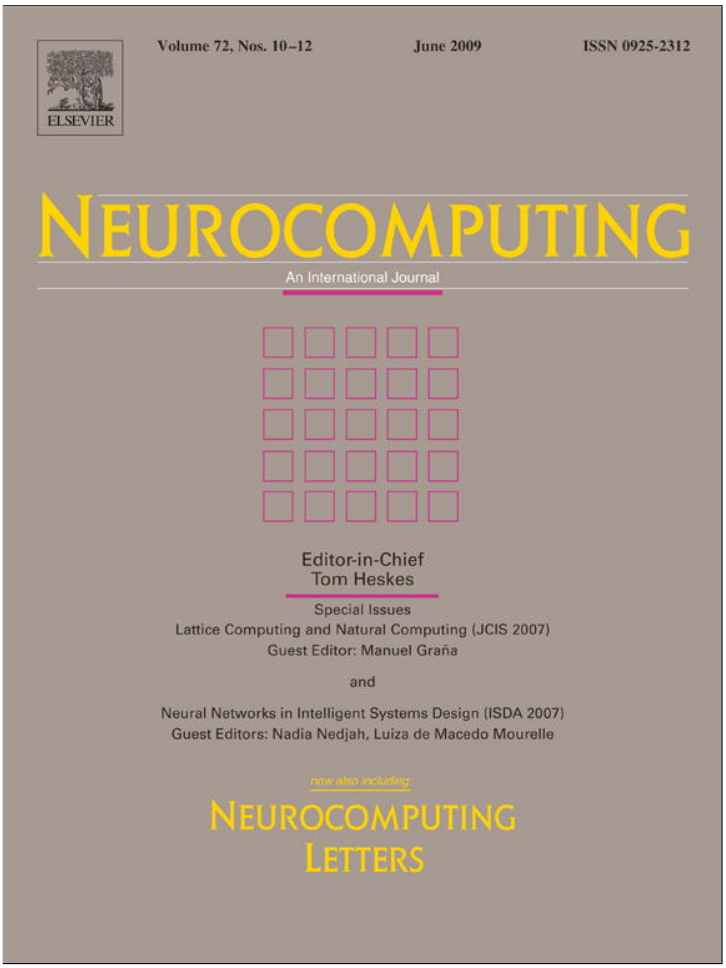

Neurocomputing 72 (2009) 2670–2681

Contents lists available at ScienceDirect

Neurocomputing

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

# Orthogonal-least-squares regression: A unified approach for data modelling

S. Chen [a,*], X. Hong [b], B.L. Luk [c], C.J. Harris [a]

[a] School of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, UK
[b] School of Systems Engineering, University of Reading, Reading RG6 6AY, UK
[c] Department of Manufacturing Engineering and Engineering Management, City University of Hong Kong, Hong Kong, China

ARTICLE INFO

*Article history:*
Received 27 May 2008
Received in revised form
29 September 2008
Accepted 5 October 2008
Communicated by K. Li
Available online 1 November 2008

*Keywords:*
Regression
Classification
Density estimation
Sparse kernel modelling
Orthogonal-least-squares algorithm
Regularisation
Leave-one-out cross-validation
Multiplicative nonnegative quadratic programming

ABSTRACT

A unified approach is proposed for data modelling that includes supervised regression and classification applications as well as unsupervised probability density function estimation. The orthogonal-least-squares regression based on the leave-one-out test criteria is formulated within this unified data-modelling framework to construct sparse kernel models that generalise well. Examples from regression, classification and density estimation applications are used to illustrate the effectiveness of this generic data-modelling approach for constructing parsimonious kernel models with excellent generalisation capability.



## 1. Introduction

The objective of modelling from data is not that the model simply fits the training data well. Rather, the goodness of a model is characterised by its generalisation capability, interpretability and ease for knowledge extraction. All these desired properties depend crucially on the ability to construct appropriate sparse models by the modelling process, and a basic principle in practical data modelling is the parsimonious principle of ensuring the smallest possible model that explains the training data. Various data modelling applications can be classified into three categories, namely, regression [5,39,1], classification [21,3,44] and probability density function (PDF) estimation [41,48,4]. 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 generalisation 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.

Recently considerable research efforts have been focused on sparse kernel data modelling techniques [6,23,19,17,40,46,45,47,50–55,20,32,38,22]. Sparse kernel modelling methods typically use every training input data as a kernel. A sparse representation is then sought based on various criteria by making as many kernel weights to (near) zero values as possible. A different approach to these sparse kernel modelling methods is the forward selection using the orthogonal-least-squares (OLS) algorithm [8,10], developed in the late 1980s for nonlinear system modelling,

* Corresponding author.
*E-mail addresses:* sqc@ecs.soton.ac.uk (S. Chen), x.hong@reading.ac.uk (X. Hong), meblluk@cityu.edu.hk (B.L. Luk), cjh@ecs.soton.ac.uk (C.J. Harris).

0925-2312/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2008.10.002

which remains highly popular for data-modelling practicians. Since its derivation, many enhanced variants of the OLS forward-selection algorithm have been proposed by incorporating the new developments from machining learning and the approach has extended its application to all the areas of data modelling, including regression, classification and kernel density estimation [9,11–16,7,25–29,42,33]. This contribution continues this theme, and it presents a unified framework for sparse kernel modelling that include all the three classes of data-modelling applications, namely, regression, classification and PDF estimation. Based on this unified data-modelling framework, the OLS forward-selection algorithm using the leave-one-out (LOO) test criteria and local regularisation (LR) is employed to construct sparse kernel models with excellent generalisation capability. Experimental results are included to demonstrate the effectiveness of the OLS forward-selection algorithm based on the LOO test criteria and regularisation within the proposed unified data-modelling framework.

## 2. A unified framework for data modelling

Regression, classification and PDF estimation can be unified under a 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 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 requirements of a density distribution, i.e. a kernel is nonnegative and the area under the kernel is unity.

### 2.1. Regression

Consider the general nonlinear data generating mechanism governed by the nonlinear model $y=f(\mathbf{x})+\varepsilon$, where $y\in\mathscr{R}$ denotes the system output, $\mathbf{x}=[x_1\ x_2\ \cdots\ x_m]^{\mathrm{T}}\in\mathscr{R}^m$ the system input, $\varepsilon$ is a white noise process representing for example the observation noise, and $f:\mathscr{R}^m\rightarrow\mathscr{R}$ 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, $\hat{f}:\mathscr{R}^m\rightarrow\mathscr{R}$, of the form

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

to capture the underlying data generating mechanism, where $\hat{y}$ denotes the model output, $\boldsymbol{\beta}_N=[\beta_1\ \beta_2\ \cdots\ \beta_N]^{\mathrm{T}}$ is the kernel weight vector and $K_\rho(\bullet,\bullet)$ is the chosen kernel function with a kernel width $\rho$. Many types of kernel functions can be employed and a commonly used one is the Gaussian function of the form

$$K_\rho(\mathbf{x},\mathbf{c}_k)=\frac{1}{(2\pi\rho^2)^{m/2}}\mathrm{e}^{-\|\mathbf{x}-\mathbf{c}_k\|^2/2\rho^2}, \tag{2}$$

where $\mathbf{c}_k\in\mathscr{R}^m$ is the $k$-th kernel centre vector. For regression and classification problems, the factor $1/(2\pi\rho^2)^{m/2}$ can be combined into kernel weights $\beta_i$. The generic kernel model (1) 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 only $N_s$ significant regressors from the full regressor set, where $N_s\ll N$.

At a training data point $(\mathbf{x}_k,y_k)$, the kernel model (1) can be expressed as

$$y_k=\hat{y}_k+\varepsilon_k=\sum_{i=1}^{N}\beta_i K_\rho(\mathbf{x}_k,\mathbf{x}_i)+\varepsilon_k=\boldsymbol{\phi}^{\mathrm{T}}(k)\boldsymbol{\beta}_N+\varepsilon_k, \tag{3}$$

where $\varepsilon_k=y_k-\hat{y}_k$ is the modelling error at $\mathbf{x}_k$ and $\boldsymbol{\phi}(k)=[K_{k,1}\ K_{k,2}\ \cdots\ K_{k,N}]^{\mathrm{T}}$ with $K_{k,i}=K_\rho(\mathbf{x}_k,\mathbf{x}_i)$. By defining $\boldsymbol{\Phi}=[\boldsymbol{\phi}_1\ \boldsymbol{\phi}_2\ \cdots\ \boldsymbol{\phi}_N]$ with $\boldsymbol{\phi}_k=[K_{1,k}\ K_{2,k}\ \cdots\ K_{N,k}]^{\mathrm{T}}$ for $1\leqslant k\leqslant N$, $\mathbf{y}=[y_1\ y_2\ \cdots\ y_N]^{\mathrm{T}}$ and $\boldsymbol{\varepsilon}=[\varepsilon_1\ \varepsilon_2\ \cdots\ \varepsilon_N]^{\mathrm{T}}$, the regression model (3) over the training data set $D_N$ can be expressed in the matrix form

$$\mathbf{y}=\boldsymbol{\Phi}\boldsymbol{\beta}_N+\boldsymbol{\varepsilon}. \tag{4}$$

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

$$\mathbf{A}=\begin{bmatrix}1 & a_{1,2} & \cdots & a_{1,N}\\ 0 & 1 & \ddots & \vdots\\ \vdots & \ddots & \ddots & a_{N-1,N}\\ 0 & \cdots & 0 & 1\end{bmatrix} \tag{5}$$

and

$$\mathbf{W}=[\mathbf{w}_1\ \ \mathbf{w}_2\ \ \cdots\ \ \mathbf{w}_N] \tag{6}$$

with orthogonal columns satisfying $\mathbf{w}_i^{\mathrm{T}}\mathbf{w}_j=0$, if $i\neq j$. The regression model (4) can alternatively be expressed as

$$\mathbf{y}=\mathbf{W}\mathbf{g}_N+\boldsymbol{\varepsilon}, \tag{7}$$

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

$$\hat{y}_k=\mathbf{w}^{\mathrm{T}}(k)\mathbf{g}_N, \tag{8}$$

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

### 2.2. Classification

For notational simplification, we only consider the two-class classification problem with the given training data set $D_N=\{\mathbf{x}_k,y_k\}_{k=1}^N$, where $\mathbf{x}_k\in\mathscr{R}^m$ 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

$$\tilde{y}_k=\mathrm{sgn}(\hat{y}_k) \tag{9}$$

with

$$\hat{y}_k=\sum_{i=1}^{N}\beta_i K_\rho(\mathbf{x}_k,\mathbf{x}_i), \tag{10}$$

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

$$\mathrm{sgn}(y)=\begin{cases}-1, & y\leqslant 0,\\ +1, & y>0.\end{cases} \tag{11}$$

