information $\mathbf{gb}^{(l)}$ are updated at each iteration as follows:

```
For ($i = 1$; $i \le S$; $i$++)
    If ($F(\mathbf{u}_i^{(l)}) < F(\mathbf{pb}_i^{(l)})$)   $\mathbf{pb}_i^{(l)} = \mathbf{u}_i^{(l)}$;
End for;
$i^* = \arg \min_{1 \le i \le S} F(\mathbf{pb}_i^{(l)})$;
If ($F(\mathbf{pb}_{i^*}^{(l)}) < F(\mathbf{gb}^{(l)})$)   $\mathbf{gb}^{(l)} = \mathbf{pb}_{i^*}^{(l)}$;
```

c) *Swarm Update:* Each particle $\mathbf{u}_i^{(l)}$ has a velocity, denoted as $\mathbf{v}_i^{(l)}$, to direct its "flying." The velocity and position of the ith particle are updated in each iteration according to

$$\begin{aligned}\mathbf{v}_i^{(l+1)} &= w_I * \mathbf{v}_i^{(l)} + rand() * c_1 * (\mathbf{pb}_i^{(l)} - \mathbf{u}_i^{(l)}) \\ &\quad + rand() * c_2 * (\mathbf{gb}^{(l)} - \mathbf{u}_i^{(l)}) \qquad (20) \\ \mathbf{u}_i^{(l+1)} &= \mathbf{u}_i^{(l)} + \mathbf{v}_i^{(l+1)} \qquad (21)\end{aligned}$$

where w_I is the inertia weight, $rand()$ denotes the uniform random number between 0 and 1, and c_1 and c_2 are the two acceleration coefficients. In order to avoid excessive roaming of particles beyond the search space [75], a velocity space

$$\prod_{j=1}^{m'} \mathsf{V}_j = \prod_{j=1}^{m'} [-V_{j,\max},\ V_{j,\max}] \qquad (22)$$

is imposed on $\mathbf{v}_i^{(l+1)}$ so that

```
If ($\mathbf{v}_i^{(l+1)}|_j > V_{j,\max}$)   $\mathbf{v}_i^{(l+1)}|_j = V_{j,\max}$;
If ($\mathbf{v}_i^{(l+1)}|_j < -V_{j,\max}$)   $\mathbf{v}_i^{(l+1)}|_j = -V_{j,\max}$;
```

where $\mathbf{v}|_j$ denotes the jth element of $\mathbf{v}$. Moreover, if the velocity as given in (20) approaches zero, it is reinitialized proportional to $V_{j,\max}$ with a small factor γ

$$\text{If } (\mathbf{v}_i^{(l+1)}|_j == 0)\ \mathbf{v}_i^{(l+1)}|_j = \pm rand() * \gamma * V_{j,\max}; \quad (23)$$

Similarly, each element of $\mathbf{u}_i^{(l+1)}$ is checked to ensure that it stays inside the search space

$$\text{If } (\mathbf{u}_i^{(l+1)}|_j > P_{j,\max}) \quad \mathbf{u}_i^{(l+1)}|_j = P_{j,\max};$$

$$\text{If } (\mathbf{u}_i^{(l+1)}|_j < P_{j,\min}) \quad \mathbf{u}_i^{(l+1)}|_j = P_{j,\min};$$

Alternatively, if a particle is outside the search space, it may be moved back inside the search space to a random position, rather than forcing it to stay at the border [75].

d) *Termination Condition Check:* If the maximum number of iterations, $I_{\max}$, is reached, terminate the algorithm with the solution $\mathbf{gb}^{(I_{\max})}$; otherwise, set $l = l + 1$ and go to Step b).

Ratnaweera and co-authors [73] reported that using a time varying acceleration coefficient (TVAC) enhances the performance of PSO. We adopt this mechanism, in which c_1 is reduced from 2.5 to 0.5 and c_2 varies from 0.5 to 2.5 during the iterative procedure

$$\begin{aligned} c_1 &= (0.5 - 2.5) * l/I_{\max} + 2.5 \\ c_2 &= (2.5 - 0.5) * l/I_{\max} + 0.5. \end{aligned} \quad (24)$$

The reason for good performance of this TVAC mechanism can be explained as follows. At the initial stages, a large cognitive component and a small social component help particles to wander around or better exploit the search space, avoiding local minima. In the later stages, a small cognitive component and a large social component help particles to converge quickly to a global minimum. We experiment with three choices of the inertia weight, namely, $w_I = 0$ as suggested in [73], which removes the influence of the previous velocity, w_I set to a small positive constant, and $w_I = rand()$. The third choice sets w_I as a uniform random number between 0 and 1 at each iteration and it typically performs better than the other two choices in our application.

The search space as given in (19) is defined by the specific problem to be solved, and the velocity limit can be set to

$$V_{j,\max} = 0.5 * (P_{j,\max} - P_{j,\min}). \quad (25)$$

An appropriate value of the small control factor γ in (23) for avoiding zero velocity is empirically found to be $\gamma = 0.1$ for our application. Our experimental results given in Section IV show that a swarm size in the range of $S = 10$ to $S = 20$ is appropriate for our application. We have also found empirically that often the maximum number of iterations can be chosen as $I_{\max} = 20$. Thus, the PSO method is generally very efficient in terms of the total required computational complexity. Let the computational complexity of evaluating the cost function $F(\mathbf{u})$ once be C_{single}. The total complexity of the above-mentioned PSO algorithm in solving the optimization problem defined in (18) is

$$C_{\text{total}} = I_{\max} \times S \times C_{\text{single}}. \quad (26)$$

B. Regression Model Construction

Consider the modeling process that has produced the n-unit RBF model. Denote the constructed n model columns as $\mathbf{\Phi}_n = [\boldsymbol{\phi}_1\ \boldsymbol{\phi}_2 \cdots \boldsymbol{\phi}_n]$, the kth model output of this n-unit model identified using the entire training set D_N as $\hat{y}_k^{[n]} = \sum_{i=1}^{n} \theta_i \phi_i(k)$, and the corresponding kth modeling error as $e_k^{[n]} = y_k - \hat{y}_k^{[n]}$. If we "remove" the kth data point from D_N and use the remaining $N - 1$ data points to identify the n-unit RBF network instead, the "test" error of the resulting model can be calculated on the data point removed from training. This LOO modeling error, denoted as $e_k^{[n,-k]}$, is given by [2]

$$e_k^{[n,-k]} = e_k^{[n]} / \eta_k^{[n]} \quad (27)$$

where $\eta_k^{[n]}$ is the LOO error weighting [2]. The LOO MSE for the n-unit RBF network is then defined by

$$J_n = \frac{1}{N} \sum_{k=1}^{N} \left(e_k^{[n,-k]} \right)^2. \quad (28)$$

This LOO MSE is a measure of the model generalization capability [2], [118]. For the orthogonal model of (8), the computation of the LOO criterion J_n is very efficient because $e_k^{[n]}$ and $\eta_k^{[n]}$ can be computed recursively using [48], [119]

$$e_k^{[n]} = y_k - \sum_{i=1}^{n} \theta_i \phi_i(k) = e_k^{[n-1]} - \theta_n \phi_n(k) \quad (29)$$

and

$$\eta_k^{[n]} = 1 - \sum_{i=1}^{n} \frac{\phi_i^2(k)}{\boldsymbol{\phi}_i^T \boldsymbol{\phi}_i + \lambda} = \eta_k^{[n-1]} - \frac{\phi_n^2(k)}{\boldsymbol{\phi}_n^T \boldsymbol{\phi}_n + \lambda} \quad (30)$$

respectively, where $\lambda \geq 0$ is a small regularization parameter. The regularization parameter λ occurs in (30) because the ROLS solution for weights is used [48]. The concept of LOO cross validation, as well as the derivation of (29) and (30), are detailed in Appendix B.

The OFR algorithm constructs the RBF nodes one by one by minimizing the LOO MSE J_n. Specifically, at the nth stage of the construction procedure, the nth RBF node is determined by minimizing J_n with respect to the node's center vector $\boldsymbol{\mu}_n$ and diagonal covariance matrix $\mathbf{\Sigma}_n$

$$\min_{\boldsymbol{\mu}_n, \mathbf{\Sigma}_n} J_n \left(\boldsymbol{\mu}_n, \mathbf{\Sigma}_n \right). \quad (31)$$

The construction procedure is automatically terminated when

$$J_M \leq J_{M+1} \quad (32)$$

yielding an M-node RBF network. Note that the LOO criterion J_n is at least locally convex with respect to the model size n, that is, there exists an optimal size M such that, for $n \leq M$, J_n decreases as the model size n increases while the condition in (32) holds [119]. After this OFR-LOO model construction, a very small model set $\mathbf{G}$, containing only M units, is obtained. At this stage, the ROLS-LOO algorithm for the fixed-node RBF model [48] may be applied to further reduce the model size and to automatically update an individual regularization parameter for each weight. This refinement requires a very small amount of computation, as $\mathbf{G}$ is completely specified with only a few columns. Note that in the OFR-LOO algorithm, the regularization parameter λ can simply be set to zero (no regularization) or a very small value (e.g., 10^{-6}). The refinement using the ROLS-LOO will automatically optimize each regularization parameter for individual weights without involving cross validation [48]. Our previous experience [68], [69] shows that for regression applications this refinement involving the ROLS-LOO is beneficial but for classification it is unnecessary (no further reduction in model size). It is also unnecessary to apply this ROLS-LOO refinement at the end of OFR-LOO construction for the tunable-node RBF density estimate, as the MNQP algorithm used in computing the weights of the constructed density model may further reduce the model size [62], [65].

How efficient this OFR construction procedure is for tunable RBF models depends crucially on the algorithm used in solving the optimization problem defined in (31). We propose to use the PSO algorithm of Section III-A to perform this optimization task. Let $\mathbf{u}$ be the parameter vector that contains $\boldsymbol{\mu}_n$ and $\boldsymbol{\Sigma}_n$. Then the dimension of $\mathbf{u}$ is $m' = 2m$ and the cost function is simply $F(\mathbf{u}) = J_n(\mathbf{u})$. The search space defined in (19) is specified by

$$\left.\begin{aligned} P_{j,\min} &= \min_{1\le k\le N}\{\mathbf{x}_k|_j\},\\ P_{j,\max} &= \max_{1\le k\le N}\{\mathbf{x}_k|_j\},\end{aligned}\right\} 1 \le j \le m \tag{33}$$

$$P_{j,\min} = \sigma^2_{\min} \text{ and } P_{j,\max} = \sigma^2_{\max},\ \ m+1 \le j \le m' \tag{34}$$

where $\sigma^2_{\min}$ and $\sigma^2_{\max}$ are the chosen lower and upper bounds for the RBF variances $\sigma^2_{n,j}$, respectively. Given the following initial conditions

$$\left.\begin{aligned} e_k^{[0]} = y_k \text{ and } \eta_k^{[0]} = 1,\ 1 \le k \le N\\ J_0 = \tfrac{1}{N}\mathbf{y}^T\mathbf{y} = \tfrac{1}{N}\sum_{k=1}^{N} y_k^2\end{aligned}\right\} \tag{35}$$

and the maximum number of iterations $I_{\max}$ as well as the swarm size S, the PSO aided OFR algorithm for constructing the nth RBF node takes the PSO procedure summarized in Section III-A with the following detailed cost evaluation in Step b):

The Orthogonalization and Cost Function Evaluation:

1) For $1 \le i \le S$, generate $\mathbf{g}_n^{i)}$ from $\mathbf{u}_i^{(l)}$, the candidates for the nth model column, according to (4), and orthogonalize them using the Gram–Schmidt orthogonalization [44]

$$\alpha_{j,n}^{i)} = \boldsymbol{\phi}_j^T\mathbf{g}_n^{i)}/\boldsymbol{\phi}_j^T\boldsymbol{\phi}_j,\ \ 1 \le j < n \tag{36}$$

$$\boldsymbol{\phi}_n^{i)} = \mathbf{g}_n^{i)} - \sum_{j=1}^{n-1}\alpha_{j,n}^{i)}\boldsymbol{\phi}_j \tag{37}$$

$$\theta_n^{i)} = \left(\boldsymbol{\phi}_n^{i)}\right)^T\mathbf{y}/\left(\left(\boldsymbol{\phi}_n^{i)}\right)^T\boldsymbol{\phi}_n^{i)} + \lambda\right). \tag{38}$$

2) For $1 \le i \le S$, calculate the LOO cost for each $\mathbf{u}_i^{(l)}$

$$e_k^{[n]}(i) = e_k^{[n-1]} - \phi_n^{i)}(k)\theta_n^{i)},\ \ 1 \le k \le N \tag{39}$$

$$\eta_k^{[n]}(i) = \eta_k^{[n-1]} - \frac{\left(\phi_n^{i)}(k)\right)^2}{\left(\boldsymbol{\phi}_n^{i)}\right)^T\boldsymbol{\phi}_n^{i)} + \lambda},\ \ 1 \le k \le N \tag{40}$$

$$F\left(\mathbf{u}_i^{(l)}\right) = \frac{1}{N}\sum_{k=1}^{N}\left(e_k^{[n]}(i)/\eta_k^{[n]}(i)\right)^2 \tag{41}$$

where $\phi_n^{i)}(k)$ is the kth element of $\boldsymbol{\phi}_n^{i)}$.

When the PSO algorithm terminates, it yields the solution $\mathbf{gb}^{(I_{\max})}$, i.e., the center vector $\boldsymbol{\mu}_n$ and diagonal covariance matrix $\boldsymbol{\Sigma}_n$ of the nth RBF node. The algorithm also generates the nth model column $\mathbf{g}_n$, the orthogonalization coefficients $\alpha_{j,n}$, $1 \le j < n$, the corresponding orthogonal model column $\boldsymbol{\phi}_n$, and the weight θ_n, as well as the n-node modeling errors $e_k^{[n]}$ and the associated LOO error weightings $\eta_k^{[n]}$ for $1 \le k \le N$. The next stage of the model construction can then commence, and the construction is automatically terminated when the condition in (32) is met.

C. Classifier Construction

The same LOO cross validation concept [2], as was used in Section III-B for regression modeling, can be adopted to provide a measure of a classifier's generalization capability. Denote the test output of the LOO n-node RBF classifier, evaluated at the kth data sample of D_N which has been "removed" from training, as $\hat{y}_k^{[n,-k]}$. The associated LOO signed decision variable is defined by

$$s_k^{[n,-k]} = \text{sgn}(y_k)\hat{y}_k^{[n,-k]} = y_k\hat{y}_k^{[n,-k]} \tag{42}$$

where $\text{sgn}(y_k) = y_k$ since $y_k \in \{-1, +1\}$. The LOO misclassification rate is defined as [63]

$$J_n = \frac{1}{N}\sum_{k=1}^{N}\mathcal{I}_d\left(s_k^{[n,-k]}\right) \tag{43}$$

where the indicator function $\mathcal{I}_d$ is defined by

$$\mathcal{I}_d(y) = \begin{cases}1, & y \le 0\\ 0, & y > 0.\end{cases} \tag{44}$$

The LOO misclassification rate J_n can be evaluated efficiently because $s_k^{[n,-k]}$ can be calculated rapidly [63]. Specifically, the LOO n-node modeling error is expressed as

$$y_k - \hat{y}_k^{[n,-k]} = \left(y_k - \hat{y}_k^{[n]}\right)/\eta_k^{[n]}. \tag{45}$$

Multiplying both sides of (45) with y_k and applying $y_k^2 = 1$ yield

$$1 - s_k^{[n,-k]} = \left(1 - y_k\hat{y}_k^{[n]}\right)/\eta_k^{[n]} \tag{46}$$

which leads to

$$\begin{aligned} s_k^{[n,-k]} &= \left(\sum_{i=1}^{n} y_k\theta_i\phi_i(k) - \sum_{i=1}^{n}\frac{\phi_i^2(k)}{\boldsymbol{\phi}_i^T\boldsymbol{\phi}_i + \lambda}\right)/\eta_k^{[n]}\\ &= \psi_k^{[n]}/\eta_k^{[n]}. \end{aligned} \tag{47}$$

The recursive formula for the LOO error weighting $\eta_k^{[n]}$ is given in (30), while $\psi_k^{[n]}$ can be represented using the following recurrence relation:

$$\psi_k^{[n]} = \psi_k^{[n-1]} + y_k\theta_n\phi_n(k) - \frac{\phi_n^2(k)}{\boldsymbol{\phi}_n^T\boldsymbol{\phi}_n + \lambda}. \tag{48}$$

As in the regression case, the OFR algorithm constructs the classifier's RBF units one by one by minimizing the LOO misclassification rate J_n defined in (43). At the nth stage of construction, the nth RBF node is determined by minimizing J_n with respect to $\boldsymbol{\mu}_n$ and $\boldsymbol{\Sigma}_n$. The procedure is automatically terminated when $J_M \leq J_{M+1}$, yielding an M-node RBF classifier.

