Engineering, 2009, 2, 55−74
doi:10.4236/eng.2009.12008 Published Online August 2009 (http://www.SciRP.org/journal/eng/).

Orthogonal-Least-Squares Forward Selection for Parsimonious Modelling from Data

Sheng CHEN
School of Electronics and Computers Sciences, University of Southampton, Southampton, UK
Email: *sqc@ecs.soton.ac.uk*
Received March 27, 2009; *revised May* 5, 2009; *accepted May* 12, 2009

Abstract

The objective of modelling from data is not that the model simply fits the training data well. Rather, the goodness of a model is characterized by its generalization capability, interpretability and ease for knowledge extraction. All these desired properties depend crucially on the ability to construct appropriate parsimonious models by the modelling process, and a basic principle in practical nonlinear data modelling is the parsimonious principle of ensuring the smallest possible model that explains the training data. There exists a vast amount of works in the area of sparse modelling, and a widely adopted approach is based on the linear-in-the-parameters data modelling that include the radial basis function network, the neurofuzzy network and all the sparse kernel modelling techniques. A well tested strategy for parsimonious modelling from data is the orthogonal least squares (OLS) algorithm for forward selection modelling, which is capable of constructing sparse models that generalise well. This contribution continues this theme and provides a unified framework for sparse modelling from data that includes regression and classification, which belong to supervised learning, and probability density function estimation, which is an unsupervised learning problem. The OLS forward selection method based on the leave-one-out test criteria is presented within this unified data-modelling framework. Examples from regression, classification and density estimation applications are used to illustrate the effectiveness of this generic parsimonious modelling approach from data.

Keywords: Data Modelling, Regression, Classification, Density Estimation, Orthogonal Least Squares Algorithm

1. Introduction

Data modelling is an important and recurrent theme in all the fields of engineering. Various data modelling 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 the two-class classification problems, and the goodness of a classifier is determined by its test error probability or misclassification rate. Despite of these differences, classifier construction can be expressed in the same framework of regression modelling. The third class of data modelling, 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 from the observation data and using it as the target function for the cumulative distribution function of the PDF estimation. This contribution adopts this unified regression framework for data modelling.

The theory and practice of linear regression modelling is well established [10−12], and the least squares (LS) method [13] has been a basic toolkit for data modelling. Since real-world phenomena that generate data are nonlinear to some extent, nonlinear models are often required in order to achieve adequate modelling accuracy. Over the past three decades, extensive efforts have been directed onto developing coherent and concise methods of nonlinear regression modelling [14−35]. A data modelling problem generally consists of two basic components: determining the model structure and estimating or fitting the model parameters. Parameter fitting is relatively straightforward if the model structure is known *a priori* but this information is rarely available in practice and must be learnt. Determining the model structure is crucial in any practical data modelling problem, and a fundamental principle is that the model should be no more complex than is required to capture the underlying data generating mechanisms. This concept known as the parsimonious principle is particular relevant in nonlinear data modelling because the size of a nonlinear model can easily become explosively large. An over complicated model may simply fit to the noise in the training data, resulting overfitting. An overfitted model does not capture the underlying system structure and will perform badly on new data. In general, a huge model not only may have poor generalisation performance but also has little practical value in data analysis and system design.

There exists a vast amount of works in the area of parsimonious nonlinear regression modelling but the most popular approach is perhaps to adopt a linear-in-the-parameters nonlinear model. This is typically achieved by placing a radial basis function (RBF) or other type of kernel on each training data sample and a sparse representation is then sought which possesses excellent generalisation performance [27−68]. Adopting a linear-in-the-parameters nonlinear model structure is attractive because many existing linear data modelling techniques can be applied successfully, providing that the model structure determination can be carried out effectively to guarantee a sufficiently parsimonious final model. Among the various linear-in-the-parameters nonlinear data modelling techniques, the support vector machine (SVM) method and other sparse kernel modelling methods [54−68] have become popular in the recent years. In particular, the SVM technique [54] is widely regarded as the state-of-the-art technique for regression and classification applications, and it has also been proposed as a promising tool for sparse kernel density estimation [69−71]. The formulation of SVM embodies the structural risk minimization principle, thus combining excellent generalisation properties with a sparse model representation. Despite of these attractive features and many good empirical results obtained using the SVM method, data modelling practicians have realized that the ability for the SVM method to produce sparse models has perhaps been overstated.

The orthogonal least squares (OLS) algorithm [34], developed in the late 1980s for nonlinear system modelling, remains highly popular for nonlinear data modelling practicians, for the reason that the algorithm is simple and efficient, and is capable of producing parsimonious linear-in-the-parameters nonlinear models with good generalisation performance. Unlike the SVM and many other sparse kernel modelling techniques, which work on the full kernel model defined on the training data set to obtain a sparse model, the OLS method [34] adopts the forward selection to build up an adequate model by only selecting significant regressors. Since its derivation, many enhanced variants of the OLS based forward regression algorithm have been proposed [37−53]. In particular, the local regularisation assisted OLS algorithm [39,41], which employs the multiple regularisers to enforce the model sparsity [60], has been shown to be capable of producing very sparse regression models that generalise well. A significant improvement to the original OLS algorithm for sparse regression modelling is to enhance the algorithm with optimal experimental design criteria [39,45,47,48,50,51]. In a traditional forward regression procedure, a separate stopping criterion is required to terminate the selection procedure at an appropriate model size in order for example to avoid an over-fitted model. Typically, information based criteria, such as the AIC [72] and the minimum description length [73], were adopted to terminate the model selection process. An information based criterion can be viewed as a model structure regularisation by using a penalty term to penalise large sized models. However, the penalty term in an information based criterion does not help to determine which model term should be selected. Multiple regularisers, i.e. local regularisation [39,41,60], and optimal experimental design criteria [39,45,47,48,50,51] offer better solutions as model structure regularisation as they are directly linked to model efficiency and parameter robustness [74].

The basic criterion for most model construction procedures, including the original OLS algorithm [34,35], is the training MSE. However, the goodness of a regression model is its generalisation capability. Therefore, a better and more natural approach is using a criterion of model generalisation performance directly in the model selection procedure rather than only using it as a measure of model complexity. The evaluation of model generalisation capability is directly based on the concept of cross validation [75], and a commonly used cross validation is the delete-one or leave-one-out (LOO) cross validation [2,76,77]. One of the most important improvements to the OLS algorithm based forward regression is the development of the OLS forward selection based on the LOO test score or MSE [40,46,47,49], which is a measure of the model generalisation performance. The use of the LOO estimate for general nonlinear-in-the- parameters models has been studied for example in [78−80]. However, even for the class of linear-in-the-parameters models, computation of the LOO MSE is normally expensive and the use of the LOO statistic in model selection is

generally prohibitive. Owing to the orthogonal property of the OLS algorithm, the calculation of the LOO statistics beomes efficient and model selection based on the LOO test score is made computationally affordable [46]. An additional advantage of adopting the LOO test score based OLS algorithm is that the model construction process becomes truly automatic without the need for the user to specify some additional terminating criterion [46]. Our empirical modelling results for regression [40], classification [52] and kernel density estimation [43,44] have demonstrated that this OLS algorithm based on the LOO test score coupled with local regularisation compares favourably with the SVM and many other existing state-of-the-art sparse kernel modelling methods, in terms of generalisation capability and model sparsity as well as the computational complexity of model construction.

This contribution is organized as follows. Section 2 presents the regression modelling framework, which unifies all the three classes of data modelling applications, namely, regression, classification and PDF estimation. In particular, the unsupervised density learning is converted into a supervised regression one by adopting the Parzen window (PW) estimate as the target function [44]. Based on this unified data-modelling framework, the OLS forward selection algorithm using the LOO test criteria and local regularisation is detailed in Section 3. More specifically, for regression modelling, the model selection criterion is based on the LOO test MSE, while for classification applications, the LOO misclassification rate is employed for model selection. In kernel density estimation, the kernel weights must satisfy the nonnegative and unity constraints, and a combined approach is adopted to tackle this constrained regression modelling. A sparse kernel density estimate is first selected by the efficient OLS algorithm based on the LOO test score and local regularisation. The kernel weights of the final model are then updated using the multiplicative nonnegative quadratic programming (MNQP) algorithm [61,81] to meet the nonnegative and unity constraints. The MNQP algorithm additionally has a desired property of forcing some kernel weights to (near) zero values, and thus further reducing the model size [61,81]. The experimental results are included in Section 4, where empirical examples taken from regression, classification and PDF estimation applications demonstrate the effectiveness of the proposed OLS algorithm based on the LOO test criteria coupled with local regularisation within the unified data-modelling framework. The concluding remarks are summarised in Section 5.

2. A Unified Data Modelling Framework

The three classes of data modelling, namely, regression, classification and PDF estimation, can be unified under the generic regression framework of sparse kernel data modelling based on the appropriate modelling criteria, where the kernel model is interpreted in a generic sense, namely, a kernel or nonlinear basis is placed on each training data sample and the model is obtained as a linear combination of all the bases defined on the training data set. For kernel density estimation, a kernel should also meet the usual requirement of a density distribution, i.e. the area under the kernel is unity. The objective is to derive a sparse model representation with excellent generalisation capability based on a training data set.

2.1. Regression Modelling

Consider the general nonlinear data generating mechanism governed by the nonlinear model

$$y = f(\mathbf{x}) + e \tag{1}$$

where $y \in \mathcal{R}$ is the system output, $\mathbf{x} = [x_1 x_2 ... x_m]^T \in \mathcal{R}^m$ the system input, e is a white noise process representing for example the observation noise, and $f : \mathcal{R}^m \to \mathcal{R}$ defines the unknown system mapping. Given a set of N training data samples, $D_N = \{\mathbf{x}_k, y_k\}_{k=1}^N$, the task is to infer a kernel model, $\hat{f} : \mathcal{R}^m \to \mathcal{R}$, of the form

$$\hat{y} = \hat{f}(\mathbf{x}) = \sum_{i=1}^{N} \beta_i K_\rho(\mathbf{x}, \mathbf{x}_i) \tag{2}$$

where $\hat{y}$ denotes the model output, β_i are the kernel weights and $K_\rho(\bullet, \bullet)$ is the chosen kernel function with a kernel width ρ. Many types of kernel function can be employed and a commonly used one is the Gaussian function of the form

$$K_\rho(\mathbf{x}, \mathbf{c}_k) = \begin{cases} e^{-\frac{\|\mathbf{x}-\mathbf{c}_k\|^2}{2\rho^2}}, & \text{for regression and classification,} \\ \dfrac{1}{(2\pi\rho^2)^{m/2}} e^{-\frac{\|\mathbf{x}-\mathbf{c}_k\|^2}{2\rho^2}}, & \text{for density estimation} \end{cases} \tag{3}$$

where $\mathbf{c}_k \in \mathcal{R}^m$ is the k-th kernel centre vector. The generic kernel model (2) is defined by placing a kernel at each of the training input samples $\mathbf{x}_k$ and forming a linear combination of all the bases defined on the training data set. A sparse representation is then sought by selecting kernel model with only N_s nonzero kernel weights, where $N_s << N$.

At a training data point $(\mathbf{x}_k, y_k)$, the kernel model (2) can be expressed as

$$y_k = \hat{y}_k + \epsilon_k = \sum_{i=1}^{N} \beta_i K_\rho(\mathbf{x}_k, \mathbf{x}_i) + \epsilon_k = \boldsymbol{\phi}_N^T(k)\boldsymbol{\beta}_N + \epsilon_k \tag{4}$$

where $\epsilon_k = y_k - \hat{y}_k$ is the modelling error at $\mathbf{x}_k$, $\boldsymbol{\beta}_N = [\beta_1 \beta_2 ... \beta_N]^T$ and $\boldsymbol{\phi}_N(k) = [K_{k,1} K_{k,2} ... K_{k,N}]^T$ with $K_{k,i}$

$= K_\rho(\mathbf{x}_k, \mathbf{x}_i)$. By defining $\mathbf{\Phi}_N = [\phi_1 \phi_2 ... \phi_N]$ with $\phi_k = [K_{1,k} K_{2,k} ... K_{N,k}]^T$ for $1 \le k \le N$, $\mathbf{y} = [y_1 y_2 ... y_N]^T$ and $\epsilon = [\epsilon_1 \epsilon_2 \ldots \epsilon_N]^T$, the regression model (4) over the training data set D_N can be expressed in the matrix form

$$\mathbf{y} = \mathbf{\Phi}_N \boldsymbol{\beta}_N + \epsilon \tag{5}$$

Note that ϕ_k is the k-th column of $\mathbf{\Phi}_N$, while $\phi_N^T(k)$ denotes the k-th row of $\mathbf{\Phi}_N$. Let an orthogonal decomposition of the regression matrix $\mathbf{\Phi}_N$ be

$$\mathbf{\Phi}_N = \mathbf{W}_N \mathbf{A}_N \tag{6}$$

where

$$\mathbf{A}_N = \begin{bmatrix} 1 & a_{1,2} & \cdots & a_{1,N} \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & a_{N-1,N} \\ 0 & \cdots & 0 & 1 \end{bmatrix} \tag{7}$$

and

$$\mathbf{W}_N = [\mathrm{w}_1 \mathrm{w}_2 \cdots \mathrm{w}_N] \tag{8}$$

with orthogonal columns satisfying $\mathrm{w}_i^T \mathrm{w}_j = 0$, if $i \ne j$. The regression model (5) can alternatively be expressed as

$$\mathrm{y} = \mathbf{W}_N \mathrm{g}_N + \epsilon \tag{9}$$

where the weight vector $\mathrm{g}_N = [g_1\ g_2 \cdots g_N]^T$ defined in the orthogonal model space satisfies the triangular system $\mathbf{A}_N \boldsymbol{\beta}_N = \mathrm{g}_N$. The space spanned by the original model bases ϕ_k, $1 \le k \le N$, is identical to the space spanned by the orthogonal model bases w_k, $1 \le k \le N$, and the model is equivalently expressed by

$$\hat{y}_k = \mathbf{w}_N^T(k) g_N \tag{10}$$

where $\mathrm{w}_N^T(k) = [w_{k,1} w_{k,2} \cdots w_{k,N}]$ is the k-th row of $\mathbf{W}_N$. A procedure that can be used to perform the orthogonalisation (6) is summarised in Appendix A.

