Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 602341, 10 pages
http://dx.doi.org/10.1155/2013/602341

*Research Article*

# A Novel Sparse Least Squares Support Vector Machines

**Xiao-Lei Xia,[1] Weidong Jiao,[1,2] Kang Li,[3] and George Irwin[3]**

[1] *School of Mechanical and Electrical Engineering, Jiaxing University, Jiaxing 314001, China*
[2] *School of Engineering, Zhejiang Normal University, Jinhua 321004, China*
[3] *School of Electronics, Electrical Engineering and Computer Science, Queen's University of Belfast, Belfast BT9 5AH, UK*

Correspondence should be addressed to Xiao-Lei Xia; xxia01@qub.ac.uk

Received 9 August 2012; Accepted 6 December 2012

Academic Editor: Huaguang Zhang

The solution of a Least Squares Support Vector Machine (LS-SVM) suffers from the problem of nonsparseness. The Forward Least Squares Approximation (FLSA) is a greedy approximation algorithm with a least-squares loss function. This paper proposes a new Support Vector Machine for which the FLSA is the training algorithm—the Forward Least Squares Approximation SVM (FLSA-SVM). A major novelty of this new FLSA-SVM is that the number of support vectors is the regularization parameter for tuning the tradeoff between the generalization ability and the training cost. The FLSA-SVMs can also detect the linear dependencies in vectors of the input Gramian matrix. These attributes together contribute to its extreme sparseness. Experiments on benchmark datasets are presented which show that, compared to various SVM algorithms, the FLSA-SVM is extremely compact, while maintaining a competitive generalization ability.

## 1. Introduction

The last decade has seen widespread applications of Least Squares Support Vector Machines (LS-SVM) [1, 2] to a variety of classification problems. The LS-SVM involves finding a separating hyperplane of maximal margin and minimizing the empirical risk via a Least Squares loss function. Here the optimization is subject to equality constraints, as opposed to the inequality ones used with a standard SVM [3, 4]. Thus the LS-SVM successfully sidesteps the quadratic programming (QP) required for the training of the standard SVM. As a result, an LS-SVM classifier is equivalent to a set of linear algebraic equations, which are usually solved by the conjugate gradient (CG) method [5]. Empirical studies suggest that an LS-SVM possesses very competitive generalization abilities compared to a regular SVM [6]; improvement has been made to Suyken's algorithm by Chu et al. [7] whose proposal is also based on CG method but with a reduced time complexity. Meanwhile Keerthi et al. advocated training an LS-SVM using the sequential minimal optimization (SMO) algorithm [8].

While these algorithms have indeed made an LS-SVM more computationally attractive, the nonsparseness of its solution still remains a major, and as yet unsolved, bottleneck. Sparseness in solutions is essential for reducing the time required in predicting the class membership of unlabelled data. The general approach to addressing this issue for an LS-SVM is iterative shrinking of the training set. Here the importance of training samples is evaluated from the weights assigned to them after training. Samples of less importance are removed, and then the remainder forms a reduced training set to be learnt again. In view of the linear relationship between the approximation error and the support value for a training sample, a straightforward pruning strategy is to remove samples whose absolute support values are trivial [9]. A later paper recommended pruning a sample which introduces the smallest error if omitted [10]. However, this necessitates the inversion of the kernel matrix, which is indicative of high computation complexity. Zeng and Chen [11] suggested a pruning method based on the SMO formulation of an LS-SVM, which leads to faster retraining. Other authors [12] proposed solving the dual form of an LS-SVM by iterative addition of basis functions from the kernel dictionary either until all the training samples are traversed or until the approximation error is lower than a preset threshold. Despite the lower computation cost with the backfitting scheme adopted, Lagrangian multipliers associated with previously selected training samples were inherited into the next pass, which could compromise the generalization abilities of the resultant LS-SVM.

This paper presents a new LS-SVM formulation called the Forward Least Squares Approximation SVMs (FLSA-SVM). It is trained by Forward Least Squares Approximation(FLSA) [13], a function approximation method using a Least Squares loss function. The FLSA-SVM loss function decreases monotonically with an increasing number of support vectors, allowing for the removal of slack variables from the equality constraints. Another novelty with the FLSA-SVM lies in the fact that it cleverly transforms the number of support vectors to be the regularization parameter, which indicates the tradeoff between generalization abilities and empirical risk. The FLSA-SVM builds a classifier by iteratively selecting a single basis function for the solution, which contributes the largest reduction to the quadratic cost function.

The paper is organized as follows. Section 2 briefly reviews Least Squares SVM principles. The new sparse LS-SVM—Forward Least Squares Approximation SVM (FLSA-SVM)—is introduced in Section 3. Experimental results are given in Section 4 and concluding remarks in Section 5.

## 2. Least Squares Support Vector Machines

For a classification problem of $\ell$ pairs of training samples $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_\ell, y_\ell)$, where $\mathbf{x} \in \mathbb{R}^N$ and $y \in \{-1, 1\}$, LS-SVM algorithms seek the optimal separating hyperplane with the orientation vector of the least $L_2$ norm.

To ensure the presence of the optimal hyperplane, the input data are translated into a reproducing kernel hilbert space (RKHS) by a mapping function denoted by $\phi(\mathbf{x})$. To avoid the curse of dimensionality, the mapping is implemented implicitly by the introduction of "kernel trick." Its idea is that dot products in the RKHS space can be represented by a Mercer kernel function in input space [14]:

$$K\left(\mathbf{x}, \mathbf{x}'\right) = \Phi(\mathbf{x})^\top \Phi\left(\mathbf{x}'\right). \tag{1}$$

Thus the linear discriminant function in the feature space can be formulated as

$$f(\mathbf{x}) = \mathbf{w}^\top \phi(\mathbf{x}) + b, \tag{2}$$

where $\mathbf{w}$ is the orientation vector and $b$ is the bias term.

An LS-SVM finds the hyperplane parameterized by $(\mathbf{w}, b)$ by solving the following optimization problem [15]:

$$\begin{aligned} &\min_{\mathbf{w},b,\mathbf{e}} \quad \frac{1}{2}\mathbf{w}^\top\mathbf{w} + \frac{1}{2}\gamma\sum_{i=1}^{\ell} e_i^2, \\ &\text{s.t.} \quad \mathbf{w}^\top\phi(\mathbf{x}_i) + b = y_i - e_i, \quad i = 1, \dots, \ell, \end{aligned} \tag{3}$$

where the slack variable $e_i$ denotes the deviation between the actual output $f(\mathbf{x})$ of the LS-SVM on sample $\mathbf{x}_i$ and its target value $y_i$. $\gamma$ is a parameter which imposes penalty on deviations.