The PSO algorithm used to construct the nth RBF node has a similar form to the regression case with small modifications. Specifically, the initial condition of (35) is replaced by

$$\psi_k^{[0]} = 0 \quad \text{and} \quad \eta_k^{[0]} = 1, \;\; 1 \leq k \leq N, \quad \text{and} \quad J_0 = 1 \tag{49}$$

while (39) and (41) in the calculation of the LOO cost for each $\mathbf{u}_i^{(l)}$ are replaced, respectively, by

$$\begin{aligned} \psi_k^{[n]}(i) \;\; = \;\; & \psi_k^{[n-1]} + y_k \theta_n^{i)} \phi_n^{i)}(k) \\ & - \left(\phi_n^{i)}(k)\right)^2 / \left(\left(\boldsymbol{\phi}_n^{i)}\right)^T \boldsymbol{\phi}_n^{i)} + \lambda\right), 1 \leq k \leq N \end{aligned} \tag{50}$$

and

$$F(\mathbf{u}_i^{(l)}) = \frac{1}{N} \sum_{k=1}^{N} \mathcal{I}_d \left(\psi_k^{[n]}(i) / \eta_k^{[n]}(i) \right). \tag{51}$$

At the nth stage of the classifier construction, when the PSO algorithm terminates, it outputs $\mathbf{gb}^{(I_{\max})}$ as the solution for $\boldsymbol{\mu}_n$ and $\boldsymbol{\Sigma}_n$. The algorithm also generates the nth model column $\mathbf{g}_n$, the orthogonalization coefficients $\alpha_{j,n}$, $1 \leq j < n$, the corresponding orthogonal model column $\boldsymbol{\phi}_n$, and the weight θ_n, as well as $\psi_k^{[n]}$ and $\eta_k^{[n]}$ for $1 \leq k \leq N$.

D. Density Estimate Construction

Since the density estimation can be expressed as a constrained regression modeling, the PSO aided OFR-LOO algorithm detailed in Section III-B can be used to construct a parsimonious density estimate. However, the model weights obtained by the OFR-LOO algorithm do not necessarily meet the nonnegative constraint of (14) and the unity constraint of (15). This "deficiency" can easily be corrected by using the MNQP algorithm to recalculate the weights of the constructed model, as in the case of selecting a fixed-node RBF density estimate [64]. Specifically, the PSO aided OFR-LOO algorithm presented in Section III-B is used to determine the structure of the density estimate by constructing M RBF nodes. This specifies the regression matrix $\mathbf{G}$ in the regression model of (5). The model weight vector $\mathbf{w}$ is then recalculated using the MNQP algorithm [62], [65], in order to meet the constraints of (14) and (15). Formally, this task is defined as follows. Given $\mathbf{y}$ and $\mathbf{G}$, find $\mathbf{w}$ for the model of (5) subject to the constraints of (14) and (15). Note that, since M is very small, the computation involved is small.

More specifically, the weight vector of the constructed density estimate is obtained by solving the following constrained nonnegative quadratic programming [65]:

$$\begin{aligned} & \min_{\mathbf{w}} \{ \tfrac{1}{2} \mathbf{w}^T \mathbf{D} \mathbf{w} - \mathbf{z}^T \mathbf{w} \} \\ & \text{s.t. } \mathbf{w}^T \mathbf{1}_M = 1 \text{ and } w_i \geq 0, \;\; 1 \leq i \leq M \end{aligned} \tag{52}$$

where $\mathbf{D} = \mathbf{G}^T \mathbf{G} = \left[d_{i,j} \right] \in \mathcal{R}^{M \times M}$ is the related design matrix and $\mathbf{z} = \mathbf{G}^T \mathbf{y} = [z_1 \; z_2 \cdots z_M]^T$. Although there exists no closed-form solution for this optimization problem, the solution can readily be obtained iteratively using a modified version of the MNQP algorithm [65]. Since the elements of $\mathbf{D}$ and $\mathbf{z}$ are strictly positive, the auxiliary function for the nonnegative quadratic programming of (52) is given by [65]

$$\frac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{M} d_{i,j} \frac{w_j^{(t)} \left(w_i^{(t+1)} \right)^2}{w_i^{(t)}} - \sum_{i=1}^{M} z_i w_i^{(t+1)} \tag{53}$$

and the Lagrangian associated with this auxiliary problem can be formed as [62]

$$\begin{aligned} \mathcal{L} = & \frac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{M} d_{i,j} \frac{w_j^{(t)} \left(w_i^{(t+1)} \right)^2}{w_i^{(t)}} - \sum_{i=1}^{M} z_i w_i^{(t+1)} \\ & - h^{(t)} \left(\sum_{i=1}^{M} w_i^{(t+1)} - 1 \right) \end{aligned} \tag{54}$$

where the index t denotes the iteration index and h is the Lagrangian multiplier. Setting

$$\frac{\partial \mathcal{L}}{\partial w_i^{(t+1)}} = 0 \text{ and } \frac{\partial \mathcal{L}}{\partial h^{(t)}} = 0 \tag{55}$$

leads to the following updating equations:

$$r_i^{(t)} = w_i^{(t)} \left(\sum_{j=1}^{M} d_{i,j} w_j^{(t)} \right)^{-1}, \;\; 1 \leq i \leq M \tag{56}$$

$$h^{(t)} = \left(\sum_{i=1}^{M} r_i^{(t)} \right)^{-1} \left(1 - \sum_{i=1}^{M} r_i^{(t)} z_i \right) \tag{57}$$

$$w_i^{(t+1)} = r_i^{(t)} \left(z_i + h^{(t)} \right). \tag{58}$$

It is easy to check that, if $\mathbf{w}^{(t)}$ meets the constraints of (14) and (15), $\mathbf{w}^{(t+1)}$ updated according to (56) to (58) also satisfies the constraints of (14) and (15). The initial condition can be set to $w_i^{(0)} = \frac{1}{M}$, $1 \leq i \leq M$. During the iterative procedure, some of the weights may be driven to (near) zero [62], [65]. The corresponding RBF units can then be removed from the model, leading to a further reduction in the model size.

E. Computational Complexity Comparison

The complexity of one LOO cost evaluation and the associated model column orthogonalization, as defined in (36) to (41), can be shown to be the order of N, $\mathcal{O}(N)$ (also see [120]). Thus, $C_{\text{single}} = \mathcal{O}(N)$, and the computational requirements of the PSO-aided OFR-LOO algorithm in constructing an M-node RBF model can readily be given as

$$C_{\text{PSO-OFR}} = (M+1) \times I_{\max} \times S \times \mathcal{O}(N). \tag{59}$$

For regression modeling, this complexity does not include the complexity of using the ROLS-LOO refinement at the end of model construction, which is negligible, while for classification application, $C_{\text{PSO-OFR}}$ accounts for the total complexity. For PDF estimate construction, $C_{\text{PSO-OFR}}$ does not include the complexity of the MNQP algorithm for updating the model weights but this complexity is small and can be neglected.

The ROLS-LOO algorithm [48], [63], [64] is an efficient construction algorithm for selecting fixed-node RBF models.

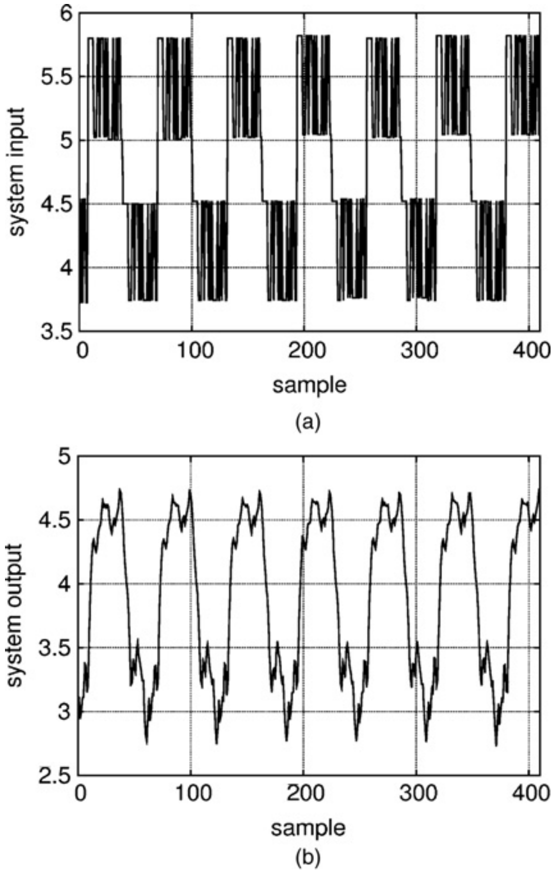


Fig. 1. Engine data set (a) input υ_k, and (b) output y_k.

The complexity of the ROLS-LOO algorithm in selecting $M^{'}$ RBF nodes from the N-candidate set with a given RBF variance is readily determined by

$$\begin{aligned} C_{\text{ROLS}} &= \left(\sum_{i=1}^{M^{'}+1} (N - (i-1)) \right) \times \mathcal{O}(N) \\ &\approx \left(M^{'} + 1 \right) \times N \times \mathcal{O}(N) \end{aligned} \tag{60}$$

where the approximation is arrived because the selected model size $M^{'}$ is usually much smaller than the training data size N. For regression application, this complexity only accounts for the complexity of the first iteration of the ROLS-LOO algorithm, but the complexity of subsequent iterations is much smaller and can be neglected [48]. For density estimation, C_{ROLS} does not take into account the complexity of the MNPQ updating, which is negligible. Only for classification application, C_{ROLS} is the total account of complexity.

We can now draw some comparison for the two methods. The number of cost function evaluations is proportional to the training data size N for the ROLS-LOO algorithm, while for the PSO aided OFR-LOO algorithm, the number of cost function evaluations is somewhat independent of N. This suggests that the PSO aided OFR-LOO algorithm has clearly computational advantage for large training data sets. Specifically, since the model size M obtained by the PSO aided OFR-LOO algorithm is generally much smaller than the model size $M^{'}$ selected by the ROLS-LOO algorithm, the PSO aided OFR-LOO algorithm will require less computation than the ROLS-LOO algorithm whenever the training data size N is larger than $I_{\max} \times S$. In our experimental investigation, we will show that $I_{\max} = 20$ and $S = 10$ are often adequate for the PSO algorithm. Thus for $N \geq 200$, we always have $C_{\text{PSO-OFR}} < C_{\text{ROLS}}$. Note that this computational advantage is largely due to the efficiency of the PSO method. In fact, we can also apply other optimization methods such as the genetic algorithm [121], [122], the adaptive simulated annealing algorithm [123], [124], or the RWBS [67] to perform the optimization task defined in (31). However, these alternative OFR-LOO algorithms typically require higher computational complexity than the ROLS-LOO algorithm, as shown in our previous works [68], [69].

A significant advantage of the OFR-LOO approach for constructing tunable RBF models is that the learning algorithm does not contain any hyperparameters which require costly cross validation to tune. The complexity $C_{\text{PSO-OFR}}$ represents the true computational requirement of the PSO-aided OFR-LOO algorithm, while the complexity C_{ROLS} is the computational requirement of the ROLS-LOO algorithm with the given RBF variance ρ^2. Since this variance is not provided by the learning algorithm, it is a hyperparameter and must be determined typically based on a grid-search cross validation. Thus, the true computational advantage of the PSO-aided OFR-LOO algorithm over the ROLS-LOO algorithm is even more significant.

IV. Empirical Data Modeling Results

Several examples, taken from regression, classification and density estimation applications, were used to demonstrate the effectiveness of the proposed unified regression modeling approach based on the PSO aided OFR-LOO algorithm. Comparison with the ROLS-LOO modeling technique was made. For the PSO aided OFR-LOO, the tunable Gaussian basis function of (3) was employed. For the ROLS-LOO, the N-candidate set was obtained by placing a Gaussian basis at each training data point with a common RBF variance ρ^2. The value of ρ^2 was determined separately via cross validation. Other modeling results, including nonlinear optimization based algorithms for constructing RBF models, were also quoted from the literature for comparison.

Our extensive experience showed that the TVAC of (24) was very effective for PSO and was used in our application. The control factor $\gamma = 0.1$ was found to be adequate for the mechanism of (23) and was used for all the examples. For our application, it was found that setting $w_I = rand()$ generally performed better than choosing a zero or constant inertia weight. The maximum number of iterations was set to $I_{\max} = 20$, as increasing $I_{\max}$ further did not improve performance but imposed higher complexity in our application. An adequate swarm size was determined in terms of algorithmic computational complexity and performance. In our application, typically $S = 10$ to 20 were found to be adequate.

TABLE I
COMPARISON OF THE TWO GAUSSIAN RBF NETWORK MODELS OBTAINED BY THE ROLS-LOO AND PSO-AIDED OFR-LOO ALGORITHMS FOR THE ENGINE DATA SET

Algorithm	RBF Type	Model Size	Training MSE	Test MSE	Complexity
ROLS-LOO	Fixed	22	0.000453	0.000490	$4830 \times \mathcal{O}(210)$
PSO OFR-LOO	Tunable	15	0.000426	0.000466	$3200 \times \mathcal{O}(210)$

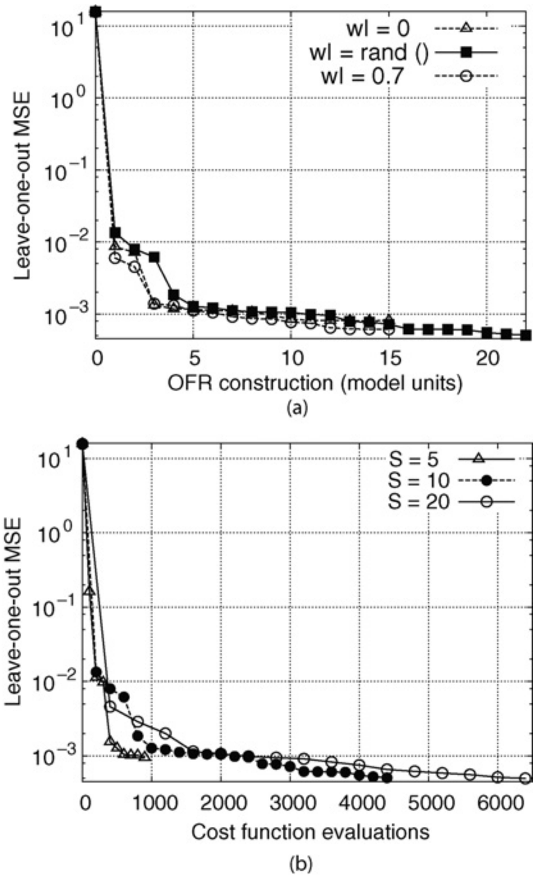


Fig. 2. Modeling of the engine data set. (a) Performance of the PSO algorithm with different w_I while fixing $S = 10$ and $I_{\max} = 20$. (b) Efficiency of the PSO algorithm with different S while fixing $w_I = rand()$ and $I_{\max} = 20$.

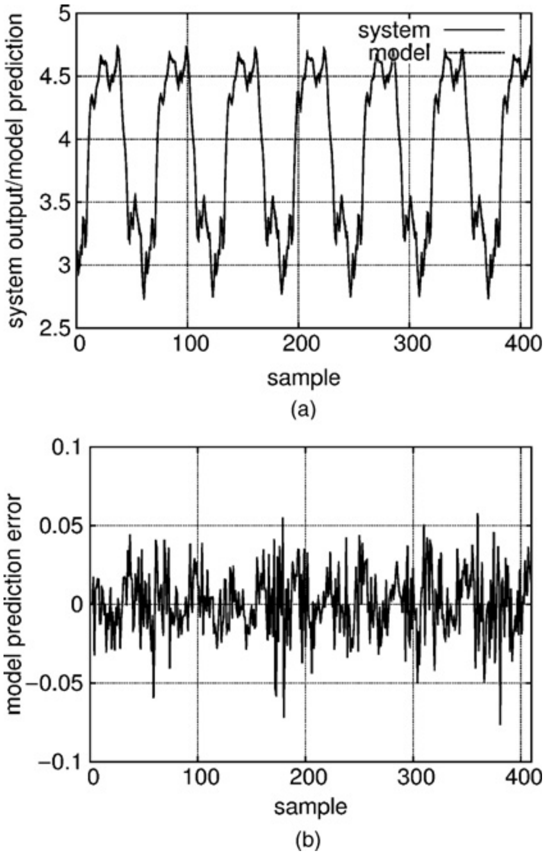


Fig. 3. Modeling of the engine data set by the 15-node RBF network constructed using the PSO aided OFR-LOO algorithm. (a) Model prediction $\hat{y}_k$ superimposed on system output y_k. (b) Model prediction error $\varepsilon_k = y_k - \hat{y}_k$.