2.2. Classification Application

Consider the two-class classification problem with the given training data set $D_N = \{\mathbf{x}_k, y_k\}_{k=1}^N$, where $\mathrm{x}_k \in \mathcal{R}^m$ 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 kernel classifier of the form

$$\tilde{y}_k = \mathrm{sgn}(\hat{y}_k) \text{ with } \hat{y}_k = \sum_{i=1}^N \beta_i K_\rho(\mathbf{x}_k, \mathbf{x}_i) \tag{11}$$

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

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

Let us define the modelling error as $\epsilon_k = y_k - \hat{y}_k$. Then the classification model over the training data set D_N can be expressed in the regression model of (5) recited here again as

$$\mathbf{y} = \mathbf{\Phi}_N \boldsymbol{\beta}_N + \epsilon \tag{13}$$

or equivalently in the orthogonal regression model of (9) rewritten here again as

$$\mathbf{y} = \mathbf{W}_N \mathrm{g}_N + \epsilon \tag{14}$$

where all the relevant notations are as defined in Subsection 2.1. It is clear that the kernel classifier construction can be expressed in the same kernel regression modelling framework of Subsection 2.1, and the only difference 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. The objective is again to derive a sparse kernel model that posses good generalisation capability and contains only N_s significant kernels.

2.3. Kernel Density Estimation

Based on a 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 kernel density estimate of the form

$$\hat{p}(\mathbf{x}; \boldsymbol{\beta}_N, \rho) = \sum_{k=1}^N \beta_k K_\rho(\mathbf{x}, \mathbf{x}_k) \tag{15}$$

with the constraints

$$\beta_k \ge 0,\ 1 \le k \le N \tag{16}$$

and

$$\boldsymbol{\beta}_N^T 1_N = 1 \tag{17}$$

where $\boldsymbol{\beta}_N = [\beta_1 \beta_2 ... \beta_N]^T$ is again the kernel weight vector, ρ the kernel width and 1_N denotes the vector of ones with dimension N. The kernel function $K_\rho(\bullet, \bullet)$ is chosen to be the Gaussian kernel in this study. However, many other kernel functions can also be used in the density estimate (15). Following the approach of [44], this unsupervised kernel density learning is transformed into a supervised learning problem.

The well-known PW estimate [7], $\hat{p}(\mathbf{x}; \boldsymbol{\beta}_{\mathrm{Par}}, \rho_{\mathrm{Par}})$, is obtained by simply setting all the elements of $\boldsymbol{\beta}_{\mathrm{Par}}$ to $\frac{1}{N}$. The optimal kernel width ρ_{Par} is typically determined via cross validation [2,75]. The PW estimate is remarkably simple and accurate [7]. The PW estimate in fact can be derived as the maximum likelihood estimator using the divergence-based criterion [83]. The negative cross-entropy or divergence between the true density

$p(\mathbf{x})$ and the estimate $\hat{p}(\mathbf{x};\boldsymbol{\beta}_{\rm N},\rho)$ is defined as

$$\int_{\mathcal{R}^m} p(\mathbf{u})\log\hat{p}(\mathbf{u};\boldsymbol{\beta}_{\rm N},\rho)d\mathbf{u} \approx \frac{1}{N}\sum_{k=1}^{N}\log\hat{p}(\mathbf{x}_k;\boldsymbol{\beta}_{\rm N},\rho)$$
$$= \frac{1}{N}\sum_{k=1}^{N}\log(\sum_{n=1}^{N}\beta_n K_\rho(\mathbf{x}_k,\mathbf{x}_n)) \quad \textbf{(18)}$$

Minimising this divergence subject to the constraints (16) and (17) leads to $\beta_n = \frac{1}{N}$ for $1 \le n \le N$, i.e. the PW estimate. 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. This high test cost has motivated the research on the sparse kernel density estimation techniques [43,44,69−71,81,82].

We may regard the PW estimate as the "observation" of the true density contaminated by some "observation noise", namely

$$\hat{p}(\mathbf{x};\boldsymbol{\beta}_{\rm Par},\rho_{\rm Par}) = p(\mathbf{x}) + \tilde{\epsilon}(\mathrm{x}) \quad \textbf{(19)}$$

Thus the generic kernel density estimation problem (15) can be viewed as the following regression problem with the PW estimate as the "desired response" or target function

$$\hat{p}(\mathbf{x};\boldsymbol{\beta}_{\rm Par},\rho_{\rm Par}) = \sum_{k=1}^{N}\beta_k K_\rho(\mathbf{x},\mathbf{x}_k) + \epsilon(\mathrm{x}) \quad \textbf{(20)}$$

subject to the constraints (16) and (17), where $\epsilon(\mathrm{x})$ is the modelling error at x. Define $y_k = \hat{p}(\mathrm{x};\boldsymbol{\beta}_{\rm Par},\rho_{\rm Par})$ and $\epsilon_k = \epsilon(\mathrm{x}_k)$. Then the generic kernel density estimation problem is expressed in the same kernel regression modelling framework of (5) recited here again as

$$\mathrm{y} = \boldsymbol{\Phi}_N\boldsymbol{\beta}_N + \boldsymbol{\epsilon} \quad \textbf{(21)}$$

subject to the nonnegative constraint (16) and the unity constrain (17), where all the relevant notations have been defined in Subsection 2.1. The regression model (21) can of course be written equivalently in the orthogonal form of (9) which is recited here again as

$$\mathrm{y} = \mathbf{W}_N\mathrm{g}_N + \boldsymbol{\epsilon} \quad \textbf{(22)}$$

The objective is to obtain a sparse *Ns*-term kernel model, satisfying the kernel weight constraints (16) and (17) and yet having a test performance comparable to that of the full-sample optimized PW estimate.

3. Orthogonal-Least-Squares Algorithm

As established in the previous section, the regression, classification and PDF estimation can all be unified within the common regression modelling framework. Therefore, the OLS forward selection based on the LOO test criteria and local regularization (OLS-LOO-LR) [40] provides an efficient algorithm to construct a sparse kernel model that generalise well. For the regression and kernel density modelling, the LOO MSE criterion is an appropriate measure of model's generalisation capability for subset model selection, while for kernel classifier construction, the LOO misclassification rate offers a proper measure of classifier's generalisation performance for selecting significant kernels [52]. Sparse kernel density (SKD) construction is special as it is formulated as a constrained regression modelling, where the kernel weights must meet the nonnegative and unity constraints. A combined OLS-LOO-LR and MNQP approach is adopted for this constrained regression modelling [44], where the OLS-LOO-LR algorithm determines the sparse kernel model structure by selecting a subset of significant kernels while the MNOP algorithm [61,81] computes the kernel weights of the selected SKD estimate.

3.1. Sparse Kernel Regression Model Construction

The local regularization aided least squares solution for the weight parameter vector g_N can be obtained by minimizing the following regularised error criterion [41]

$$J_R(g_N,\boldsymbol{\lambda}_N) = \boldsymbol{\epsilon}^T\boldsymbol{\epsilon} + \sum_{i=1}^{N}\lambda_i g_i^2 = \boldsymbol{\epsilon}^T\boldsymbol{\epsilon} + g_N^T\boldsymbol{\Lambda}_N g_N \quad \textbf{(23)}$$

where $\boldsymbol{\lambda}_N = [\lambda_1\lambda_2...\lambda_N]^T$ is the vector of regularisation parameters, and $\boldsymbol{\Lambda}_N = \mathrm{diag}\{\lambda_1\lambda_2...\lambda_N\}$. In fact setting $\frac{\partial J_R}{\partial g_N} = 0$ leads to the normal equation

$$\mathbf{W}_N^T\mathbf{y} = (\mathbf{W}_N^T\mathbf{W}_N + \boldsymbol{\Lambda}_N)g_N \quad \textbf{(24)}$$

Because $\mathbf{W}_N^T\mathbf{W}_N + \boldsymbol{\Lambda}_N$ is diagonal, the solution is $g_l = \mathrm{w}_l^T\mathrm{y}/(\mathrm{w}_l^T\mathrm{w}_l + \lambda_l)$ for $1 \le l \le N$, which is also given in Appendix A. The criterion (23) is rooted in the Bayesian learning framework. According to the Bayesian learning theory [24,39,60], the optimal g_N is obtained by maximizing the posterior probability of g_N, which can be shown to be

$$p(\mathbf{g}_N \mid \mathbf{y},\mathbf{h}_N,\omega) = \frac{p(\mathbf{y}\mid\mathbf{g}_N,\mathbf{h}_N,\omega)p(\mathbf{g}_N\mid\mathbf{h}_N,\omega)}{p(\mathbf{y}\mid\mathbf{h}_N,\omega)} \quad \textbf{(25)}$$

where $p(\mathbf{g}_N\mid\mathbf{h}_N,\omega)$ is the prior with $\mathbf{h}_N = [h_1h_2...h_N]^T$ denoting the vector of hyper-parameteres and ω a noise parameter (the inverse of the variance of ϵ), $p(\mathbf{y}\mid\mathbf{g}_N,\mathbf{h}_N,\omega)$ is called the likehood, and $p(\mathbf{y}\mid\mathbf{h}_N,\omega)$ is the evidence which does not depend on $\mathbf{g}_N$ explicitly. Under the assumption that ϵ is the white and has a

Gaussian distribution, the likelihood is given by

$$p(\mathbf{y} \mid \mathbf{g}_N, \mathbf{h}_N, \omega) = \prod_{i=1}^{N} \left(\frac{\omega}{2\pi} \right)^{N/2} e^{-\frac{\omega}{2} \boldsymbol{\epsilon}^T \boldsymbol{\epsilon}} \tag{26}$$

If the Gaussian prior is chosen, i.e.

$$p(\mathbf{g}_N \mid \mathbf{h}_N, \omega) = \prod_{i=1}^{N} \frac{\sqrt{h_i}}{\sqrt{2\pi}} e^{-\frac{h_i g_i^2}{2}} \tag{27}$$

maximising $\log(\, p(\mathbf{g}_N \mid \mathbf{h}_N, \omega)\,)$ with respect to $\mathbf{g}_N$ is equivalent to minimising the following Bayesian cost function

$$J_B(\mathbf{g}_N, \mathbf{h}_N, \omega) = \omega \boldsymbol{\epsilon}^T \boldsymbol{\epsilon} + \mathbf{g}_N^T \mathbf{H}_N \mathbf{g}_N \tag{28}$$

where $\mathbf{H}_N = \mathrm{diag}\{h_1, h_2, ..., h_N\}$. It is obvious that the criterion (23) is equivalent to the criterion (28) with the relationship

$$\lambda_i = \frac{h_i}{\omega}, 1 \le i \le N \tag{29}$$

The hyperparameters specify the prior distributions of $\mathbf{g}_N$. Since initially the optimal value of $\mathbf{g}_N$ is unknown, λ_i should be initialised to the same small value, and this corresponds to choose a same flat distribution for each prior of g_i in (27). The beauty of the Bayesian learning framework is that it learns not only the model parameters $\mathbf{g}_N$ but also the related hyperparameters $\mathbf{h}_N$. This can be done by iteratively optimizing $\mathbf{g}_N$ and $\mathbf{h}_N$ using the evidence procedure [24,39,60]. Applying this evidence procedure results in the following iterative updating formulas for the regularization parameters [39]

$$\lambda_i^{\text{new}} = \frac{\gamma_i^{\text{old}}}{N - \gamma^{\text{old}}} \frac{\boldsymbol{\epsilon}^T \boldsymbol{\epsilon}}{g_i^2}, 1 \le i \le N \tag{30}$$

where g_i for $1 \le i \le N$ denote the current estimated parameter values, and

$$\gamma = \sum_{i=1}^{N} \gamma_i \quad \text{with} \quad \gamma_i = \frac{\mathbf{w}_i^T \mathbf{w}_i}{\lambda_i + \mathbf{w}_i^T \mathbf{w}_i} \tag{31}$$

Usually a few iterations (typically less than 10) are sufficient to find a (near) optimal λ_N. The detailed derivation of the updating Formulas (30) and (31), quoted from [39], can be found in Appendix B. The use of multiple- regularisers or local regularisation is known to be capable of providing very sparse solutions [41,60].