Let us define the modelling error as $\varepsilon_k=y_k-\hat{y}_k$. Then the classification model over the training data set $D_N$ can be expressed in the regression model of (4), recited here $\mathbf{y}=\boldsymbol{\Phi}\boldsymbol{\beta}_N+\boldsymbol{\varepsilon}$, or equivalently in the orthogonal regression model of (7), rewritten here $\mathbf{y}=\mathbf{W}\mathbf{g}_N+\boldsymbol{\varepsilon}$, where all the relevant notations are as defined in Section 2.1. It is clear that the kernel classifier construction can be expressed in the same kernel regression modelling framework of Section 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 $\mathbf{x}_k \in \mathscr{R}^m$, the task is to estimate the unknown density $p(\mathbf{x})$ using the kernel density estimate of the form

$$\hat{p}(\mathbf{x}; \boldsymbol{\beta}_N, \rho) = \sum_{k=1}^{N} \beta_k K_\rho(\mathbf{x}, \mathbf{x}_k) \tag{12}$$

with the constraints

$$\beta_k \geqslant 0, \quad 1 \leqslant k \leqslant N \tag{13}$$

and

$$\boldsymbol{\beta}_N^{\mathrm{T}} \mathbf{1}_N = 1, \tag{14}$$

where $\mathbf{1}_N$ denotes the vector of ones with dimension $N$. The kernel function $K_\rho(\bullet, \bullet)$ is chosen to be the Gaussian kernel (2) in this study. However, many other kernel functions can also be used in the density estimate (12). This kernel density estimation is an unsupervised learning problem, as the desired response for the training data points $\mathbf{x}_k$ are unknown.

The well-known Parzen window (PW) estimate [41], denoted as $\hat{p}(\mathbf{x}; \boldsymbol{\beta}_{N_{\mathrm{Par}}}, \rho_{\mathrm{Par}})$, is obtained by simply setting all the elements of $\boldsymbol{\beta}_{N_{\mathrm{Par}}}$ to $1/N$, and the kernel width $\rho_{\mathrm{Par}}$ is typically determined via cross-validation [39,49]. The PW estimate is remarkably simple and accurate [41]. The PW estimate in fact can be derived as the maximum likelihood estimator using the divergence-based criterion [35]. The negative cross-entropy or divergence between the true density $p(\mathbf{x})$ and the estimate $\hat{p}(\mathbf{x}; \boldsymbol{\beta}_N, \rho)$, calculated over the training data set $D_N$, is defined as

$$\begin{aligned} \int_{\mathscr{R}^m} p(\mathbf{u}) \log \hat{p}(\mathbf{u}; \boldsymbol{\beta}_N, \rho)\, \mathrm{d}\mathbf{u} &\approx \frac{1}{N} \sum_{k=1}^{N} \log \hat{p}(\mathbf{x}_k; \boldsymbol{\beta}_N, \rho) \\ &= \frac{1}{N} \sum_{k=1}^{N} \log \left( \sum_{n=1}^{N} \beta_n K_\rho(\mathbf{x}_k, \mathbf{x}_n) \right). \end{aligned} \tag{15}$$

Minimising this divergence subject to constraints (13) and (14) leads to $\beta_n = 1/N$ for $1 \leqslant n \leqslant N$, 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 (SKD) estimation techniques [55,38,53,22,12,13,18].

Following the approach of [13], the unsupervised density learning is transformed into a supervised learning problem. The PW estimate can be regarded as the "observation" of the true density contaminated by some "observation noise", namely

$$\hat{p}(\mathbf{x}; \boldsymbol{\beta}_{N_{\mathrm{Par}}}, \rho_{\mathrm{Par}}) = p(\mathbf{x}) + \tilde{\varepsilon}(\mathbf{x}). \tag{16}$$

Thus the generic kernel density estimation problem (12) can be viewed as the following regression problem with the PW estimate as the "desired response" or target function:

$$\hat{p}(\mathbf{x}; \boldsymbol{\beta}_{N_{\mathrm{Par}}}, \rho_{\mathrm{Par}}) = \sum_{k=1}^{N} \beta_k K_\rho(\mathbf{x}, \mathbf{x}_k) + \varepsilon(\mathbf{x}) \tag{17}$$

subject to constraints (13) and (14), where $\varepsilon(\mathbf{x})$ denotes the modelling error at $\mathbf{x}$. Define $y_k = \hat{p}(\mathbf{x}_k; \boldsymbol{\beta}_{N_{\mathrm{Par}}}, \rho_{\mathrm{Par}})$ and $\varepsilon_k = \varepsilon(\mathbf{x}_k)$. Then the generic kernel density estimation problem is expressed in the same kernel regression modelling framework of (4), recited here again $\mathbf{y} = \boldsymbol{\Phi}\boldsymbol{\beta}_N + \boldsymbol{\varepsilon}$, subject to the nonnegative constraint (13) and the unity constraint (14), where all the relevant notations have been defined in Section 2.1. The regression model (4) can of course be written equivalently in the form of (7), which is recited here again $\mathbf{y} = \mathbf{W}\mathbf{g}_N + \boldsymbol{\varepsilon}$. The objective is to obtain a sparse $N_s$-term kernel model, satisfying the kernel weight constraints (13) and (14) and yet having a test performance comparable to that of the full-sample optimised PW estimate.

## 3. OLS 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 LR (OLS-LOO-LR) [14,26,13] provides an efficient algorithm to construct a sparse kernel model that generalise well. For the regression and density modelling, the LOO MSE criterion is an appropriate measure of model's generalisation capability for subset model selection [14,13], while for classifier construction, the LOO misclassification rate offers a proper measure of classifier's generalisation performance for selecting significant kernels [26]. SKD construction is special as it is formulated as a constrained regression modelling, where the kernel weights must meet the nonnegative and unity constraints. We will adopt the combined approach of [13] to this constrained regression modelling, in which the OLS-LOO-LR algorithm determines the number of kernels in the SKD estimate while a modified multiplicative nonnegative quadratic programming (MNQP) algorithm [47,22] computes the kernel weights of the selected SKD estimate.

### 3.1. Sparse kernel regression model construction

The LR aided least-squares (LS) solution for the weight parameter vector $\mathbf{g}_N$ is obtained by minimising the following regularised error criterion [7]:

$$J_R(\mathbf{g}_N, \boldsymbol{\lambda}_N) = \boldsymbol{\varepsilon}^{\mathrm{T}}\boldsymbol{\varepsilon} + \mathbf{g}_N^{\mathrm{T}}\boldsymbol{\Lambda}\mathbf{g}_N, \tag{18}$$

where $\boldsymbol{\lambda}_N = [\lambda_1 \ \lambda_2 \ \cdots \ \lambda_N]^{\mathrm{T}}$ is the vector of regularisation parameters, and $\boldsymbol{\Lambda} = \mathrm{diag}\{\lambda_1, \lambda_2, \ldots, \lambda_N\}$. In fact setting $\partial J_R / \partial \mathbf{g}_N = \mathbf{0}$ leads to the normal equation $\mathbf{W}^{\mathrm{T}}\mathbf{y} = (\mathbf{W}^{\mathrm{T}}\mathbf{W} + \boldsymbol{\Lambda})\mathbf{g}_N$. Because $\mathbf{W}^{\mathrm{T}}\mathbf{W} + \boldsymbol{\Lambda}$ is diagonal, the solution is $g_l = \mathbf{w}_l^{\mathrm{T}}\mathbf{y}/(\mathbf{w}_l^{\mathrm{T}}\mathbf{w}_l + \lambda_l)$ for $1 \leqslant l \leqslant N$, as is given in Appendix A. Criterion (18) is rooted in the Bayesian learning framework. According to the Bayesian learning theory [51,7,34], the optimal $\mathbf{g}_N$ is obtained by maximising the posterior probability of $\mathbf{g}_N$, which can be shown to be

$$p(\mathbf{g}_N | \mathbf{y}, \mathbf{h}_N, \omega) = \frac{p(\mathbf{y} | \mathbf{g}_N, \mathbf{h}_N, \omega) p(\mathbf{g}_N | \mathbf{h}_N, \omega)}{p(\mathbf{y} | \mathbf{h}_N, \omega)}, \tag{19}$$

where $p(\mathbf{g}_N | \mathbf{h}_N, \omega)$ is the prior with $\mathbf{h}_N = [h_1 \ h_2 \ \cdots \ h_N]^{\mathrm{T}}$ denoting the vector of hyperparameters and $\omega$ a noise parameter (the inverse of the variance of $\varepsilon$), $p(\mathbf{y} | \mathbf{g}_N, \mathbf{h}_N, \omega)$ is called the likelihood, and $p(\mathbf{y} | \mathbf{h}_N, \omega)$ is the evidence which does not depend on $\mathbf{g}_N$ explicitly. Under the assumption that $\varepsilon$ is white and has a Gaussian distribution, the likelihood is given by

$$p(\mathbf{y} | \mathbf{g}_N, \mathbf{h}_N, \omega) = \prod_{i=1}^{N} \left( \frac{\omega}{2\pi} \right)^{N/2} \mathrm{e}^{-(\omega/2)\boldsymbol{\varepsilon}^{\mathrm{T}}\boldsymbol{\varepsilon}}. \tag{20}$$

If the Gaussian prior is chosen, i.e.

$$p(\mathbf{g}_N | \mathbf{h}_N, \omega) = \prod_{i=1}^{N} \frac{\sqrt{h_i}}{\sqrt{2\pi}} \mathrm{e}^{-h_i g_i^2/2}, \tag{21}$$

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

$$J_B(\mathbf{g}_N, \mathbf{h}_N, \omega) = \omega\boldsymbol{\varepsilon}^{\mathrm{T}}\boldsymbol{\varepsilon} + \mathbf{g}_N^{\mathrm{T}}\mathbf{H}\mathbf{g}_N, \tag{22}$$

where $\mathbf{H} = \mathrm{diag}\{h_1, h_2, \ldots, h_N\}$. It is obvious that criterion (18) is equivalent to criterion (22) with the relationship

$$\lambda_i = h_i/\omega, \quad 1 \leqslant i \leqslant N. \tag{23}$$

The hyperparameters specify the prior distributions of $\mathbf{g}_N$. Since initially the optimal value of $\mathbf{g}_N$ is unknown, $\lambda_i$ should be initialised to the same small value, and this corresponds to choose a same flat distribution for each prior of $g_i$ in (21). 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 is done by iteratively optimising $\mathbf{g}_N$ and $\mathbf{h}_N$ using the evidence procedure [51,7,34]. Applying this evidence procedure results in the following iterative updating formulas for the regularisation parameters [7]:

$$\lambda_i^{\text{new}} = \frac{\gamma_i^{\text{old}}}{N-\gamma^{\text{old}}}\frac{\boldsymbol{\varepsilon}^{\mathrm{T}}\boldsymbol{\varepsilon}}{g_i^2}, \quad 1\leqslant i\leqslant N, \tag{24}$$

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

$$\gamma = \sum_{i=1}^{N}\gamma_i \quad \text{with } \gamma_i = \frac{\mathbf{w}_i^{\mathrm{T}}\mathbf{w}_i}{\lambda_i+\mathbf{w}_i^{\mathrm{T}}\mathbf{w}_i}. \tag{25}$$

Usually a few iterations (typically less than 10) are sufficient to find a (near) optimal $\boldsymbol{\lambda}_N$. The derivation of the updating formulas (24) and (25), quoted from [7], can be found in Appendix B. The use of multiple-regularisers or LR is known to be capable of providing very sparse solutions [51,7].