A. Regression Applications

1) *Engine Data Set:* This example constructed a model representing the relationship between the fuel rack position (input υ_k) and the engine speed (output y_k) for a Leyland TL11 turbocharged, direct injection diesel engine operated at low engine speed [125]. Detailed system description and experimental setup can be found in [125]. The data set, depicted in Fig. 1, contained 410 samples. The first 210 data points were used in modeling and the last 200 points in model validation. The previous results [125] have shown that this data set can be modeled adequately as

$$y_k = f_s(\mathbf{x}_k) + \varepsilon_k \tag{61}$$

where $f_s(\bullet)$ describes the unknown underlying system, ε_k denotes the system noise, and $\mathbf{x}_k = [y_{k-1} \ \upsilon_{k-1} \ \upsilon_{k-2}]^T$. We first applied the ROLS-LOO algorithm [48] to fit a fixed-node RBF model with a common RBF variance ρ^2 to the data set. An appropriate value for the RBF variance was found to be $\rho^2 = 1.69$ via cross validation. Given $\rho^2 = 1.69$, the algorithm automatically selected $M^{'} = 22$ RBF nodes, and this obtained fixed-node RBF model is listed in Table I.

We next applied the PSO-aided OFR-LOO algorithm to construct a tunable-node RBF model. The maximum number of iterations was set to $I_{\max} = 20$. Fig. 2(a) shows the performance of the PSO algorithm, in terms of LOO MSE, for three different choices of the inertia weight while fixing the swarm size $S = 10$. It can be seen that for this example the case $w_I = rand()$, although less efficient, attained a smaller LOO MSE than the choices of $w_I = 0$ and $w_I = 0.7$. Fig. 2(b) depicts the efficiency of the PSO algorithm, in terms of the total number of cost function evaluations, for three different swarm sizes, while fixing $w_I = rand()$. It can be seen that $S = 5$ was too small for guaranteeing adequate performance, while $S = 10$ achieved the same excellent performance as $S = 20$ but imposed much lower complexity. Hence, $w_I = rand()$, $S = 10$, and $I_{\max} = 20$ were used for this example.

TABLE II

Comparison of the Two Gaussian RBF Network Models Obtained by the ROLS-LOO and PSO-Aided OFR-LOO Algorithms for the Gas Furnace Data Set

Algorithm	RBF Type	Model Size	Training MSE	Test MSE	Complexity
ROLS-LOO	Fixed	12	0.047448	0.080491	$1924 \times \mathcal{O}(148)$
PSO OFR-LOO	Tunable	8	0.041639	0.078884	$1800 \times \mathcal{O}(148)$

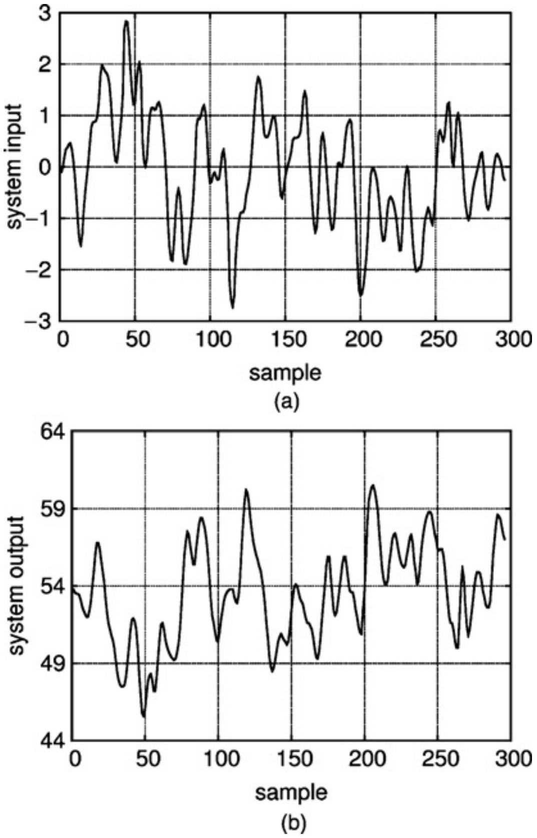


Fig. 4. Gas furnace data set (a) input υ_k and (b) output y_k.

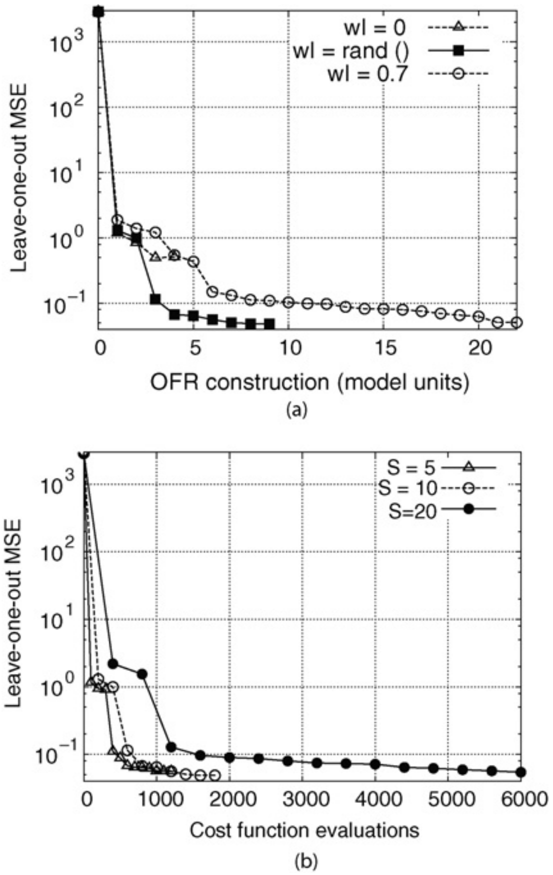


Fig. 5. Modeling of the gas furnace data set. (a) Performance of the PSO algorithm with different w_I while fixing S = 10 and $I_{\max}$ = 20. (b) Efficiency of the PSO algorithm with different S while fixing $w_I = rand()$ and $I_{\max}$ = 20.

The PSO algorithm automatically constructed M = 15 RBF nodes, and this model is also listed in Table I, in comparison with the 22-term RBF model selected by the ROLS-LOO algorithm. Fig. 3 depicts the model prediction $\hat{y}_k$ and the prediction error $\varepsilon_k = y_k - \hat{y}_k$ generated by the 15-node RBF model constructed by the PSO aided OFR-LOO algorithm. In comparison with the benchmark ROLS-LOO algorithm, the PSO aided OFR-LOO algorithm not only produced a smaller RBF model with better test MSE performance but also was more efficient in modeling process. Note that the computational advantage of the PSO aided OFR-LOO was much more significant than shown in Table I, as we did not count the computational requirements for determining the RBF variance ρ^2 = 1.69 needed by the ROLS-LOO algorithm.

2) *Gas Furnace Data Set:* The gas furnace data set (the time series J in [1]) contained 296 pairs of input-output points as depicted in Fig. 4, where the input υ_k was the coded input gas feed rate and the output y_k represented the CO$_2$ concentration from the gas furnace. From the 296 pairs of input and output data, we constructed 296 data points $\{\mathbf{x}_k, y_k\}$ with $\mathbf{x}_k$ given by

$$\mathbf{x}_k = [y_{k-1}\ y_{k-2}\ y_{k-3}\ \upsilon_{k-1}\ \upsilon_{k-2}\ \upsilon_{k-3}]^T. \qquad (62)$$

From Fig. 4, it can be observed that the second half of the data set was different from the first half. Therefore, we used the even-number pairs of $\{\mathbf{x}_k, y_k\}$ for training and the odd-number pairs of $\{\mathbf{x}_k, y_k\}$ for testing. Both the training and testing sets had 148 data points. For the fixed-node RBF modeling, an adequate RBF variance was found to be ρ^2 = 1000.0 after a grid search based cross validation for the ROLS-LOO algorithm, and the algorithm automatically selected a model of 12 RBF nodes. The performance of this fixed-node RBF model is given in Table II.

For constructing the tunable-node RBF model, we set S = 10 and $I_{\max}$ = 20 while using $w_I = rand()$. This set of the PSO algorithmic parameters was appropriate, as clearly demonstrated by the results shown in Fig. 5(a) and (b). The model construction using the PSO-aided OFR-LOO algorithm

TABLE III

COMPARISON OF THE TWO GAUSSIAN RBF NETWORK MODELS OBTAINED BY THE ROLS-LOO AND PSO-AIDED OFR-LOO ALGORITHMS FOR THE LIQUID LEVEL DATA SET

Algorithm	RBF Type	Model Size	Training MSE	Test MSE	Complexity
ROLS-LOO	Fixed	30	0.001400	0.002532	$15500 \times \mathcal{O}(500)$
PSO OFR-LOO	Tunable	20	0.001461	0.002463	$4200 \times \mathcal{O}(500)$

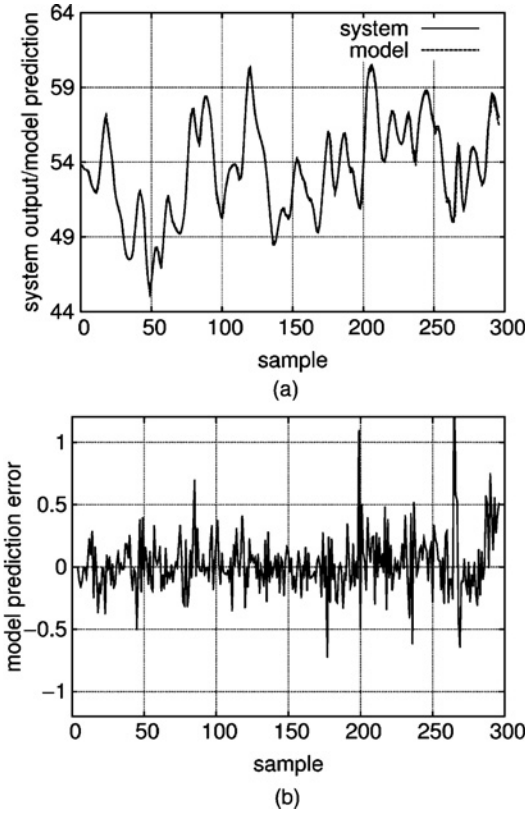


Fig. 6. Modeling of the gas furnace data set by the 8-node RBF network constructed using the PSO aided OFR-LOO algorithm. (a) Model prediction $\hat{y}_k$ superimposed on system output y_k. (b) Model prediction error $\varepsilon_k = y_k - \hat{y}_k$.

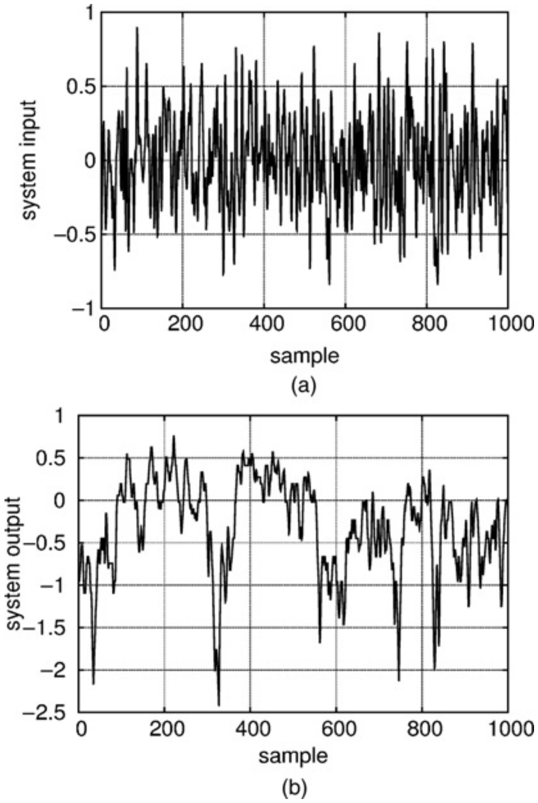


Fig. 7. Liquid level data set (a) input υ_k, and (b) output y_k.

automatically terminated with a 8-node RBF network, and this 8-node tunable RBF model is listed in Table II, in comparison with the 12-term fixed-node RBF model selected by the ROLS-LOO algorithm. Fig. 6 depicts the model prediction $\hat{y}_k$ and the prediction error $\varepsilon_k = y_k - \hat{y}_k$ produced by the 8-node RBF model constructed using the PSO-aided OFR-LOO algorithm. Even though this example had a relatively small training data size $N = 148$, the PSO-aided OFR-LOO algorithm still offered clear advantages in producing a more parsimonious model and better generalization performance as well as less computation in model construction, compared with the benchmark ROLS-LOO algorithm. The true complexity of the ROLS-LOO algorithm, including the cost of determining $\rho^2 = 1000.0$, was several times larger than C_{ROLS} listed in Table II, since several values of ρ^2 needed to be tested in the grid search based cross validation.

3) *Liquid Level Data Set:* The data set was collected from a nonlinear liquid level system, which consisted of a DC water pump feeding a conical flask which in turn fed a square tank. The system input υ_k was the voltage to the pump motor and the system output y_k was the water level in the conical flask. A description of this nonlinear process can be found in [126], and Fig. 7 shows the 1000 data points of the data set used in this experiment. From the data set, 1000 data points $\{\mathbf{x}_k, y_k\}$ were constructed with $\mathbf{x}_k$ given by

$$\mathbf{x}_k = [y_{k-1}\ y_{k-2}\ y_{k-3}\ \upsilon_{k-1}\ \upsilon_{k-2}\ \upsilon_{k-3}\ \upsilon_{k-4}]^T. \tag{63}$$

The first 500 pairs of the data were used for training and the remaining 500 pairs for testing. For the fixed-node RBF model with every training input data used as a candidate RBF center vector, an appropriate RBF variance was found to be $\rho^2 = 2.0$ via a grid search based cross validation for the ROLS-LOO algorithm. With $\rho^2 = 2.0$, the ROLS-LOO algorithm automatically selected a model set of $M' = 30$ nodes, and this fixed-node model is shown in Table III.

Based on the empirical results shown in Fig. 8(a) and (b), we again set $S = 10$ and $I_{\max} = 20$ while adopting $w_I = rand()$. The PSO-aided OFR-LOO algorithm automatically produced a model set of $M = 20$ nodes. The results produced by the PSO-aided OFR-LOO and the ROLS-LOO are compared in Table III. Fig. 9 shows the model prediction $\hat{y}_k$ and the prediction error $\varepsilon_k = y_k - \hat{y}_k$ produced by the 20-node RBF

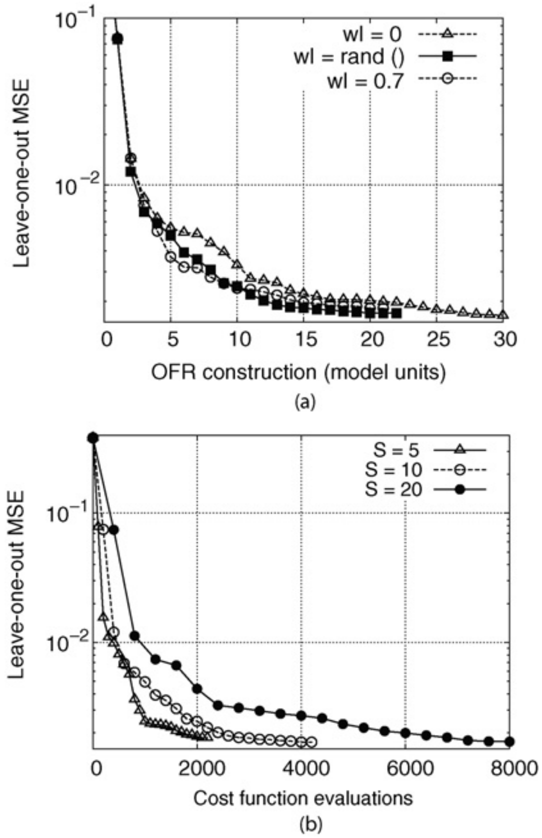


Fig. 8. Modeling of the liquid level data set. (a) Performance of the PSO algorithm with different w_I while fixing $S = 10$ and $I_{\max} = 20$. (b) Efficiency of the PSO algorithm with different S while fixing $w_I = rand()$ and $I_{\max} = 20$.

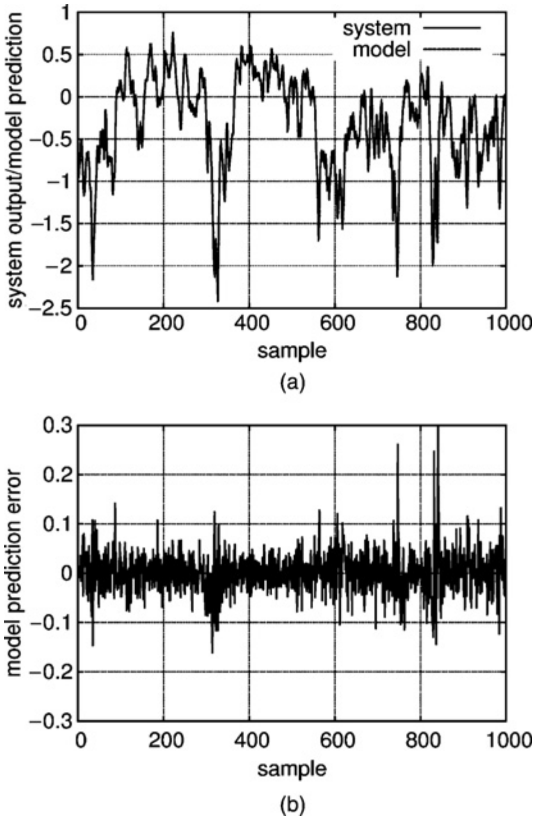


Fig. 9. Modeling of the liquid level data set by the 20-node RBF network constructed using the PSO aided OFR-LOO algorithm. (a) Model prediction $\hat{y}_k$ superimposed on system output y_k. (b) Model prediction error $\varepsilon_k = y_k - \hat{y}_k$.

model constructed using the PSO-aided OFR-LOO algorithm. For this example, the PSO-aided OFR-LOO algorithm has clear advantages over the benchmark ROLS-LOO algorithm, in terms of model size and generalization capability, as well as complexity of model construction.