Introducing the Lagrange multipliers $\alpha_i$ $(i = 1, \dots, l)$ for each of the equality constraints gives

$$\begin{aligned} \mathscr{L}(\mathbf{w}, b, \mathbf{e}, \alpha) = {} & \frac{1}{2}\mathbf{w}^\top\mathbf{w} + \frac{1}{2}\gamma\sum_{i=1}^{\ell} e_i^2 \\ & - \sum_{i=1}^{\ell} \alpha_i \left[\mathbf{w}^\top\phi(\mathbf{x}_i) + b - y_i + e_i\right]. \end{aligned} \tag{4}$$

Due to the equality constraints, $\alpha_i$ can either be positive or negative according to the Karush-Kuhn-Tucker (KKT) conditions [16]:

$$\mathbf{w} = \sum_{i=1}^{\ell} \alpha_i \phi(\mathbf{x}_i), \tag{5}$$

$$\sum_{i=1}^{\ell} \alpha_i = 0, \tag{6}$$

$$\alpha_i = \gamma e_i, \tag{7}$$

$$\mathbf{w}^\top\phi(\mathbf{x}_i) + b = y_i - e_i. \tag{8}$$

The linear equations can be further simplified to

$$\begin{bmatrix} \mathbf{H} + \gamma^{-1}\mathbf{I} & \overrightarrow{\mathbf{1}} \\ \overrightarrow{\mathbf{1}}^\top & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ b \end{bmatrix} = \begin{bmatrix} \mathbf{y} \\ 0 \end{bmatrix}, \tag{9}$$

where $\mathbf{H} \in \mathbb{R}^{\ell\times\ell}$ and $\mathbf{H}_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$, $\mathbf{y} = [y_1, \dots, y_\ell]^\top$, $\boldsymbol{\alpha} = [\alpha_1, \dots, \alpha_\ell]^\top$, $\mathbf{I}$ is unity matrix of rank $n$, and $\overrightarrow{\mathbf{1}}$ is a column vector of 1s of $n$ length. The solution of an LS-SVM is a linear combination of basis function $K(\cdot, \mathbf{x}_i)$ whose associated $\alpha_i$ is nonzero:

$$f(\mathbf{x}) = \sum_i \alpha_i K(\mathbf{x}, \mathbf{x}_i) + b. \tag{10}$$

It has been noted that the LS-SVM is almost equivalent to ridge regression [17] since the two methods corresponds to an identical optimization problem. Meanwhile, the equivalence between the LS-SVM and the Kernel Fisher Discriminant method [18] has also been established [19].

## 3. Least Squares Approximation Sparse SVM

Equation (3) shows that an LS-SVM can also be viewed as a ridge regression model. And the optimality conditions (7) indicate that the introduction of slack variable $e_i$ is the root of the nonsparseness problem, since in regression applications, most slack variables end as nonzero [20]. But the introduction of $e_i$ seems inevitable for the representation of training cost and thus the penalty parameter to indicate the tradeoff between training cost and generalization abilities.

The section introduces a new formulation of Least Squares SVM, namely, Forward Least Squares Approximation SVM (FLSA-SVM). The proposed FLSA-SVM algorithm is motivated by the Least Squares "Forward Approximation & Backward Refinement" method for function approximation, which was first developed in the dynamic system identification community by Li et al.

FLSA-SVM is rid of slack variables by employing an approximation function of Least Squares loss function which is Forward Least Squares Approximation (FLSA) algorithm. FLSA approximation function iteratively selects a basis function which causes the steepest reduction in the loss function. The features enable the number of support vectors (SVs), which equals to the number of basis functions, to be the tradeoff between training cost and generalization abilities. In FLSA-SVMs, the sparseness of its solution is ensured and confirmed by the experiments section in which FLSA-SVMs are more sparse than standard SVMs.

In this section, a description of "Forward Approximation" is given, from a machine learning perspective, as a preface to the introduction of LSA-SVM algorithm. The strategy to reduce the computation complexity of FLSA-SVMs is then presented.

*3.1. Forward Least Squares Approximation [13].* Given $\ell$ values $(y_1, \dots, y_\ell)$ of an unknown target function $f$ in a Hilbert space at $\ell$ input data $(\mathbf{x}_1, \dots, \mathbf{x}_l)$. A "dictionary" $\{\varphi_1, \dots, \varphi_N\}$ of $N$ functions in the Hilbert space is also given in which $\varphi_i$ $(i = 1, \dots, N)$ is termed as a "basis function." Forward Least Squares Approximation (FLSA) addresses the estimation of $f$ with a linear combinations of $m$ basis functions chosen from the dictionary:

$$f_m = \sum_{i=1}^{m} \theta_i \phi_i, \tag{11}$$

where

$m$ is the number of basis functions which expand $f_m$,

$\{\phi_1, \dots, \phi_m\}$ is the set of selected basis functions, and

$(\theta_1, \dots, \theta_m) \in \mathbb{R}^m$ is the associated weight vector,

so that the squared norm of the residue vector, denoted by $L$, is minimized:

$$L = \left(\overrightarrow{f_m} - \mathbf{y}\right)^\top \left(\overrightarrow{f_m} - \mathbf{y}\right), \tag{12}$$

where $\mathbf{y} = (y_1, \dots, y_\ell)$ and $\overrightarrow{f_m} = (f_m(\mathbf{x}_1), \dots, f_m(\mathbf{x}_\ell))^\top$ is the output vector of $f_m$ on the $\ell$ input data:

$$\overrightarrow{f_m} = \sum_{i=1}^{m} \theta_i \phi_i = \mathbf{\Omega\Theta}, \tag{13}$$

where $\phi_i = (\phi_i(\mathbf{x}_1), \dots, \phi_i(\mathbf{x}_\ell))^\top$ is the vector of decision values of basis function $\phi_i$ on the input data and $\mathbf{\Omega} = [\phi_1, \dots, \phi_m] \in \mathbb{R}^{\ell \times m}$. Equation (13) suggests that FLSA can be understood as working entirely in the $\mathbb{R}^\ell$ space.

The weight vector $\mathbf{\Theta}$ that minimizes the loss function (12) is given as

$$\mathbf{\Theta} = \left(\mathbf{\Omega}^\top \mathbf{\Omega}\right)^{-1} \mathbf{\Omega}^\top \mathbf{y} \tag{14}$$

for a matrix $\Omega$ which is of full column rank [21]. The resultant minimal loss function is

$$\begin{aligned} L(\mathbf{\Theta}) &= \mathbf{y}^\top \mathbf{y} - \mathbf{\Theta}^\top \mathbf{\Omega}^\top \mathbf{y} \\ &= \mathbf{y}^\top \left(\mathbf{I} - \mathbf{\Omega}\left(\mathbf{\Omega}^\top \mathbf{\Omega}\right)^{-1} \mathbf{\Omega}^\top\right)\mathbf{y}\right). \end{aligned} \tag{15}$$