It is highly desired to select a sparse model by directly optimizing the model generalisation capability, rather than minimising the training MSE. The OLS-LOO-LR algorithm achieves this objective by incrementally minimizing the LOO MSE criterion, which is a measure of the model's generalization performance [2,40,46,47,78–80]. At the *n*-th stage of the OLS forward selection procedure, an *n*-term model is selected. It can be shown that the LOO test error, denoted as $\boldsymbol{\epsilon}_k^{(n,-k)}$, for the selected *n*-term model is [40,46,47]

$$\epsilon_k^{(n,-k)} = \frac{\epsilon_k^{(n)}}{\eta_k^{(n)}} \tag{32}$$

where $\boldsymbol{\epsilon}_k^{(n)}$ is the usual *n*-term modelling error and $\eta_k^{(n)}$ is the associated LOO error weighting. The LOO MSE for the model with a size *n* is then defined by

$$J_n = \frac{1}{N} \sum_{k=1}^{N} (\boldsymbol{\epsilon}_k^{(n,-k)})^2 = \frac{1}{N} \sum_{k=1}^{N} \frac{\left(\epsilon_k^{(n)}\right)^2}{\left(\eta_k^{(n)}\right)^2} \tag{33}$$

The LOO MSE can be computed efficiently due to the fact that the *n*-term model error $\boldsymbol{\epsilon}_k^{(n)}$ and associated LOO error weighting $\eta_k^{(n)}$ can be calculated recursively according to [40,46,47]

$$\boldsymbol{\epsilon}_k^{(n)} = \boldsymbol{\epsilon}_k^{(n,-k)} - w_{k,n} \mathbf{g}_n \tag{34}$$

and

$$\eta_k^{(n)} = \eta_k^{(n-1)} - \frac{w_{k,n}^2}{w_n^T w_n + \lambda_n} \tag{35}$$

respectively, where $w_{k,n}$ is the *k*-th element of w_n. The derivation of the LOO test error (32) together with the recursive Formulas (34) and (35) is detailed in Appendix C.

The subset model selection procedure is carried out as follows. At the *n*-th stage of the selection procedure, a model term is selected among the remaining *n* to *N* candidates if the resulting *n*-term model produces the smallest LOO MSE J_n. The selection procedure is terminated when

$$J_{N_s+1} \ge J_{N_s} \tag{36}$$

yielding an N_s-term sparse model. It has been shown in [46] that the LOO statistic J_n is at least locally convex with respect to the model size *n*. That is, there exists an "optimal" model size N_s such that for $n \le N_s J_n$ decreases as *n* increases while the condition (36) holds. This property is extremely useful, as it enables the selection procedure to be automatically terminated with an N_s-term model, without the need for the user to specify a separate termination criterion. The sparse regression model selection procedure based on the OLS-LOO-LR algorithm is now summarised as follows.

Initialisation: Set $\lambda_i = 10^{-6}$ for $1 \le i \le N$, and set iteration index $I = 1$.

Step 1: Given the current $\boldsymbol{\lambda}_N$ and with the following initial conditions

$$\begin{cases} \epsilon_k^{(0)} = y_k \text{ and } \eta_k^{(0)} = 1, 1 \le k \le N, \\ J_0 = \dfrac{1}{N} \mathbf{y}^T \mathbf{y} = \dfrac{1}{N} \sum_{k=1}^{N} y_k^2 \end{cases} \tag{37}$$

use the procedure described in Appendix D to select

model with N_I terms.

Step 2: Update $\boldsymbol{\lambda}_N$ using (30) and (31) with $N = N_I$. If the pre-set maximum iteration number (e.g. 10) is reached, stop; otherwise set $I += 1$ and go to *Step* 1.

3.2. Sparse Kernel Classifier Construction

Since the generic kernel classifier construction takes the same form of regression modelling, the OLS-LOO-LR algorithm described in the previous subsection can be applied to select a sparse kernel classifier. However, the goal of a classifier is to minimise the misclassification or error rate, and the MSE in general is not an appropriate criterion for classifier construction. Note that the class label $y_k \in \{-1, +1\}$. Define the signed decision variable

$$s_k = \text{sgn}(y_k)\hat{y}_k = y_k\hat{y}_k \tag{38}$$

Then the misclassification rate over the training data set $D_N = \{\mathbf{x}_k, y_k\}_{k=1}^N$ is evaluated as

$$J_{\mathcal{M}} = \frac{1}{N}\sum_{k=1}^{N} \mathcal{I}_d(s_k) \tag{39}$$

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

$$\mathcal{I}_d(y) = \begin{cases} 1, & y \le 0, \\ 0, & y > 0. \end{cases} \tag{40}$$

The classifier's generalisation capability however is measured by the test error rate over data unseen in training. The same LOO cross validation concept [2,76,77] is adopted to provide a measure of classifier's generalisation capability.

Let the k-th data sample be removed from the training data set D_N, and the resulting LOO data set is used to construct and n-term classifier. The test output of the obtained LOO n-term model evaluated at the k-th data sample not used in training is again denoted by $\hat{y}_k^{(n,-k)}$. The associated LOO signed decision variable is then defined by

$$s_k^{(n,-k)} = y_k\hat{y}_k^{(n,-k)} \tag{41}$$

and the LOO misclassification rate can be computed by

$$J_n = \frac{1}{N}\sum_{k=1}^{N} \mathcal{I}_d(s_k^{(n,-k)}) \tag{42}$$

This LOO misclassification rate is a measure of the classifier's generalisation capability. Moreover, the LOO signed decision variable $s_k^{(n,-k)}$ can be calculated very fast owning to the orthogonal decomposition and, therefore, the LOO misclassification rate J_n can be evaluated efficiently [52]. Specifically, the LOO n-term modelling error is expressed by (also see Appendix C)

$$y_k - \hat{y}_k^{(n,-k)} = \frac{y_k - \hat{y}_k^{(n)}}{1 - \sum_{i-1}^{n} \dfrac{w_{k,i}^2}{\mathbf{w}_i^T\mathbf{w}_i + \lambda_i}} \tag{43}$$

Multiplying the both sides of (43) with y_k and applying $y_k^2 = 1$ yields

$$1 - s_k^{(n,-k)} = \frac{1 - y_k\hat{y}_k^{(n)}}{1 - \sum_{i-1}^{n} \dfrac{w_{k,i}^2}{\mathbf{w}_i^T\mathbf{w}_i + \lambda_i}} \tag{44}$$

From (44), the LOO n-term signed decision variable is given by

$$s_k^{(n,-k)} = \frac{\sum_{i=1}^{n} y_k g_i w_{k,i} - \sum_{i-1}^{n} \dfrac{w_{k,i}^2}{\mathbf{w}_i^T\mathbf{w}_i + \lambda_i}}{1 - \sum_{i-1}^{n} \dfrac{w_{k,i}^2}{\mathbf{w}_i^T\mathbf{w}_i + \lambda_i}} = \frac{\psi_k^{(n)}}{\eta_k^{(n)}} \tag{45}$$

The recursive formula for the LOO error weighting $\eta_k^{(n)}$ is given in (35), while $\psi_k^{(n)}$ can be represented using the following recursive formula [52]

$$\psi_k^{(n)} = \psi_k^{(n-1)} + y_k g_n w_{k,n} - \frac{w_{k,n}^2}{\mathbf{w}_n^T\mathbf{w}_n + \lambda_n} \tag{46}$$

The OLS-LOO-LR algorithm described in Subsection 3.1 can readily be applied to select a sparse kernel classifier with some minor modifications. These modifications are due to the fact that the selection criterion is the LOO misclassification rate (42) rather than the LOO MSE (33). Extensive empirical experience has also suggested that, for constructing sparse kernel classifier, multiple regularisers or local regularisation, which is so effective in further enforcing model sparsity in regression, becomes unnecessary. Thus, all the regularisation parameters λ_i, $1 \le i \le N$, can be set to a small positive constant λ, and there is no need to update them using the evidence procedure. The sparse kernel classifier selection procedure based on this OLS-LOO algorithm is summarised as follows.

Setting λ to a small positive number, and with the following initial conditions

$$\psi_k^{(0)} = 0 \text{ and } \eta_k^{(0)} = 1 \text{ for } 1 \le k \le N, \text{ and } J_0 = 1 \tag{47}$$

use the procedure described in Appendix E to select a subset model with N_s terms.

The selection procedure of Appendix E is essentially the same one as described in Appendix D, with only minor modifications connected with the computation of the LOO misclassification rate J_n. Note that the LOO misclassification rate J_n is also locally convex with respect to the classifier's size n. Thus there exists an optimal model size N_s such that for $n \le N_s$ J_n decreases as n increases, while

$$J_{N_s} \le J_{N_s+1} \tag{48}$$

Therefore the selection procedure is automatically

terminated with a subset classifier containing only N_s significant kernels.

3.3. Sparse Kernel Density Estimator Construction

As shown in Subsection 2.3, the generic kernel density estimation problem can be expressed as a constrained regression modelling, and the regression modelling part itself is identical to that of regression described in Subsection 2.1. Therefore, the OLS-LOO-LR algorithm detailed in Subsection 3.1 can be used to select a sparse kernel density estimate. The only problem is that the kernel weights obtained by the OLS-LOO-LR algorithm for this sparse kernel density estimate do not necessarily meet the nonnegative constraint (16) and the unity constraint (17). This "deficiency" however can easily be corrected by using the MNQP algorithm to modify or update the kernel weights of the selected sparse model [44]. This combined OLS-LOO-LR and MNQP algorithm offers an effective means of obtaining sparse kernel density estimates with excellent generalisation capability. The detailed OLS-LOO-LR algorithm has been described in Subsection 3.1 and, therefore, only the MNQP part needs to be discussed.

After the structure determination using the OLS-LOO-LR algorithm of Subsection 3.1, a sparse N_s-term subset kernel model is obtained, where $N_s \ll N$. Let $\mathbf{A}_{N_s}$ denote the subset matrix of $\mathbf{A}_N$, corresponding to the selected N_s-term subset model. The kernel weight vector $\boldsymbol{\beta}_{N_s}$, computed from $\mathbf{A}_{N_s}\boldsymbol{\beta}_{N_s} = \mathbf{g}_{N_s}$, may not satisfy the nonnegative constraint (16) and the unity constraint (17). Thus $\boldsymbol{\beta}_{N_s}$ must be recalculated using for example the MNQP algorithm [61,81]. Note that, since N_s is very small, the extra computation involved is small. Formally, this task is defined as follows. Find $\boldsymbol{\beta}_{N_s}$ for the regression model

$$\mathbf{y} = \boldsymbol{\Phi}_{N_s}\boldsymbol{\beta}_{N_s} + \boldsymbol{\epsilon} \tag{49}$$

subject to the constraints

$$\beta_i \ge 0,\ 1 \le i \le N_s \tag{50}$$

$$\boldsymbol{\beta}_{N_s}^T \mathbf{1}_{N_s} = 1 \tag{51}$$

where $\boldsymbol{\Phi}_{N_s}$ denotes the selected subset regression matrix and $\boldsymbol{\beta}_{N_s}^{\mathrm{T}} = [\beta_1 \beta_2 ... \beta_{N_s}]$. The kernel weight vector can be obtained by solving the following constrained nonnegative quadratic programming

$$\min_{\boldsymbol{\beta}_{N_s}} \left\{ \frac{1}{2}\boldsymbol{\beta}_{N_s}^T \mathbf{C}_{N_s}\boldsymbol{\beta}_{N_s} - \mathbf{v}_{N_s}^T\boldsymbol{\beta}_{N_s} \right\}$$
$$\text{s.t. } \boldsymbol{\beta}_{N_s}^T \mathbf{1}_{N_s} = 1 \text{ and } \beta_i \ge 0,\ 1 \le i \le N_s \tag{52}$$

where $\mathbf{C}_{N_s} = \boldsymbol{\Phi}_{N_s}^T\boldsymbol{\Phi}_{N_s} = [c_{i,j}] \in \mathcal{R}^{N_s \times N_s}$ is the related design matrix and $\mathbf{v}_{N_s} = \boldsymbol{\Phi}_{N_s}^{\mathrm{T}}\mathbf{y} = [v_1\ v_2 ... v_{N_s}]^T$. Although there exists no closed-form solution for this optimisation problem, the solution can readily be obtained iteratively using a modified version of the MNQP algorithm [61].

Since the elements of $\mathbf{C}_{N_s}$ and $\mathbf{v}_{N_s}$ are strictly positive, the Lagrangian for the above problem can be formed as [81]

$$\mathcal{L} = \frac{1}{2}\sum_{i=1}^{N_s}\sum_{j=1}^{N_s} c_{i,j}\frac{\beta_j^{<t>}(\beta_i^{<t+1>})^2}{\beta_i^{<t>}} - \sum_{i=1}^{N_s} v_i\beta_i^{<t+1>} - h^{<t>}\left(\sum_{i=1}^{N_s}\beta_i^{<t+1>} - 1\right) \tag{53}$$

where the superindex $^{<t>}$ denotes the iteration index and h is the Lagrangian multiplier. Setting

$$\frac{\partial \mathcal{L}}{\partial \beta_i^{<t+1>}} = 0 \quad \text{and} \quad \frac{\partial \mathcal{L}}{\partial h^{<t>}} = 0 \tag{54}$$

leads to the following updating equations

$$r_i^{<t>} = \beta_i^{<t>}\left(\sum_{j=1}^{N_s} c_{i,j}\beta_j^{<t>}\right)^{-1},\ 1 \le i \le N_s \tag{55}$$

$$h^{<t>} = \left(\sum_{i=1}^{N_s} r_i^{<t>}\right)^{-1}\left(1 - \sum_{i=1}^{N_s} r_i^{<t>}v_i\right) \tag{56}$$

$$\beta_i^{<t+1>} = r_i^{<t>}\left(v_i + h^{<t>}\right) \tag{57}$$

It is easy to check that, if $\boldsymbol{\beta}_{N_s}^{<t>}$ meets the constraints (50) and (51), $\boldsymbol{\beta}_{N_s}^{<t+1>}$ updated according to (55) to (57) also satisfies (50) and (51). The initial condition can be set as $\beta_i^{<0>} = \frac{1}{N_s}, 1 \le i \le N_s$. Alternative, $\boldsymbol{\beta}_{N_s}^{<0>}$ can be chosen as follows. First, if the kernel weight vector obtained by the OLS-LOO-LR algorithm contains negative elements, these elements are replaced by a small positive number. The resulting kernel weight vector is then normalised and used as $\boldsymbol{\beta}_{N_s}^{<0>}$. During the iterative procedure, some of the kernel weights may be driven to (near) zero [61,81]. The corresponding kernels can then be removed from the kernel model, leading to a further reduction in the subset model size.