B. Classification Applications

1) *Breast Cancer Data:* This classification benchmark data set was originated in the UCI repository [127] and the data set used in our experiment was obtained from [128]. The feature input space dimension was $m = 9$. The data set contained 100 realizations, each having 200 training patterns and 77 test patterns. In [129], seven existing state-of-the-art RBF and kernel classifier construction algorithms were compared and the performance averaged over all the 100 realizations was given. For the first five methods studied in [129], the RBF network with five optimized nonlinear Gaussian units was used. The kernel Fisher discriminant was the optimal non-sparse method that placed a Gaussian kernel on every training data sample. For the SVM method with the Gaussian kernel, no average model size was given in [129] but it was certainly larger than, say, 50. We applied both the ROLS-LOO algorithm to select sparse fixed-node Gaussian RBF classifiers and the PSO-aided OFR-LOO algorithm to construct small tunable-node Gaussian RBF classifiers, and the results obtained are listed in Table IV, in comparison with the benchmark results quoted from [129]. For the PSO-aided OFR-LOO, $S = 10$, $I_{\max} = 20$, and $w_I = rand()$ were empirically found to be appropriate. From Table IV, it can be seen that the PSO aided OFR-LOO algorithm compared favorably with other RBF modeling methods, in terms of classification accuracy and model size. The PSO aided OFR-LOO algorithm is also seen to impose less computational complexity in model construction than the efficient ROLS-LOO algorithm.

2) *Diabetes Data:* This was another benchmark data set originated from the UCI repository [127] and we obtained the data set from [128]. The feature space dimension was $m = 8$. There were 100 realizations of the data set, each having 468 training patterns and 300 test patterns. We applied both the ROLS-LOO and PSO-aided OFR-LOO algorithms to the data set. For the PSO-aided OFR-LOO, the swarm size and the maximum number of iterations were set to $S = 10$ and $I_{\max} = 20$, respectively, while the random inertia weight $w_I = rand()$ was adopted. The results obtained by these two algorithms are listed Table V, in comparison with the seven benchmark RBF classifiers studied in [129]. For the first five methods studied in [129], the Gaussian RBF network with 15 optimized nonlinear RBF units was used. For the SVM with RBF kernel, no average model size was given in [129] but we could safely assume that it might be larger than 100. It can be seen from Table V that the PSO-aided OFR-LOO method produced the best classification accuracy with the smallest RBF classifier. It is also seen to impose much lower complexity in model construction than the ROLS-LOO method.

TABLE IV

COMPARISON OF AVERAGE CLASSIFICATION TEST ERROR RATES IN % OVER THE 100 REALIZATIONS OF THE BREAST CANCER DATA SET OBTAINED BY NINE METHODS

Method	RBF Type	Test Error Rate	Model Size	Complexity
RBF-Network	Tunable	27.64 ± 4.71	5	NA
AdaBoost with RBF-Network	Tunable	30.36 ± 4.73	5	NA
LP-Reg-AdaBoost (-"-)	Tunable	26.79 ± 6.08	5	NA
QP-Reg-AdaBoost (-"-)	Tunable	25.91 ± 4.61	5	NA
AdaBoost-Reg (-"-)	Tunable	26.51 ± 4.47	5	NA
SVM with RBF-Kernel	Fixed	26.04 ± 4.74	Not available	NA
Kernel Fisher Discriminant	Fixed	24.77 ± 4.63	200	NA
ROLS-LOO	Fixed	25.74 ± 5.00	6.0 ± 2.0	$1400 \times \mathcal{O}(200)$
PSO OFR-LOO	Tunable	23.04 ± 3.41	2.8 ± 0.9	$760 \times \mathcal{O}(200)$

The first seven results were quoted from [129].

TABLE V

COMPARISON OF AVERAGE CLASSIFICATION TEST ERROR RATES IN % OVER THE 100 REALIZATIONS OF THE DIABETES DATA SET OBTAINED BY NINE METHODS

Method	RBF Type	Test Error Rate	Model Size	Complexity
RBF-Network	Tunable	24.29 ± 1.88	15	NA
AdaBoost with RBF-Network	Tunable	26.47 ± 2.29	15	NA
LP-Reg-AdaBoost (-"-)	Tunable	24.11 ± 1.90	15	NA
QP-Reg-AdaBoost (-"-)	Tunable	25.39 ± 2.20	15	NA
AdaBoost-Reg (-"-)	Tunable	23.79 ± 1.80	15	NA
SVM with RBF-Kernel	Fixed	23.53 ± 1.73	Not available	NA
Kernel Fisher Discriminant	Fixed	23.21 ± 1.63	468	NA
ROLS-LOO	Fixed	23.00 ± 1.70	6.0 ± 1.0	$3276 \times \mathcal{O}(468)$
PSO OFR-LOO	Tunable	21.87 ± 1.24	3.5 ± 1.4	$900 \times \mathcal{O}(468)$

The first seven results were quoted from [129].

TABLE VI

COMPARISON OF AVERAGE CLASSIFICATION TEST ERROR RATES IN % OVER THE 100 REALIZATIONS OF THE THYROID DATA SET OBTAINED BY NINE METHODS

Method	RBF Type	Test Error Rate	Model Size	Complexity
RBF-Network	Tunable	4.52 ± 2.12	8	NA
AdaBoost with RBF-Network	Tunable	4.40 ± 2.18	8	NA
LP-Reg-AdaBoost (-"-)	Tunable	4.59 ± 2.22	8	NA
QP-Reg-AdaBoost (-"-)	Tunable	4.35 ± 2.18	8	NA
AdaBoost-Reg (-"-)	Tunable	4.55 ± 2.19	8	NA
SVM with RBF-Kernel	Fixed	4.80 ± 2.19	Not available	NA
Kernel Fisher Discriminant	Fixed	4.20 ± 2.07	140	NA
ROLS-LOO	Fixed	4.80 ± 2.20	4.6 ± 1.0	$784 \times \mathcal{O}(140)$
PSO OFR-LOO	Tunable	2.48 ± 1.41	3.5 ± 0.8	$1800 \times \mathcal{O}(140)$

The first seven results were quoted from [129].

3) *Thyroid Data:* This was also a benchmark data set in the UCI repository [127] and again we obtained the data set from [128]. The input space dimension was $m = 5$. There were 100 realizations of this data set, each containing 140 training patterns and 75 test patterns. Nine RBF classifiers are compared in Table VI, with the first seven quoted from [129]. Again it is seen that the PSO-aided OFR-LOO method produced the best classification accuracy with the smallest RBF classifier. The PSO algorithmic parameters were found empirically to be $S = 20$, $I_{\max} = 20$, and $w_I = rand()$. For this example, the complexity of the PSO aided OFR-LOO algorithm is seen to be higher than that of the ROLS-LOO algorithm when the latter's RBF variance ρ^2 was given. However, several points of ρ^2 needed to be tested via grid search for the ROLS-LOO algorithm and, therefore, its true complexity was likely to be higher than that of the PSO aided OFR-LOO algorithm even for this example of small N.

C. Density Estimation Applications

For each density estimation case, a data set of N randomly drawn samples was used to construct a density estimate, and a separate test data set of $N_{\text{test}} = 10\,000$ samples was used to calculate the L_1 test error for the resulting estimate according to

$$L_1 = \frac{1}{N_{\text{test}}} \sum_{k=1}^{N_{\text{test}}} |p(\mathbf{x}_k) - \hat{p}(\mathbf{x}_k)| . \tag{64}$$

The Kullback–Leibler divergence (KLD) is a measure of the difference between the two probability distributions, $p(\mathbf{x})$ and $\hat{p}(\mathbf{x})$, and is defined by

$$D_{\text{KL}}(p|\hat{p}) = \int_{\mathcal{R}^m} p(\mathbf{x}) \log \frac{p(\mathbf{x})}{\hat{p}(\mathbf{x})} \, d\mathbf{x}. \tag{65}$$

For a 1-D problem, by partitioning the integration range $[x_{\min},\ x_{\max}]$ into the N_{par} small equal-length intervals, the

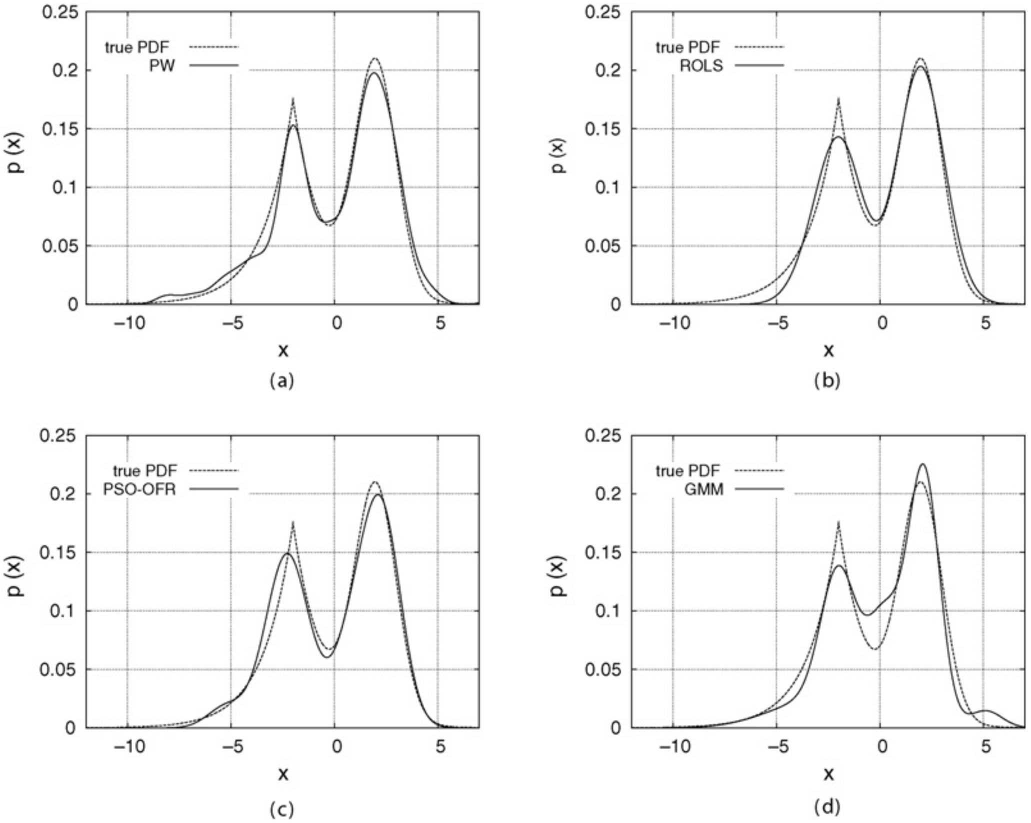


Fig. 10. (a) A PW estimate, (b) a ROLS-LOO estimate, (c) a PSO-aided OFR-LOO estimate, and (d) a GMM estimate, in comparison with the true density, for the 1-D example of Gaussian and Laplacian mixture.

KLD can be approximated accurately using the summation

$$D_{\text{KL}}(p|\hat{p}) \approx \sum_{k=1}^{N_{\text{par}}} p(k) \log \frac{p(k)}{\hat{p}(k)} \Delta x \tag{66}$$

where $\Delta x = (x_{\max} - x_{\min})/N_{\text{par}}$, $p(k) = p(x_{\min} + k\Delta x)$, and $\hat{p}(k) = \hat{p}(x_{\min} + k\Delta x)$. In the experiment, $N_{\text{par}} \geq 10\,000$ was used to ensure the accuracy of approximation. For a 2-D problem, by partitioning the integration range $[x_{1,\min},\ x_{1,\max}] \times [x_{2,\min},\ x_{2,\max}]$ into the $N_{\text{par}} \times N_{\text{par}}$ small equal-area intervals, the KLD is approximated by the summation

$$D_{\text{KL}}(p|\hat{p}) \approx \sum_{k=1}^{N_{\text{par}}} \sum_{l=1}^{N_{\text{par}}} p(k,l) \log \frac{p(k,l)}{\hat{p}(k,l)} (\Delta x)^2 \tag{67}$$

where $\Delta x = (x_{1,\max} - x_{1,\min})/N_{\text{par}} = (x_{2,\max} - x_{2,\min})/N_{\text{par}}$, $p(k,l) = p(x_{1,\min} + k\Delta x, x_{2,\min} + l\Delta x)$, and $\hat{p}(k,l) = \hat{p}(x_{1,\min} + k\Delta x, x_{2,\min} + l\Delta x)$. To ensure the accuracy of approximation, we chose $N_{\text{par}} > 100$. For higher-dimensional problems, calculation of the KLD becomes too expensive. The experiment was repeated by N_{run} different random runs for each example.

Four PDF estimators, the PW estimator, the ROLS-LOO estimator, the PSO-aided OFR-LOO estimator and the Gaussian mixture model (GMM) based on the EM algorithm (see Appendix C) were compared. The optimal values of the kernel variances ρ^2_{Par} and ρ^2 for the PW and ROLS-LOO estimators, respectively, with the fixed-kernel model were found via cross validation. Instead of using costly cross validation to determine the number of mixture components for the GMM, we simply set the number of mixture components to the average model size obtained by the PSO-aided OFR-LOO estimator, rounded to an integer.

1) *1-D Example:* The density to be estimated was the mixture of Gaussian and Laplacian distributions defined by

$$p(x) = \frac{1}{2\sqrt{2\pi}} e^{-\frac{(x-2)^2}{2}} + \frac{0.7}{4} e^{-0.7|x+2|}. \tag{68}$$

The number of data points for density estimation was $N = 100$, and the experiment was repeated $N_{\text{run}} = 100$ times. For the PSO-aided OFR-LOO estimator, $S = 9$, $I_{\max} = 20$ and $w_I = rand()$ were found to be adequate. For this example, the average model size obtained by the PSO-aided OFR-LOO estimator was 4.8. Therefore, for the GMM, we set $M = 5$. Table VII lists the performance of the four density estimators, in terms of the L_1 test error and the KLD, as well as the number of kernels required. Fig. 10(a) to (d) plot a PW estimate obtained, a ROLS-LOO estimate selected, a PSO-aided OFR-LOO estimate constructed and a typical GMM estimate obtained, in comparison with the true density. For this example, it is seen that the PSO-aided OFR-LOO estimator achieved the best test performance with the most compact estimate. The complexity of the ROLS-LOO estimate listed in Table VII was for the given RBF variance. Since several points of ρ^2 needed to be tested for the ROLS-LOO algorithm, its true computational complexity was likely to be higher than that of the PSO-aided OFR-LOO algorithm.