Starting from (15), FLSA defines a set of residue matrices $\mathbf{R}_i \in \mathbb{R}^{\ell \times \ell}$ $(i = 1, \dots, m)$ for the measurement of the contribution of each basis function to the reduction of loss function:

$$\mathbf{R}_i = \mathbf{I} - \mathbf{\Omega}_i \left(\mathbf{\Omega}_i^\top \mathbf{\Omega}_i\right)^{-1} \mathbf{\Omega}_i^\top, \tag{16}$$

where matrix $\mathbf{\Omega}_i = (\phi_1, \dots, \phi_i)$ is full column rank and is composed of output vectors of $i$ basis function. $\mathbf{I}$ is the unity matrix and set $\mathbf{R}_0 = \mathbf{I}$. Then, the following equation holds:

$$(1)\ \ \mathbf{R}_i \phi_j = 0 \quad 1 \le j \le i, \tag{17}$$

$$(2)\ \ \mathbf{R}_i^\top = \mathbf{R}_i, \qquad \mathbf{R}_i^2 = \mathbf{R}_i. \tag{18}$$

For a $\varphi_i = (\varphi_i(\mathbf{x}_1), \dots, \varphi_i(\mathbf{x}_\ell))^\top$ $(1 \le k \le N)$ which produces a full column rank $(\mathbf{\Omega}_i, \varphi_k)$, it is proved that $\mathbf{R}_{i+1}$ has the following properties:

$$(3)\ \ \mathbf{R}_{i+1} = \mathbf{R}_i - \frac{\mathbf{R}_i \varphi_k \varphi_k^\top \mathbf{R}_i^\top}{\varphi_k^\top \mathbf{R}_i \varphi_k}, \tag{19}$$

$$(4)\ \ \mathbf{R}_i \mathbf{R}_{i+1} = \mathbf{R}_{i+1} \mathbf{R}_i = \mathbf{R}_{i+1}. \tag{20}$$

Meanwhile, the introduction of $\mathbf{R}_i$ simplifies the formulation of $L_i$ which is the evaluation of loss function $L$ at $\mathbf{\Omega}_i$:

$$L_i = \mathbf{y}^\top \mathbf{R}_i \mathbf{y}. \tag{21}$$

And thus

$$L_{i+1} = \mathbf{y}^\top \mathbf{R}_{i+1} \mathbf{y} = L_i - \frac{\mathbf{y}^\top \mathbf{R}_i \varphi_k \varphi_k^\top \mathbf{R}_i^\top \mathbf{y}}{\varphi_k^\top \mathbf{R}_i \varphi_k}. \tag{22}$$

The contribution, denoted by $\delta L_k$, of a column vector $\varphi_k$ makes the loss function $L_i$ can able to be explicitly expressed:

$$\delta L_k = \frac{\mathbf{y}^\top \mathbf{R}_i \varphi_k \varphi_k^\top \mathbf{R}_i^\top \mathbf{y}}{\varphi_k^\top \mathbf{R}_i \varphi_k} = \frac{\left[(\mathbf{R}_i \varphi_k)^\top (\mathbf{R}_i \mathbf{y})\right]^2}{(\mathbf{R}_i \varphi_k)^\top (\mathbf{R}_i \varphi_k)}. \tag{23}$$

FLSA algorithm proceeds in a greedy manner which selects one basis function per iteration. The $i$th iteration identifies the index of the $i$th basis function $\phi_i$ by solving the optimization problem:

$$\underset{k=1,\dots,N}{\arg\max}\, \delta L_k = \frac{\left[\left(\varphi_k^{(i-1)}\right)^\top \mathbf{y}^{(i-1)}\right]^2}{\left(\varphi_k^{(i-1)}\right)^\top \varphi_k^{(i-1)}}, \tag{24}$$

where $\varphi_k^{(i-1)} = \mathbf{R}_{i-1} \varphi_k$ and $\mathbf{y}^{(i-1)} = \mathbf{R}_{i-1} \mathbf{y}$.

The $i$th iteration of FLSA algorithm, in actual fact, establishes the following linear system whose solutions are the last $m - i + 1$ elements of $\theta$ in (13):

$$\begin{bmatrix} \phi_i^{(i-1)} & \phi_{i+1}^{(i-1)} & \cdots & \phi_m^{(i-1)} \end{bmatrix} \begin{bmatrix} \theta_i \\ \theta_{i+1} \\ \vdots \\ \theta_m \end{bmatrix} = \mathbf{y}^{(i-1)}. \tag{25}$$

Thus $[\phi_i^{(i-1)}\varphi_j^{(i-1)}, \phi_i^{(i-1)}\mathbf{y}^{(i-1)}]$ ($\varphi_j \notin \{\phi_1, \ldots, \phi_i\}$) which represents a linear equation is stored for the $i$th iteration. Eventually, $m$ iterations build up an upper triangle which represents a linear system of (26). The solution $\boldsymbol{\Theta}$ can be computed by performing a back substitution procedure that is used by a typical Gaussian elimination process:

$$\begin{bmatrix} \phi_1^\top \phi_1 & \phi_1^\top \phi_2 & \cdots & \phi_1^\top \phi_m \\ 0 & \left(\phi_2^{(1)}\right)^\top \phi_2^{(1)} & \cdots & \left(\phi_2^{(1)}\right)^\top \phi_m^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \left(\phi_m^{(m-1)}\right)^\top \phi_m^{(m-1)} \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_m \end{bmatrix} = \begin{bmatrix} \phi_1^\top \mathbf{y} \\ \left(\phi_2^{(1)}\right)^\top \mathbf{y}^{(1)} \\ \vdots \\ \left(\phi_m^{(m-1)}\right)^\top \mathbf{y}^{(m-1)} \end{bmatrix}. \tag{26}$$

The FLSA algorithm, in fact, is closely related to the Orthogonal Least Squares (OLS) method [22–24], which also allows for the explicit formulation of the contribution of a basis function made to the reduction of the squared error loss.

*3.2. Forward Least Squares Approximation SVMs.* As with standard SVMs, the formulation of LS-SVMs embodies the Structural Risk Minimization (SRM) principle which can be illustrated by Figure 1, in which the dotted line represents the upper bound on the complexity term of function set from which solution is chosen and the dash line the empirical risk. SRM minimizes the upper bound on the expected risk (generalization error), for which the best tradeoff between the complexity term and the empirical risk is required to be found.

To this end, a regularization term is introduced to indicate the tradeoff between the complexity term and the empirical risk, both of which LS-SVMs explicitly formulate. Model selection is performed in the domain of $\mathbb{R} > 0$ in search for its optimal value.

While in FLSA algorithm, it can be concluded from (22) that the training cost $L$ monotonously decreases to the increase in the number of selected basis functions $m$. The training cost $L$ can be then plotted against $m$ into a curve similar to that of the empirical risk in Figure 1.