It is highly desired to select a sparse model by directly optimising the model generalisation capability, rather than minimising the training MSE. The OLS-LOO-LR algorithm achieves this objective by incrementally minimising the LOO MSE criterion, which is a measure of the model's generalisation performance [39,14,29,24,36,37]. 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 $\varepsilon_k^{(n,-k)}$, for the selected $n$-term model is [14,29]

$$\varepsilon_k^{(n,-k)} = \varepsilon_k^{(n)}/\eta_k^{(n)}, \tag{26}$$

where $\varepsilon_k^{(n)}$ 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

$$J_n = \frac{1}{N}\sum_{k=1}^{N}(\varepsilon_k^{(n,-k)})^2 = \frac{1}{N}\sum_{k=1}^{N}(\varepsilon_k^{(n)})^2/(\eta_k^{(n)})^2. \tag{27}$$

This LOO MSE can be computed efficiently due to the fact that the $n$-term model error $\varepsilon_k^{(n)}$ and the associated LOO error weighting $\eta_k^{(n)}$ can be calculated recursively according to [14,29]

$$\varepsilon_k^{(n)} = \varepsilon_k^{(n-1)} - w_{k,n}g_n \tag{28}$$

and

$$\eta_k^{(n)} = \eta_k^{(n-1)} - \frac{w_{k,n}^2}{\mathbf{w}_n^{\mathrm{T}}\mathbf{w}_n+\lambda_n}, \tag{29}$$

respectively, where $w_{k,n}$ is the $k$-th element of $\mathbf{w}_n$. The derivation of the LOO test error (26) together with the recursive formulas (28) and (29) 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

$$J_{N_s+1} \geqslant J_{N_s}, \tag{30}$$

yielding an $N_s$-term sparse model. It has been shown in [29] that the LOO statistic $J_n$ has the following desired property with respect to the model size $n$: There exists an "optimal" model size $N_s$ such that for $n\leqslant N_s$ $J_n$ decreases as $n$ increases while condition (30) 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.

- *Initialisation*: Set $\lambda_i = 10^{-6}$ for $1\leqslant i\leqslant N$, and set iteration index $I = 1$.
- *Step* 1: Given the current $\boldsymbol{\lambda}_N$ and with the following initial conditions

$$\begin{cases} \varepsilon_k^{(0)} = y_k \quad \text{and} \quad \eta_k^{(0)} = 1, \ \ 1\leqslant k\leqslant N, \\ J_0 = \dfrac{1}{N}\mathbf{y}^{\mathrm{T}}\mathbf{y} = \dfrac{1}{N}\sum_{k=1}^{N} y_k^2, \end{cases} \tag{31}$$

  use the procedure described in Appendix D to select a subset model with $N_I$ terms.
- *Step* 2: Update $\boldsymbol{\lambda}_N$ using (24) and (25) with $N = N_I$. If the preset maximum iteration number (e.g. 10) is reached, stop; otherwise set $I+=1$ and go to *Step* 1.

### 3.2. Sparse kernel classifier construction

The same LOO cross-validation concept [39] is adopted to provide a measure of classifier's generalisation capability. Denote the test output of the LOO $n$-term model evaluated at the $k$-th data sample of $D_N$ not used in training as $\hat{y}_k^{(n,-k)}$. The associated LOO signed decision variable is defined by

$$s_k^{(n,-k)} = \text{sgn}(y_k)\hat{y}_k^{(n,-k)} = y_k\hat{y}_k^{(n,-k)}, \tag{32}$$

where $\text{sgn}(y_k) = y_k$ since $y_k\in\{-1,+1\}$. The LOO misclassification rate can be computed by

$$J_n = \frac{1}{N}\sum_{k=1}^{N}\mathscr{I}_d(s_k^{(n,-k)}), \tag{33}$$

where the indication function is defined by $\mathscr{I}_d(y) = 1$ if $y\leqslant 0$ and $\mathscr{I}_d(y) = 0$ if $y>0$. The LOO misclassification rate $J_n$ can be evaluated efficiently because $s_k^{(n,-k)}$ can be calculated very fast [26]. Specifically, the LOO $n$-term modelling error is expressed by (also see Appendix C)

$$y_k - \hat{y}_k^{(n,-k)} = \frac{y_k-\hat{y}_k^{(n)}}{1-\sum_{i=1}^{n}\dfrac{w_{k,i}^2}{\mathbf{w}_i^{\mathrm{T}}\mathbf{w}_i+\lambda_i}}, \tag{34}$$

where $\hat{y}_k^{(n)}$ is the $n$-term model output. Multiplying the both sides of (34) with $y_k$ and applying $y_k^2 = 1$ yields

$$1 - s_k^{(n,-k)} = \frac{1-y_k\hat{y}_k^{(n)}}{1-\sum_{i=1}^{n}\dfrac{w_{k,i}^2}{\mathbf{w}_i^{\mathrm{T}}\mathbf{w}_i+\lambda_i}}. \tag{35}$$

From (35), the LOO $n$-term signed decision variable is given by

$$s_k^{(n,-k)} = \frac{\sum_{i=1}^{n}y_kg_iw_{k,i} - \sum_{i=1}^{n}\dfrac{w_{k,i}^2}{\mathbf{w}_i^{\mathrm{T}}\mathbf{w}_i+\lambda_i}}{1-\sum_{i=1}^{n}\dfrac{w_{k,i}^2}{\mathbf{w}_i^{\mathrm{T}}\mathbf{w}_i+\lambda_i}} = \frac{\psi_k^{(n)}}{\eta_k^{(n)}}. \tag{36}$$

The recursive formula for the LOO error weighting $\eta_k^{(n)}$ is given in (29), while $\psi_k^{(n)}$ can be represented using the following recursive formula [26]

$$\psi_k^{(n)} = \psi_k^{(n-1)} + y_kg_nw_{k,n} - \frac{w_{k,n}^2}{\mathbf{w}_n^{\mathrm{T}}\mathbf{w}_n+\lambda_n}. \tag{37}$$

The OLS-LOO-LR algorithm described in Section 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 (33) rather than the LOO MSE (27). Moreover, extensive empirical experience has suggested that all the regularisation parameters $\lambda_i$, $1 \leqslant i \leqslant N$, can be set to a small positive constant $\lambda$, and there is no need to update them using the evidence procedure. This significantly reduces the computational cost. The sparse kernel classifier selection procedure based on this OLS-LOO algorithm is now summarised.

- Setting $\lambda$ to a small positive number, and with the following initial conditions

$$\psi_k^{(0)} = 0, \quad \eta_k^{(0)} = 1 \ \text{ for } 1 \leqslant k \leqslant N \text{ and } J_0 = 1, \tag{38}$$

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

The LOO misclassification rate $J_n$ also has the desired property with respect to the classifier's size $n$, namely, there exists an optimal model size $N_s$ such that for $n \leqslant N_s$ $J_n$ decreases as $n$ increases, while $J_{N_s} \leqslant J_{N_s+1}$. Therefore the selection procedure is automatically terminated with a subset classifier containing only $N_s$ significant kernels.

### 3.3. SKD estimator construction

Since the density estimation can be expressed as a constrained regression modelling, the OLS-LOO-LR algorithm detailed in Section 3.1 can be used to select a sparse kernel model. The only problem is that the kernel weights obtained by the OLS-LOO-LR algorithm do not necessarily meet the nonnegative constraint (13) and the unity constraint (14). This "deficiency", however, can easily be corrected by using the modified MNQP algorithm to update the kernel weights of the selected sparse model [13]. This combined OLS-LOO-LR and MNQP algorithm offers an effective means of obtaining SKD estimates with excellent generalisation capability. The detailed OLS-LOO-LR algorithm has been described in Section 3.1 and, therefore, we only need to discuss the MNQP part.

After the structure determination using the OLS-LOO-LR algorithm of Section 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 $\mathbf{A}$, corresponding to the selected $N_s$-term subset model. The kernel weight vector $\boldsymbol{\beta}_{N_s} = [\beta_1 \ \beta_2 \ \cdots \ \beta_{N_s}]^{\mathrm{T}}$, computed from $\mathbf{A}_{N_s}\boldsymbol{\beta}_{N_s} = \mathbf{g}_{N_s}$, may not satisfy constraints (13) and (14). Thus $\boldsymbol{\beta}_{N_s}$ must be recalculated using for example the modified version of the MNQP algorithm [47,22]. 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 model

$$\mathbf{y} = \boldsymbol{\Phi}_{N_s}\boldsymbol{\beta}_{N_s} + \boldsymbol{\varepsilon} \tag{39}$$

subject to the constraints

$$\beta_i \geqslant 0, \quad 1 \leqslant i \leqslant N_s, \tag{40}$$

$$\boldsymbol{\beta}_{N_s}^{\mathrm{T}}\mathbf{1}_{N_s} = 1, \tag{41}$$

where $\boldsymbol{\Phi}_{N_s}$ denotes the selected subset regression matrix. The kernel weight vector can be obtained by solving the following constrained nonnegative quadratic programming:

$$\min_{\boldsymbol{\beta}_{N_s}} \ \{\tfrac{1}{2}\boldsymbol{\beta}_{N_s}^{\mathrm{T}}\mathbf{C}_{N_s}\boldsymbol{\beta}_{N_s} - \mathbf{v}_{N_s}^{\mathrm{T}}\boldsymbol{\beta}_{N_s}\}$$
$$\text{s.t.} \quad \boldsymbol{\beta}_{N_s}^{\mathrm{T}}\mathbf{1}_{N_s} = 1 \quad \text{and} \quad \beta_i \geqslant 0, \ 1 \leqslant i \leqslant N_s, \tag{42}$$

where $\mathbf{C}_{N_s} = \boldsymbol{\Phi}_{N_s}^{\mathrm{T}}\boldsymbol{\Phi}_{N_s} = [c_{i,j}] \in \mathscr{R}^{N_s \times N_s}$ is the related design matrix and $\mathbf{v}_{N_s} = \boldsymbol{\Phi}_{N_s}^{\mathrm{T}}\mathbf{y} = [v_1 \ v_2 \ \cdots \ v_{N_s}]^{\mathrm{T}}$. 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 [47].