TABLE VII

PERFORMANCE OF THE PW, ROLS-LOO, PSO-AIDED OFR-LOO, AND GMM ESTIMATORS FOR THE 1-D EXAMPLE OF GAUSSIAN AND LAPLACIAN MIXTURE, QUOTED AS MEAN ± STANDARD DEVIATION OVER 100 RUNS

Estimator	PW	ROLS-LOO	PSO OFR-LOO	GMM
Kernel type	Fixed Gaussian, $\rho_{\rm Par} = 0.54$	Fixed Gaussian, $\rho = 1.1$	Tunable Gaussian	Tunable Gaussian
L_1 error	$(1.9963 \pm 0.6179) \times 10^{-2}$	$(2.0213 \pm 0.6535) \times 10^{-2}$	$(1.9784 \pm 0.7039) \times 10^{-2}$	$(2.4597 \pm 0.8117) \times 10^{-2}$
KLD	$(8.0003 \pm 5.1662) \times 10^{-2}$	$(8.1419 \pm 5.0102) \times 10^{-2}$	$(6.5097 \pm 4.0160) \times 10^{-2}$	$(12.7724 \pm 9.5317) \times 10^{-2}$
Kernel no.	100	5.1 ± 1.2	4.8 ± 0.8	5
Complexity	NA	$610 \times \mathcal{O}(100)$	$1044 \times \mathcal{O}(100)$	NA

TABLE VIII

PERFORMANCE OF THE PW, ROLS-LOO, PSO-AIDED OFR-LOO, AND GMM ESTIMATORS FOR THE 2-D EXAMPLE OF GAUSSIAN AND LAPLACIAN MIXTURE, QUOTED AS MEAN ± STANDARD DEVIATION OVER 100 RUNS

Estimator	PW	ROLS-LOO	PSO OFR-LOO	GMM
Kernel type	Fixed Gaussian, $\rho_{\rm Par} = 0.42$	Fixed Gaussian, $\rho = 1.1$	Tunable Gaussian	Tunable Gaussian
L_1 error	$(4.0358 \pm 0.6925) \times 10^{-3}$	$(3.8379 \pm 0.7797) \times 10^{-3}$	$(3.8550 \pm 0.8658) \times 10^{-3}$	$(3.1319 \pm 0.8567) \times 10^{-3}$
KLC	$(1.4661 \pm 0.2281) \times 10^{-1}$	$(1.4028 \pm 0.5337) \times 10^{-1}$	$(1.0279 \pm 0.3848) \times 10^{-1}$	$(0.4380 \pm 0.1328) \times 10^{-1}$
Kernel no.	500	15.3 ± 3.9	6.1 ± 1.6	7
Complexity	NA	$8150 \times \mathcal{O}(500)$	$2840 \times \mathcal{O}(500)$	NA

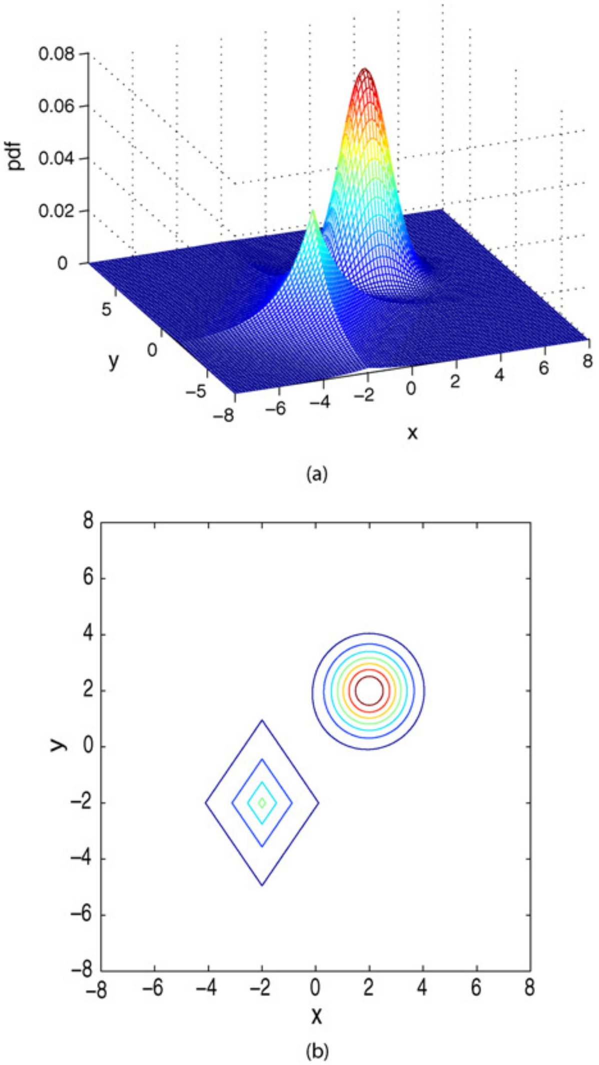


Fig. 11. (a) True density and (b) contour plot for the 2-D example of the Gaussian and the Laplacian mixture.

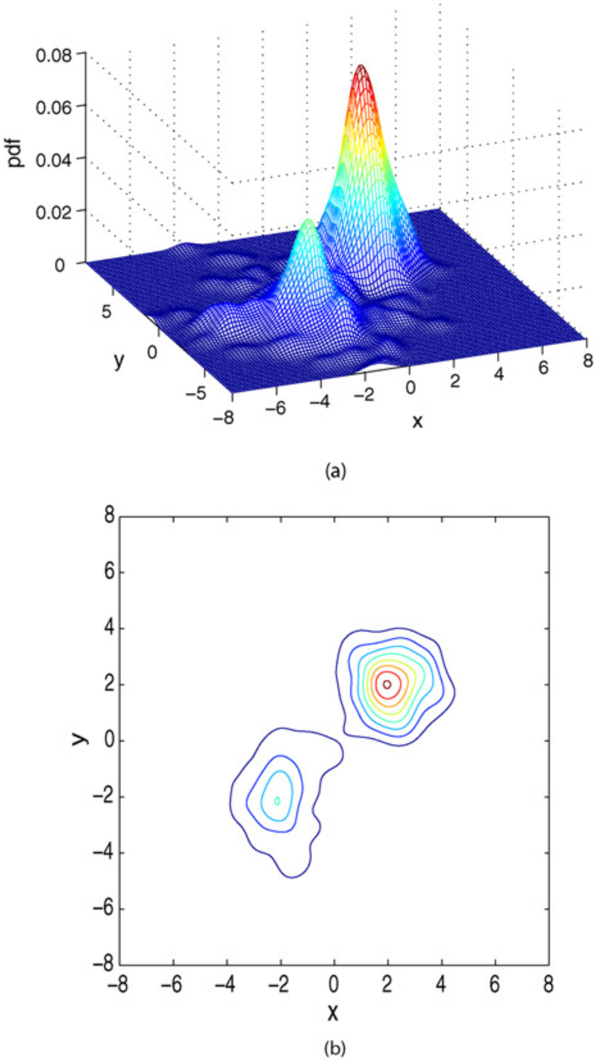
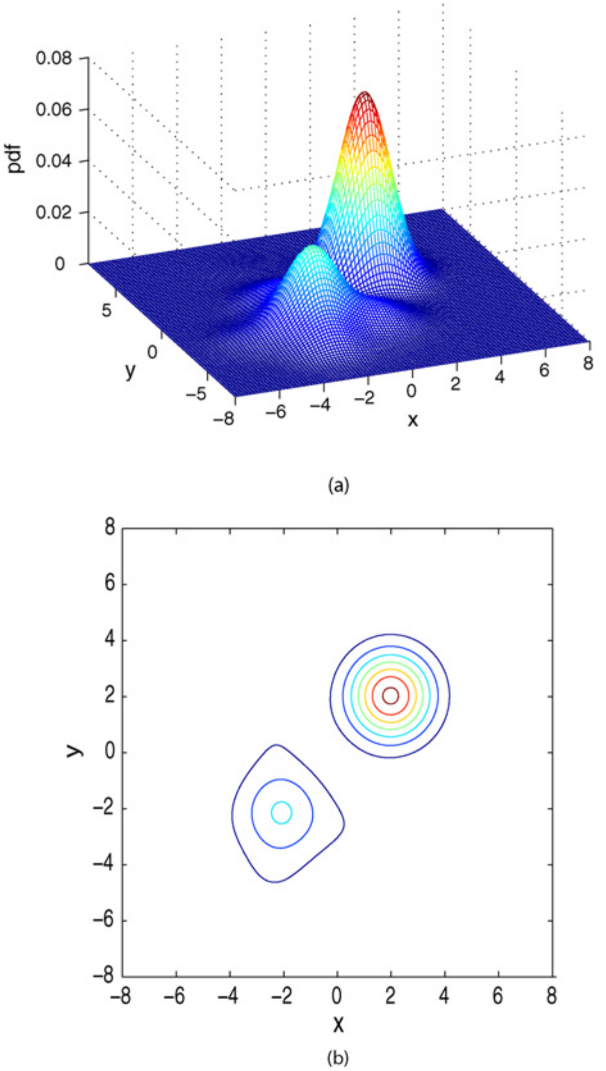


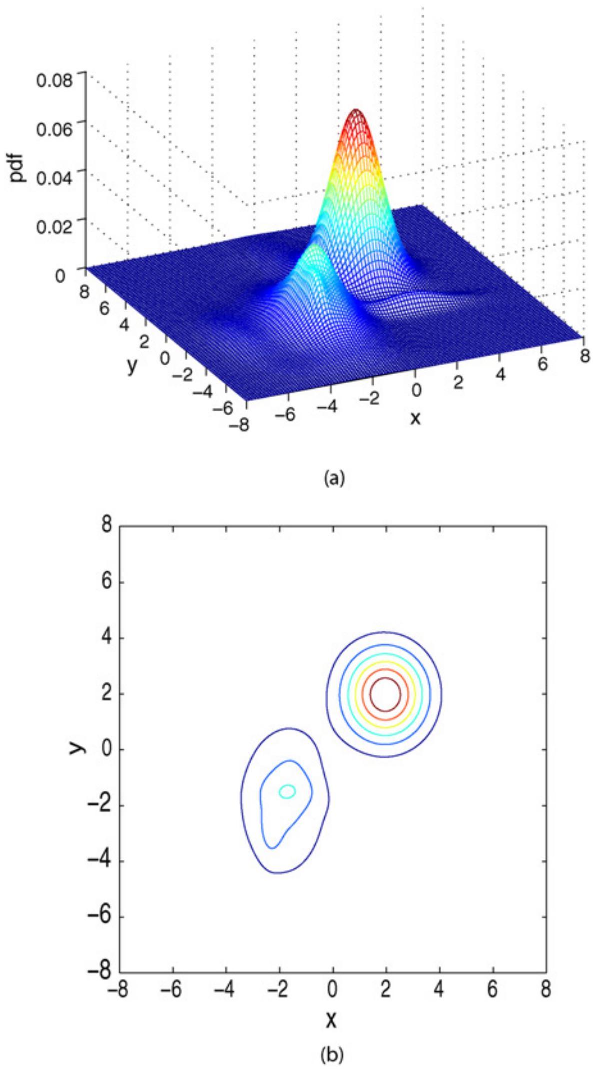
Fig. 12. (a) A PW estimate and (b) its contour plot for the 2-D example of the Gaussian and the Laplacian mixture.



(a)

(b)

Fig. 13. (a) A ROLS-LOO estimate and (b) its contour plot for the 2-D example of the Gaussian and the Laplacian mixture.



(a)

(b)

Fig. 14. (a) A PSO-aided OFR-LOO estimate and (b) its contour plot for the 2-D example of the Gaussian and the Laplacian mixture.

2) *2-D Example:* The density to be estimated was defined by the mixture of Gaussian and Laplacian distributions given as follows

$$p(x_1, x_2) = \frac{1}{4\pi} e^{-\frac{(x_1-2)^2}{2}} e^{-\frac{(x_2-2)^2}{2}} + \frac{0.35}{8} e^{-0.7|x_1+2|} e^{-0.5|x_2+2|}. \tag{69}$$

Fig. 11 shows this density distribution and its contour plot. The estimation data set contained $N = 500$ samples, and the experiment was repeated $N_{\text{run}} = 100$ times. The swarm size and the maximum number of iterations were set to $S = 20$ and $I_{\max} = 20$ while the inertia weight was chosen as $w_I = rand()$ for the PSO-aided OFR-LOO algorithm. Because an average model size of 6.1 was obtained by the PSO-aided OFR-LOO estimator, $M = 7$ was used for the GMM. Table VIII lists the L_1 test errors and the KLD values as well as the numbers of kernels required for the four density estimates, namely, the PW, ROLS-LOO, PSO-aided OFR-LOO and GMM estimators. For this example, the GMM estimator achieved the best test performance. The PSO-aided OFR-LOO estimator also did well with the second best test performance, and it imposed much lower complexity than the ROLS-LOO estimator. Typical PW, ROLS-LOO, PSO-aided OFR-LOO and GMM estimates obtained are depicted in Figs. 12–15, respectively.

3) *6-D Example:* The underlying density to be estimated was given by the mixture of three Gaussian distributions

$$p(\mathbf{x}) = \frac{1}{3} \sum_{i=1}^{3} \frac{1}{(2\pi)^{6/2}} \frac{1}{\det^{1/2} |\bar{\boldsymbol{\Gamma}}_i|} e^{-\frac{1}{2}(\mathbf{x}-\bar{\boldsymbol{\mu}}_i)^T \bar{\boldsymbol{\Gamma}}_i^{-1} (\mathbf{x}-\bar{\boldsymbol{\mu}}_i)} \tag{70}$$

with

$$\begin{aligned} \bar{\boldsymbol{\mu}}_1 &= [1.0\ 1.0\ 1.0\ 1.0\ 1.0\ 1.0]^T \\ \bar{\boldsymbol{\Gamma}}_1 &= \text{diag}\{1.0, 2.0, 1.0, 2.0, 1.0, 2.0\} \end{aligned} \tag{71}$$

$$\begin{aligned} \bar{\boldsymbol{\mu}}_2 &= [-1.0\ \ -1.0\ \ -1.0\ \ -1.0\ \ -1.0\ \ -1.0]^T \\ \bar{\boldsymbol{\Gamma}}_2 &= \text{diag}\{2.0, 1.0, 2.0, 1.0, 2.0, 1.0\} \end{aligned} \tag{72}$$

$$\begin{aligned} \bar{\boldsymbol{\mu}}_3 &= [0.0\ 0.0\ 0.0\ 0.0\ 0.0\ 0.0]^T \\ \bar{\boldsymbol{\Gamma}}_3 &= \text{diag}\{2.0, 1.0, 2.0, 1.0, 2.0, 1.0\}. \end{aligned} \tag{73}$$

TABLE IX

PERFORMANCE OF THE PW, ROLS-LOO, PSO-AIDED OFR-LOO AND GMM ESTIMATORS FOR THE 6-D EXAMPLE OF THREE-GAUSSIAN MIXTURE, QUOTED AS MEAN ± STANDARD DEVIATION OVER 100 RUNS

Estimator	PW	ROLS-LOO	PSO OFR-LOO	GMM
Kernel type	Fixed Gaussian, $\rho_{\text{Par}} = 0.65$	Fixed Gaussian, $\rho = 1.2$	Tunable Gaussian	Tunable Gaussian
L_1 error	$(3.5195 \pm 0.1616) \times 10^{-5}$	$(3.1134 \pm 0.5335) \times 10^{-5}$	$(2.4979 \pm 0.2749) \times 10^{-5}$	$(1.5309 \pm 0.2995) \times 10^{-5}$
Kernel no.	600	9.4 ± 1.9	4.3 ± 0.9	5
Complexity	NA	$6240 \times \mathcal{O}(600)$	$2120 \times \mathcal{O}(600)$	NA

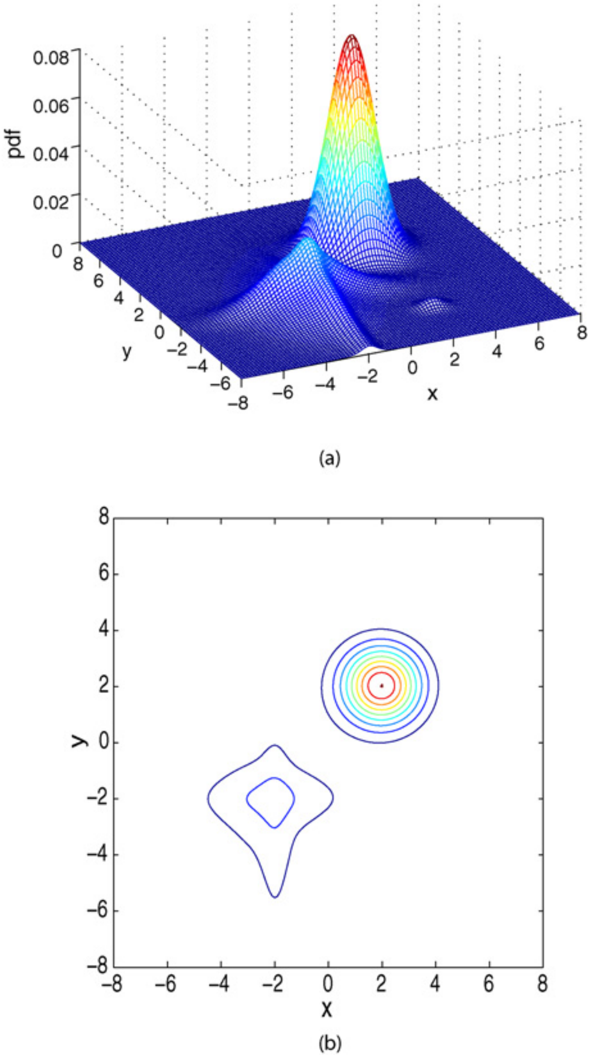


Fig. 15. (a) A GMM estimate and (b) its contour plot for the 2-D example of the Gaussian and the Laplacian mixture.

The estimation data set contained $N = 600$ samples, and the experiment was repeated $N_{\text{run}} = 100$ times. For the PSO-aided OFR-LOO algorithm, $S = 20$, $I_{\max} = 20$ and $w_I = rand()$ were again found to be sufficient. For the GMM estimator, $M = 5$ was used. The results obtained by the four density estimators are summarized in Table IX, where it can be seen that both the PSO-aided OFR-LOO and GMM estimators performed well. It can also be seen from Table IX that for this multi-dimensional example the PSO-aided OFR-LOO algorithm for constructing tunable-node density estimate offered clear advantages over the ROLS-LOO algorithm for selecting fixed-node density estimate, in terms of achievable test performance and estimator model size, as well as complexity in model construction.