This fact motivates the following two innovations to traditional formulation of LS-SVMs: (1) the employment of the parameter $m$ as the regularization term; (2) the avoidance of the term to represent empirical risk by using the FLSA as the training algorithm, which minimizes the summed squared residues for any value of $m$. As a result, a new formulation of LS-SVMs—namely, Forward Least Squares Approximation Sparse SVM (FLSA-SVM), which is restricted to be trained by FLSA algorithm, is proposed:

$$\min_{\mathbf{w},b} \quad \frac{1}{2}\left(\mathbf{w}^\top\mathbf{w} + b^2\right), \tag{27}$$

$$\text{s.t.} \quad \mathbf{w}^\top \phi(\mathbf{x}_i) + b = y_i, \quad i = 1, \ldots, \ell, \tag{28}$$

$$|\Gamma| = m, \tag{29}$$

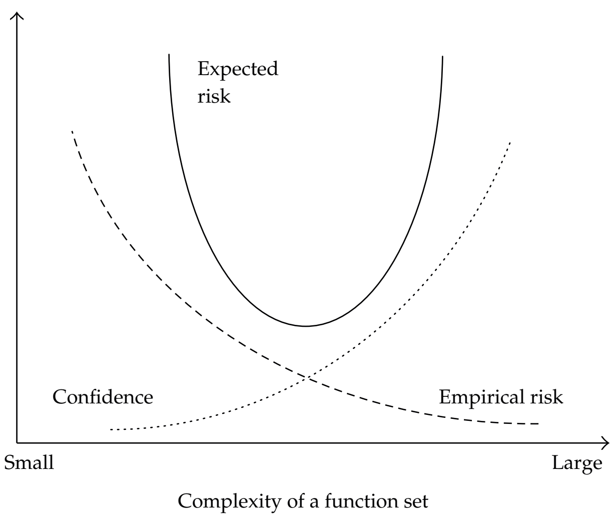

Figure 1: Structural risk minimization (SRM) principle.

where $\Gamma = \gamma_1, \ldots, \gamma_m$ is composed of the indices of the support vectors and $|\Gamma|$ is the cardinality of the set. The addition of term $b^2$ to (27) is first introduced by [25].

In an FLSA-SVM, the parameter $m$ is interpreted as the number of nonzeros of Lagrangian multipliers, that is, the number of support vectors (SVs), which is seen more obviously in its Lagrangian by introducing the Lagrange multipliers $\alpha_i$ ($i = 1, \ldots, \ell$) for each of the equality constraints giving

$$\mathscr{L}(\mathbf{w}, b, \alpha) = \frac{1}{2}\left(\mathbf{w}^\top\mathbf{w} + b^2\right) - \sum_{i=1}^{m} \alpha_i \left[\mathbf{w}^\top \phi(\mathbf{x}_i) + b - y_i\right] - \beta|\Gamma|. \tag{30}$$

The optimal point requires that

$$\begin{gathered} \mathbf{w} = \sum_{i=1}^{m} \alpha_i \phi(\mathbf{x}_i), \\ \sum_{i=1}^{m} \alpha_i = b, \\ \mathbf{w}^\top \phi(\mathbf{x}_i) + b = \mathbf{y}_i, \\ |\Gamma| = m. \end{gathered} \tag{31}$$

The linear equations can be further simplified to

$$\left(\mathbf{K} + \overrightarrow{\mathbf{1}}\right)\boldsymbol{\alpha} = \mathbf{D}\boldsymbol{\alpha} = \mathbf{y}, \tag{32}$$

$$|\Gamma| = m, \tag{33}$$

where $\mathbf{K} \in \mathbb{R}^{\ell\times\ell}$ and $\mathbf{K}_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$, $\mathbf{y} = [y_1, \ldots, y_\ell]^\top$, $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_\ell]^\top$, $\overrightarrow{\mathbf{1}}$ is a $\ell$-by-$\ell$ matrix of 1s, and $\mathbf{D} \in \mathbb{R}^{\ell\times\ell}$, where $\mathbf{D}_{ij} = \mathbf{K}_{ij} + 1$.

It is worth attention that, in this paper, $\mathbf{D}$ in (32) is also referred to as $\mathbf{D}_m$ as it has $m$ support vectors. Thus, if all $\ell$ data samples are used as support vectors, then $\mathbf{D}$ becomes $\mathbf{D}_\ell$.

FLSA can be applied to train an FLSA-SVM using a kernel-based dictionary $\mathbf{D}_\ell = \{d_i = \mathbf{K}(\cdot, j) + 1 \mid i = 1, \ldots, \ell\}$ of candidate basis functions. Since $m$ is defined to be the number of SVs, it thus achieved a more direct control of the sparseness of its solution.

*3.3. Automatic Detection of Linear Dependencies.* Since the kernel matrix $\mathbf{K}$ in (32) is a semipositive definite, it is easy to prove the semipositive definiteness of the matrix $\mathbf{D}$. Then it is likely the occurrence of linear dependencies among column vectors, each being evaluation of a basis function on the training data. Assume that columns $(\mathbf{d}_{\gamma_1}, \mathbf{d}_{\gamma_2}, \ldots, \mathbf{d}_{\gamma_k})$ are selected iteratively and linearly independent, where $\gamma_{1,\ldots,k}$ are their column indices in chronological sequence. Denote $\mathbf{d}$ to be any column which remains as available candidates in $\mathbf{D}$ and meanwhile can be expressed as a linear combination of $(\mathbf{d}_{\gamma_1}, \mathbf{d}_{\gamma_2}, \ldots, \mathbf{d}_{\gamma_k})$. It is thus naturally desired to remove $\mathbf{d}$ as candidates in order to (1) ensure the sparseness of the solution; (2) avoid any undue computation concerning $\mathbf{d}$ since

$$\mathbf{d} = \alpha_1 \mathbf{d}_{\gamma_1} + \alpha_2 \mathbf{d}_{\gamma_2} + \cdots + \alpha_k \mathbf{d}_{\gamma_k}, \tag{34}$$

where $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_k]^\top \neq \mathbf{0}$.

With the introduction of the residue matrix at each iteration as $\mathbf{R}_{\gamma_i}$ $(i = 1, \ldots, k)$, the following property holds according to (18) and (20):

$$\begin{aligned}\mathbf{R}_{\gamma_k}\mathbf{d} &= \alpha_1 \mathbf{R}_{\gamma_k}\mathbf{d}_{\gamma_1} + \alpha_2 \mathbf{R}_k \mathbf{d}_{\gamma_2} + \cdots + \alpha_k \mathbf{R}_{\gamma_{k-1}}\mathbf{d}_{\gamma_k} \\ &= \alpha_1 \mathbf{R}_{\gamma_k}\left(\mathbf{R}_{\gamma_1}\mathbf{d}_{\gamma_1}\right) + \alpha_2 \mathbf{R}_{\gamma_k}\left(\mathbf{R}_{\gamma_2}\mathbf{d}_{\gamma_2}\right) + \cdots + \mathbf{0} \\ &= \mathbf{0}.\end{aligned} \tag{35}$$

Thus updating the dictionary $\mathbf{D}$ by $\mathbf{R}_{\gamma_k}$, the column vector(s) $\mathbf{d}$ becomes $\mathbf{0}$, that is, automatically pruned. Hence, at each iteration of the FLSA-SVM, any column vector(s) which can be represented by a linear combination of previously selected columns can be automatically pruned. This merit of the FLSA-SVM is one contributor to the sparseness of the resultant solution.