Since the elements of $\mathbf{C}_{N_s}$ and $\mathbf{v}_{N_s}$ are strictly positive, the auxiliary function [47] for the above problem is given by

$$\frac{1}{2}\sum_{i=1}^{N_s}\sum_{j=1}^{N_s} c_{i,j}\frac{\beta_j^{\langle t\rangle}(\beta_i^{\langle t+1\rangle})^2}{\beta_i^{\langle t\rangle}} - \sum_{i=1}^{N_s} v_i\beta_i^{\langle t+1\rangle} \tag{43}$$

and the Lagrangian associated with this auxiliary problem can be formed as [22]

$$\mathscr{L} = \frac{1}{2}\sum_{i=1}^{N_s}\sum_{j=1}^{N_s} c_{i,j}\frac{\beta_j^{\langle t\rangle}(\beta_i^{\langle t+1\rangle})^2}{\beta_i^{\langle t\rangle}} - \sum_{i=1}^{N_s} v_i\beta_i^{\langle t+1\rangle} - h^{\langle t\rangle}\left(\sum_{i=1}^{N_s}\beta_i^{\langle t+1\rangle} - 1\right), \tag{44}$$

where the superindex $\langle t\rangle$ denotes the iteration index and $h$ is the Lagrangian multiplier. Setting

$$\frac{\partial\mathscr{L}}{\partial\beta_i^{\langle t+1\rangle}} = 0 \quad \text{and} \quad \frac{\partial\mathscr{L}}{\partial h^{\langle t\rangle}} = 0 \tag{45}$$

leads to the following updating equations:

$$r_i^{\langle t\rangle} = \beta_i^{\langle t\rangle}\left(\sum_{j=1}^{N_s} c_{i,j}\beta_j^{\langle t\rangle}\right)^{-1}, \quad 1 \leqslant i \leqslant N_s, \tag{46}$$

$$h^{\langle t\rangle} = \left(\sum_{i=1}^{N_s} r_i^{\langle t\rangle}\right)^{-1}\left(1 - \sum_{i=1}^{N_s} r_i^{\langle t\rangle}v_i\right), \tag{47}$$

$$\beta_i^{\langle t+1\rangle} = r_i^{\langle t\rangle}(v_i + h^{\langle t\rangle}). \tag{48}$$

It is easy to check that, if $\boldsymbol{\beta}_{N_s}^{\langle t\rangle}$ meets the constraints (40) and (41), $\boldsymbol{\beta}_{N_s}^{\langle t+1\rangle}$ updated according to (46)–(48) also satisfies (40) and (41). The initial condition can be set as $\beta_i^{\langle 0\rangle} = 1/N_s$, $1 \leqslant i \leqslant N_s$. Alternative, $\boldsymbol{\beta}_{N_s}^{\langle 0\rangle}$ 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 $\boldsymbol{\beta}_{N_s}^{\langle 0\rangle}$. During the iterative procedure, some of the kernel weights may be driven to (near) zero [47,22]. The corresponding kernels can then be removed from the kernel model, leading to a further reduction in the subset model size.

## 4. Empirical data-modelling results

Several examples, taken from regression, classification and density estimation applications, were used to demonstrate the effectiveness of the proposed unified regression modelling approach. For each data-modelling example, the full regression model set was formed by placing a Gaussian kernel (2) on each training data sample, and a sparse kernel model was then selected using the OLS-LOO-LR/OLS-LOO algorithm. For SKD estimation, additionally, the modified MNQP algorithm described in Section 3.3 was applied to update the kernel weight vector. The appropriate value for the kernel width $\rho$ 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. Comparison 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

*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 [2]. Detailed system description and experimental setup can be found in [2]. The data set, depicted in Fig. 1, contained 410 samples. The first 210 data points were used in modelling and the last 200 points in model validation. The previous results [2] have shown that this data set can be modelled adequately as

$$y_k = f(\mathbf{x}_k) + e_k, \tag{49}$$

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

$$\mathbf{x}_k = [y_{k-1} \;\; u_{k-1} \;\; u_{k-2}]^{\mathrm{T}}. \tag{50}$$

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 (4).

Both the OLS-LOO-LR algorithm and the support vector machine (SVM) algorithm with the $\varepsilon$-insensitive cost function [23] were applied to this data set, and the two sparse Gaussian kernel models obtained are compared in Table 1. The model

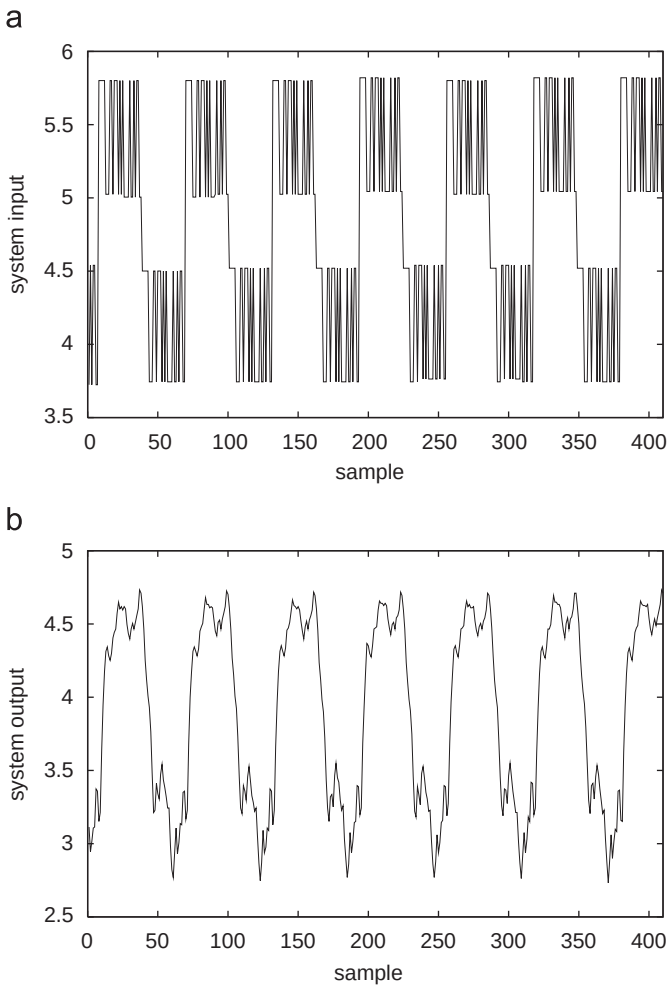


**Fig. 1.** Engine data set (a) input $u_k$ and (b) output $y_k$.

**Table 1**
Comparison of modelling accuracy for the engine data set.

| Algorithm | Model size | Training MSE | Test MSE |
|---|---|---|---|
| OLS-LOO-LR | 22 | 0.000453 | 0.000490 |
| SVM | 92 | 0.000447 | 0.000498 |

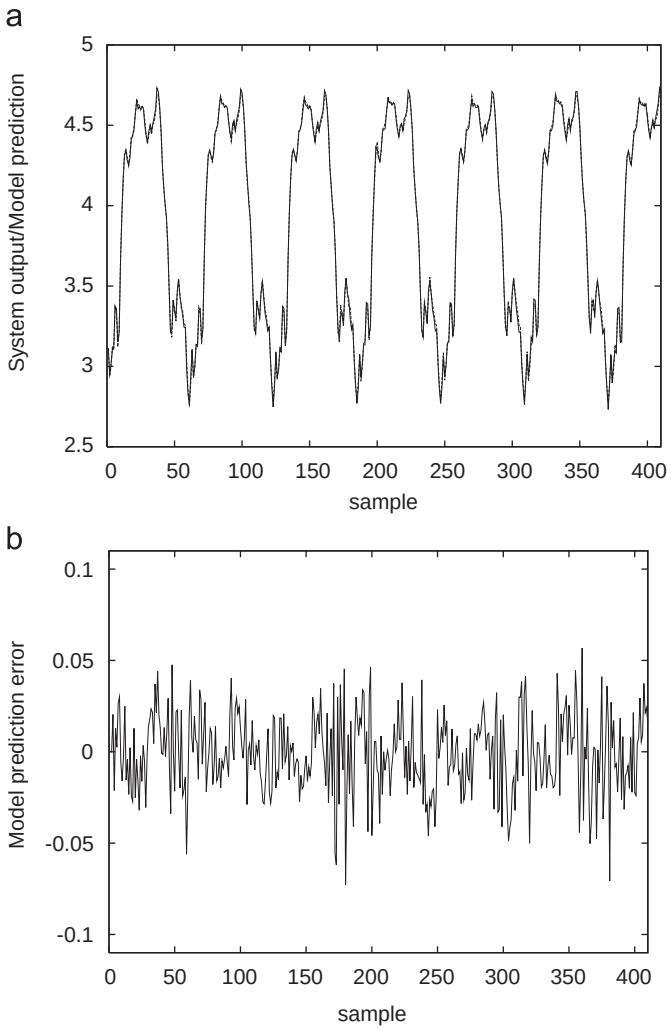


**Fig. 2.** Modelling performance for the engine data set: (a) model prediction $\hat{y}_k$ (dashed) superimposed on system output $y_k$ (solid) and (b) model prediction error $\varepsilon_k = y_k - \hat{y}_k$. The 22-term model was constructed by the proposed OLS-LOO-LR algorithm.

output $\hat{y}_k$ and modelling error $\varepsilon_k = y_k - \hat{y}_k$ generated by the 22-term kernel model obtained using the OLS-LOO-LR algorithm are depicted in Fig. 2. The modelling performance of the 92-term kernel model constructed by the SVM algorithm, not shown here, are very similar to those shown in Fig. 2. 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, as 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 $\rho$. However, the SVM method has two more learning parameters, namely the error-band and trade-off parameters [23], that require tuning. Therefore, the OLS-LOO-LR algorithm is easier to tune and computationally more efficient than the SVM algorithm.