4. Experimental Results

Several examples, taken from the regression, classification and density estimation applications, were used to demonstrate the effectiveness of the proposed unified regression modelling approach. For each of these data modelling examples, the full regression model set was formed by placing a Gaussian kernel (3) on each training data sample, and a sparse kernel model was then selected

using the OLS-LOO-LR algorithm. For sparse kernel density estimate, additionally, the MNQP algorithm described in Subsection 3.3 was applied to update the kernel weight vector. The appropriate value for the kernel width ρ was found empirically via cross validation. The obtained model's generalisation performance was evaluated based on a separate test data set not used for training. Comparsion with some existing sparse kernel modelling techniques was made, in terms of the model generalisation performance, model sparsity and the complexity of model construction process.

4.1. Regression Applications

4.1.1. Scalar Function Modelling

This example used a Gaussian kernel model to fit the scalar function

$$f(x)=\frac{\sin(x)}{x},\ -10\le x\le 10 \qquad \textbf{(58)}$$

based on noisy data. Four hundred data points were generated from $y=f(x)+e$, where the input x was uniformly distributed in (−10, 10) and the noise e was Gaussian distributed with zero mean and standard deviation 0.2. The first 200 data points were used for training and the other 200 samples for model validation. The kernel variance $\rho^2=10.0$ was found to be optimal empirically for this example. As a Gaussian kernel was placed on each training data x, there were $N=200$ candidate regressors in the regression model (5). The training data were very noisy. In addition to use the noisy test data set for evaluating the model's generalisation performance, two hundred noise-free data $f(x)$ with equally spaced x in (−10, 10) were also generated as the second test data set. The OLS-LOO-LR algorithm was applied to the noisy training data set, and the algorithm automatically selected a 7-term kernel model. The modelling accuracy of the resulting 7-term kernel model is summarised in Table 1, and the corresponding model mapping generated by this 7-term kernel model is depicted in Figure 1, in comparison with the true scalar function (58).

The relevance vector machine (RVM) algorithm [60] is an existing sparse kernel modelling algorithm that is often regarded as the state-of-the-art. It has the same excellent generalisation performance as the SVM algorithm but achieves a dramatically sparser kernel model than the SVM method. A drawback of the RVM method is a significant increase in computational complexity, compared with the SVM method. The iterative procedure for updating multiple regularisers in the RVM method converges much slower and may even suffer from numerical instability, compared with the efficient OLS-LOO-LR algorithm. The detailed comparison for these two sparse kernel modelling algorithms is given in [40]. The RVM algorithm was also applied to fit a sparse Gaussian kernel model for this example, and the algorithm produced the 15-term kernel model as listed in Table 1. The model mapping generated by the 15-term kernel model constructed using the RVM algorithm is shown in Figure 2. It can be seen that the OLS-LOO-LR algorithm and the RVM algorithm both had the same excellent generalisation performance, but the former produced a much sparser model than the latter. The OLS-LOO-LR algorithm additionally had significantly computational advantages in model construction.

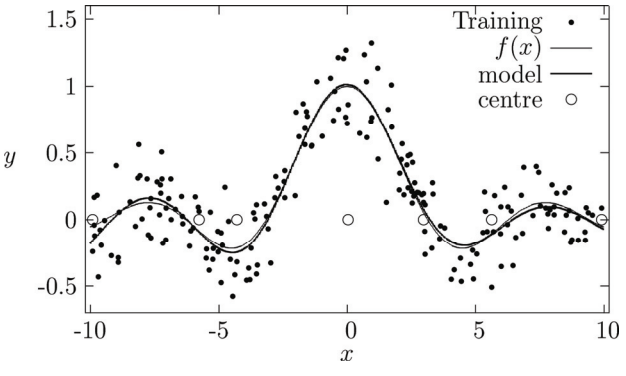

Figure 1. Scalar function modelling: noisy training data y (dots), underlying function $f(x)$ (thin curve), model mapping (thick curve), and selected kernel centres (circles). The 7-term kernel model was identified by the proposed OLS algorithm.

Table 1. Comparison of modelling accuracy for the scalar function modelling.

algorithm	model size	training MSE	noisy test MSE	noisefree test MSE
OLS	7	0.038792	0.042001	0.000736
RVM	15	0.038784	0.041827	0.000668

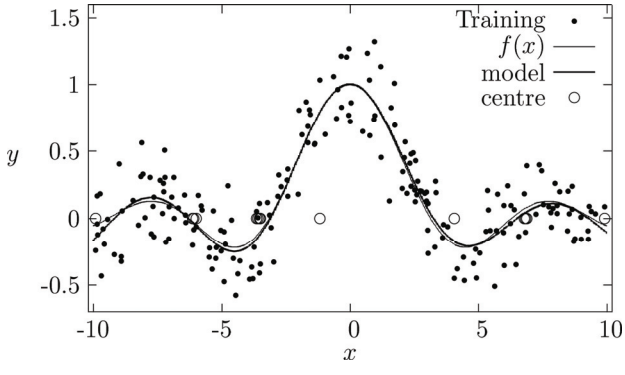

Figure 2. Scalar function modelling: noisy training data y (dots), underlying function $f(x)$ (thin curve), model mapping (thick curve), and selected kernel centres (circles). The 15-term kernel model was identified by the RVM algorithm.

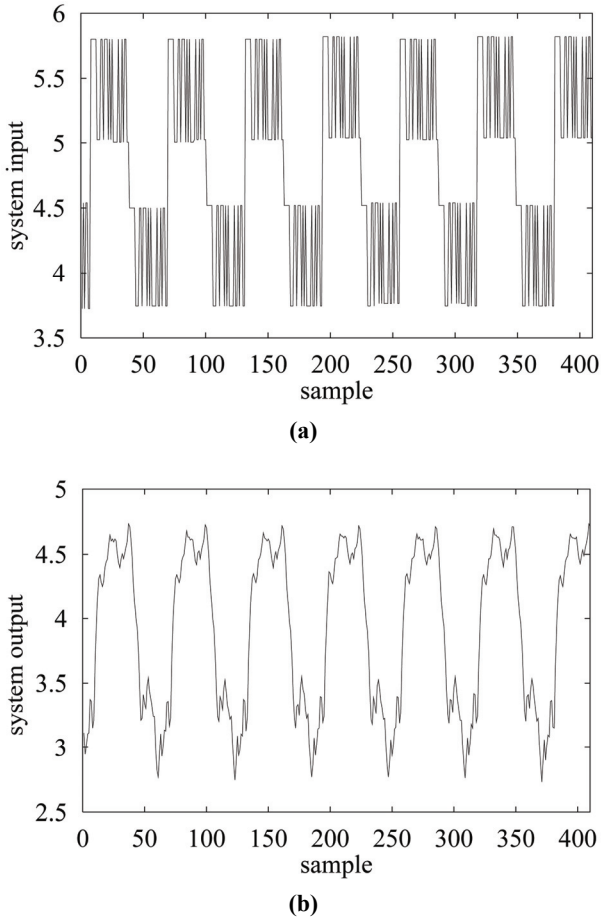

Figure 3. Engine data set (a) input u_k and (b) output y_k.

4.1.2. Engine Data Set

This example constructed a model representing the relationship between the fuel rack position (input u_k) and the engine speed (output y_k) for a Leyland TL11 turbocharged, direct injection diesel engine operated at low engine speed. It is known that at low engine speed, the relationship between the input and output is nonlinear [84]. Detailed system description and experimental setup can be found in [84]. The data set, depicted in Figure 3, contained 410 samples. The first 210 data points were used in modelling and the last 200 points in model validation. The previous results [84] have shown that this data set can be modelled adequately as

$$y_k = f(\mathbf{x}_k) + e_k \tag{59}$$

where $f(\bullet)$ describes the unknown system to be identified, e_k denotes the system noise, and

$$\mathbf{x}_k = [y_{k-1}\, u_{k-1}\, u_{k-2}]^T \tag{60}$$

Table 2. Comparison of modelling accuracy for the engine data set.

algorithm	model size	training MSE	test MSE
OLS	22	0.000453	0.000490
SVM	92	0.000447	0.000498

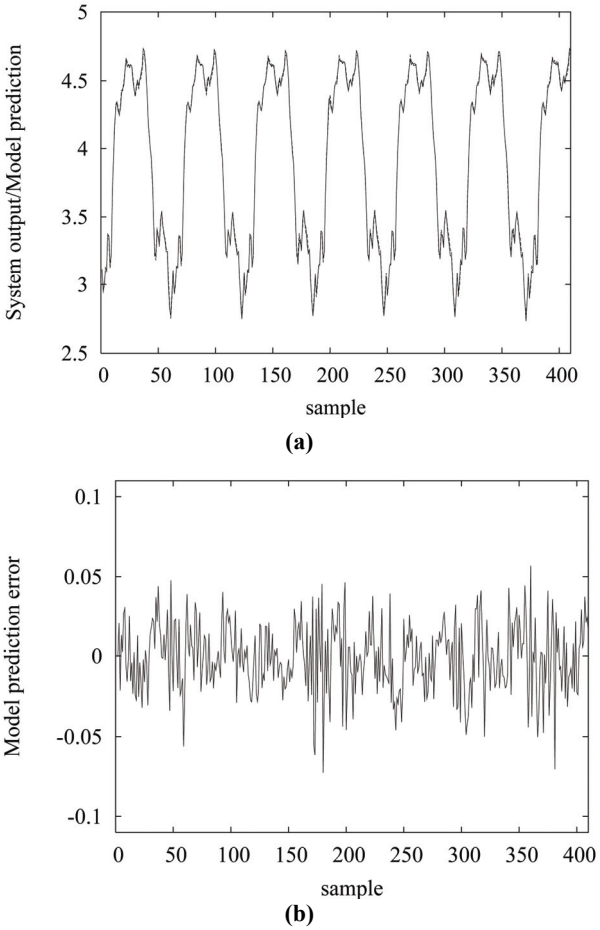

Figure 4. Modelling performance for the engine data set: (a) model prediction $\hat{y}_k$ (dashed) superimposed on system output y_k (solid), and (b) model prediction error $\epsilon_k = y_k - \hat{y}_k$. The 22-term model was constructed by the proposed OLS algorithm.

The optimal value of the kernel variance for the Gaussian kernel was found empirically to be $\rho^2 = 1.69$. As each $\mathbf{x}_k$ in the training data set was considered as a candidate kernel centre, there were N = 210 candidate kernel regressors in the full regression model (5).

Both the OLS-LOO-LR algorithm and the SVM algorithm [56] were applied to this data set, and the two sparse Gaussian kernel models obtained are compared in

Table 2. The model output $\hat{y}_k$ and modelling error $\epsilon_k = y_k - \hat{y}_k$ generated by the 22-term kernel model obtained by the OLS-LOO-LR algorithm are depicted in Figure 4. The modelling performance of the 92-term kernel model constructed by the SVM algorithm, not shown here, are very similar to those shown in Figure 4. It can be seen that the two sparse regression modelling techniques achieved the same excellent generalisation performance but the OLS-LOO-LR method obtained a much sparser model than the SVM method. It should be emphasised that the model size is critically important for this particular example. The main purpose of identifying a model for this engine system is to use it for designing a controller. A large model will make the controller design a very complex task and, moreover, the resulting controller will be difficult to implement in the real system. It is also worth emphasising that the OLS-LOO-LR algorithm has considerably computational advantages over the SVM algorithm. Both the algorithms require to determine the kernel width ρ. However, the SVM method has two more learning parameters, namely the error-band and trade-off parameters [56], that require tuning. Therefore, the OLS-LOO-LR algorithm is easier to tune and computationally more efficient than the SVM algorithm.

4.1.3. Boston Housing Data Set

This was a regression benchmark data set, available at the UCI repository [85]. The data set comprised 506 data points with 14 variables. The task was to predict the median house value from the remaining 13 attributes. From the data set, 456 data points were randomly selected for training and the remaining 50 data points were used to form the test set. Because a Gaussian kernel was placed at each training data sample, there were $N = 456$ candidate regressors in the full regression model (5). The kernel width for the OLS-LOO-LR algorithm was determined via a grid-search based cross validation. Similarly, the three learning parameters of the SVM, the kernel width, error-band and trade-off parameters, were tuned via cross validation. Average results were given over 100 repetitions, and the two sparse Gaussian kernel models obtained by the OLS-LOO-LR and SVM algorithms, respectively, are compared in Table 3.

For the particular computational platform used in the experiment, the recorded average run time for the OLS-LOO-LR algorithm when the kernel width was fixed was 200 times faster than the SVM algorithm[1] when the kernel width, error-band and trade-off parameters were chosen. It can be seen from Table 3 that the OLS-LOO-LR algorithm achieved better modelling accuracy with a much sparser model than the SVM algorithm.

[1]One could argue that by adopting fast implementation of the SVM algorithm significant reduction in run time can be achieved.