Algorithm 1 gives the pseudocode of the FLSA-SVM algorithm in detail.

*3.4. Computation Complexity.* As discussed in [26], for a single round of training, the computational complexity of FLSA-SVM is $O(m\ell^2)$ and space complexity for FLSA-SVM is $O(\ell^2)$.

## 4. Experimental Results

A set of experiments were performed to evaluate the performance of the proposed FLSA-SVM algorithm. The FLSA-SVMs were first applied to the two-spiral benchmark [27] for an illustrative view of their generalization abilities. The following Gaussian kernel function was used for the experiments in this paper:

$$K\left(\mathbf{X}_i, \mathbf{X}_j\right) = e^{-\lambda\|\mathbf{X}_i - \mathbf{X}_j\|^2}. \tag{36}$$

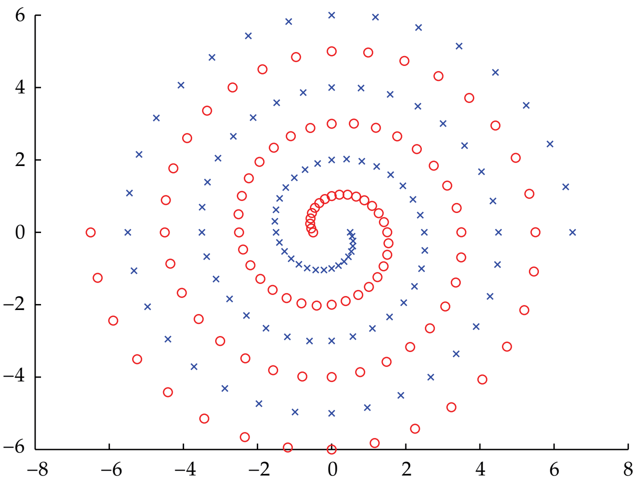

Figure 2: Two-spiral dataset.

The standard SVMs were implemented by LIBSVM [28]. The LS-SVM trained by CG method was implemented by the toolbox of LS-SVMlab [29] and all experiments were run on a Pentium 4 3.2 GHz processor under Windows XP with 2 GB of RAM.

*4.1. Experiments on Two-Spiral Dataset.* The 2D "two-spiral" benchmark is known to be difficult for pattern recognition algorithms and poses great challenges to neural networks [30]. The training set consists of 194 points of the $X$-$Y$ plane, half of which has a target value of +1 output and half a target value of −1. These training points describe two intertwining spirals that go around the origin three times, as shown in Figure 2, where the two categories are marked, respectively, by "+" and "o."

For the FLSA-SVM, the parameter setting of Gaussian kernel was $\lambda = 1$ as in (36). With a feasible range of $[85, 193]$, the optimal regularization parameter was found to be 134 which gave a leave-one-out cross-validation (LOOCV) accuracy of 97.94%. With standard SVMs, the parameter setting was $C = 32$ and $\lambda = 0.5$, whose LOOCV accuracy is 98.97%. The SVM classifier was required in 164 support vectors (SVs). The graphical outputs of the FLSA-SVM and the SVM were given by Figure 3. It showed that generally both SVM and FLSA-SVM algorithms recognized the pattern successfully, outputting "two-spiral" in a very smooth and regular manner. But at the area around the coordinate of $[0, 0]$, the FLSA-SVM showed better performance than the conventional SVM whose decision hyperplane was biased towards the "o" class.

Despite the reduction in the number of SVs which is very notable for such a "hard" classification problem, the FLSA-SVMs are comparatively superior to the following two aspects. In SVMs, it often occurred that given a fixed kernel parameter, the optimal CV accuracy can be obtained from multiple value settings on the regularization parameter. The "two-spiral" problem is a case in point. For the best LOOCV accuracy, with a fixed $\lambda = 0.5$, the value of $C$ can also opt

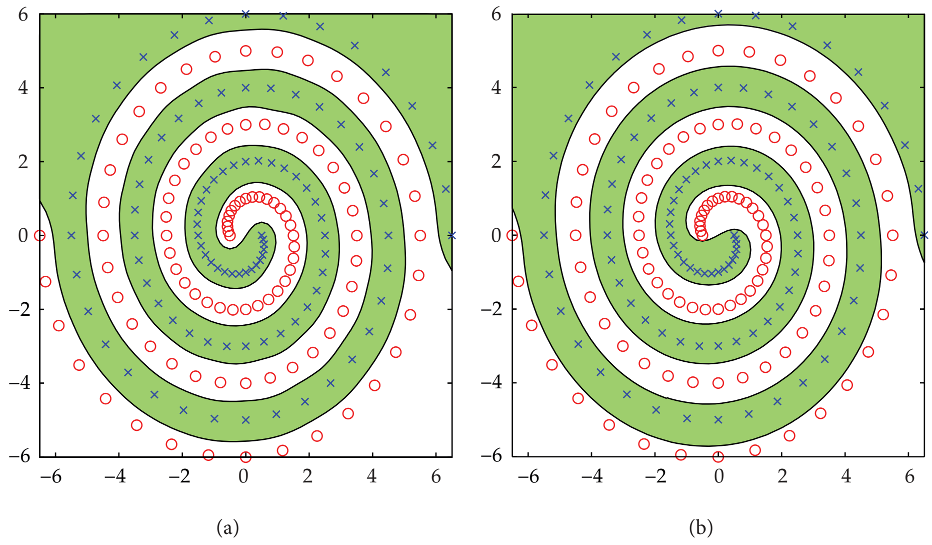

Figure 3: Comparison between standard SVMs and FLSA-SVMs: (a) two-spiral pattern recognized by the FLSA-SVM trained on 134 data points with $\lambda = 1$ in (36); (b) two-spiral pattern recognized by the standard SVM trained on 164 data points. The penalty constant $C = 32$ and $\sigma = 0.5$.