*Boston housing data set*: This was a regression benchmark data set, available at the UCI repository [30]. 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 (4). The kernel width for the OLS-LOO-LR algorithm was determined via a grid-search-based cross-validation. The SVM algorithm [23] was also used to construct the regression model for this data set, as a comparison. The three learning parameters of the SVM algorithm, the kernel width, error-band and trade-off parameters, were also 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 2.

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 when the kernel width, error-band and trade-off parameters were chosen. One could argue that by adopting fast implementation of the SVM algorithm significant reduction in run time can be achieved but it would still be less efficient than the OLS-LOO-LR algorithm. It can be seen from Table 2 that the OLS-LOO-LR algorithm achieved better modelling accuracy with a much sparser model than the SVM algorithm. 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

*Diabetes data*: This two-class classification benchmark data set was originated in the UCI repository [30] and the data set used in the experiment was obtained from [31]. 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 existing state-of-the-art radial basis function (RBF) and kernel classifiers were compared in [31,43]. The results given in [31] were reproduced in Table 3. For the first five methods studied in [31], 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 [31] but it could safely be assumed that it was larger than, said, 100. The kernel Fisher discriminant was the nonsparse optimal classifier using all the $N = 468$ training data samples as kernels. 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 3. It can be seen that the proposed OLS-LOO method compared favourably with the existing benchmark RBF and kernel classifier construction algorithms, both in terms of classification accuracy and model size.

**Table 2**
Comparison of modelling accuracy for the Boston housing data set.

| Algorithm | OLS-LOO-LR | SVM |
|---|---|---|
| Model size | $58.6 \pm 11.3$ | $243.2 \pm 5.3$ |
| Training MSE | $12.9690 \pm 2.6628$ | $6.7986 \pm 0.4444$ |
| Test MSE | $17.4157 \pm 4.6670$ | $23.1750 \pm 9.0459$ |

The results were averaged over 100 realisations and quoted as the mean± standard deviation.

**Table 3**
Average classification test error rate in % over the 100 realisations of the diabetes data set.

| Algorithm | Test error rate | Model size |
|---|---|---|
| RBF-network | $24.29 \pm 1.88$ | 15 |
| AdaBoost RBF-network | $26.47 \pm 2.29$ | 15 |
| LP-Reg-AdaBoost | $24.11 \pm 1.90$ | 15 |
| QP-Reg-AdaBoost | $25.39 \pm 2.20$ | 15 |
| AdaBoost-Reg | $23.79 \pm 1.80$ | 15 |
| SVM | $23.53 \pm 1.73$ | Not available |
| Kernel Fisher discriminant | $23.21 \pm 1.63$ | 468 |
| OLS-LOO | $23.00 \pm 1.70$ | $6.0 \pm 1.0$ |

The first seven results were quoted from [31].

*Breast cancer data*: This classification benchmark data set was also originated in the UCI repository [30] and the actual data set used in the experiment was obtained from [31]. The feature input space dimension was $m = 9$. There were 100 realisations of this data set, each containing 200 training patterns and 77 test patterns. In [31,43], seven existing state-of-the-art RBF and kernel classifier construction algorithms were compared and the performance averaged over all the 100 realisations were given. For the first five methods studied in [31], the RBF network with five 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, again no average model size was given in [31] but it was certainly larger than, said, 30. The OLS-LOO algorithm was applied to all the 100 realisations of the data set to construct sparse Gaussian kernel classifiers and the results obtained are given in Table 4, in comparison with the benchmark results quoted from [31,43]. From Table 4, it can be seen that the OLS-LOO algorithm compared favourably with these existing state-of-the-art RBF and kernel modelling methods, both in terms of classification accuracy and model size.

### 4.3. Density estimation applications

*Synthetic two-dimensional classification data*: This was a two-class classification problem in a two-dimensional feature space [44]. 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%. We first estimated the two conditional density functions $\hat{p}(\mathbf{x}; \boldsymbol{\beta}_N, \rho|\mathrm{C0})$ and $\hat{p}(\mathbf{x}; \boldsymbol{\beta}_N, \rho|\mathrm{C1})$ from the training data, and then applied the Bayes decision rule

$$\left.\begin{array}{ll} \text{if } \hat{p}(\mathbf{x}; \boldsymbol{\beta}_N, \rho|\mathrm{C0}) \geqslant \hat{p}(\mathbf{x}; \boldsymbol{\beta}_N, \rho|\mathrm{C1}), & \mathbf{x} \text{ belongs to class 0,} \\ \text{else,} & \mathbf{x} \text{ belongs to class 1} \end{array}\right\} \quad (51)$$

to the test data set and calculated the corresponding error rate. Table 5 lists the results obtained by the three kernel density

estimates, the PW estimator, the proposed SKD estimator based on the combined OLS-LOO-LR and MNQP algorithm, and our previous SKD estimator [12], where the value of the kernel width $\rho$ for each kernel density estimate was determined via cross-validation.

The algorithm of [12], 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. Moreover, in the work of [12], 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, which imposes considerable computational cost. It can be seen from Table 5 that the proposed SKD 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. Note that the computational complexity of the proposed SKD estimator was much smaller than the SKD estimator of [12]. Fig. 3(a) and (b) depict the decision boundaries of the classifier (51) for the PW estimate and the SKD estimate obtained by the combined OLS-LOO-LR and MNQP algorithm, respectively.

**Table 4**
Average classification test error rate in % over the 100 realisations of the breast cancer data set.

| Algorithm | Test error rate | Model size |
|---|---|---|
| RBF-network | $27.64 \pm 4.71$ | 5 |
| AdaBoost RBF-network | $30.36 \pm 4.73$ | 5 |
| LP-Reg-AdaBoost | $26.79 \pm 6.08$ | 5 |
| QP-Reg-AdaBoost | $25.91 \pm 4.61$ | 5 |
| AdaBoost-Reg | $26.51 \pm 4.47$ | 5 |
| SVM | $26.04 \pm 4.74$ | Not available |
| Kernel Fisher discriminant | $24.77 \pm 4.63$ | 200 |
| OLS-LOO | $25.74 \pm 5.00$ | $6.0 \pm 2.0$ |

The first seven results were quoted from [31].

*Six-dimensional density estimation*: The underlying density to be estimated was given by

$$p(\mathbf{x}) = \frac{1}{3}\sum_{i=1}^{3} \frac{1}{(2\pi)^{6/2}} \frac{1}{\det^{1/2} |\boldsymbol{\Gamma}_i|} \mathrm{e}^{-(1/2)(\mathbf{x}-\boldsymbol{\mu}_i)^{\mathrm{T}} \boldsymbol{\Gamma}_i^{-1} (\mathbf{x}-\boldsymbol{\mu}_i)} \tag{52}$$

with

$$\boldsymbol{\mu}_1 = [1.0 \;\; 1.0 \;\; 1.0 \;\; 1.0 \;\; 1.0 \;\; 1.0]^{\mathrm{T}},$$
$$\boldsymbol{\Gamma}_1 = \mathrm{diag}\{1.0, 2.0, 1.0, 2.0, 1.0, 2.0\}, \tag{53}$$

$$\boldsymbol{\mu}_2 = [-1.0 \;\; -1.0 \;\; -1.0 \;\; -1.0 \;\; -1.0 \;\; -1.0]^{\mathrm{T}},$$
$$\boldsymbol{\Gamma}_2 = \mathrm{diag}\{2.0, 1.0, 2.0, 1.0, 2.0, 1.0\}, \tag{54}$$

$$\boldsymbol{\mu}_3 = [0.0 \;\; 0.0 \;\; 0.0 \;\; 0.0 \;\; 0.0 \;\; 0.0]^{\mathrm{T}},$$
$$\boldsymbol{\Gamma}_3 = \mathrm{diag}\{2.0, 1.0, 2.0, 1.0, 2.0, 1.0\}. \tag{55}$$

A training data set of $N = 600$ randomly drawn samples was used to construct kernel density estimates, and a separate test data set of $N_{\mathrm{test}} = 10,000$ samples was used to calculate the $L_1$ test error for the resulting estimate according to

$$L_1 = \frac{1}{N_{\mathrm{test}}} \sum_{k=1}^{N_{\mathrm{test}}} |p(\mathbf{x}_k) - \hat{p}(\mathbf{x}_k; \boldsymbol{\beta}_N, \rho)|. \tag{56}$$

The experiment was repeated $N_{\mathrm{run}} = 100$ different random runs. The optimal kernel width was found to be $\rho = 0.65$ for the PW estimate and $\rho = 1.2$ for both the previous SKD algorithm [12] and the combined OLS-LOO-LR and MNQP algorithm, respectively, via cross-validation. The results obtained by the three density

**Table 5**
Performance comparison for the synthetic two-class two-dimensional classification data set.

| Method | $\hat{p}(\bullet\|C0)$ | $\rho$ | $\hat{p}(\bullet\|C1)$ | $\rho$ | Test error rate (%) |
|---|---|---|---|---|---|
| Parzen window estimate | 125 kernels | 0.24 | 125 kernels | 0.23 | 8.0 |
| OLS-LOO-LR/MNQP | 6 kernels | 0.28 | 5 kernels | 0.28 | 8.0 |
| Previous SKD estimate of [12] | 5 kernels | 0.20 | 4 kernels | 0.20 | 8.3 |

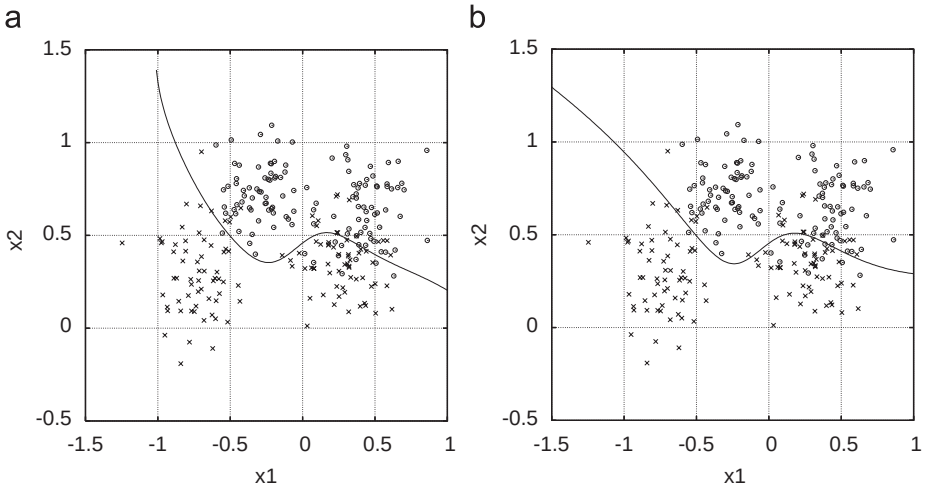


**Fig. 3.** (a) Decision boundary of the Parzen window estimate and (b) decision boundary of the sparse kernel density estimate obtained by the combined OLS-LOO-LR and MNQP algorithm, for the synthetic two-class two-dimensional classification example, where circles represent the class-1 training data and crosses the class-0 training data.