V. CONCLUSION

A unified regression framework has been proposed for data modeling applications that include supervised regression and classification, as well as unsupervised density estimation. A novel algorithm has been developed for constructing the RBF network with tunable nodes. Unlike most of the sparse RBF or kernel modeling methods, the RBF centers are not restricted to the training input data points and each RBF node has an individually adjusted diagonal covariance matrix. On the other hand, it does not attempt to optimize all the RBF network's parameters together using nonlinear optimization. Rather the RBF units are optimized one by one using the PSO assisted OFR algorithm based on LOO criteria. The RBF network construction is fully automatic and the user does not need to specify any additional termination criterion. Compared with the state-of-the-art ROLS-LOO algorithm for selecting fixed-node RBF models, the proposed PSO-aided OFR-LOO algorithm offers significant advantages in terms of better generalization performance and smaller model size, as well as imposes lower complexity in model construction process. Moreover, the proposed approach gains further computational advantages because the model construction algorithm does not have any hyperparameter that requires costly tuning based on cross validation. Several examples taken from regression and classification, as well as PDF estimation applications have been used in our experimental study, and the results obtained have demonstrated that the proposed PSO-aided tunable RBF network construction algorithm compares favorably with many existing benchmark RBF network learning algorithms.

APPENDIX A
DENSITY ESTIMATION AS A CONSTRAINED REGRESSION

One way of transferring the unsupervised density estimation problem into a supervised learning problem is to convert the kernels into the associated CDFs and to adopt the EDF calculated using the training data as the desired response for the unknown CDF of the PDF $p(\mathbf{x})$ to be estimated, as in the various fixed-kernel density estimation methods of [59]–[61], [130], [131]. The true CDF of the PDF $p(\mathbf{x})$ is defined as

$$F(\mathbf{x}) = \int_{-\infty}^{\mathbf{x}} p(\mathbf{u})\, d\mathbf{u} \tag{74}$$

and the CDF associated with kernel $g_i(\mathbf{x})$ is given by

$$q_i(\mathbf{x}) = \int_{-\infty}^{\mathbf{x}} g_i(\mathbf{u})\, d\mathbf{u} \tag{75}$$

where $\mathbf{x} = [x_1\ x_2 \cdots x_m]^T \in \mathcal{R}^m$. Further define the EDF on the training set D_N as

$$\hat{F}(\mathbf{x}) = \frac{1}{N} \sum_{k=1}^{N} \prod_{j=1}^{m} \omega\left(x_j - x_{j,k}\right) \tag{76}$$

with

$$\omega(x) = \begin{cases} 1, & x > 0 \\ 0, & x \leq 0. \end{cases} \tag{77}$$

Using $\hat{F}(\mathbf{x})$ as the desired response for $F(\mathbf{x})$, the density estimation can be expressed as a regression modeling

$$\hat{F}(\mathbf{x}) = \sum_{i=1}^{M} w_i q_i(\mathbf{x}) + \hat{e}(\mathbf{x}) \tag{78}$$

subject to the constraints of (14) and (15), where $\hat{e}(\mathbf{x})$ denotes the modeling error at $\mathbf{x}$.

An alternative approach is the direct modeling in the PDF space by using the PW estimate defined in (16) as the desired response for the unknown true PDF $p(\mathbf{x})$ to be estimated, as in the fixed-kernel density estimation method of [64]. Using $\hat{p}_{\text{Par}}(\mathbf{x})$ as the desired response for $p(\mathbf{x})$, the density estimation can be expressed as a regression modeling

$$\hat{p}_{\text{Par}}(\mathbf{x}) = \sum_{i=1}^{M} w_i g_i(\mathbf{x}) + \tilde{e}(\mathbf{x}) \tag{79}$$

subject to the constraints of (14) and (15), where $\tilde{e}(\mathbf{x})$ is the modeling error at $\mathbf{x}$.

Reformulating the density estimation as a regression problem by using the PW estimate as the target function of the true PDF has some advantages over the regression approach based on using the EDF as the target function of the true CDF. The former approach can use any type of kernel function and it is computationally simpler, as it does not need to compute the values of regressors of (75) on the training data set D_N. Computing the associated CDFs for the kernels can be inconvenient and may be difficult for certain types of kernels. Computing the values of the PW estimator on D_N is no more complex than calculating the values of $\hat{F}(\mathbf{x})$ on D_N. The only drawback of using the PW estimate is that the kernel variance for the PW estimator must be determined. Although we develop the tunable kernel model using the PW estimate as the target function in this contribution, the construction algorithm developed is equally applicable for the both approaches.

Appendix B
Leave-One-Out Cross Validation

A commonly used cross validation for model selection is the LOO cross validation [2], [118]. Consider the model selection problem where a set of m_S models have been identified using the training data set D_N. Denote these models, identified using all the N data points of D_N, as $\hat{y}_k^{[j]}$ and the corresponding modeling error as $e_k^{[j]} = y_k - \hat{y}_k^{[j]}$ with index $j = 1, 2, \cdots, m_S$. The concept of LOO cross validation is as follows. For every model, each data point in the training set D_N is sequentially set aside in turn, a model is estimated using the remaining $N-1$ data points, and the test error is derived using the single data point that was removed from training. Specifically, let $D_N \setminus (\mathbf{x}_l, y_l)$ be the resulting data set by removing the lth data point from D_N, and denote the jth model estimated using $D_N \setminus (\mathbf{x}_l, y_l)$ as $\hat{y}_k^{[j,-l]}$. The test error of the model $\hat{y}_k^{[j,-l]}$ calculated on the data point $(\mathbf{x}_l, y_l)$ not used in training is given by

$$e_l^{[j,-l]} = y_l - \hat{y}_l^{[j,-l]}. \tag{80}$$

The mean square LOO test error for the jth model is obtained by averaging all these test errors

$$E\left[\left(e_l^{[j,-l]}\right)^2\right] \approx \frac{1}{N} \sum_{l=1}^{N} \left(e_l^{[j,-l]}\right)^2 \tag{81}$$

where $E[\bullet]$ denotes the expectation. The LOO MSE is a measure of the mode generalization capability [2], [118]. To select the best model from the m_S model candidates $\hat{y}_k^{[j]}$, $1 \leq j \leq m_S$, the same LOO cross validation procedure is applied to each of the m_S models, and the model with the minimum LOO MSE is selected.

For the linear-in-the-weights models, which the model of (1) is when the regression matrix $\mathbf{G}$ has been determined, the LOO test errors can be generated, without actually sequentially splitting the training data set and repeatedly estimating the associated models, by applying the Sherman–Morrison–Woodbury theorem [2], [118]. Furthermore, within the OFR model selection procedure, the LOO test errors for the n-term model can be computed very efficiently [48], [119]. More specifically, consider the n-term model with the associated orthogonal regression matrix

$$\mathbf{\Phi}_n = [\boldsymbol{\phi}_1\ \boldsymbol{\phi}_2 \cdots \boldsymbol{\phi}_n]. \tag{82}$$

The regularized least squares solution for the parameter vector $\boldsymbol{\theta}_n = [\theta_1\ \theta_2 \cdots \theta_n]^T$ is [48]

$$\boldsymbol{\theta}_n = \left(\mathbf{\Phi}_n^T \mathbf{\Phi}_n + \lambda \mathbf{I}_n\right)^{-1} \mathbf{\Phi}_n^T \mathbf{y} = \tilde{\mathbf{B}}_n^{-1} \mathbf{\Phi}_n^T \mathbf{y} \tag{83}$$

where λ is the regularization parameter, $\mathbf{I}_n$ denotes the $n \times n$ identity matrix, $\tilde{\mathbf{B}}_n = \mathbf{\Phi}_n^T \mathbf{\Phi}_n + \lambda \mathbf{I}_n$ is diagonal. The modeling error at the kth training data sample is given by

$$e_k^{[n]} = y_k - \boldsymbol{\theta}_n^T \boldsymbol{\phi}_n(k) = y_k - \mathbf{y}^T \mathbf{\Phi}_n \tilde{\mathbf{B}}_n^{-1} \boldsymbol{\phi}_n(k) \tag{84}$$

where $\boldsymbol{\phi}_n^T(k)$ denotes the kth row of $\mathbf{\Phi}_n$. Let the kth data sample be deleted from the training data set D_N, and the resulting LOO training set is used to estimate the model parameter vector. The corresponding regularized least squares solution is defined by

$$\begin{aligned} \boldsymbol{\theta}_n^{[-k]} &= \left(\left(\mathbf{\Phi}_n^{[-k]}\right)^T \mathbf{\Phi}_n^{[-k]} + \lambda \mathbf{I}_n\right)^{-1} \left(\mathbf{\Phi}_n^{[-k]}\right)^T \mathbf{y}^{[-k]} \\ &= \left(\tilde{\mathbf{B}}_n^{[-k]}\right)^{-1} \left(\mathbf{\Phi}_n^{[-k]}\right)^T \mathbf{y}^{[-k]} \end{aligned} \tag{85}$$

where $\mathbf{\Phi}_n^{[-k]}$ and $\mathbf{y}^{[-k]}$ denote the resulting LOO regression matrix and LOO desired output vector, respectively. The model output for this LOO n-term model evaluated at the kth data sample not used in training is given by

$$\hat{y}_k^{[n,-k]} = \left(\boldsymbol{\theta}_n^{[-k]}\right)^T \boldsymbol{\phi}_n(k). \tag{86}$$

By definition, it can be shown that

$$\tilde{\mathbf{B}}_n^{[-k]} = \tilde{\mathbf{B}}_n - \boldsymbol{\phi}_n(k)\boldsymbol{\phi}_n^T(k), \tag{87}$$

$$\left(\mathbf{y}^{[-k]}\right)^T \boldsymbol{\Phi}_n^{[-k]} = \mathbf{y}^T\boldsymbol{\Phi}_n - y_k\boldsymbol{\phi}_n^T(k). \tag{88}$$

The LOO test error evaluated at the kth data sample not used for training, denoted as $e_k^{[n,-k]} = y_k - \hat{y}_k^{[n,-k]}$, is given by

$$\begin{aligned} e_k^{[n,-k]} &= y_k - \left(\boldsymbol{\theta}_n^{[-k]}\right)^T \boldsymbol{\phi}_n(k) \\ &= y_k - \left(\mathbf{y}^{[-k]}\right)^T \boldsymbol{\Phi}_n^{[-k]} \left(\tilde{\mathbf{B}}_n^{[-k]}\right)^{-1} \boldsymbol{\phi}_n(k). \end{aligned} \tag{89}$$

Applying the matrix inversion lemma to (87) yields

$$\left(\tilde{\mathbf{B}}_n^{[-k]}\right)^{-1} = \tilde{\mathbf{B}}_n^{-1} + \frac{\tilde{\mathbf{B}}_n^{-1}\boldsymbol{\phi}_n(k)\boldsymbol{\phi}_n^T(k)\tilde{\mathbf{B}}_n^{-1}}{1 - \boldsymbol{\phi}_n^T(k)\tilde{\mathbf{B}}_n^{-1}\boldsymbol{\phi}_n(k)} \tag{90}$$

and

$$\left(\tilde{\mathbf{B}}_n^{[-k]}\right)^{-1} \boldsymbol{\phi}_n(k) = \frac{\tilde{\mathbf{B}}_n^{-1}\boldsymbol{\phi}_n(k)}{1 - \boldsymbol{\phi}_n^T(k)\tilde{\mathbf{B}}_n^{-1}\boldsymbol{\phi}_n(k)}. \tag{91}$$

Substituting (88) and (91) into (89) results in

$$\begin{aligned} e_k^{[n,-k]} &= y_k - \frac{\left(\mathbf{y}^T\boldsymbol{\Phi}_n - y_k\boldsymbol{\phi}_n^T(k)\right)\tilde{\mathbf{B}}_n^{-1}\boldsymbol{\phi}_n(k)}{1 - \boldsymbol{\phi}_n^T(k)\tilde{\mathbf{B}}_n^{-1}\boldsymbol{\phi}_n(k)} \\ &= \frac{y_k - \mathbf{y}^T\boldsymbol{\Phi}_n\tilde{\mathbf{B}}_n^{-1}\boldsymbol{\phi}_n(k)}{1 - \boldsymbol{\phi}_n^T(k)\tilde{\mathbf{B}}_n^{-1}\boldsymbol{\phi}_n(k)} \\ &= \frac{e_k^{[n]}}{1 - \boldsymbol{\phi}_n^T(k)\tilde{\mathbf{B}}_n^{-1}\boldsymbol{\phi}_n(k)} = \frac{e_k^{[n]}}{\eta_k^{[n]}} \end{aligned} \tag{92}$$

where the n-term modeling error

$$e_k^{[n]} = y_k - \sum_{i=1}^{n} \phi_i(k)\theta_i = e_k^{[n-1]} - \phi_n(k)\theta_n \tag{93}$$

and the associated LOO error weighting

$$\begin{aligned} \eta_k^{[n]} &= 1 - \boldsymbol{\phi}_n^T(k)\left(\boldsymbol{\Phi}_n^T\boldsymbol{\Phi}_n + \lambda\mathbf{I}_n\right)^{-1}\boldsymbol{\phi}_n(k) \\ &= 1 - \sum_{i=1}^{n} \frac{\phi_i^2(k)}{\boldsymbol{\phi}_i^T\boldsymbol{\phi}_i + \lambda} \\ &= \eta_k^{[n-1]} - \frac{\phi_n^2(k)}{\boldsymbol{\phi}_n^T\boldsymbol{\phi}_n + \lambda}. \end{aligned} \tag{94}$$

Appendix C
Gaussian Mixture Model

A general finite mixture model (FMM) [132] is described by

$$\hat{p}(\mathbf{x};\boldsymbol{\Omega}) = \sum_{i=1}^{M} w_i K\left(\mathbf{x};\boldsymbol{\mu}_i,\boldsymbol{\Sigma}_i\right) \tag{95}$$

where M is the number of mixture components, the weights w_i satisfy the constraints given in (14) and (15), $\boldsymbol{\mu}_i = [\mu_{i,1} \cdots \mu_{i,m}]^T$ and $\boldsymbol{\Sigma}_i = \text{diag}\{\sigma_{i,1}^2, \cdots, \sigma_{i,m}^2\}$ are the mean vector and covariance matrix of the ith component, respectively, and $\boldsymbol{\Omega} = \{w_l, \boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l\}_{l=1}^M$ denotes all the parameters of the FMM. When the Gaussian function of (3) is used, the FMM of (95) is the GMM.

The EM algorithm for estimating the parameters of the GMM takes an explicit iterative form [133]. Given a value of $\boldsymbol{\Omega}$, labeled as $\boldsymbol{\Omega}^{\text{old}}$, define

$$P(l|\mathbf{x}_k,\boldsymbol{\Omega}^{\text{old}}) = \frac{w_l^{\text{old}}K(\mathbf{x}_k;\boldsymbol{\mu}_l^{\text{old}},\boldsymbol{\Sigma}_l^{\text{old}})}{\sum_{i=1}^{M} w_i^{\text{old}}K(\mathbf{x}_k;\boldsymbol{\mu}_i^{\text{old}},\boldsymbol{\Sigma}_i^{\text{old}})} \tag{96}$$

for $1 \le l \le M$ and $1 \le k \le N$. Then a new value of $\boldsymbol{\Omega}$ is obtained according to [133]

$$w_l^{\text{new}} = \frac{1}{N}\sum_{k=1}^{N} P(l|\mathbf{x}_k,\boldsymbol{\Omega}^{\text{old}}) \tag{97}$$

$$\boldsymbol{\mu}_l^{\text{new}} = \frac{\sum_{k=1}^{N} \mathbf{x}_k P(l|\mathbf{x}_k,\boldsymbol{\Omega}^{\text{old}})}{\sum_{k=1}^{N} P(l|\mathbf{x}_k,\boldsymbol{\Omega}^{\text{old}})} \tag{98}$$

$$\boldsymbol{\Sigma}_l^{\text{new}} = \frac{\sum_{k=1}^{N} P(l|\mathbf{x}_k,\boldsymbol{\Omega}^{\text{old}})\text{diag}\{(\Delta_l x_{k,1})^2,\cdots,(\Delta_l x_{k,m})^2\}}{\sum_{k=1}^{N} P(l|\mathbf{x}_k,\boldsymbol{\Omega}^{\text{old}})} \tag{99}$$

where

$$\Delta_l x_{k,i} = x_{k,i} - \mu_{l,i}^{\text{new}} \tag{100}$$

denotes the ith element of $\mathbf{x}_k - \boldsymbol{\mu}_l^{\text{new}}$.

This simple EM algorithm for the GMM, however, is generally ill-posed. In particular, the updating (99) may cause numerical problems, which leads to divergence. Often more complicated robust techniques such as the bootstrap [32], [134] may need to be used to overcome numerical difficulties. The choice of the initial $\boldsymbol{\Omega}$ is also critical, as the algorithm can only converge to local minima, and whether or not the algorithm converges may depend on the initial parameter value. Based on our previous experience [32] we found that it is necessary to impose a minimum bound, σ_{min}^2, for all the variances $\sigma_{l,i}^2$, $1 \le i \le m$ and $1 \le l \le M$. During the iteration process, any $\sigma_{l,i}^2$ goes below the value σ_{min}^2 is reset to this minimum value. This helps to alleviate numerical problems and improve the chance of convergence. Appropriate σ_{min}^2 values are problem dependant and can only be found by experiment.