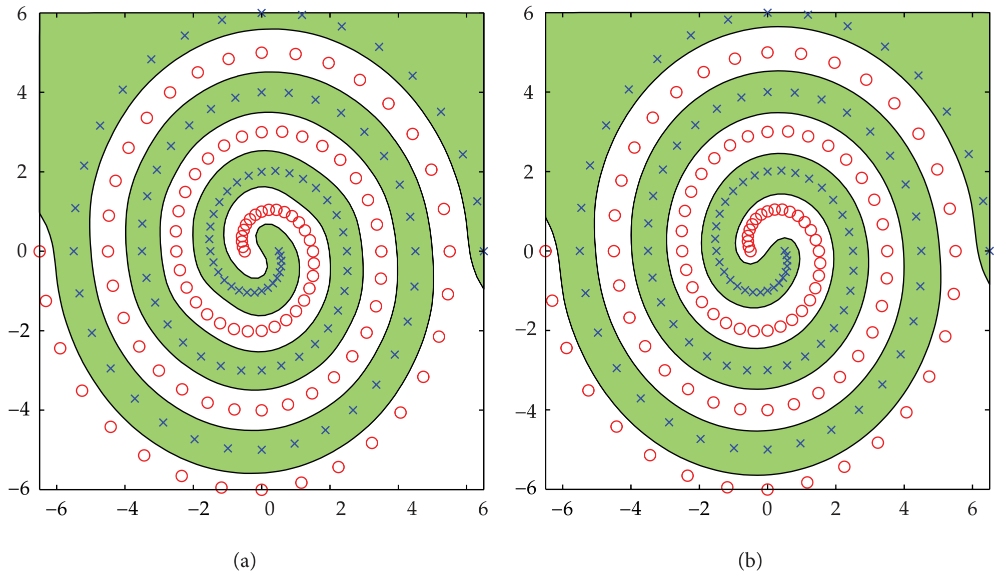

Figure 4: Comparison between LS-SVMs and FLSA-SVMs: (a) two-spiral pattern recognized by the conventional LS-SVM trained on 194 data points with the penalty parameter $\gamma = 173.7427$ and $\lambda = 5.4899$ for Gaussian RBF; (b) two-spiral pattern recognized by the FLSA-SVM trained on 180 data points with $\lambda = 1$ in (36).

for $(2^6, 2^7, 2^8, 2^9, 2^{10})$ beyond $2^5$. There is no specific rule as to which option is the best, and normally the smallest is chosen for a scaled-down feasible region. While in FLSA-SVMs, the setting for the value of the regularization term $m$ is more tractable since different options correspond to different learning errors which can be easily tracked with (23). For multiple values of $m$ which produce the same optimal CV accuracy, the largest $m$ is chosen for a smaller learning error.

In fact, for $m \in [160, 193]$, the LOOCV accuracy remained stable at 96.91%. Thus an FLSA-SVM was also trained with the parameter settings of $m = 193$ which was depicted in Figure 4(a). For comparisons, an LS-SVM was trained whose parameter settings are $\gamma = 173.7427$ and $\lambda = 5.4899$ for Gaussian RBF. The solution was parameterized by the entire 194 points and illustrated in Figure 4(b). The decision boundaries of both of the LS-SVM and the

```
INPUT:
(i) The data set $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l)\}$
(ii) $m$ which is the number of support vectors desired in the expansion of the solution and
    $1 \le m \le l$
(iii) A dictionary of $l$ basis functions $\mathbf{D}_0 = \{\mathbf{d}_1, \ldots, \mathbf{d}_l\}$
INITIALIZATION:
(i) Current residue vector $\mathbf{y}$, current dictionary $\mathbf{D}$ which is initially a matrix of evaluations
    of $l$ candidate basis functions on training data:
        $\mathbf{y} \longleftarrow \begin{pmatrix} y_1 \\ \vdots \\ y_l \end{pmatrix}, \qquad \mathbf{D} \longleftarrow \begin{pmatrix} d_1(\mathbf{x}_1) & \cdots & d_l(\mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ d_1(\mathbf{x}_l) & \cdots & d_l(\mathbf{x}_l) \end{pmatrix}$
(ii) The matrix $\mathbf{A}$ and the vector $\mathbf{b}$ both starts as empty $\mathbf{A}$ is appended a row and $\mathbf{b}$ grows
    by one extra element at each iteration, which in the end forms a linear system.
(iii) A variable $n$ which is the count of candidate basis functions and a vector $\mathbf{\Gamma} = \{\gamma_1, \ldots, \gamma_l\}$
    which contains the indices of basis functions. At the start, $\gamma_i = i$ for $1 \le i \le l$ and $n = l$.
FOR $i = 1, \ldots, m$
        $k \longleftarrow \underset{k=i,\ldots,n}{\arg\max} \left| \frac{\mathbf{D}(\cdot, \gamma_k)^\top \mathbf{y}}{\|\mathbf{D}(\cdot, \gamma_k)\|} \right|$
(iv) $\gamma_i$ is made a pointer to the current selected basis functions:
    $\gamma_k \leftrightarrow \gamma_i$
        $b_i \longleftarrow \frac{\mathbf{D}(\cdot, \gamma_i)^\top \mathbf{y}}{\|\mathbf{D}(\cdot, \gamma_i)\|^2}$
(v) The residue vector is reduced by $b_i \mathbf{d}_{\gamma_i}$ as the target values for the next linear system of size $l$:
        $\mathbf{y} \longleftarrow \mathbf{y} - b_i \mathbf{D}(\cdot, \gamma_i)$
(vi) Update the dictionary matrix and prune the candidate basis functions which can be
    represented as a linear combinations of the previously selected ones:
    FOR $j = i + 1, \ldots, n$
            $\beta_{\gamma_j} \longleftarrow \frac{\mathbf{D}(\cdot, \gamma_i)^\top \mathbf{D}(\cdot, \gamma_j)}{\|\mathbf{D}(\cdot, \gamma_i)\|^2}$
        $\mathbf{D}(\cdot, \gamma_j) \leftarrow \mathbf{D}(\cdot, \gamma_j) - \beta_j \mathbf{D}(\cdot, \gamma_i)$
      IF $\mathbf{D}(\cdot, \gamma_j) = \mathbf{0}$
            $\gamma_l \leftrightarrow \gamma_i$
            $n \leftarrow n - 1$
    $\beta_{\gamma_i} \leftarrow 1$
    $\mathbf{D}(\cdot, \gamma_j) \leftarrow \mathbf{0}$
    $\mathbf{A} \leftarrow \begin{pmatrix} \beta_1, \ldots, \beta_l \\ \mathbf{A} \end{pmatrix}$
    $\mathbf{b} \leftarrow (b_i \quad \mathbf{b})^\top$
(vii) If equations $i = n$ and $m > n$ hold where $i$ is the number of selected basis functions and
    $n$ the count of available candidates, it suggests that the initial value setting on $m$ has
    exceeded the rank of $\mathbf{D}_0$. Terminate the loop and reset $m$:
            $m \leftarrow n$
BACK SUBSTITUTION:
(i) $m$ basis functions are chosen whose indices $\{\gamma_1, \ldots, \gamma_m\}$ are the first $m$ elements of $\mathbf{\Gamma}$. $m_1$
    columns of matrix $\mathbf{A}$ with the indices $\{\gamma_1, \ldots, \gamma_m\}$ and $\mathbf{b}$ forms a linear system, on which
    the process of back substitution is performed for the solution:
            $\alpha_m \leftarrow b_i$
        FOR $i = m - 1, \ldots, 1$
            $\alpha_i \leftarrow b_i - \sum_{j=i+1}^{m} \alpha_j \mathbf{A}(i, \gamma_j)$
OUTPUT:
    (i) The solution is defined by $f(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i \mathbf{d}_{\gamma_i}(\mathbf{x})$
```

Algorithm 1: Forward Least Squares Approximation SVMs.