**Table 6**
Performance comparison for the six-dimensional three-Gaussian mixture.

| Method | $L_1$ test error | Kernel number |
|---|---|---|
| Parzen window estimate | $(3.5195 \pm 0.1616) \times 10^{-5}$ | $600 \pm 0$ |
| SKD estimate of [12] | $(4.4781 \pm 1.2292) \times 10^{-5}$ | $14.9 \pm 2.1$ |
| OLS-LOO-LR/MNQP | $(3.1134 \pm 0.5335) \times 10^{-5}$ | $9.4 \pm 1.9$ |

estimators are summarised in Table 6. It can be seen that the proposed combined OLS-LOO-LR and MNQP algorithm yielded sparser kernel density estimates with better test performance.

## 5. Conclusions

A regression framework has been proposed for sparse kernel modelling, which unifies the supervised regression and classification learning problems as well as the unsupervised PDF learning problem. An OLS algorithm has been developed for selecting sparse kernel models that generalise well, based on the LOO test criteria and coupled with local regularisation. For sparse kernel density estimation, a combined approach of the OLS-LOO-LR algorithm and multiplicative nonnegative quadratic programming (MNQP) 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 model to meet the constraints for density estimate. Empirical data-modelling results involving regression, classification and density estimation have been presented to demonstrate the effectiveness of the proposed unified data-modelling framework based on the OLS-LOO-LR algorithm, and the results shown have confirmed that this unified sparse kernel regression framework offers a state-of-the-art for data-modelling applications.

## Appendix A

The modified Gram–Schmidt orthogonalisation procedure [8] calculates the $\mathbf{A}$ matrix row by row and orthogonalises $\mathbf{\Phi}$ as follows: At the $l$-th stage make the columns $\boldsymbol{\phi}_j$, $l+1 \leqslant j \leqslant N$, orthogonal to the $l$-th column and repeat the operation for $1 \leqslant l \leqslant N-1$. Specifically, denoting $\boldsymbol{\phi}_j^{[0]} = \boldsymbol{\phi}_j$, $1 \leqslant j \leqslant N$, then for $l = 1, 2, \ldots, N-1$,

$$\left.\begin{aligned} &\mathbf{w}_l = \boldsymbol{\phi}_l^{[l-1]}, \\ &a_{l,j} = \mathbf{w}_l^{\mathrm{T}} \boldsymbol{\phi}_j^{[l-1]} / (\mathbf{w}_l^{\mathrm{T}} \mathbf{w}_l), \quad l+1 \leqslant j \leqslant N, \\ &\boldsymbol{\phi}_j^{[l]} = \boldsymbol{\phi}_j^{[l-1]} - a_{l,j} \mathbf{w}_l, \qquad l+1 \leqslant j \leqslant N. \end{aligned}\right\} \tag{57}$$

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

$$\left.\begin{aligned} &g_l = \mathbf{w}_l^{\mathrm{T}} \mathbf{y}^{[l-1]} / (\mathbf{w}_l^{\mathrm{T}} \mathbf{w}_l + \lambda_l), \\ &\mathbf{y}^{[l]} = \mathbf{y}^{[l-1]} - g_l \mathbf{w}_l, \end{aligned}\right\} 1 \leqslant l \leqslant N, \tag{58}$$

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

## Appendix B

It can be shown that the log evidence for $\mathbf{h}_N$ and $\omega$ is [34]

$$\begin{aligned} \log(p(\mathbf{y}|\mathbf{h}_N, \omega)) \approx & \sum_{i=1}^{N} \frac{1}{2} \log(h_i) - \frac{N}{2} \log(\pi) + \frac{N}{2} \log(\omega) \\ & - \sum_{i=1}^{N} \frac{1}{2} h_i g_i^2 - \frac{1}{2} \omega \boldsymbol{\varepsilon}^{\mathrm{T}} \boldsymbol{\varepsilon} - \frac{1}{2} \log(\det(\mathbf{B})), \end{aligned} \tag{59}$$

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

$$\begin{aligned} \mathbf{B} &= \mathbf{H} + \omega \mathbf{W}^{\mathrm{T}} \mathbf{W} \\ &= \mathrm{diag}\{h_1 + \omega \mathbf{w}_1^{\mathrm{T}} \mathbf{w}_1, h_2 + \omega \mathbf{w}_2^{\mathrm{T}} \mathbf{w}_2, \ldots, h_N + \omega \mathbf{w}_N^{\mathrm{T}} \mathbf{w}_N\}. \end{aligned} \tag{60}$$

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

$$\omega \boldsymbol{\varepsilon}^{\mathrm{T}} \boldsymbol{\varepsilon} = N - \sum_{i=1}^{N} \frac{\omega \mathbf{w}_i^{\mathrm{T}} \mathbf{w}_i}{h_i + \omega \mathbf{w}_i^{\mathrm{T}} \mathbf{w}_i}. \tag{61}$$

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

$$h_i = \frac{\omega \mathbf{w}_i^{\mathrm{T}} \mathbf{w}_i}{g_i^2 (h_i + \omega \mathbf{w}_i^{\mathrm{T}} \mathbf{w}_i)}. \tag{62}$$

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

$$\gamma = \sum_{i=1}^{N} \gamma_i \tag{63}$$

with

$$\gamma_i = \frac{\omega \mathbf{w}_i^{\mathrm{T}} \mathbf{w}_i}{h_i + \omega \mathbf{w}_i^{\mathrm{T}} \mathbf{w}_i} = \frac{\mathbf{w}_i^{\mathrm{T}} \mathbf{w}_i}{\lambda_i + \mathbf{w}_i^{\mathrm{T}} \mathbf{w}_i}. \tag{64}$$

Then the recalculation formula for $\lambda_i$ is

$$\lambda_i = \frac{\gamma_i}{N - \gamma} \frac{\boldsymbol{\varepsilon}^{\mathrm{T}} \boldsymbol{\varepsilon}}{g_i^2}, \quad 1 \leqslant i \leqslant N. \tag{65}$$

## Appendix C

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

$$\mathbf{g}_N = (\mathbf{W}^{\mathrm{T}} \mathbf{W} + \mathbf{\Lambda})^{-1} \mathbf{W}^{\mathrm{T}} \mathbf{y} = \tilde{\mathbf{B}}^{-1} \mathbf{W}^{\mathrm{T}} \mathbf{y}, \tag{66}$$

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

$$\varepsilon_k = \varepsilon_k^{(N)} = y_k - \mathbf{g}_N^{\mathrm{T}} \mathbf{w}(k) = y_k - \mathbf{y}^{\mathrm{T}} \mathbf{W} \tilde{\mathbf{B}}^{-1} \mathbf{w}(k). \tag{67}$$

Let the $k$-th data sample be deleted from the training set $D_N$, and the resulting LOO training set is used to estimate the model parameter vector. The corresponding regularised LS solution is defined by

$$\mathbf{g}_N^{(N,-k)} = (\tilde{\mathbf{B}}^{(N,-k)})^{-1} (\mathbf{W}^{(N,-k)})^{\mathrm{T}} \mathbf{y}^{(N,-k)}, \tag{68}$$

where $\tilde{\mathbf{B}}^{(N,-k)} = (\mathbf{W}^{(N,-k)})^{\mathrm{T}} \mathbf{W}^{(N,-k)} + \mathbf{\Lambda}$, $\mathbf{W}^{(N,-k)}$ and $\mathbf{y}^{(N,-k)}$ 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

$$\hat{y}_k^{(N,-k)} = (\mathbf{g}_N^{(N,-k)})^{\mathrm{T}} \mathbf{w}(k). \tag{69}$$

By definition, it can be shown that

$$\tilde{\mathbf{B}}^{(N,-k)} = \tilde{\mathbf{B}} - \mathbf{w}(k) \mathbf{w}^{\mathrm{T}}(k), \tag{70}$$

$$(\mathbf{y}^{(N,-k)})^{\mathrm{T}} \mathbf{W}^{(N,-k)} = \mathbf{y}^{\mathrm{T}} \mathbf{W} - y_k \mathbf{w}^{\mathrm{T}}(k). \tag{71}$$

The LOO test error evaluated at the $k$-th data sample not used for training, denoted as $\varepsilon_k^{(N,-k)} = y_k - \hat{y}_k^{(N,-k)}$, is given by

$$\begin{aligned} \varepsilon_k^{(N,-k)} &= y_k - (\mathbf{g}_N^{(N,-k)})^{\mathrm{T}} \mathbf{w}(k) \\ &= y_k - (\mathbf{y}^{(N,-k)})^{\mathrm{T}} \mathbf{W}^{(N,-k)} (\tilde{\mathbf{B}}^{(N,-k)})^{-1} \mathbf{w}(k). \end{aligned} \tag{72}$$

Applying the matrix inversion lemma to (70) yields

$$(\tilde{\mathbf{B}}^{(N,-k)})^{-1} = \tilde{\mathbf{B}}^{-1} + \frac{\tilde{\mathbf{B}}^{-1}\mathbf{w}(k)\mathbf{w}^{\mathrm{T}}(k)\tilde{\mathbf{B}}^{-1}}{1-\mathbf{w}^{\mathrm{T}}(k)\tilde{\mathbf{B}}^{-1}\mathbf{w}(k)} \tag{73}$$

and

$$(\tilde{\mathbf{B}}^{(N,-k)})^{-1}\mathbf{w}(k) = \frac{\tilde{\mathbf{B}}^{-1}\mathbf{w}(k)}{1-\mathbf{w}^{\mathrm{T}}(k)\tilde{\mathbf{B}}^{-1}\mathbf{w}(k)}. \tag{74}$$