Table 3. Comparison of modelling accuracy for the Boston housing data set. The results were averaged over 100 realizations and quoted as the mean±standard deviation.

algorithm	model size	training MSE	test MSE
OLS	58.6 ± 11.3	12.9690 ± 2.6628	17.4157 ± 4.6670
SVM	243.2 ± 5.3	6.7986 ± 0.4444	23.1750 ± 9.0459

Table 4. Comparison of classification accuracy for the first 10 realizations of the breast cancer data set. The results for the SVM and RVM algorithms were quoted from [60].

algorithm	average model size	average test error rate
SVM	116.7	26.9%
RVM	6.3	29.9%
OLS	5.8	27.4%

The test MSE of the SVM algorithm was poor. This was probably because the three learning parameters, namely the kernel width, error-band and trade-off parameters, were not tuned to the optimal values. For this regression problem of input dimension 13 and data size $N \approx 500$, the grid search required by the SVM algorithm to tune the three learning parameters was expensive and the optimal values of the three learning parameters were hard to find, compared with for example the previous smaller engine data set.

4.2. Classification Applications

4.2.1. Breast Cancer Data

This classification benchmark data set was originated in the UCI repository [85] and the actual data set used in the experiment was obtained from [86]. The feature input space dimension was m =9. There were 100 realizations of this data set, each containing 200 training patterns and 77 test patterns. In [60], the SVM and RVM algorithms were applied to the first 10 realizations of this data set, and the results given in [60] were reproduced in Table 4. The OLS-LOO algorithm described in Subsection 3.2 was also applied to construct Gaussian kernel classifiers for the same first 10 realisations of this data set, and the results obtained are summarised in Table 4, in comparison with those obtained by the SVM and RVM algorithms. In [86,87], seven existing state-of-the-art RBF and kernel classifier construction algorithms were compared and the performance averaged over all the 100 realizations were given. The OLS-LOO algorithm was applied to all the 100 realizations of the data set to construct sparse Gaussian kernel classifiers and the results obtained are given in Table 5, in comparison with the benchmark results quoted from [86,87]. For the first 5

methods studied in [86], the RBF network with 5 optimised nonlinear Gaussian units was used. The kernel Fisher discriminant was the optimal nonsparse 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 [86] but it was certainly much larger than 50. From Table 5, it can be seen that the proposed OLS-LOO algorithm compared favourably with these benchmark RBF and kernel classifier construction algorithms, both in terms of classification accuracy and model size.

4.2.2. Diabetes Data

This was another classification benchmark data set in the UCI repository [85] and the data set used in the experiment was obtained from [86]. The feature space dimension was $m = 8$. There were 100 realisations of the data set, each having 468 training patterns and 300 test patterns. Seven benchmark RBF and kernel classifiers were studied in [86,87], and the results given in [86] were reproduced in Table 6. For the first 5 methods studied in [86], the nonlinear RBF network with 15 optimised Gaussian units was used. For the SVM algorithm with Gaussian kernel, no average model size was given in [86] but it could safely be assumed that it was much larger than 100. The OLS-LOO algorithm was applied to construct sparse Gaussian kernel classifiers for this data set, and the results averaged over the 100 realisations are also listed in Table 6. It can be seen that the proposed OLS-LOO method produced the best classification accuracy with the smallest classifier.

Table 5. Average classification test error rate in % over the 100 realizations of the breast cancer data set. The first 7 results were quoted from [86].

algorithm	test error rate	model size
RBF-Network	27.64 ± 4.71	5
AdaBoost RBF-Network	30.36 ± 4.73	5
LP-Reg-AdaBoost	26.79 ± 6.08	5
QP-Reg-AdaBoost	25.91 ± 4.61	5
AdaBoost-Reg	26.51 ± 4.47	5
SVM	26.04 ± 4.74	not available
Kernel Fisher Discriminant	24.77 ± 4.63	200
OLS	25.74 ± 5.00	6.0 ± 2.0

Table 6. Average classification test error rate in % over the 100 realizations of the diabetes data set. The first 7 results were quoted from [86].

algorithm	test error rate	model size
RBF-Network	24.29± 1.88	15
AdaBoost RBF-Network	26.47 ± 2.29	15
LP-Reg-AdaBoost	24.11 ± 1.90	15
QP-Reg-AdaBoost	25.39 ± 2.20	15
AdaBoost-Reg	23.79 ± 1.80	15
SVM	23.53 ± 1.73	not available
Kernel Fisher Discriminant	23.21 ± 1.63	468
OLS	23.00 ± 1.70	6.0 ± 1.0

Table 7. Average classification test error rate in % over the 100 realizations of the thyroid data set. The first 7 results were quoted from [86].

algorithm	test error rate	model size
RBF-Network	4.52 ± 2.12	8
AdaBoost RBF-Network	4.40 ± 2.18	8
LP-Reg-AdaBoost	4.59 ± 2.22	8
QP-Reg-AdaBoost	4.35 ± 2.18	8
AdaBoost-Reg	4.55 ± 2.19	8
SVM	4.80 ± 2.19	not available
Kernel Fisher Discriminant	4.20 ± 2.07	140
OLS	4.80 ± 2.20	4.6 ± 1.0

4.2.3. Thyroid Data

This classification benchmark data set was originated in the UCI repository [85] and the data set used in the experiment was obtained from [86]. The input space dimension was $m = 5$. There were 100 realizations of this data set, each containing 140 training patterns and 75 test patterns. Eight RBF and kernel classifiers are compared in Table 7, with the first seven methods quoted from [86,87]. It can be seen that the classification accuracy of the proposed OLS-LOO method is comparable to that of the SVM method, but the former achieved a much smaller model size than the latter.

4.3. Density Estimation Applications

Simulation was used to test the proposed combined OLS-LOO-LR and MNQP algorithm and to compare its performance with the Parzen window estimator as well as the previous sparse kernel density estimation algorithm [43]. The algorithm presented in [43], although also based on the OLS-LOO-LR regression framework, is very different from the current combined OLS-LOO-LR and MNQP algorithm. In particular, it transfers the kernels into the corresponding cumulative distribution functions and uses the empirical distribution function calculated on the training data set as the target function of the unknown cumulative distribution function. In other words, the regression framework is defined in the cumulative distribution function "space", not the original PDF "space". Converting the kernels into corresponding cumulative distribution functions can be inconvenient and may be difficult for certain types of kernels. Moreover, in the work [43], the unity constraint is met by normalising the kernel weight vector of the final selected model, which is nonoptimal, and the nonnegative constraint is ensured by adding a test to the OLS forward selection procedure. In each selection stage, a candidate that causes the resulting kernel weight vector to have negative elements, if included, will not be considered at all. This nonnegative test imposes considerable computational cost to the OLS selection procedure. The proposed combined OLS-LOO-LR and MNQP algorithm in com-

parison is computationally simpler.

The first and third examples in the simulation were one-dimensional and six-dimensional density estimation problems, respectively, where a data set of N randomly drawn samples was used to construct kernel density estimates based on the regression model (21), and a separate test data set of N_{test} = 10,000 samples was used to calculate the L_1 test error for the resulting estimate according to

$$L_1 = \frac{1}{N_{test}} \sum_{k=1}^{N_{test}} | p(\mathbf{x}_k) - \hat{p}(\mathbf{x}_k; \boldsymbol{\beta}_N, \rho) | \qquad (61)$$

The experiment was repeated N_{run} different random runs. The second example was a two-class two-dimensional classification problem taken from [6]. For all the three example, the value of the kernel width ρ was determined via cross validation.

4.3.1. One-Dimensional Density Estimation

The one-dimensional density to be estimated was the mixture of Gaussian and Laplacian given by

$$p(x) = \frac{1}{2\sqrt{2\pi}} e^{-\frac{(x-2)^2}{2}} + \frac{0.7}{4} e^{-0.7|x+2|} \qquad (62)$$

The number of data points for density estimation was N = 100. The optimal kernel widths were found to be ρ = 0.54 and ρ = 1.1 empirically for the Parzen window estimate and the proposed sparse kernel density estimate, respectively. The experiment was repeated N_{run} = 200 times. Table 8 compares the performance of these two kernel density estimates, in terms of the L_1 test error and the number of kernels required. Figure 5(a) plots a Parzen window estimate obtained while Figure 5(b) illustrates a sparse kernel density estimate obtained by the combined OLS-LOO-LR and MNQP algorithm, in comparison with the true distribution. It can be seen that the accuracy of the proposed sparse kernel density estimate was comparable to that of the Parzen window estimate, and the combined OLS-LOO-LR and MNQP algorithm achieved sparse estimate with an average kernel number less than 6% of the data samples. The maximum and minimum number of kernels over 200 runs were 9 and 2, respectively, for the sparse kernel density estimator.

The previous sparse kernel density estimator using the empirical distribution function as the desired response and based on the OLS-LOO-LR algorithm only [43] was also applied to this example. Under the identical experimental conditions, the results obtained by this sparse kernel density estimator are also given in Table 8, where it can be seen that the both sparse kernel density estimators had a very similar performance, in terms of the L_1 test error and average number of kernels required.

Table 8. Performance comparison for the one-dimensional Gaussian and Laplacian mixture.

method	L_1 test error	kernel number
Parzen window estimate	$(1.9503 \pm 0.5881) \times 10^{-2}$	100 ± 0
proposed SKD estimate	$(1.9436 \pm 0.6208) \times 10^{-2}$	5.1 ± 1.3
SKD estimate of [43]	$(2.1785 \pm 0.7468) \times 10^{-2}$	4.8 ± 0.9

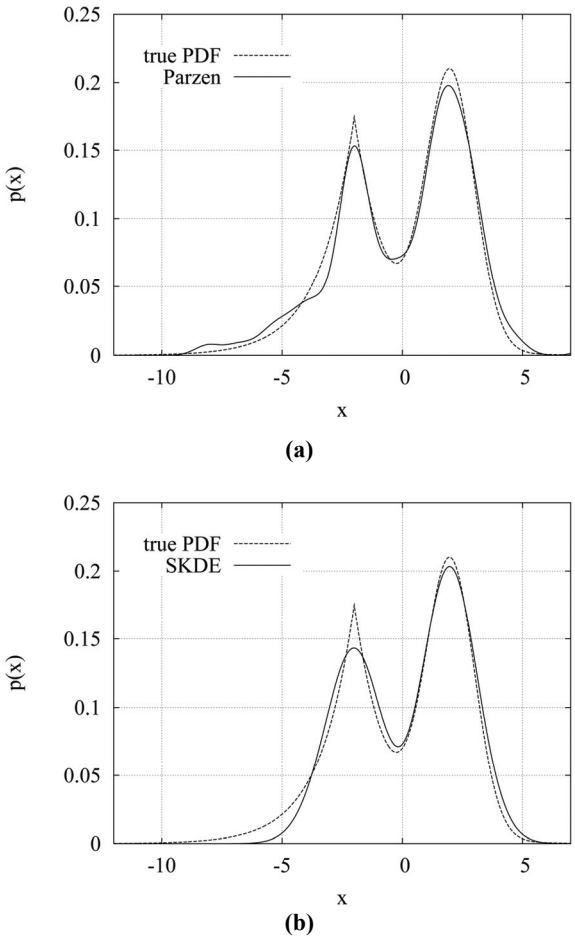

Figure 5. (a) true density (dashed) and a Parzen window estimate (solid), and (b) true density (dashed) and a sparse kernel density estimate (solid) obtained by the combined OLS-LOO-LR and MNQP algorithm, for the one-dimensional Gaussian and Laplacian mixture.

4.3.2. Synthetic Classification Data

This was a two-class classification problem in a two-dimensional feature space [6]. The training data set contained 250 samples with 125 points for each class, and the test data set had 1000 points with 500 samples for each class. The optimal Bayes test error rate based on the true underlying probability distribution for this example was known to be 8%. The task was first to esti-

mate the two conditional density functions $\hat{p}(\mathbf{x};\boldsymbol{\beta}_{N_{C0}}, \rho \,|\, \boldsymbol{C}0)$ and $\hat{p}(\mathbf{x};\boldsymbol{\beta}_{N_{C1}}, \rho \,|\, \boldsymbol{C}1)$ from the training data, and then applied the Bayes decision rule

$$\left.\begin{array}{ll} \text{if } \hat{p}(\mathbf{x};\boldsymbol{\beta}_{N_{C0}}, \rho \,|\, \mathrm{C}0) \geq \hat{p}(\mathbf{x};\boldsymbol{\beta}_{N_{C1}}, \rho \,|\, \mathrm{C}1), & \mathbf{x} \text{ belongs to class } 0 \\ \text{else,} & \mathbf{x} \text{ belongs to class } 1 \end{array}\right\} \tag{63}$$

to the test data set and calculated the corresponding error rate, where N_{C0} and N_{C1} are the number of the class $C0$ and class $C1$ training data points, respectively.

Table 9 lists the results obtained by the three kernel density estimates, the Parzen window estimator, the current sparse kernel density estimator based on the combined OLS-LOO-LR and MNQP algorithm, and the previous sparse kernel density estimator with the empirical distribution function as the desired response and based on the OLS-LOO-LR algorithm only [43], where the value of the kernel width was determined by minimizing the test error rate. It can be seen that the proposed sparse kernel density estimation method yielded the very sparse conditional density estimates and achieved the optimal Bayes classification performance. This clearly demonstrated the accuracy of the density estimates. Figure 6(a) and (b) depict the decision boundaries of the classifier (63) for the Parzen window estimate and the sparse kernel density estimate obtained by the combined OLS-LOO-LR and MNQP algorithm, respectively.