Table 1: Time cost of 10-fold cross-validation on two-spiral data by FLSA-SVMS, SVMs, and LS-SVMs with fixed kernel parameters: $\lambda = 1$ for FLSA-SVMs, $\lambda = 0.5$ for SVMs, and $\lambda = 5.4899$ for LS-SVMs.

| No. of folds | FLSA-SVMs | SVMs | LS-SVMs $\sigma^2 = 1$ | |
|---|---|---|---|---|
| | $\lambda = 1$ | $\lambda = 0.5$ | SMO | CG |
| 1 | 0.218 | 0.843 | 0.125 | 2.915 |
| 2 | 0.203 | 0.766 | 0.141 | 2.832 |
| 3 | 0.219 | 0.859 | 0.125 | 2.815 |
| 4 | 0.219 | 0.735 | 0.140 | 2.916 |
| 5 | 0.203 | 0.703 | 0.125 | 2.821 |
| 6 | 0.219 | 0.890 | 0.125 | 2.861 |
| 7 | 0.219 | 0.954 | 0.141 | 2.941 |
| 8 | 0.218 | 1.000 | 0.125 | 2.842 |
| 9 | 0.219 | 0.968 | 0.125 | 2.847 |
| 10 | 0.219 | 0.907 | 0.125 | 2.847 |
| $\sum$ | 2.156 | 8.625 | 1.297 | 28.637 |

FLSA-SVM were generally smooth and followed the pattern satisfactorily, despite the slightly biased segments around the coordinates of [1.5, 1.5] and [−1.5, −1.5]. But still the FLSA-SVM performed much better at the origin area than the LS-SVM.

Table 1 also compares the time cost of 10-fold cross-validation (CV) on the regularization parameter by FLSA-SVMs, SVMs and LS-SVMs. For FLSA-SVMs, the regularization term $m$ was assigned 21 integers evenly with an interval of 10 within the range of (1, 194). For SVMs and LS-SVMs, the parameter $C$ was sequentially increased from $2^{-10}$ to $2^{10}$ in multiple of 2. Each fold was split into a training set and a validation set, with a division pattern of 176 training versus 18 validation or 174 training versus 20 validation. The total computation time of the 21 classifiers on various folds for each algorithm was reported in Table 1. The time cost of the 10-fold altogether was given in the last row entry of Table 1. It was clearly shown that the 10-fold cross-validation (CV) of FLSA-SVMs is over 4 times faster than SVMs. The time complexity of FLSA-SVMs remained competitive to that of LS-SVMs using the SMO algorithm and much more reduced than LS-SVMs using the CG method.

These comparisons prove that the FLSA-SVM is very promising in easing the nonsparseness problem of an LS-SVM, in addition to its outstanding generalization performance. And it can obtain a solution whose sparseness is competitive, or even superior, to that of a standard SVM.

*4.2. More Benchmark Problems.* The FLSA-SVM algorithm was applied to 3 small-scale binary problems: Banana, Image, and Splice which are accessible at http://theoval.cmp.uea.ac.uk/matlab/#benchmarks/. Among all the realizations for each benchmark, the first one of them was used. Experiments were also performed on Banana dataset [31], which is a medium-scale binary learning problem. The detailed information of the datasets was given in Table 2.

Table 2: Benchmark information.

| | No. of trainings | No. of tests | No. of features |
|---|---|---|---|
| Banana | 400 | 4900 | 2 |
| Splice | 1000 | 2175 | 60 |
| Image | 1300 | 1010 | 18 |
| Ringnorm | 3000 | 4400 | 20 |

FLSA-SVMs were compared with SVMs, LS-SVMs, the fast sparse approximation scheme for LS-SVM (FSALS-SVM), and its variant, called PFSALS-SVM, both of which were proposed by Jiao et al. [12]. The parameter $\epsilon$ of FSALS-SVMs and PFSALS-SVMs was uniformly set to be 0.5 which was empirically proved to work well with most datasets [12]. Comparisons were also made against D-optimality orthogonal forward regression (D-OFR) [32] which is a technique for nonlinear function estimation, promised to yield sparse solutions. The parameters, which were the penalty constant and $\lambda$ in (36), were tuned by tenfold cross-validation (CV).

Table 3 presented the classification accuracy of the SVM algorithms. The best results among the four SVM algorithms were highlighted. It can be seen that the FLSA-SVM achieved comparable classification accuracy to the standard SVM, the conventional SVM, FSAL-SVM, PFSALS-SVM, and D-optimality OFR.

The numbers of SVs were compared in Table 4. For all the learning problems, the FLSA-SVM required much less SVs than the SVM and the conventional SVM. In particular, the reduction in SVs reached over 98% and 80%, respectively, on Ringnorm and Banana datasets compared with the SVM. FLSA-SVM has maintained its edges over FSAL-SVM and PFSALS-SVM, particularly with the Ringnorm and Banana datasets. Although the D-OFR method falls into the category of unsupervised learning algorithms, FLSA-SVM, mathematically, has the closest link with the D-OFR method. The results in Table 3 demonstrated that the D-OFR method remained competitive compared with FLSA-SVM on Ringnorm and Banana datasets. However, on the Splice and Image datasets, the D-OFR method failed to achieve any sparse solutions.

## 5. Conclusions

The paper proposed a new LS-SVM algorithm—the FLSA-SVM which is trained specifically by the FLSA method of minimized squared error loss. The FLSA-SVM iteratively selects an optimal basis function which is associated with a specific training example into the solution. The algorithm cleverly adapts the number of SVs into the regularization term as the tradeoff between generalization abilities and training cost. As a result, the solution of the FLSA-SVMs is extremely sparse compared to LS-SVMs. Experiments showed that the FLSA-SVM algorithm maintained a comparable accuracy compared to the standard SVM, the LS-SVM and a number of recently developed sparse learning algorithms. Yet the FLSA-SVM showed definite advantages to its counterparts regarding the sparseness of the solution. On

Table 3: Test correctness (%).

| | FLSA-SVM<br>$(m, \lambda)$ | SVM<br>$(C, \lambda)$ | LS-SVM<br>$(\gamma, \lambda)$ | FSALS-SVM<br>$(\gamma, \lambda)$ | PFSALS-SVM<br>$(\gamma, \lambda)$ | D-OFR<br>$(\gamma, \lambda)$ |
|---|---|---|---|---|---|---|
| Banana | **89.33** $(20, 1)$ | **89.33** $(2^5, 2^{-1})$ | 88.92 $(2^3, 0.6369)$ | 89.14 $(2^5, 2^{-1})$ | 89.12 $(2^3, 2^{-1})$ | 89.10 $(40, 2^{-2})$ |
| Splice | **89.98** $(220, 2^{-6})$ | 89.75 $(2^3, 2^{-7})$ | 89.84 $(2^3, 0.0135)$ | 89.93 $(2^3, 2^{-6})$ | 89.93 $(2^8, 2^{-6})$ | 89.33 $(380, 2^{-6})$ |
| Image | 97.92 $(180, 2^{-5})$ | 97.82 $(2^7, 2^{-3})$ | 97.92 $(2^7, 0.0135)$ | **98.32** $(2^4, 2^{-2})$ | 98.02 $(2^5, 2^{-2})$ | 97.92 $(480, 2^{-3})$ |
| Ringnorm | 98.66 $(27, 2^{-5})$ | 98.68 $(2^{-6}, 2^{-5})$ | 97.07 $(2^9, 0.1192)$ | **98.70** $(2^{-4}, 2^{-5})$ | **98.70** $(2^{-5}, 2^{-5})$ | 98.59 $(47, 2^{-5})$ |