Substituting (71) and (74) into (72) results in

$$\begin{aligned}\varepsilon_k^{(N,-k)} &= y_k - \frac{(\mathbf{y}^{\mathrm{T}}\mathbf{W} - y_k\mathbf{w}^{\mathrm{T}}(k))\tilde{\mathbf{B}}^{-1}\mathbf{w}(k)}{1-\mathbf{w}^{\mathrm{T}}(k)\tilde{\mathbf{B}}^{-1}\mathbf{w}(k)} \\ &= \frac{y_k - \mathbf{y}^{\mathrm{T}}\mathbf{W}\tilde{\mathbf{B}}^{-1}\mathbf{w}(k)}{1-\mathbf{w}^{\mathrm{T}}(k)\tilde{\mathbf{B}}^{-1}\mathbf{w}(k)} = \frac{y_k - \hat{y}_k^{(N)}}{1-\mathbf{w}^{\mathrm{T}}(k)\tilde{\mathbf{B}}^{-1}\mathbf{w}(k)} \\ &= \frac{\varepsilon_k^{(N)}}{1-\mathbf{w}^{\mathrm{T}}(k)\tilde{\mathbf{B}}^{-1}\mathbf{w}(k)} = \frac{\varepsilon_k^{(N)}}{\eta_k^{(N)}},\end{aligned} \tag{75}$$

where the $N$-term model output $\hat{y}_k^{(N)} = \mathbf{g}_N^{\mathrm{T}}\mathbf{w}(k) = \mathbf{y}^{\mathrm{T}}\mathbf{W}\tilde{\mathbf{B}}^{-1}\mathbf{w}(k)$, the $N$-term modelling error

$$\varepsilon_k^{(N)} = y_k - \sum_{i=1}^{N} w_{k,i}g_i \tag{76}$$

and the associated LOO error weighting

$$\begin{aligned}\eta_k^{(N)} &= 1 - \mathbf{w}^{\mathrm{T}}(k)(\mathbf{W}^{\mathrm{T}}\mathbf{W}+\mathbf{\Lambda})^{-1}\mathbf{w}(k) \\ &= 1 - \sum_{i=1}^{N}\frac{w_{k,i}^2}{\mathbf{w}_i^{\mathrm{T}}\mathbf{w}_i+\lambda_i}.\end{aligned} \tag{77}$$

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

$$\mathbf{W}^{(n)} = [\mathbf{w}_1 \ \ \mathbf{w}_2 \ \ \cdots \ \ \mathbf{w}_n] \tag{78}$$

and the $k$-th row of $\mathbf{W}^{(n)}$ as

$$(\mathbf{w}^{(n)}(k))^{\mathrm{T}} = [w_{k,1} \ \ w_{k,2} \ \ \cdots \ \ w_{k,n}]. \tag{79}$$

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

$$\begin{aligned}\varepsilon_k^{(n)} &= y_k - \sum_{i=1}^{n} w_{k,i}g_i = \left(y_k - \sum_{i=1}^{n-1} w_{k,i}g_i\right) - w_{k,n}g_n \\ &= \varepsilon_k^{(n-1)} - w_{k,n}g_n\end{aligned} \tag{80}$$

and

$$\begin{aligned}\eta_k^{(n)} &= 1 - \sum_{i=1}^{n}\frac{w_{k,i}^2}{\mathbf{w}_i^{\mathrm{T}}\mathbf{w}_i+\lambda_i} \\ &= \left(1 - \sum_{i=1}^{n-1}\frac{w_{k,i}^2}{\mathbf{w}_i^{\mathrm{T}}\mathbf{w}_i+\lambda_i}\right) - \frac{w_{k,n}^2}{\mathbf{w}_n^{\mathrm{T}}\mathbf{w}_n+\lambda_i} \\ &= \eta_k^{(n-1)} - \frac{w_{k,n}^2}{\mathbf{w}_n^{\mathrm{T}}\mathbf{w}_n+\lambda_i}\end{aligned} \tag{81}$$

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

$$\varepsilon_k^{(n,-k)} = y_k - \hat{y}_k^{(n,-k)} = \frac{y_k - \hat{y}_k^{(n)}}{1-\sum_{i=1}^{n}\frac{w_{k,i}^2}{\mathbf{w}_i^{\mathrm{T}}\mathbf{w}_i+\lambda_i}} = \frac{\varepsilon_k^{(n)}}{\eta_k^{(n)}}, \tag{82}$$

where $\hat{y}_k^{(n)} = \sum_{i=1}^{n} w_{k,i}g_i$ is the $n$-term model output.

## 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

$$\mathbf{\Phi}^{[l-1]} = [\mathbf{w}_1 \ \ \cdots \ \ \mathbf{w}_{l-1} \ \ \boldsymbol{\phi}_l^{[l-1]} \ \ \cdots \ \ \boldsymbol{\phi}_N^{[l-1]}]. \tag{83}$$

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 (31), the $l$-th stage of the selection procedure is given as follows.

*Step* 1: For $l \leqslant j \leqslant N$:

- **Test**—Conditioning number check. If $(\boldsymbol{\phi}_j^{[l-1]})^{\mathrm{T}}\boldsymbol{\phi}_j^{[l-1]} < T_z$, the $j$-th candidate is not considered.
- Compute

$$g_l^{\{j\}} = \frac{(\boldsymbol{\phi}_j^{[l-1]})^{\mathrm{T}}\mathbf{y}^{[l-1]}}{((\boldsymbol{\phi}_j^{[l-1]})^{\mathrm{T}}\boldsymbol{\phi}_j^{[l-1]}+\lambda_j)}, \tag{84}$$

$$\left.\begin{aligned}\varepsilon_k^{(l)\{j\}} &= y_k^{[l-1]} - \phi_j^{[l-1]}(k)g_l^{\{j\}}, \\ \eta_k^{(l)\{j\}} &= \eta_k^{(l-1)} - \frac{(\phi_j^{[l-1]}(k))^2}{(\boldsymbol{\phi}_j^{[l-1]})^{\mathrm{T}}\boldsymbol{\phi}_j^{[l-1]}+\lambda_j},\end{aligned}\right\} \quad 1 \leqslant k \leqslant N, \tag{85}$$

$$J_l^{\{j\}} = \frac{1}{N}\sum_{k=1}^{N}\left(\frac{\varepsilon_k^{(l)\{j\}}}{\eta_k^{(l)\{j\}}}\right)^2, \tag{86}$$

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

$$\mathscr{J}_l = \{l \leqslant j \leqslant N \text{ and } j \text{ passes } \mathbf{Test}\}. \tag{87}$$

*Step* 2: Find

$$J_l = J_l^{\{j_l\}} = \min\{J_l^{\{j\}}, j \in \mathscr{J}_l\}. \tag{88}$$

Then the $j_l$-th column of $\mathbf{\Phi}^{[l-1]}$ is interchanged with the $l$-th column of $\mathbf{\Phi}^{[l-1]}$, the $j_l$-th column of $\mathbf{A}$ is interchanged with the $l$-th column of $\mathbf{A}$ up to the $(l-1)$-th row, and the $j_l$-th element of $\boldsymbol{\lambda}_N$ is interchanged with the $l$-th element of $\boldsymbol{\lambda}_N$. 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 $J_l \geqslant J_{l-1}$. Otherwise, perform the orthogonalisation as indicated in (57) to derive the $l$-th row of $\mathbf{A}$ and to transform $\mathbf{\Phi}^{[l-1]}$ into $\mathbf{\Phi}^{[l]}$; let $g_l = g_l^{\{j_l\}}$ and update $\mathbf{y}^{[l-1]}$ into $\mathbf{y}^{[l]}$ in the way shown in (58); update the LOO error weightings

$$\eta_k^{(l)} = \eta_k^{(l-1)} - \frac{w_{k,l}^2}{\mathbf{w}_l^{\mathrm{T}}\mathbf{w}_l+\lambda_l}, \quad 1 \leqslant k \leqslant N \tag{89}$$

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 (38), and all $\lambda_i$ are fixed to the constant $\lambda$. In *Step* 1, the calculation of the candidates' LOO MSE (85) and (86) is replaced by the

following candidates' LOO misclassification rate

$$\left.\begin{aligned}\psi_k^{(l)\{j\}} &= \psi_k^{(l-1)} + y_k g_l^{\{j\}} \phi_j^{[l-1]}(k) - \frac{(\phi_j^{[l-1]}(k))^2}{(\boldsymbol{\phi}_j^{[l-1]})^{\mathrm{T}} \boldsymbol{\phi}_j^{[l-1]} + \lambda},\\ \eta_k^{(l)\{j\}} &= \eta_k^{(l-1)} - \frac{(\phi_j^{[l-1]}(k))^2}{(\boldsymbol{\phi}_j^{[l-1]})^{\mathrm{T}} \boldsymbol{\phi}_j^{[l-1]} + \lambda},\\ s_k^{(l,-k)\{j\}} &= \frac{\psi_k^{(l)\{j\}}}{\eta_k^{(l)\{j\}}},\end{aligned}\right\} \quad 1 \leqslant k \leqslant N, \tag{90}$$

$$J_l^{\{j\}} = \frac{1}{N} \sum_{k=1}^{N} \mathscr{J}_d(s_k^{(l,-k)\{j\}}). \tag{91}$$

In *Step* 3, in addition to update $\eta_k^{(l-1)}$ according to (89), $\psi_k^{(l-1)}$ are also updated according to

$$\psi_k^{(l)} = \psi_k^{(l-1)} + y_k g_l w_{k,l} - \frac{w_{k,l}^2}{\mathbf{w}_l^{\mathrm{T}} \mathbf{w}_l + \lambda}, \quad 1 \leqslant k \leqslant N. \tag{92}$$

## References


[1] S.A. Billings, 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, vol. 7, Academic Press, San Diego, CA, 1998, pp. 231–278.

[2] S.A. Billings, S. Chen, R.J. Backhouse, The identification of linear and non-linear models of a turbocharged automotive diesel engine, Mech. Syst. Signal Process. 3 (20) (1989) 123–142.

[3] C.M. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, Oxford, UK, 1995.

[4] A.W. Bowman, A. Azzalini, Applied Smoothing Techniques for Data Analysis, Oxford University Press, Oxford, UK, 1997.