4.3.3. Six-Dimensional Density Estimation

The underlying density to be estimated was given by

$$p(\mathbf{x}) = \frac{1}{3}\sum_{i=1}^{3} \frac{1}{(2\pi)^{6/2}} \frac{1}{\det^{1/2} |\,\boldsymbol{\Gamma}_i\,|} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^T \boldsymbol{\Gamma}_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)} \tag{64}$$

with

$$\begin{aligned} \boldsymbol{\mu}_1 &= [1.0\ 1.0\ 1.0\ 1.0\ 1.0\ 1.0]^T \\ \boldsymbol{\Gamma}_1 &= \mathrm{diag}\{1.0,\ 2.0,\ 1.0,\ 2.0,\ 1.0,\ 2.0,\} \end{aligned} \tag{65}$$

$$\begin{aligned} \boldsymbol{\mu}_2 &= [-1.0\ -1.0\ -1.0\ -1.0\ -1.0\ -1.0]^T \\ \boldsymbol{\Gamma}_2 &= \mathrm{diag}\{2.0, 1.0,\ 2.0, 1.0,\ 2.0, 1.0\} \end{aligned} \tag{66}$$

$$\begin{aligned} \boldsymbol{\mu}_3 &= [0.0\ 0.0\ 0.0\ 0.0\ 0.0\ 0.0]^T \\ \boldsymbol{\Gamma}_3 &= \mathrm{diag}\{2.0, 1.0,\ 2.0, 1.0,\ 2.0, 1.0\} \end{aligned} \tag{67}$$

The estimate data set contained $N = 600$ samples. The optimal kernel width was found to be $\rho = 0.65$ for the Parzen window estimate and $\rho = 1.2$ for the sparse kernel density estimate based on the combined OLS- LOO-LR and MNQP algorithm, respectively, via cross validation. The experiment was repeated $N_{run} = 100$ times. The results obtained by the two density estimator are summarized in Table 10. For this example, again, the two density estimates were seen to have comparable accuracies,

Table 9. Performance comparison for the two-dimensional classification data set.

method	$\hat{p}(\bullet \,|\, \mathrm{C}0)$	ρ	$\hat{p}(\bullet \,|\, \mathrm{C}1)$	ρ	test error rate
Parzen window estimate	125 kenels	0.24	125 kernels	0.23	8.0%
proposed SKD estimate	6 kernels	0.28	5 kernels	0.28	8.0%
SKD estimate of [43]	5 kernels	0.20	4 kernels	0.20	8.3%

Table 10. Performance comparison for the six-dimensional three-Gaussian mixture.

method	L_1 test error	kernel number
Parzen window estimate	$(3.5195 \pm 0.1616) \times 10^{-5}$	600 ± 0
proposed SKD estimate	$(3.1134 \pm 0.5335) \times 10^{-5}$	9.4 ± 1.9
SKD estimate of [43]	$(4.4781 \pm 1.2292) \times 10^{-5}$	14.9 ± 2.1

but the proposed sparse kernel density estimate method achieved very sparse estimates with an average number of required kernels less than 2% of the data samples. The maximum and minimum numbers of kernels over 100 runs were 16 and 7, respectively, for the proposed sparse kernel density estimator.

This example was used to test the sparse kernel density estimation method of [43] under the same experimental conditions. The results obtained by this previous sparse kernel density estimator, quoted from [43], are also given in Table 10 for comparison. It is seen from Table 10 that for this high dimensional example the proposed sparse kernel density estimator outperformed the previous sparse kernel density estimator in terms of both the test performance and the level of model sparsity.

5. Conclusions

A regression framework has been proposed for sparse modelling from data, which unifies the supervised regression and classification problems as well as the unsupervised probability density function learning problem in the same kernel regression model. A powerful orthogonal-least-squares algorithm has been developed for selecting very sparse kernel models that generalise well, based on the leave-one-out test criteria and coupled with local regularisation. For sparse kernel density estimation, in particular, a combined approach of the OLS-LOO-LR algorithm and multiplicative nonnegative quadratic programming has been proposed, with the OLS-LOO-LR algorithm selecting a sparse kernel density estimate while the MNQP algorithm computing the kernel weights of the selected final model to meet the constraints for den-

sity estimate. Empirical datamodelling results involving all the three classes of data modelling, namely regression, classification and density estimation, have been presented to demonstrate the effectiveness of the proposed unified kernel regression modelling framework based on the OLS-LOO-LR algorithm, and the results shown have clearly confirmed that this proposed unified sparse kernel regression framework offers a truly state-of-art for data modelling applications.

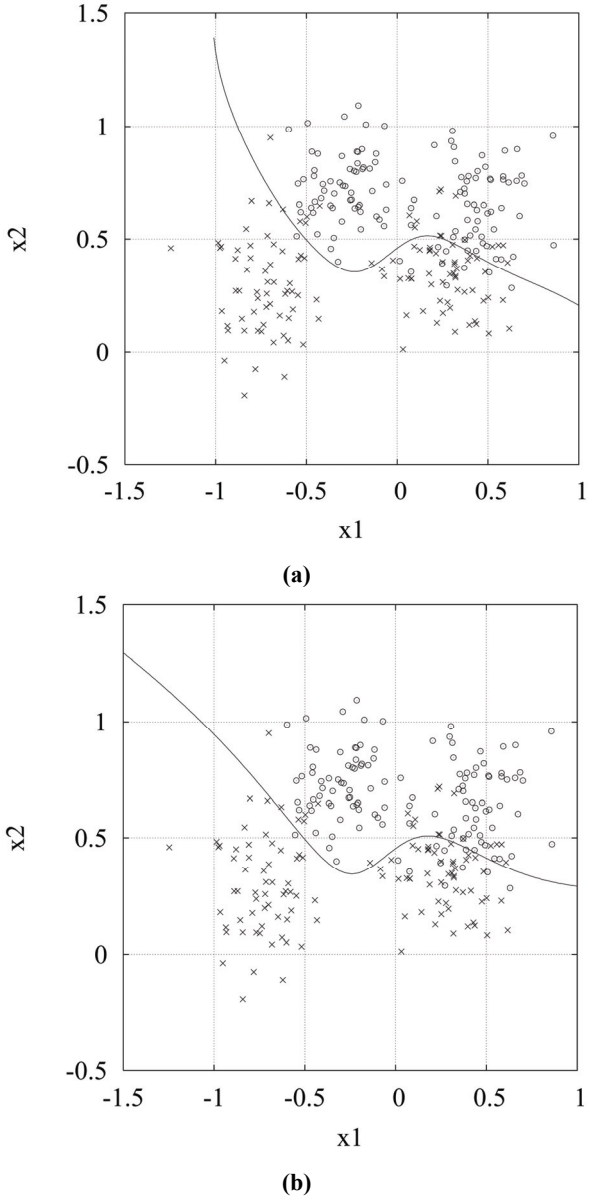

Figure 6. (a) decision boundary of the Parzen window estimate, and (b) decision boundary of the sparse kernel density estimate obtained by the combined OLS-LOO-LR and MNQP algorithm, for the two-class two-dimensional classification example, where circles represent the class-1 training data and crosses the class-0 training data.

The unified regression framework developed in this contribution is based on the linear-in-the-parameters kernel model, where the full candidate kernel set is obtained by placing a kernel at each training data point and employing a fixed kernel width for all the kernel regressors. Further reseach has been conducted to develop a nonlinear-in-the-parameters regression model, where each regressor has tunable base centre vector and diagonal covariance matrix. A powerful orthogonal-least-squares assisted forward selection procedure can be developed based on the leave-one-out test criteria and local regulation. At each stage of the construction procedure, a nonlinear base is constructed by optimising the appropriate LOO test criterion to determine the base's centre vector and diagonal covariance matrix. Such sparse data modelling techniques based on tunable nonlinear base units have been proposed for regression data modelling [88] and classification application [89]. Sparse density estimation based on this novel tunable regression modelling framework is currently under investigation [90].

6. Acknowledgements

The author acknowledges the contributions of Dr. Xia Hong and Professor Chris J. Harris to the topic reported in this work.

Appendix A

The modified Gram-Schmidt orthogonalisation procedure [34] calculate the $\mathbf{A}_N$ matrix row by row and orthogonalises $\mathbf{\Phi}_N$ as follows: At the l-th stage make the columns $\boldsymbol{\phi}_j$, $l+1 \le j \le N$, orthogonal to the l-th column and repeat the operation for $l \le j \le N-1$. Specifically, denoting $\boldsymbol{\phi}_j^{[0]} = \boldsymbol{\phi}_j$, $l \le j \le N$, then for l = 1, 2, …, $N-1$,

$$\left.\begin{aligned} &\mathbf{w}_l = \boldsymbol{\phi}_l^{[l-1]}, \\ &a_{l,j} = \mathbf{w}_l^T \boldsymbol{\phi}_j^{[l-1]} / (\mathbf{w}_l^T \mathbf{w}_l), l+1 \le j \le N, \\ &\boldsymbol{\phi}_j^{[l]} = \boldsymbol{\phi}_j^{[l-1]} - a_{l,j} \mathbf{w}_l, l+1 \le j \le N. \end{aligned}\right\} \quad \textbf{(68)}$$

The last stage of the procedure is simply $\mathbf{w}_N = \boldsymbol{\phi}_N^{[N-1]}$. The elements of $\mathbf{g}_N$ are computed by transforming $\mathbf{y}^{[0]} = \mathbf{y}$ in a similar way

$$\left.\begin{aligned} &g_l = \mathbf{w}_l^T \mathbf{y}^{[l-1]} / (\mathbf{w}_l^T \mathbf{w}_l + \lambda_l), \\ &\mathbf{y}^{[l]} = \mathbf{y}^{[l-1]} - g_l \mathbf{w}_l, \end{aligned}\right\} 1 \le l \le N \quad \textbf{(69)}$$

where $\lambda_l, 1 \le l \le N$, are the regularisation parameters.

Appendix B

It can be shown that the log evidence for $\mathbf{h}_N$ and ω is [24]

$$\begin{aligned}\log(p(\mathbf{y}\,|\,\mathbf{h}_N,\omega)) \approx & \sum_{i=1}^{N}\frac{1}{2}\log(h_i)-\frac{N}{2}\log(\pi)\\ &+\frac{N}{2}\log(\omega)-\sum_{i=1}^{N}\frac{1}{2}h_i g_i^2\\ &-\frac{1}{2}\omega\boldsymbol{\epsilon}^T\boldsymbol{\epsilon}-\frac{1}{2}\log(\det(\mathbf{B}_N))\end{aligned} \tag{70}$$

where $\mathbf{g}_N$ is set to the maximum *a posterior* probability solution, and the Hessian matrix $\mathbf{B}_N$ is diagonal and is given by

$$\begin{aligned}\mathbf{B}_N &= \mathbf{H}_N+\omega\mathbf{W}_N^T\mathbf{W}_N\\ &= \mathrm{diag}\{h_1+\omega\mathbf{w}_1^T\mathbf{w}_1, h_2+\omega\mathbf{w}_2^T\mathbf{w}_2, ..., h_N+\omega\mathbf{w}_N^T\mathbf{w}_N\}\end{aligned} \tag{71}$$

Setting $\partial\log(p(\mathbf{y}\,|\,\mathbf{h}_N,\omega))/\partial\omega=0$ yields the recalculation formula for ω

$$\omega\boldsymbol{\epsilon}^T\boldsymbol{\epsilon} = N-\sum_{i=1}^{N}\frac{\omega\mathbf{w}_i^T\mathbf{w}_i}{h_i+\omega\mathbf{w}_i^T\mathbf{w}_i} \tag{72}$$

Setting $\partial\log(p(\mathbf{y}\,|\,\mathbf{h}_N,\omega))/\partial h_i=0$ yields the recalculation formula for h_i

$$h_i=\frac{\omega\mathbf{w}_i^T\mathbf{w}_i}{g_i^2(h_i+\omega\mathbf{w}_i^T\mathbf{w}_i)} \tag{73}$$

Note $\lambda_i=h_i/\omega$ and define

$$\gamma=\sum_{i=1}^{N}\gamma_i \tag{74}$$

with

$$\gamma_i=\frac{\omega\mathbf{w}_i^T\mathbf{w}_i}{h_i+\omega\mathbf{w}_i^T\mathbf{w}_i}=\frac{\mathbf{w}_i^T\mathbf{w}_i}{\lambda_i+\omega\mathbf{w}_i^T\mathbf{w}_i} \tag{75}$$

Then the recalculation formula λ_i for is

$$\lambda_i=\frac{\gamma_i}{N-\gamma}\frac{\epsilon^T\epsilon}{g_i^2},\ 1\le i\le N \tag{76}$$

Appendix C

Consider the full N-term model first. The regularised least squares solution for the parameter vector is

$$\mathbf{g}_N=(\mathbf{W}_N^T\mathbf{W}_N+\boldsymbol{\Lambda}_N)^{-1}\mathbf{W}_N^T\mathrm{y}=\tilde{\mathrm{B}}_N^{-1}\mathbf{W}_N^T\mathbf{y} \tag{77}$$

with $\tilde{\mathbf{B}}_N=\mathbf{W}_N^T\mathbf{W}_N+\boldsymbol{\Lambda}_N$. The modelling error at the k-th training sample is given by

$$\epsilon_k = y_k-\mathbf{g}_N^T\mathbf{w}_N(k)=y_k-\mathbf{y}^T\mathbf{W}_N\tilde{\mathrm{B}}_N^{-1}\mathbf{w}_N(k) \tag{78}$$