In the experiment study, all the initial mixing weights w_l can be set to $\frac{1.0}{M}$, the initial center vectors $\boldsymbol{\mu}_l$ are randomly chosen from the region $[a,\ b]^m \in \mathcal{R}^m$, and all the initial variances $\sigma_{l,i}^2$ are set to the same value σ_{ini}^2. If some runs of the EM algorithm are observed to diverge, the region $[a,\ b]^m$, the values of σ_{ini}^2 and/or σ_{min}^2 are re-chosen until all the N_{run} of the EM algorithm are converged.

References

[1] G. E. P. Box and G. M. Jenkins, *Time Series Analysis, Forecasting and Control*. San Francisco, CA: Holden Day, 1976.

[2] R. H. Myers, *Classical and Modern Regression with Applications*, 2nd ed. Boston, MA: PWS-Kent, 1990.

[3] S. A. Billings and S. Chen, "The determination of multivariable nonlinear models for dynamic systems," in *Neural Network Systems Techniques and Applications* (Control and Dynamic Systems), vol. 8, C. T. Leondes, Ed. San Diego, CA: Academic, 1998, pp. 231–278.

[4] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. New York: Wiley, 1973.

[5] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford, U.K.: Oxford Univ. Press, 1995.

[6] B. D. Ripley, *Pattern Recognition and Neural Networks*. Cambridge, U.K.: Cambridge Univ. Press, 1996.

[7] E. Parzen, "On estimation of a probability density function and mode," *Ann. Mathematical Statistics*, vol. 33, no. 3, pp. 1066–1076, 1962.

[8] B. W. Silverman, *Density Estimation for Statistics and Data Analysis*. London, U.K.: Chapman & Hall, 1986.

[9] A. W. Bowman and A. Azzalini, *Applied Smoothing Techniques for Data Analysis*. Oxford, U.K.: Oxford Univ. Press, 1997.

[10] S. Chen, S. A. Billings, C. F. N. Cowan, and P. M. Grant, "Non-linear systems identification using radial basis functions," *Int. J. Syst. Sci.*, vol. 21, no. 12, pp. 2513–2539, 1990.

[11] J. A. Leonard and M. A. Kramer, "Radial basis function networks for classifying process faults," *IEEE Control Syst. Mag.*, vol. 11, no. 3, pp. 31–38, Apr. 1991.

[12] S. Chen, B. Mulgrew, and P. M. Grant, "A clustering technique for digital communications channel equalization using radial basis function networks," *IEEE Trans. Neural Netw.*, vol. 4, no. 4, pp. 570–579, Jul. 1993.

[13] A. Caiti and T. Parisini, "Mapping ocean sediments by RBF networks," *IEEE J. Oceanic Eng.*, vol. 19, no. 4, pp. 577–582, Oct. 1994.

[14] D. Gorinevsky, A. Kapitanovsky, and A. Goldenberg, "Radial basis function network architecture for nonholonomic motion planning and control of free-flying manipulators," *IEEE Trans. Robot. Autom.*, vol. 12, no. 3, pp. 491–496, Jun. 1996.

[15] I. Cha and S. A. Kassam, "RBFN restoration of nonlinearly degraded images," *IEEE Trans. Image Process.*, vol. 5, no. 6, pp. 964–975, Jun. 1996.

[16] M. Rosenblum and L. S. Davis, "An improved radial basis function network for visual autonomous road following," *IEEE Trans. Neural Netw.*, vol. 7, no. 5, pp. 1111–1120, Sept. 1996.

[17] J. A. Refaee, M. Mohandes, and H. Maghrabi, "Radial basis function networks for contingency analysis of bulk power systems," *IEEE Trans. Power Syst.*, vol. 14, no. 2, pp. 772–778, May 1999.

[18] S. Muraki, T. Nakai, Y. Kita, and K. Tsuda, "An attempt for coloring multichannel MR imaging data," *IEEE Trans. Vis Comput. Graphics*, vol. 7, no. 3, pp. 265–274, Jul.–Sep. 2001.

[19] R. Mukai, V. A. Vilnrotter, P. Arabshahi, and V. Jamnejad, "Adaptive acquisition and tracking for deep space array feed antennas," *IEEE Trans. Neural Netw.*, vol. 13, no. 5, pp. 1149–1162, Sep. 2002.

[20] C.-T. Su, T. Yang, and C.-M. Ke, "A neural-network approach for semiconductor wafer post-sawing inspection," *IEEE Trans. Semicond. Manuf.*, vol. 15, no. 2, pp. 260–266, May 2002.

[21] Y. Li, N. Sundararajan, P. Saratchandran, and Z. Wang, "Robust neuro-H_∞ controller design for aircraft auto-landing," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 40, no. 1, pp. 158–167, Jan. 2004.

[22] M.-J. Lee and Y.-K. Choi, "An adaptive neurocontroller using RBFN for robot manipulators," *IEEE Trans. Ind. Electron.*, vol. 51, no. 3, pp. 711–717, Jun. 2004.

[23] S. X. Ng, M.-S. Yee, and L. Hanzo, "Coded modulation assisted radial basis function aided turbo equalization for dispersive Rayleigh-fading channels," *IEEE Trans. Wireless Commun.*, vol. 3, no. 6, pp. 2198–2206, Nov. 2004.

[24] Y.-J. Oyang, S.-C. Hwang, Y.-Y. Ou, C.-Y. Chen, and Z. W. Chen, "Data classification with radial basis function networks based on a novel kernel density estimation algorithm," *IEEE Trans. Neural Netw.*, vol. 16, no. 1, pp. 225–236, Jan. 2005.

[25] N. Acir, I. Oztura, M. Kuntalp, B. Baklan, and C. Guzelis, "Automatic detection of epileptiform events in EEG by a three-stage procedure based on artificial neural networks," *IEEE Trans. Biomed. Eng.*, vol. 52, no. 1, pp. 30–40, Jan. 2005.

[26] K. K. Tan, S. Zhao, and S. Huang, "Iterative reference adjustment for high-precision and repetitive motion control applications," *IEEE Trans. Control Syst. Technol.*, vol. 13, no. 1, pp. 85–97, Jan. 2005.

[27] S. Chen, C. F. N. Cowan, S. A. Billings, and P. M. Grant, "Parallel recursive prediction error algorithm for training layered neural networks," *Int. J. Control*, vol. 51, no. 6, pp. 1215–1228, 1990.

[28] P. E. An, M. Brown, S. Chen, and C. J. Harris, "Comparative aspects of neural network algorithms for on-line modeling of dynamic processes," *Proc. Inst. Mech. Eng., part I.—J. Syst. Control Eng.*, vol. 207, pp. 223–241, 1993.

[29] S. McLoone, M. D. Brown, G. Irwin, and A. Lightbody, "A hybrid linear/nonlinear training algorithm for feedforward neural networks," *IEEE Trans. Neural Netw.*, vol. 9, no. 4, pp. 669–684, Jul. 1998.

[30] N. B. Karayiannis and M. M. Randolph-Gips, "On the construction and training of reformulated radial basis function neural networks," *IEEE Trans. Neural Netw.*, vol. 14, no. 4, pp. 835–846, Jul. 2003.

[31] H. Peng, T. Ozaki, V. Haggan-Ozaki, and Y. Toyoda, "A parameter optimization method for radial basis function type models," *IEEE Trans. Neural Netw.*, vol. 14, no. 4, pp. 432–438, Mar. 2003.

[32] Z. R. Yang and S. Chen, "Robust maximum likelihood training of heteroscedastic probabilistic neural networks," *Neural Netw.*, vol. 11, pp. 739–747, Jun. 1998.

[33] M.-W. Mak and S.-Y. Kung, "Estimation of elliptical basis function parameters by the EM algorithm with application to speaker verification," *IEEE Trans. Neural Netw.*, vol. 11, no. 4, pp. 961–969, Jul. 2000.

[34] B. A. Whitehead and T. D. Choate, "Evolving space-filling curves to distribute radial basis functions over an input space," *IEEE Trans. Neural Netw.*, vol. 5, no. 1, pp. 15–23, Jan. 1994.

[35] B. A. Whitehead, "Genetic evolution of radial basis function coverage using orthogonal niches," *IEEE Trans. Neural Netw.*, vol. 7, no. 6, pp. 1525–1528, Nov. 1996.

[36] J. Gonzalez, I. Rojas, J. Ortega, H. Pomares, F. J. Fernandez, and A. F. Diaz, "Multiobjective evolutionary optimization of the size, shape, and position parameters of radial basis function networks for function approximation," *IEEE Trans. Neural Netw.*, vol. 14, no. 6, pp. 1478–1495, Nov. 2003.

[37] H.-M. Feng, "Self-generation RBFNs using evolutional PSO learning," *Neurocomputing*, vol. 70, pp. 241–251, Dec. 2006.

[38] F. A. Guerra and L. S. Coelho, "Multi-step ahead nonlinear identification of Lorenz's chaotic system using radial basis function neural network with learning by clustering and particle swarm optimization," *Chaos, Solitons Fractals*, vol. 35, pp. 967–979, Mar. 2008.

[39] J. Moody and C. J. Darken, "Fast learning in networks of locally-tuned processing units," *Neural Comput.*, vol. 1, no. 2, pp. 281–294, 1989.

[40] S. Chen, S. A. Billings, and P. M. Grant, "Recursive hybrid algorithm for non-linear system identification using radial basis function networks," *Int. J. Control*, vol. 55, no. 5, pp. 1051–1070, 1992.

[41] S. Chen, "Nonlinear time series modeling and prediction using Gaussian RBF networks with enhanced clustering and RLS learning," *Electron. Lett.*, vol. 31, no. 2, pp. 117–118, 1995.

[42] Z. Uykan, "Clustering-based algorithms for single-hidden-layer sigmoid perceptron," *IEEE Trans. Neural Netw.*, vol. 14, no. 3, pp. 708–715, May 2003.

[43] D. Shi, D. S. Yeung, and J. Gao, "Sensitivity analysis applied to the construction of radial basis function networks," *Neural Netw.*, vol. 18, pp. 951–957, Sep. 2005.

[44] S. Chen, S. A. Billings, and W. Luo, "Orthogonal least squares methods and their application to non-linear system identification," *Int. J. Control*, vol. 50, no. 5, pp. 1873–1896, 1989.

[45] S. Chen, C. F. N. Cowan, and P. M. Grant, "Orthogonal least squares learning algorithm for radial basis function networks," *IEEE Trans. Neural Netw.*, vol. 2, no. 2, pp. 302–309, Mar. 1991.

[46] S. Chen, Y. Wu, and B. L. Luk, "Combined genetic algorithm optimization and regularised orthogonal least squares learning for radial basis function networks," *IEEE Trans. Neural Netw.*, vol. 10, no. 5, pp. 1239–1243, Sep. 1999.

[47] S. Chen, X. Hong, and C. J. Harris, "Sparse kernel regression modeling using combined locally regularized orthogonal least squares and D-optimality experimental design," *IEEE Trans. Automat. Control*, vol. 48, no. 6, pp. 1029–1036, Jun. 2003.

[48] S. Chen, X. Hong, C. J. Harris, and P. M. Sharkey, "Sparse modeling using orthogonal forward regression with PRESS statistic and regularization," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 34, no. 2, pp. 898–911, Apr. 2004.

[49] V. Vapnik, *The Nature of Statistical Learning Theory*. New York: Springer-Verlag, 1995.

[50] S. Gunn, "Support vector machines for classification and regression," School Electron. Comput. Sci., Univ. Southampton, Southampton, U.K., Tech. Rep., May 1998.

[51] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *Soc. Ind. Appl. Math. Rev.*, vol. 43, no. 1, pp. 129–159, 2001.

[52] M. E. Tipping, "Sparse Bayesian learning and the relevance vector machine," *J. Mach. Learning Res.*, vol. 1, pp. 211–244, Jun. 2001.

[53] B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA: MIT Press, 2002.

[54] P. Vincent and Y. Bengio, "Kernel matching pursuit," *Mach. Learning*, vol. 48, nos. 1–3, pp. 165–187, 2002.

[55] G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan, "Learning the kernel matrix with semidefinite programming," *J. Mach. Learning Res.*, vol. 5, pp. 27–72, Dec. 2004.

[56] F. Schwenker, H. A. Kestler, and G. Palm, "Three learning phases for radial-basis-function networks," *Neural Netw.*, vol. 14, pp. 439–458, May 2001.

[57] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining Knowl. Discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[58] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," Royal-Holloway College, Univ. London, London, U.K., Tech. Rep. TR-98-030, 1998.

[59] J. Weston, A. Gammerman, M. O. Stitson, V. Vapnik, V. Vovk, and C. Watkins, "Support vector density estimation," in *Advances in Kernel Methods—Support Vector Learning*, B. Schölkopf, C. Burges, and A. J. Smola, Eds. Cambridge, MA: MIT Press, 1999, pp. 293–306.

[60] S. Mukherjee and V. Vapnik, "Support vector method for multivariate density estimation," MIT AI Lab., Cambridge, MA, Tech. Rep. A.I. Memo 1653, 1999.

[61] V. Vapnik and S. Mukherjee, "Support vector method for multivariate density estimation," in *Advances in Neural Information Processing Systems*, S. Solla, T. Leen, and K. R. Müller, Eds. Cambridge, MA: MIT Press, 2000, pp. 659–665.

[62] M. Girolami and C. He, "Probability density estimation from optimally condensed data samples," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 10, pp. 1253–1264, Oct. 2003.

[63] X. Hong, S. Chen, and C. J. Harris, "A fast linear-in-the-parameters classifier construction algorithm using orthogonal forward selection to minimize leave-one-out misclassification rate," *Int. J. Syst. Sci.*, vol. 39, no. 2, pp. 119–125, 2008.

[64] S. Chen, X. Hong, and C. J. Harris, "An orthogonal forward regression technique for sparse kernel density estimation," *Neurocomputing*, vol. 71, pp. 931–943, Jan. 2008.

[65] F. Sha, L. K. Saul, and D. D. Lee, "Multiplicative updates for nonnegative quadratic programming in support vector machines," Univ. Pennsylvania, Philadelphia, Tech. Rep. MS-CIS-02-19, 2002.

[66] S. Chen, X. Hong, C. J. Harris, and X. X. Wang, "Identification of nonlinear systems using generalized kernel models," *IEEE Trans. Control Syst. Technol.*, vol. 13, no. 3, pp. 401–411, May 2005.

[67] S. Chen, X. X. Wang, and C. J. Harris, "Experiments with repeating weighted boosting search for optimization in signal processing applications," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 35, no. 4, pp. 682–693, Aug. 2005.

[68] S. Chen, X. Hong, and C. J. Harris, "Orthogonal forward selection for constructing the radial basis function network with tunable nodes," in *Proc. Int. Conf. Intell. Comput.*, Hefei, China, Aug. 23–26, 2005, pp. 777–786.

[69] S. Chen, X. Hong, and C. J. Harris, "Construction of RBF classifiers with tunable units using orthogonal forward selection based on leave-one-out misclassification rate," in *Proc. Int. Joint Conf. Neural Netw.*, Vancouver, B.C., Canada, Jul. 16–21, 2006, pp. 6390–6394.

[70] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, vol. 4. Perth, Australia, Nov. 27–Dec. 1, 1995, pp. 6390–6394.

[71] J. Kennedy, and R. C. Eberhart, *Swarm Intelligence*. San Mateo, CA: Morgan Kaufmann, 2001.

[72] D. W. van der Merwe and A. P. Engelbrecht, "Data clustering using particle swarm optimization," in *Proc. Congr. Evol. Comput.*, Canberra, Australia, Dec. 8–12, 2003, pp. 215–220.

[73] A. Ratnaweera, S. K. Halgamuge, and H. C. Watson, "Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 240–255, Jun. 2004.

[74] M. G. H. Omran, "Particle swarm optimization methods for pattern recognition and image processing," Ph.D. dissertation, Dept. Comput. Sci., Univ. Pretoria, Pretoria, South Africa, 2005.

[75] S. M. Guru, S. K. Halgamuge, and S. Fernando, "Particle swarm optimisers for cluster formation in wireless sensor networks," in *Proc. Int. Conf. Intell. Sensors, Sensor Netw. Inform. Process.*, Melbourne, Australia, Dec. 5–8, 2005, pp. 319–324.

[76] K. W. Chau, "Particle swarm optimization training algorithm for ANNs in stage prediction of Shing Mun River," *J. Hydrol.*, vol. 329, pp. 363–367, Oct. 2006.