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

[6] O. Chapelle, V. Vapnik, O. Bousquet, S. Mukherjee, Choosing multiple parameters for support vector machines, Machine Learning 46 (1–3) (2002) 131–159.

[7] S. Chen, Local regularization assisted orthogonal least squares regression, Neurocomputing 69 (4–6) (2006) 559–585.

[8] S. Chen, S.A. Billings, W. Luo, Orthogonal least squares methods and their application to non-linear system identification, Int. J. Control 50 (5) (1989) 1873–1896.

[9] S. Chen, E.S. Chng, K. Alkadhimi, Regularised orthogonal least squares algorithm for constructing radial basis function networks, Int. J. Control 64 (5) (1996) 829–837.

[10] S. Chen, C.F.N. Cowan, P.M. Grant, Orthogonal least squares learning algorithm for radial basis function networks, IEEE Trans. Neural Networks 2 (2) (1991) 302–309.

[11] S. Chen, X. Hong, C.J. Harris, Sparse kernel regression modelling using combined locally regularized orthogonal least squares and D-optimality experimental design, IEEE Trans. Autom. Control 48 (6) (2003) 1029–1036.

[12] S. Chen, X. Hong, C.J. Harris, Sparse kernel density construction using orthogonal forward regression with leave-one-out test score and local regularization, IEEE Trans. Syst Man Cybern Part B 34 (4) (2004) 1708–1717.

[13] S. Chen, X. Hong, C.J. Harris, An orthogonal forward regression technique for sparse kernel density estimation, Neurocomputing 71 (4–6) (2008) 931–943.

[14] S. Chen, X. Hong, C.J. Harris, P.M. Sharkey, Sparse modelling using orthogonal forward regression with PRESS statistic and regularization, IEEE Trans. Systems Man Cybern. Part B 34 (2) (2004) 898–911.

[15] S. Chen, J. Wigger, Fast orthogonal least squares algorithm for efficient subset model selection, IEEE Trans. Signal Process. 43 (7) (1995) 1713–1715.

[16] S. Chen, Y. Wu, B.L. Luk, Combined genetic algorithm optimisation and regularised orthogonal least squares learning for radial basis function networks, IEEE Trans. Neural Networks 10 (5) (1999) 1239–1243.

[17] S.S. Chen, D.L. Donoho, M.A. Saunders, Atomic decomposition by basis pursuit, SIAM Rev. 43 (1) (2001) 129–159.

[18] A. Choudhury, Fast machine learning algorithms for large data, Ph.D. Thesis, Computational Engineering and Design Center, School of Engineering Sciences, University of Southampton, UK, 2002.

[19] N. Cristianini, J. Shawe-Taylor, An Introduction to Support Vector Machines: And Other Kernel-Based Learning Methods, Cambridge University Press, Cambridge, UK, 2000.

[20] K. Duan, S.S. Keerthi, A.N. Poo, Evaluation of simple performance measures for tuning SVM hyperparameters, Neurocomputing 51 (2003) 41–59.

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

[22] M. Girolami, C. He, Probability density estimation from optimally condensed data samples, IEEE Trans. Pattern Anal. Mach. Intell. 25 (10) (2003) 1253–1264.

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

[24] L.K. Hansen, J. Larsen, Linear unlearning for cross-validation, Adv. Comput. Math. 5 (1996) 269–280.

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

[26] X. Hong, S. Chen, C.J. Harris, A fast linear-in-the-parameters classifier construction algorithm using orthogonal forward selection to minimize leave-one-out misclassification rate, Int. J. Syst. Sci. 39 (2) (2008) 119–125.

[27] X. Hong, S. Chen, C.J. Harris, A forward-constrained regression algorithm for sparse kernel density estimation, IEEE Trans. Neural Networks 19 (1) (2008) 193–198.

[28] X. Hong, R.J. Mitchell, S. Chen, C.J. Harris, K. Li, G.W. Irwin, Model selection approaches for non-linear system identification: a review, Int. J. Syst. Sci. 39 (10) (2008) 925–946.

[29] X. Hong, P.M. Sharkey, K. Warwick, Automatic nonlinear predictive model construction algorithm using forward regression and the PRESS statistic, IEE Proc. Control Theory Appl. 150 (3) (2003) 245–254.

[30] ⟨http://www.ics.uci.edu/~mlearn/MLRepository.html⟩

[31] ⟨http://ida.first.fhg.de/projects/bench/benchmarks.htm⟩

[32] G.R.G. Lanckriet, N. Cristianini, P. Bartlett, L.E. Ghaoui, M.I. Jordan, Learning the kernel matrix with semidefinite programming, J. Mach. Learn. Res. 5 (2004) 27–72.

[33] K. Li, J. Peng, E.-W. Bai, Two-stage mixed discrete-continuous identification of radial basis function (RBF) neural models for nonlinear systems, IEEE Trans. Circuits Syst. (2008), to appear.

[34] D.J.C. MacKay, Bayesian interpretation, Neural Comput. 4 (3) (1992) 415–447.

[35] G. McLachlan, D. Peel, Finite Mixture Models, Wiley, New York, 2000.

[36] G. Monari, G. Dreyfus, Withdrawing an example from the training set: an analytic estimation of its effect on a non-linear parameterised model, Neurocomputing 35 (2000) 195–201.

[37] G. Monari, G. Dreyfus, Local overfitting control via leverages, Neural Comput. 14 (2002) 1481–1506.

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

[39] R.H. Myers, Classical and Modern Regression with Applications, second ed., PWS Pub. Co., Boston, MA, 1990.

[40] C.S. Ong, A.J. Smola, R.C. Williamson, Hyperkernels, in: Neural Information Processing Systems, vol. 15, MIT Press, Cambridge, MA, 2002.

[41] E. Parzen, On estimation of a probability density function and mode, Ann. Math. Stat. 33 (1962) 1066–1076.

[42] J. Peng, K. Li, D.S. Huang, A hybrid forward algorithm for RBF neural network construction, IEEE Trans. Neural Networks 17 (6) (2006) 1439–1451.

[43] G. Rätsch, T. Onoda, K.R. Müller, Soft margins for AdaBoost, Mach. Learn. 42 (3) (2001) 287–320.

[44] B.D. Ripley, Pattern Recognition and Neural Networks, Cambridge University Press, Cambridge, UK, 1996.

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

[46] B. Schölkopf, A.J. Smola, R.C. Williamson, P.L. Bartlett, New support vector algorithms, Neural Comput. 12 (5) (2000) 1207–1245.

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

[48] B.W. Silverman, Density Estimation for Statistics and Data Analysis, Chapman & Hall, London, 1986.

[49] M. Stone, Cross validation choice and assessment of statistical predictions, J. R. Stat. Soc. Ser. B 36 (1974) 117–147.

[50] J.A.K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, J. Vandewalle, Least Squares Support Vector Machines, World Scientific Publishing Co., Singapore, 2002.

[51] M.E. Tipping, Sparse Bayesian learning and the relevance vector machine, J. Mach. Learn. Res. 1 (2001) 211–244.

[52] V. Vapnik, The Nature of Statistical Learning Theory, Springer, New York, 1995.

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

[54] P. Vincent, Y. Bengio, Kernel matching pursuit, Mach. Learn. 48 (1) (2002) 165–187.

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

**Sheng Chen** received his B.Eng. degree from Huadong Petroleum Institute, Dongying, China, in January 1982 and Ph.D. degree from the City University, London, UK, in September 1986, both in control engineering. He was awarded the Doctor of Sciences (D.Sc.) degree by the University of Southampton, Southampton, UK, in 2005.

He joined the School of Electronics and Computer Science, the University of Southampton, in September 1999. He previously held research and academic appointments at the University of Sheffield, Sheffield, the University of Edinburgh, Edinburgh, and the University of Portsmouth, Portsmouth, all in UK. Professor Chen's research interests include wireless communications, machine learning and neural networks, finite-precision digital controller design, and evolutionary computation methods. He has published over 350 research papers.

In the database of the world's most highly cited researchers, compiled by Institute for Scientific Information (ISI) of the USA, Dr. Chen is on the list of the highly cited researchers in the engineering category.

**Xia Hong** received her university education at National University of Defence Technology, China (B.Sc., 1984, M.Sc., 1987), and University of Sheffield, UK (Ph.D., 1998), all in automatic control.

She worked as a research assistant in Beijing Institute of Systems Engineering, Beijing, China from 1987 to 1993. She worked as a research fellow in the Department of Electronics and Computer Science at the University of Southampton from 1997 to 2001. She is currently a lecturer at the School of Systems Engineering, the University of Reading, UK. She is actively engaged in research into nonlinear systems identification, data modelling, estimation and intelligent control, neural networks, pattern recognition, learning theory and their applications. She has published over 100 research papers, and coauthored a research book.

Dr. Hong was awarded a Donald Julius Groen Prize by IMechE in 1999.

**Bing L. Luk** received his B.Sc. Degree in Electrical and Electronic Engineering from Portsmouth Polytechnic, UK, in 1985, M.Sc. Degree in Digital Computer Systems from Brunel University, UK, in 1986 and Ph.D. Degree in Robotics from the University of Portsmouth, UK, in 1991.

He joined the Department of Manufacturing Engineering and Engineering Management at City University of Hong Kong, Hong Kong, China, in 2000. He previously held research and academic appointments at the University of Portsmouth, UK and engineering consultant position at Portsmouth Technology Consultant Ltd. and also other industrial companies. His recent research works include mobile robotics, telemedicine research, nondestructive test methods, machine learning and evolutionary computation methods.

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

He previously held appointments at the University of Hull, Hull, the UMIST, Manchester, the University of Oxford, Oxford, and the University of Cranfield, Cranfield, all in UK, as well as being employed by the UK Ministry of Defence. He returned to the University of Southampton as the Lucas Professor of Aerospace Systems Engineering in 1987 to establish the Advanced Systems Research Group and, later, Image, Speech and Intelligent Systems Group. His research interests lie in the general area of intelligent and adaptive systems theory and its application to intelligent autonomous systems such as autonomous vehicles, management infrastructures such as command and control, intelligent control, and estimation of dynamic processes, multi-sensor data fusion, and systems integration. He has authored and co-authored 12 research books and over 400 research papers, and he is the associate editor of numerous international journals.

Dr. Harris was elected to Fellow of the Royal Academy of Engineering in 1996, was awarded the IEE Senior Achievement medal in 1998 for his work in autonomous systems, and the highest international award in IEE, the IEE Faraday medal, in 2001 for his work in intelligent control and neurofuzzy systems.