Table 4: Number of support vectors (best in bold).

| | FLSA-SVM | SVM | LS-SVM | FSALS-SVM | PFSALS-SVM | D-OFR |
|---|---|---|---|---|---|---|
| Banana | **20** | 94 | 400 | 145 | 141 | 40 |
| Splice | **220** | 595 | 1000 | 507 | 539 | 380 |
| Image | **180** | 221 | 1300 | 272 | 278 | 480 |
| Ringnorm | **27** | 1624 | 3000 | 556 | 575 | 47 |

small datasets like the two-spiral benchmark, the FLSA-SVM training algorithm also proved to be more efficient than the CG method.

## References

[1] J. A. K. Suykens, T. V. Gestel, J. Vandewalle, and B. D. Moor, "A support vector machine formulation to PCA analysis and its kernel version," *IEEE Transactions on Neural Networks*, vol. 14, no. 2, pp. 447–450, 2003.

[2] J. A. K. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural Processing Letters*, vol. 9, no. 3, pp. 293–300, 1999.

[3] V. Vapnik, *The Nature of Statistical Learning Theory*, Springer, New York, NY, USA, 1995.

[4] V. Vapnik, *Statistical Learning Theory*, John Wiley & Sons, New York, NY, USA, 1998.

[5] J. A. K. Suykens, L. Lukas, P. V. Dooren, B. D. Moor, and J. Vandewalle, "Least squares support vector machine classifiers: a large scale algorithm," in *Proceedings of the European Conference on Circuit Theory and Design (ECCTD '99)*, pp. 839–842, Stresa, Italy, September 1999.

[6] T. V. Gestel, J. A. K. Suykens, B. Baesens et al., "Benchmarking least squares support vector machine classifiers," *Machine Learning*, vol. 54, no. 1, pp. 5–32, 2004.

[7] W. Chu, C. J. Ong, and S. S. Keerthi, "An improved conjugate gradient scheme to the solution of least squares SVM," *IEEE Transactions on Neural Networks*, vol. 16, no. 2, pp. 498–501, 2005.

[8] S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy, "Improvements to Platt's SMO algorithm for SVM classifier design," *Neural Computation*, vol. 13, no. 3, pp. 637–649, 2001.

[9] J. A. K. Suykens, J. de Brabanter, L. Lukas, and J. Vandewalle, "Weighted least squares support vector machines: robustness and sparce approximation," *Neurocomputing*, vol. 48, no. 1, pp. 85–105, 2002.

[10] B. J. de Kruif and T. J. A. de Vries, "Pruning error minimization in least squares support vector machines," *IEEE Transactions on Neural Networks*, vol. 14, no. 3, pp. 696–702, 2003.

[11] X. Zeng and X. W. Chen, "SMO-based pruning methods for sparse least squares support vector machines," *IEEE Transactions on Neural Networks*, vol. 16, no. 6, pp. 1541–1546, 2005.

[12] L. Jiao, L. Bo, and L. Wang, "Fast sparse approximation for least squares support vector machine," *IEEE Transactions on Neural Networks*, vol. 18, no. 3, pp. 685–697, 2007.

[13] K. Li, J. X. Peng, and G. W. Irwin, "A fast nonlinear model identification method," *IEEE Transactions on Automatic Control*, vol. 50, no. 8, pp. 1211–1216, 2005.

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

[15] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*, Cambridge University Press, New York, NY, USA, 2000.

[16] R. Fletcher, *Practical Methods of Optimization*, John Wiley & Sons, New York, NY, USA, 1987.

[17] C. Saunders, A. Gammerman, and V. Vovk, "Ridge regression learning algorithm in dual variables," in *Proceedings of the 15th International Conference on Machine Learning (ICML '98)*, pp. 515–521, Morgan Kaufmann, 1998.

[18] S. Mika, G. Ratsch, and K. Muller, "A mathematical programming approach to the kernel fisher algorithm," in *Advances in Neural Information Processing Systems*, pp. 591–597, 2001.

[19] T. Gestel, J. A. K. Suykens, G. Lanckriet, A. Lambrechts, B. Moor, and J. Vandewalle, "Bayesian framework for least-squares support vector machine classifiers, gaussian processes, and kernel fisher discriminant analysis," *Neural Computation*, vol. 14, no. 5, pp. 1115–1147, 2002.

[20] X. Xia, K. Li, and G. Irwin, "Improved training of an optimal sparse least squares support vector machine," in *Proceedings of the 17th World Congress The International Federation of Automatic Control (IFAC '08)*, Seoul, Korea, July 2008.

[21] C. Lawson and R. Hanson, "Solving least squares problems," in *Prentice-Hall Series in Automatic Computation*, Prentice Hall, Englewood Cliffs, NJ, USA, 1974.

[22] S. Chen, S. 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.

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

[24] 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.

[25] O. L. Mangasarian and D. R. Musicant, "Successive overrelaxation for support vector machines," *IEEE Transactions on Neural Networks*, vol. 10, no. 5, pp. 1032–1037, 1999.

[26] K. Li, J. X. Peng, and E. Bai, "A two-stage algorithm for identification of nonlinear dynamic systems," *Automatica*, vol. 42, no. 7, pp. 1189–1197, 2006.

[27] S. Fahlman and C. Lebiere, "The cascade-correlation learning architecture," in *Advances in Neural Information Processing Systems 2*, D. S. Touretzky, Ed., 1990.

[28] C. Chang and C. Lin, "LIBSVM: a library for support vector machines," *Software*, vol. 80, pp. 604–611, 2001, http://www.csie.ntu.edu.tw/~cjlin/libsvm/ .

[29] K. Pelckmans, J. Suykens, T. van Gestel et al., "LSSVMlab: a matlab/C toolbox for least squares support vector machines," in *Tutorial*, KULeuven-ESAT, Leuven, Belgium, 2002.

[30] J. Garcke, M. Griebel, and M. Thess, "Data mining with sparse grids," *Computing*, vol. 67, no. 3, pp. 225–253, 2001.

[31] L. Breiman, "Arcing classifier (with discussion and a rejoinder by the author)," *The Annals of Statistics*, vol. 26, no. 3, pp. 801–849, 1998.

[32] S. Chen, X. Hong, and C. J. Harris, "Regression based D-optimality experimental design for sparse kernel density estimation," *Neurocomputing*, vol. 73, no. 4–6, pp. 727–739, 2010.