Let the k-th data sample be deleted from the training data set D_N, and the resulting leave-one-out training set is used to estimate the model parameter vector. The corresponding regularised least squares solution is defined by

$$\begin{aligned}\mathbf{g}_N^{(-k)} &= ((\mathbf{W}_N^{(-k)})^T\mathbf{W}_N^{(-k)}+\boldsymbol{\Lambda}_N)^{-1}(\mathbf{W}_N^{(-k)})^T\mathbf{y}^{(-k)}\\ &= \left(\tilde{\mathbf{B}}_N^{(-k)}\right)^{-1}\left(\mathbf{W}_N^{(-k)}\right)^T\mathbf{y}^{(-k)}\end{aligned} \tag{79}$$

where $\mathbf{W}_N^{(-k)}$ and $\mathrm{y}^{(-k)}$ denote the resulting LOO regression matrix and LOO desired output vector, respectively. The model output for this LOO model evaluated at the k-th data sample not used in training is given by

$$\hat{y}_k^{(-k)}=\left(g_N^{(-k)}\right)^T\mathbf{w}_N(k) \tag{80}$$

By definition, it can be shown that

$$\tilde{\mathbf{B}}_N^{(-k)}=\tilde{\mathbf{B}}_N-\mathbf{w}_N(k)\mathbf{w}_N^T(k) \tag{81}$$

$$\left(\mathbf{y}^{(-k)}\right)^T\mathbf{W}_N^{(-k)}=\mathbf{y}^T\mathbf{W}_N-y_k\mathbf{w}_N^T(k) \tag{82}$$

The LOO test error evaluated at the k-th data sample not used for training, denoted as , is given by

$$\begin{aligned}\epsilon_k^{(-k)} &= y_k-\left(\mathbf{g}_N^{(-k)}\right)^T\mathbf{w}_N(k)\\ &= y_k-\left(\mathbf{y}^{(-k)}\right)^T\mathbf{W}_N^{(-k)}\left(\tilde{\mathbf{B}}_N^{(-k)}\right)^{-1}\mathbf{w}_N(k)\end{aligned} \tag{83}$$

Applying the matrix inversion lemma to (81) yields

$$\left(\tilde{\mathbf{B}}_N^{(-k)}\right)^{-1}=\tilde{\mathbf{B}}_N^{-1}+\frac{\tilde{\mathbf{B}}_N^{-1}\mathbf{w}_N(k)\mathbf{w}_N^T(k)\tilde{\mathbf{B}}_N^{-1}}{1-\mathbf{w}_N^T(k)\tilde{\mathbf{B}}_N^{-1}\mathbf{w}_N(k)} \tag{84}$$

and

$$\left(\tilde{\mathbf{B}}_N^{(-k)}\right)^{-1}\mathbf{w}_N(k)=\frac{\tilde{\mathbf{B}}_N^{-1}\mathbf{w}_N(k)}{1-\mathbf{w}_N^T(k)\tilde{\mathbf{B}}_N^{-1}\mathbf{w}_N(k)} \tag{85}$$

Substituting (82) and (85) into (83) results in

$$\begin{aligned}\epsilon_k^{(-k)} &= y_k-\frac{\left(\mathbf{y}^T\mathbf{W}_N-y_k\mathbf{w}_N^T(k)\right)\tilde{\mathbf{B}}_N^{-1}\mathbf{w}_N(k)}{1-\mathbf{w}_N^T(k)\tilde{\mathbf{B}}_N^{-1}\mathbf{w}_N(k)}\\ &= \frac{y_k-\mathbf{y}^T\mathbf{W}_N\tilde{\mathbf{B}}_N^{-1}\mathbf{w}_N(k)}{1-\mathbf{w}_N^T(k)\tilde{\mathbf{B}}_N^{-1}\mathbf{w}_N(k)}\\ &= \frac{\epsilon_k}{1-\mathbf{w}_N^T(k)\tilde{\mathbf{B}}_N^{-1}\mathbf{w}_N(k)}=\frac{\epsilon_k}{\eta_k}\end{aligned} \tag{86}$$

where the N-term modelling error

$$\epsilon_k = y_k-\sum_{i=1}^{N}w_{k,i}g_i \tag{87}$$

and the associated LOO error weighting

$$\eta_k = 1 - \mathbf{w}_N^T(k)(\mathbf{W}_N^T\mathbf{W}_N + \mathbf{\Lambda}_N)^{-1}\mathbf{w}_N(k)$$
$$= 1 - \sum_{i=1}^{N} \frac{w_{k,i}^2}{\mathbf{w}_i^T\mathbf{w}_i + \lambda_i} \quad (88)$$

Now consider the subset model consisting of n model columns. Denote the corresponding n-column regression matrix as

$$\mathbf{W}_n = [\mathbf{w}_1 \mathbf{w}_2 ... \mathbf{w}_n] \quad (89)$$

and the k-th row of $\mathbf{W}_n$ as

$$\mathbf{w}_n^T(k) = [w_{k,1} w_{k,2} ... w_{k,n}] \quad (90)$$

Further denote the n-term modelling error at the k-th data sample as $\epsilon_k^{(n)}$ and the associated LOO error weighting as $\eta_k^{(n)}$. Substituting $\mathbf{W}_n$ for $\mathbf{W}_N$ and $\mathbf{w}_n(k)$ for $\mathbf{w}_N(k)$, respectively, in the above derivation leads naturally to

$$\epsilon_k^{(n)} = y_k - \sum_{i=1}^{n} w_{k,i} g_i = \left(y_k - \sum_{i=1}^{n-1} w_{k,i} g_i \right) - w_{k,n} g_n$$
$$= \epsilon_k^{(n-1)} - w_{k,n} g_n \quad (91)$$

and

$$\eta_k^{(n)} = 1 - \sum_{i=1}^{n} \frac{w_{k,i}^2}{\mathbf{w}_i^T\mathbf{w}_i + \lambda_i}$$
$$= \left(1 - \sum_{i=1}^{n-1} \frac{w_{k,i}^2}{\mathbf{w}_i^T\mathbf{w}_i + \lambda_i} \right) - \frac{w_{k,n}^2}{\mathbf{w}_n^T\mathbf{w}_n + \lambda_i}$$
$$= \eta_k^{(n-1)} - \frac{w_{k,n}^2}{\mathbf{w}_n^T\mathbf{w}_n + \lambda_i} \quad (92)$$

as well as the LOO test error of the n-term model evaluated at the k-th data sample not used for training

$$\epsilon_k^{(n,-k)} = y_k - \hat{y}_k^{(n,-k)} = \frac{\epsilon_k^{(n)}}{\eta_k^{(n)}} \quad (93)$$

Appendix D

At the beginning of the l-th stage of the OLS forward selection procedure, the l–1 regressors have been selected and the regression matrix is expressed as

$$\mathbf{\Phi}_N^{[l-1]} = \left[\mathbf{w}_1 ... \mathbf{w}_{l-1} \phi_l^{l-1} ... \phi_N^{[l-1]} \right] \quad (94)$$

Let a very small positive number T_z be given, which specifies the zero threshold and is used to automatically avoiding any ill-conditioning or singular problem. With the initial conditions as specified in (37), the l-th stage of the selection procedure is given as follows.

Step 1. For $l \le j \le N$:

- **Test** – Conditioning number check. If $\left(\phi_j^{[l-1]}\right)^T \phi_j^{[l-1]} < T_z$, the j-th candidate is not considered.
- Compute

$$g_l^{\{j\}} = \left(\phi_j^{[l-1]}\right)^T \mathbf{y}^{[l-1]} / \left(\left(\phi_j^{[l-1]}\right)^T \phi_j^{[l-1]} + \lambda_j \right) \quad (95)$$

$$\left.\begin{aligned} \epsilon_k^{(l)\{j\}} &= y_k^{[l-1]} - \phi_j^{[l-1]}(k) g_l^{\{j\}} \\ \eta_k^{(l)\{j\}} &= \eta_k^{(l-1)} - \frac{\left(\phi_j^{[l-1]}(k)\right)^2}{\left(\phi_j^{[l-1]}\right)^T \phi_j^{[l-1]} + \lambda_j} \end{aligned}\right\} k = 1, 2, ..., N \quad (96)$$

$$J_l^{\{j\}} = \frac{1}{N} \sum_{k=1}^{N} \left(\frac{\epsilon_k^{(l)\{j\}}}{\eta_k^{(l)\{j\}}} \right)^2 \quad (97)$$

where $y_k^{[l-1]}$ and $\phi_j^{[l-1]}(k)$ are the k-th elements of $y^{[l-1]}$ and $\phi_j^{[l-1]}$, respectively. Let the index set $\mathcal{J}_l$ be

$$\mathcal{J}_l = \{ l \le j \le N \text{ and } j \text{ passes } \mathbf{Test} \} \quad (98)$$

Step 2. Find

$$J_l = J_l^{\{j_l\}} = \min\{ J_l^{\{j\}}, j \in \mathcal{J}_l \} \quad (99)$$

Then the j_l-th column of $\mathbf{\Phi}_N^{[l-1]}$ is interchanged with the l-th column of $\mathbf{\Phi}_N^{[l-1]}$, the the j_l-th column of $\mathbf{A}_N$ is interchanged with the l-th column of $\mathbf{A}_N$ up to the (l-1)-th row, and the j_l-th element of $\mathbf{\lambda}_N$ is inter- changed with the l-th element of $\mathbf{\lambda}_N$. This effectively selects the j_l-th candidate as the l-th regressor in the subset model.

Step 3. The selection procedure is terminated with a (l – 1)-term model, if $J_l \ge J_{l-1}$. Otherwise, perform the orthogonalisation as indicated in (68) to derive the l-th row of $\mathbf{A}_N$ and to transform $\mathbf{\Phi}_N^{[l-1]}$ into $\mathbf{\Phi}_N^{[l]}$; Let $g_l = g_l^{\{j_l\}}$ and update $\mathbf{y}^{[l-1]}$ into $\mathbf{y}^{[l]}$ in the way shown in (69); Update the LOO error weightings

$$\eta_k^{(l)} = \eta_k^{(l-1)} - \frac{w_{k,l}^2}{\mathbf{w}_l^T\mathbf{w}_l + \lambda_l}, 1 \le k \le N \quad (100)$$

and go to *Step* 1.

Appendix E

The OLS-LOO algorithm for selecting a subset kernel classifier is basically the same one described in Appendix D, except for some minor modifications. These required modifications are explicitly given here. The initial condition is now defined by (47), and all λ_i are fixed to the constant λ. In *Step* 1, the calculation of the candidates' LOO misclassification rate

$$\left.\begin{aligned}\psi_k^{(l)\{j\}} &= \psi_k^{(l-1)} + y_k g_l^{\{j\}} \phi_j^{[l-1]}(k) - \frac{\left(\phi_j^{[l-1]}(k)\right)^2}{\left(\phi_j^{[l-1]}\right)^T \phi_j^{[l-1]} + \lambda} \\ \eta_k^{(l)\{j\}} &= \eta_k^{(l-1)} - \frac{\left(\phi_j^{[l-1]}(k)\right)^2}{\left(\phi_j^{[l-1]}\right)^T \phi_j^{[l-1]} + \lambda} \\ s_k^{(l,-k)\{j\}} &= \frac{\psi_k^{(l)\{j\}}}{\eta_k^{(l)\{j\}}}\end{aligned}\right\} 1 \le k \le N \quad \textbf{(101)}$$

$$J_l^{\{j\}} = \frac{1}{N}\sum_{k=1}^{N} \mathcal{I}_d\left(\left(s_k^{(l,-k)\{j\}}\right)^2\right) \quad \textbf{(102)}$$

In *Step* 3, in addition to update $\eta_k^{(l-1)}$ according to (100), $\psi_k^{(l-1)}$ are also updated according to

$$\psi_k^{(l)} = \psi_k^{(l-1)} + y_k g_l w_{k,l} - \frac{w_{k,l}^2}{\mathbf{w}_l^T \mathbf{w}_l + \lambda}, 1 \le k \le N \quad \textbf{(103)}$$

7. 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 Edition)," Boston, MA: PWS Publishing Company, 1990.

[3] S. A. Billings and S. Chen, "The determination of multivariable nonlinear models for dynamic systems," in: C. T. Leondes (Ed.), Control and dynamic systems, Neural Network Systems Techniques and Applications, San Diego, CA: Academic Press, Vol. 7, pp. 231–278, 1998.

[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 University Press, 1995.

[6] B. D. Ripley, "Pattern recognition and neural networks," Cambridge, U. K.: Cambridge University Press, 1996.

[7] E. Parzen, "On estimation of a probability density function and mode," The Annals of Mathematical Statistics, Vol. 33, pp. 1066–1076, 1962.

[8] B. W. Silverman, "Density estimation for statistics and data analysis," London: Chapman Hall, 1986.

[9] A. W. Bowman and A. Azzalini, "Applied smoothing techniques for data analysis," Oxford, U.K.: Oxford University Press, 1997.

[10] P. Eykhoff, "System identification: Parameter and state estimation," Chichester, England: Wiley, 1974.

[11] G. C. Goodwin and R. L. Payne, "Dynamic system identification: Experiment design and data analysis," New York: Academic Press, 1977.

[12] L. Ljung and T. Söderström, "Theory and practice of recursive identification," Cambridge, MA: MIT Press, 1983.

[13] N. R. Draper and H. Smith, "Applied regression analysis," New York: Wiley, 1981.

[14] I. J. Leontaritis and S. A. Billings, "Input-output parametric models for non-linear system – Part I: deterministic nonlinear systems; Part II: stochastic non-linear systems," International Journal of Control, Vol. 41, No. 2, pp. 303–344, 1985.

[15] I. J. Leontaritis and S. A. Billings, "Model selection and validation methods for non-linear systems," International Journal of Control, Vol. 45, No. 1, pp. 311–341, 1987.