[77] K. K. Soo, Y. M. Siu, W. S. Chan, L. Yang, and R. S. Chen, "Particle-swarm-optimization-based multiuser detector for CDMA communications," *IEEE Trans. Veh. Technol.*, vol. 56, no. 5, pp. 3006–3013, Sep. 2007.

[78] J.-R. Zhang, J. Zhang, T.-M. Lok, and M. R. Lyu, "A hybrid particle swarm optimization-back-propagation algorithm for feedforward neural network training," *Appl. Math. Comput.*, vol. 185, pp. 1026–1037, Feb. 2007.

[79] T.-Y. Sun, C.-C. Liu, T.-Y. Tsai, and S.-T. Hsieh, "Adequate determination of a band of wavelet threshold for noise cancellation using particle swarm optimization," in *Proc. Congr. Evol. Comput.*, Hong Kong, Jun. 2008, pp. 1168–1175.

[80] J. Y. Chen, Z. Qin, and J. Jia, "A PSO-based subtractive clustering technique for designing RBF neural networks," in *Proc. Congr. Evol. Comput.*, Hong Kong, Jun. 2008, pp. 2047–2052.

[81] A. A. Cardi, H. A. Firpi, M. T. Bement, and S. Y. Liang, "Workpiece dynamic analysis and prediction during chatter of turning process," *Mech. Syst. Signal Process.*, vol. 22, pp. 1481–1494, Aug. 2008.

[82] H.-W. Ge, F. Qian, Y.-C. Liang, W.-L. Du, and L. Wang, "Identification and control of nonlinear systems by a dissimilation particle swarm optimization-based Elman neural network," *Nonlinear Anal.: Real World Applicat.*, vol. 9, pp. 1345–1360, Sep. 2008.

[83] H. Liu and J. Li, "A particle swarm optimization-based multiuser detection for receive-diversity-aided STBC systems," *IEEE Signal Process. Lett.*, vol. 15, pp. 29–32, 2008.

[84] W.-F. Leong and G. G. Yen, "PSO-based multiobjective optimization with dynamic population size and adaptive local archives," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 38, no. 5, pp. 1270–1293, Oct. 2008.

[85] C.-J. Lin, C.-H. Chen, and C.-T. Lin, "A hybrid of cooperative particle swarm optimization and cultural algorithm for neural fuzzy networks and its prediction applications," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 39, no. 1, pp. 55–68, Jan. 2009.

[86] Z.-H. Zhan, J. Zhang, Y. Li, and H. S.-H. Chung, "Adaptive particle swarm optimization," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 39, no. 6, pp. 1362–1381, Dec. 2009.

[87] V. S. Pappala, I. Erlich, K. Rohrig, and J. Dobschinski, "A stochastic model for the optimal operation of a wind-thermal power system," *IEEE Trans. Power Syst.*, vol. 24, no. 2, pp. 940–950, May 2009.

[88] K. V. Deligkaris, Z. D. Zaharis, D. G. Kampitaki, S. K. Goudos, I. T. Rekanos, and M. N. Spasos, "Thinned planar array design using Boolean PSO with velocity mutation," *IEEE Trans. Magnet.*, vol. 45, no. 3, pp. 1490–1493, Mar. 2009.

[89] B. Biswal, P. K. Dash, and B. K. Panigrahi, "Power quality disturbance classification using fuzzy C-means algorithm and adaptive particle swarm optimization," *IEEE Trans. Ind. Electron.*, vol. 56, no. 1, pp. 212–220, Jan. 2009.

[90] H. Wu, J. Geng, R. Lin, J. Qiu, W. Liu, J. Chen, and S. Liu, "An improved comprehensive learning particle swarm optimization and its application to the semiautomatic design of antennas," *IEEE Trans. Antennas and Propagation*, vol. 57, pp. 3018–3028, Oct. 2009.

[91] M. T. Hagh, H. Taghizadeh, and K. Razi, "Harmonic minimization in multilevel inverters using modified species-based particle swarm optimization," *IEEE Trans. Power Electron.*, vol. 24, no. 10, pp. 2259–2267, Oct. 2009.

[92] M. R. AlRashidi and M. E. El-Hawary, "A survey of particle swarm optimization applications in electric power systems," *IEEE Trans. Evolutionary Computat.*, vol. 13, pp. 913–918, Aug. 2009.

[93] F.-J. Lin, S.-Y. Chen, L.-T. Teng, and H. Chu, "Recurrent functional-link-based fuzzy neural network controller with improved particle swarm optimization for a linear synchronous motor drive," *IEEE Trans. Magnetics*, vol. 45, no. 8, pp. 3151–3165, Aug. 2009.

[94] M. Pedrasa, T. D. Spooner, and I. F. MacGill, "Scheduling of demand side resources using binary particle swarm optimization," *IEEE Trans. Power Syst.*, vol. 24, pp. 1173–1181, Aug. 2009.

[95] N. He, D.-G. Xu, and L. Huang, "The application of particle swarm optimization to passive and hybrid active power filter design," *IEEE Trans. Ind. Electron.*, vol. 56, no. 8, pp. 2841–2851, Aug. 2009.

[96] Y.-X. Liao, J.-H. She, and M. Wu, "Integrated hybrid-PSO and fuzzy-NN decoupling control for temperature of reheating furnace," *IEEE Trans. Ind. Electron.*, vol. 56, no. 7, pp. 2704–2714, Jul. 2009.

[97] P. Demarcke, H. Rogier, R. Goossens, and P. De Jaeger, "Beamforming in the presence of mutual coupling based on constrained particle swarm

optimization," *IEEE Trans. Antennas Propag.*, vol. 57, no. 6, pp. 1655–1666, Jun. 2009.

[98] N. C. Chauhan, M. V. Kartikeyan, and A. Mittal, "CAD of RF windows using multiobjective particle swarm optimization," *IEEE Trans. Plasma Sci.*, vol. 37, no. 6, pp. 1104–1109, Jun. 2009.

[99] G. G. Yen and W. F. Leong, "Dynamic multiple swarms in multiobjective particle swarm optimization," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 39, pp. 890–911, Jul. 2009.

[100] F.-J. Lin, L.-T. Teng, J.-W. Lin, and S.-Y. Chen, "Recurrent functional-link-based fuzzy-neural-network-controlled induction-generator system using improved particle swarm optimization," *IEEE Trans. Ind. Electron.*, vol. 56, no. 5, pp. 1557–1577, May 2009.

[101] M. Ramezani, M.-R. Haghifam, C. Singh, H. Seifi, and M. P. Moghaddam, "Determination of capacity benefit margin in multiarea power systems using particle swarm optimization," *IEEE Trans Power Systems*, vol. 24, no. 2, pp. 631–641, May 2009.

[102] H. R. Marateb and K. C. McGill, "Resolving superimposed MUAPs using particle swarm optimization," *IEEE Trans. Biomed. Eng.*, vol. 56, no. 3, pp. 916–919, Mar. 2009.

[103] H.-L. Wei, S. A. Billings, Y.-F. Zhao, and L.-Z. Guo, "Lattice dynamical wavelet neural networks implemented using particle swarm optimization for spatio-temporal system identification," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 181–185, Jan. 2009.

[104] K. D. Sharma, A. Chatterjee, and A. Rakshit, "A hybrid approach for design of stable adaptive fuzzy controllers employing Lyapunov theory and particle swarm optimization," *IEEE Trans. Fuzzy Syst.*, vol. 17, no. 2, pp. 329–342, Apr. 2009.

[105] S. K. Goudos, Z. D. Zaharis, D. G. Kampitaki, I. T. Rekanos, and C. S. Hilas, "Pareto optimal design of dual-band base station antenna arrays using multiobjective particle swarm optimization with fitness sharing," *IEEE Trans. Magnetics*, vol. 45, no. 3, pp. 1522–1525, Mar. 2009.

[106] L.-F. Wang and C. Singh, "Multicriteria design of hybrid power generation systems based on a modified particle swarm optimization algorithm," *IEEE Trans. Energy Conversion*, vol. 24, no. 1, pp. 163–172, Mar. 2009.

[107] K. Zielinski, P. Weitkemper, R. Laur, and K.-D. Kammeyer, "Optimization of power allocation for interference cancellation with particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 13, no. 1, pp. 128–150, Feb. 2009.

[108] R. Saeidi, H. R. S. Mohammadi, T. Ganchev, and R. D. Rodman, "Particle swarm optimization for sorted adapted Gaussian mixture models," *IEEE Trans. Audio Speech Language Process.*, vol. 17, no. 2, pp. 344–353, Feb. 2009.

[109] N. M. Kwok, Q. P. Ha, D.-K. Liu, and G. Fang, "Contrast enhancement and intensity preservation for gray-level images using multiobjective particle swarm optimization," *IEEE Trans. Autom. Sci. Eng.*, vol. 6, no. 1, pp. 145–155, Jan. 2009.

[110] S. Maheswararajah, S. K. Halgamuge, and M. Premaratne, "Sensor scheduling for target tracking by suboptimal algorithms," *IEEE Trans. Vehicular Technol.*, vol. 58, no. 3, pp. 1467–1479, Mar. 2009.

[111] M. A. Montes de Oca, T. Stützle, M. Birattari, and M. Dorigo, "Frankenstein's PSO: A composite particle swarm optimization algorithm," *IEEE Trans. Evol. Comput.*, vol. 13, no. 5, pp. 1120–1132, Oct. 2009.

[112] W. Qiao, G. K. Venayagamoorthy, and R. G. Harley, "Missing-sensor-fault-tolerant control for SSSC FACTS device with real-time implementation," *IEEE Trans. Power Delivery*, vol. 24, no. 2, pp. 740–750, Apr. 2009.

[113] T. Ince, S. Kiranyaz, and M. Gabbouj, "A generic and robust system for automated patient-specific classification of ECG signals," *IEEE Trans. Biomed. Eng.*, vol. 56, no. 5, pp. 1415–1426, May 2009.

[114] S. Kwak, S. Lee, S. Lee, W.-S. Kim, J.-K. Lee, C. Park, J. Bae, J.-B. Song, H. Lee, K. Choi, K. Seong, H. Jung, and S.-Y. Hahn, "Design of HTS magnets for a 2.5 MJ SMES," *IEEE Trans. Appl. Superconductivity*, vol. 19, no. 3, pp. 1985–1988, Jun. 2009.

[115] S. Y. Kim, N. H. Myung, and M. J. Kang, "Antenna mask design for SAR performance optimization," *IEEE Geosci. Remote Sensing Lett.*, vol. 6, pp. 443–447, Jul. 2009.

[116] S. Das, A. Abraham, U. K. Chakraborty, and A. Konar, "Differential evolution using a neighborhood-based mutation operator," *IEEE Trans. Evol. Comput.*, vol. 13, no. 3, pp. 526–553, Jun. 2009.

[117] M. Donelli, D. Franceschini, P. Rocca, and A. Massa, "Three-dimensional microwave imaging problems solved through an efficient multiscaling particle swarm optimization," *IEEE Trans. Geosci. Remote Sensing*, vol. 47, no. 5, pp. 1467–1481, May 2009.

[118] M. Stone, "Cross validation choice and assessment of statistical predictions," *J. Roy. Statist. Soc. Ser. B*, vol. 36, no. 2, pp. 111–147, 1974.

[119] X. Hong, P. M. Sharkey, and K. Warwick, "Automatic nonlinear predictive model construction algorithm using forward regression and the PRESS statistic," *IEE Proc. Control Theory Applicat.*, vol. 150, no. 3, pp. 245–254, May 2003.

[120] S. Chen and J. Wigger, "Fast orthogonal least squares algorithm for efficient subset model selection," *IEEE Trans. Signal Process.*, vol. 43, pp. 1713–1715, Jul. 1995.

[121] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Mach. Learning*. Reading, MA: Addison-Wesley, 1989.

[122] K. F. Man, K. S. Tang, and S. Kwong, *Genetic Algorithms: Concepts and Design*. London, U.K.: Springer-Verlag, 1998.

[123] L. Ingber, "Simulated annealing: Practice versus theory," *Math. Comput. Modeling*, vol. 18, no. 11, pp. 29–57, 1993.

[124] S. Chen and B. L. Luk, "Adaptive simulated annealing for optimization in signal processing applications," *Signal Process.*, vol. 79, no. 1, pp. 117–128, Nov. 1999.

[125] S. A. Billings, S. Chen, and R. J. Backhouse, "The identification of linear and non-linear models of a turbocharged automotive diesel engine," *Mech. Syst. Signal Process.*, vol. 3, no. 2, pp. 123–142, 1989.

[126] S. A. Billings and W. S. F. Voon, "A prediction-error and stepwise-regression estimation algorithm for non-linear systems," *Int. J. Control*, vol. 44, no. 3, pp. 803–822, 1986.

[127] A. Asuncion and D. J. Newman, *UCI Mach. Learning Repository* [Online]. Available: http://www.ics.uci.edu/∼mlearn/MLRepository.html

[128] G. Rätsch, T. Onoda, and K. R. Müller, *Data Sets in [129]* [Online]. Available: http://ida.first.fhg.de/projects/bench/benchmarks.htm

[129] G. Rätsch, T. Onoda, and K. R. Müller, "Soft margins for AdaBoost," *Mach. Learning*, vol. 42, no. 3, pp. 287–320, Mar. 2001.

[130] A. Choudhury, "Fast machine learning algorithms for large data," Ph.D. dissertation, School Eng. Sci., Univ. Southampton, Southampton, U.K., 2002.

[131] S. Chen, X. Hong, and C. J. Harris, "Sparse kernel density construction using orthogonal forward regression with leave-one-out test score and local regularization," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 34, no. 4, pp. 1708–1717, Aug. 2004.

[132] G. McLachlan and D. Peel, *Finite Mixture Models*. New York: Wiley, 2000.

[133] J. A. Bilmes, "A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models," Univ. Berkeley, Berkeley, CA, Tech. Rep. ICSI-TR-97-021, 1997.

[134] B. Efron and R. J. Tibshirani, *An Introduction to Bootstrap*. London: Chapman and Hall, 1993.

Sheng Chen (M'90–SM'97–F'08) received the B.E. degree from the Huadong Petroleum Institute, Dongying, China, in January 1982, and the Ph.D. degree from City University, London, U.K., in September 1986, both in control engineering. He received the D.Sc. degree from the University of Southampton, Southampton, U.K., in 2004.

From October 1986 to August 1999, he held research and academic appointments with the University of Sheffield, Sheffield, U.K., the University of Edinburgh, Edinburgh, U.K., and the University of Portsmouth, Portsmouth, U.K. Since September 1999, he has been with the School of Electronics and Computer Science, University of Southampton. He has published over 400 research papers. His research interests include wireless communications, adaptive signal processing for communications, machine learning, and evolutionary computation methods.

Dr. Chen is a Fellow of the Institution of Engineering and Technology. In the database of the world's most highly cited researchers, compiled by the Institute for Scientific Information, Philadelphia, PA, he is on the list of the most highly cited researchers in the engineering category.

Xia Hong (SM'02) received the B.S. and M.S. degrees from the National University of Defense Technology, Changsha, Hunan, China, in 1984 and 1987, respectively, and the Ph.D. degree from the University of Sheffield, Sheffield, U.K., in 1998, all in automatic control.

She was a Research Assistant with the Beijing Institute of Systems Engineering, Beijing, China from 1987 to 1993. She was a Research Fellow with the School of Electronics and Computer Science, University of Southampton, Southampton, U.K., from 1997 to 2001. Since 2001, she has been with the School of Systems Engineering, University of Reading, Reading, U.K., where she currently holds a Readership Post. She has published over 180 research papers, and coauthored a research book. She is actively engaged in research into nonlinear systems identification, data modeling, estimation and intelligent control, neural networks, pattern recognition, learning theory, and their applications.

Dr. Hong was awarded the Donald Julius Groen Prize by the Institution of Mechanical Engineers in 1999.

Chris J. Harris received the Ph.D. degree from the University of Southampton, Southampton, U.K., in 1972, and the Doctor of Sciences (D.Sc.) degree from the University of Southampton, in 2001.

He has held appointments with the University of Hull, Hull, U.K., the University of Manchester Institute of Science and Technology, Manchester, U.K., the University of Oxford, Oxford, U.K., and the University of Cranfield, Cranfield, U.K. He was employed by the U.K. Ministry of Defense. In 1987 he returned to the School of Electronics and Computer Science, University of Southampton, as the Lucas Professor of Aerospace Systems Engineering to establish the Advanced Systems Research Group and, more recently, the Image, Speech and Intelligent Systems Group. He has authored and co-authored 12 research books and over 400 research papers, and is the Associate Editor of numerous international journals. His research interests include the general area of intelligent and adaptive systems theory and its application to intelligent autonomous systems such as autonomous vehicles, management infrastructures such as command and control, intelligent control, and estimation of dynamic processes, multi-sensor data fusion, and systems integration.

Dr. Harris was elected to the Royal Academy of Engineering in 1996, and was awarded the Institution of Electrical Engineers (IEE) Senior Achievement Medal in 1998 for his work in autonomous systems and the IEE Faraday Medal in 2001 for his work in intelligent control and neurofuzzy systems.