[16] S. Chen and S. A. Billings, "Representation of non-linear systems: The NARMAX model," International Journal of Control, Vol. 49, No. 3, pp. 1013–1032, 1989.

[17] J. Moody and C. J. Darken, "Fast learning in networks of locally-tuned processing units," Neural Computation, Vol. 1, pp. 281–294, 1989.

[18] Y. H. Pao, "Adaptive pattern recognition and neural networks," Reading, MA: Addison-Wesley, 1989.

[19] S. Chen, S. A. Billings and P. M. Grant, "Non-linear system identification using neural networks," International Journal of Control, Vol. 51, No. 6, pp. 1191–1214, 1990.

[20] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamic systems using neural networks," IEEE Transactions on Neural Networks, Vol. 1, No. 1, pp. 4–27, 1990.

[21] T. Poggio and F. Girosi, "Networks for approximation and learning," Proceedings of IEEE, Vol. 78, No. 9, pp. 1481–1497, 1990.

[22] T. D. Sanger, "A tree-structured adaptive network for function approximation in high-dimensional space," IEEE Transactions on Neural Networks, Vol. 2, No. 2, pp. 285–293, 1991.

[23] S. Chen and S. A. Billings, "Neural networks for non-linear dynamic system modelling and identification," International Journal of Control, Vol. 56, No. 2, pp. 319–346, 1992.

[24] D. J. C. MacKay, "Bayesian interpretation," Neural Computation, Vol. 4, No. 3, pp. 415–447, 1992.

[25] L. X. Wang, "Adaptive fuzzy systems and control: Design and stability analysis," Prentice Hall, 1994.

[26] J. Sjoberg, Q. Zhang, L. Ljung, A. Benveniste, B. Delyon, P. Glorennec, H. Hjalmarsson, and A. Juditsky, "Nonlinear black-box modelling in system identification: A unified overview," Automatica, Vol. 31, No. 12, pp. 1691–1724, 1995.

[27] D. S. Broomhead and D. Lowe, "Multivariable function interpolation and adaptive networks," Complex Systems, Vol. 2, pp. 321–355, 1988.

[28] S. A. Billings and S. Chen, "Extended model set, global data and threshold model identification of severely non-linear systems," International Journal of Control, Vol. 50, No. 5, pp. 1897–1923, 1989.

[29] J. H. Friedman, "Multivariate adaptive regression splines," The Annals of Statistics, Vol. 19, No. 1, pp. 1–141, 1991.

[30] L. X. Wang and J. M. Mendel, "Fuzzy basis functions, universal approximation, and orthogonal least-squares

learning," IEEE Transactions on Neural Networks, Vol. 3, No. 5, pp. 807–814, 1992.

[31] T. Kavli, "ASMOD: An algorithm for adaptive spline modelling of observation data," International Journal of Control, Vol. 58, No. 4, pp. 947–968, 1993.

[32] M. Brown and C. J. Harris, "Neurofuzzy adaptive modelling and control," Englewood Cliffs, NJ: Prentice-Hall, 1994.

[33] Q. H. Zhang, "Using wavelet network in nonparametric estimation," IEEE Transactions on Neural Networks, Vol. 8, No. 2, pp. 227–236, 1997.

[34] S. Chen, S. A. Billings, and W. Luo, "Orthogonal least squares methods and their application to non-linear system identification," International Journal of Control, Vol. 50, No. 5, pp. 1873–1896, 1989.

[35] S. Chen, C. F. N. Cowan, and P. M. Grant, "Orthogonal least squares learning algorithm for radial basis function network," IEEE Transactions on Neural Networks, Vol. 2, No. 2, pp. 302–309, 1991.

[36] S. Chen and J. Wigger, "Fast orthogonal least squares algorithm for efficient subset model selection," IEEE Transactions on Signal Processing, Vol. 43, No. 7, pp. 1713–1715, 1995.

[37] S. Chen, E. S. Chng, and K. Alkadhimi, "Regularised orthogonal least squares algorithm for constructing radial basis function networks," International Journal of Control, Vol. 64, No. 5, pp. 829–837, 1996.

[38] S. Chen, Y. Wu, and B. L. Luk, "Combined genetic algorithm optimisation and regularised orthogonal least squares learning for radial basis function networks," IEEE Transactions on Neural Networks, Vol. 10, No. 5, pp. 1239–1243, 1999.

[39] S. Chen, X. Hong, and C. J. Harris, "Sparse kernel regression modelling using combined locally regularized orthogonal least squares and D-optimality experimental design," IEEE Transactions on Automatic Control, Vol. 48, No. 6, pp. 1029–1036, 2003.

[40] S. Chen, X. Hong, C. J. Harris, and P. M. Sharkey, "Sparse modelling using orthogonal forward regression with PRESS statistic and regularization," IEEE Transactions on Systems, Man and Cybernetics, Part B, Vol. 34, No. 2, pp. 898–911, 2004.

[41] S. Chen, "Local regularization assisted orthogonal least squares regression," Neurocomputing, Vol. 69, No. 4–6, pp. 559–585, 2006.

[42] S. Chen, X. X. Wang, X. Hong, and C. J. Harris, "Kernel classifier construction using orthogonal forward selection and boosting with Fisher ratio class separability measure," IEEE Transactions on Neural Networks, Vol. 17, No. 6, pp. 1652–1656, 2006.

[43] 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 Transactions on Systems, Man and Cybernetics, Part B, Vol. 34, No. 4, pp. 1708–1717, 2004.

[44] S. Chen, X. Hong, and C. J. Harris, "An orthogonal forward regression technique for sparse kernel density estimation," Neurocomputing, Vol. 71, No. 4–6, pp. 931–943, 2008.

[45] X. Hong and C. J. Harris, "Nonlinear model structure design and construction using orthogonal least squares and D-optimality design," IEEE Transactions on Neural Networks, Vol. 13, No. 5, pp. 1245–1250, 2002.

[46] X. Hong, P. M. Sharkey, and K. Warwick, "Automatic nonlinear predictive model construction algorithm using forward regression and the PRESS statistic," IEE Proceedings of Control Theory and Applications, Vol. 150, No. 3, pp. 245–254, 2003.

[47] X. Hong, C. J. Harris, S. Chen, and P. M. Sharkey, "Robust nonlinear model identification methods using forward regression," IEEE Transactions on Systems, Man and Cybernetics, Part A, Vol. 33, No. 4, pp. 514–523, 2003.

[48] X. Hong, M. Brown, S. Chen, and C. J. Harris, "Sparse model identification using orthogonal forward regression with basis pursuit and D-optimality," IEE Proceedings of Control Theory and Applications, Vol. 151, No. 4, pp. 491–498, 2004.

[49] X. Hong, S. Chen, and P. M. Sharkey, "Automatic kernel regression modelling using combined leave-one-out test score and regularised orthogonal least squares," International Journal of Neural Systems, Vol. 14, No. 1, pp. 27–37, 2004.

[50] X. Hong, C. J. Harris, and S. Chen, "Robust neurofuzzy rule base knowledge extraction and estimation using subspace decomposition combined with regularization and D-optimality," IEEE Transactions on Systems, Man and Cybernetics, Part B, Vol. 34, No. 1, pp. 598–608, 2004.

[51] X. Hong and S. Chen, "M-estimator and D-optimality model construction using orthogonal forward regression," IEEE Transactions on Systems, Man and Cybernetics, Part B, Vol. 35, No. 1, pp. 155–162, 2005.

[52] X. Hong, S. Chen, and C. J. Harris, "Fast kernel classifier construction using orthogonal forward selection to minimize leave-one-out misclassification rate," in: Proceedings of 2nd International Conference Intelligent Computing, Kunming, China, pp. 106–114, August 16–19, 2006.

[53] X. Hong, S. Chen, and C. J. Harris, "A kernel-based two-class classifier for imbalanced data sets," IEEE Transactions on Neural Networks, Vol. 18, No. 1, pp. 28–41, 2007.

[54] V. Vapnik, "The nature of statistical learning theory," New York: Springer-Verlag, 1995.

[55] B. Schölkopf, K. K. Sung, C. J. C. Burges, F. Girosi, P. Niyogi, T. Poggio, and V. Vapnik, "Comparing support vector machines with Gaussian kernels to radial basis function classifiers," IEEE Transactions on Signal Processing, Vol. 45, No. 11, pp. 2758–2765, 1997.

[56] S. Gunn, "Support vector machines for classification and regression," Technical Report, ISIS Research Group, School of Electronics and Computer Science, University of Southampton, U.K., 1998.

[57] N. Cristianini and J. Shawe-Taylor, "An introduction to support vector machines: And other kernel-based learn-

ing methods," Cambridge, U.K.: Cambridge University Press, 2000.

[58] B. Schölkopf, A. J. Smola, R. C. Williamson, and P. L. Bartlett, "New support vector algorithms," Neural Computation, Vol. 12, No. 5, pp. 1207–1245, 2000.

[59] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," SIAM Review, Vol. 43, No. 1, pp. 129–159, 2001.

[60] M. E. Tipping, "Sparse Bayesian learning and the relevance vector machine," Journal of Machine Learning Research, Vol. 1, pp. 211–244, 2001.

[61] F. Sha, L. K. Saul, and D. D. Lee, "Multiplicative updates for nonnegative quadratic programming in support vector machines," Technical Report, MS-CIS-02-19, University of Pennsylvania, USA, 2002.

[62] B. Schölkopf and A. J. Smola, "Learning with kernels: Support vector machines, regularization, optimization, and beyond," Cambridge, MA: MIT Press, 2002.

[63] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee, "Choosing multiple parameters for support vector machines," Machines Learning, Vol. 46, No. 1–3, pp. 131–159, 2002.

[64] P. Vincent and Y. Bengio, "Kernel matching pursuit," Machine Learning, Vol. 48, No. 1, pp. 165–187, 2002.

[65] J. A. K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle, "Least squares support vector machines," Singapore: World Scientific Publishing Company, 2002.

[66] C. S. Ong, A. J. Smola, and R. C. Williamson, "Hyperkernels," Neural Information Processing Systems, Vol. 15, MIT Press, 2002.

[67] K. Duan, S. S. Keerthi, and A. N. Poo, "Evaluation of simple performance measures for tuning SVM hyperparameters," Neurocomputing, Vol. 51, pp. 41–59, 2003.

[68] G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan, "Learning the kernel matrix with semidefinite programming," Journal of Machine Learning Research, Vol. 5, pp. 27–72, 2004.

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

[70] S. Mukherjee and V. Vapnik, "Support vector method for multivariate density estimation," Technical Report, A. I. Memo, No. 1653, MIT AI Laboratory, 1999.

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

[72] H. Akaike, "A new look at the statistical model identification," IEEE Transactions on Automatic Control, AC-19, pp. 716–723, 1974.

[73] M. H. Hansen and B. Yu, "Model selection and the principle of minimum description length," Journal of American Statistical Association, Vol. 96, No. 454, pp. 746–774, 2001.

[74] A. C. Atkinson and A. N. Donev, "Optimum experimental designs," Oxford, U.K.: Clarendon Press, 1992.

[75] M. Stone, "Cross validation choice and assessment of statistical predictions," Journal of Royal Statistics Society Series B, Vol. 36, pp. 117–147, 1974.

[76] D. H. Wolpert, "Stacked generalization," Neural Networks, Vol. 5, No. 2, pp. 241–259, 1992.

[77] L. Breiman, "Stacked regressions," Machine Learning, Vol. 24, pp. 49–64, 1996.

[78] L. K. Hansen and J. Larsen, "Linear unlearning for cross-validation," Advances in Computational Mathematics, Vol. 5, pp. 269–280, 1996.

[79] G. Monari and G. Dreyfus, "Withdrawing an example from the training set: An analytic estimation of its effect on a non-linear parameterised model," Neurocomputing, Vol. 35, pp. 195–201, 2000.

[80] G. Monari and G. Dreyfus, "Local overfitting control via leverages," Neural Computation, Vol. 14, pp. 1481–1506, 2002.

[81] M. Girolami and C. He, "Probability density estimation from optimally condensed data samples," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 25, No. 10, pp. 1253–1264, 2003.

[82] A. Choudhury, "Fast machine learning algorithms for large data," PhD Thesis, Computational Engineering and Design Center, School of Engineering Sciences, University of Southampton, U.K., 2002.

[83] G. McLachlan and D. Peel, "Finite mixture models," John Wiley, 2000.

[84] S. A. Billings, S. Chen, and R. J. Backhouse, "The identification of linear and non-linear models of a turbocharged automotive diesel engine," Mechanical Systems and Signal Processing, Vol. 3, No. 20, pp. 123–142, 1989.

[85] http://www.ics.uci.edu/~mlearn/MLRepository.html.

[86] http://ida.first.fhg.de/projects/bench/benchmarks.htm.

[87] G. Rätsch, T. Onoda, and K. R. Müller, "Soft margins for AdaBoost," Machine Learning, Vol. 42, No. 3, pp. 287–320, 2001.

[88] S. Chen, X. Hong, and C. J. Harris, "Orthogonal forward selection for constructing the radial basis function network with tunable nodes," in: Proceedings of 1st International Conference on Intelligent Computing, Hefei, China, Part I, pp. 777–786, August 23–26, 2005.

[89] 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: Proceedings of International Joint Conference on Neural Networks, Vancouver, Canada, pp. 6390–6394, July 16–21, 2006.

[90] S. Chen, X. Hong, and C. J. Harris, "Probability density estimation with tunable kernels using orthogonal forward regression," submitted for publication, 2009.

