Fast Sparse Approximation for Least Squares Support Vector Machine

Licheng Jiao, *Senior Member, IEEE*, Liefeng Bo, and Ling Wang

Abstract—**In this paper, we present two fast sparse approximation schemes for least squares support vector machine (LS-SVM), named FSALS-SVM and PFSALS-SVM, to overcome the limitation of LS-SVM that it is not applicable to large data sets and to improve test speed. FSALS-SVM iteratively builds the decision function by adding one basis function from a kernel-based dictionary at one time. The process is terminated by using a flexible and stable epsilon insensitive stopping criterion. A probabilistic speedup scheme is employed to further improve the speed of FSALS-SVM and the resulting classifier is named PFSALS-SVM. Our algorithms are of two compelling features: low complexity and sparse solution. Experiments on benchmark data sets show that our algorithms obtain sparse classifiers at a rather low cost without sacrificing the generalization performance.**

Index Terms—**Fast algorithm, greedy algorithm, least squares support vector machine (LS-SVM), sparse approximation.**

I. Introduction

In a classification problem, we are given a set of samples of input vectors $\{\mathbf{x}_i\}_{i=1}^l$ along with corresponding class labels $\{y_i\}_{i=1}^l$, and the task is to find a deterministic function that best represents the relation between input vectors and class labels. A successful method for classification problem is least squares support vector machine (LS-SVM) [1], [2], which attempts to minimize the least square error on the training samples while simultaneously maximizing the margin between two classes.

Extensive empirical comparisons [3] show that LS-SVM obtains good performance on various classification problems, but two obvious limitations still exist. First, the training procedure of LS-SVM amounts to solving a set of linear equations [4]. Although the training problem is, in principle, solvable, in practice, it is intractable for a large data set by the classical techniques, e.g., Gaussian elimination, because their computational complexity usually scales cubically with the size of training samples. Second, the solution of LS-SVM lacks the sparseness and, hence, the test speed is significantly slower than other learning algorithms such as support vector machine (SVM) [5], [6] and neural networks (NNs) [7], [8].

Many researchers have identified the aforementioned two limitations and tried to give their solutions. As for the fast algorithms for LS-SVM, Suykens *et al.* [9] presented a conjugate gradient algorithm. Chu *et al.* [10] gave an improved conjugate gradient algorithm. Keerthi and Shevade [11] proposed a sequential minimal optimization (SMO) algorithm. Although these algorithms achieve low complexity, their resulting solutions are not sparse. Hence, the second limitation still exists. As for the sparseness of LS-SVM, Suykens *et al.* [12], [13] proposed a simple approach to introduce the sparseness by sorting the support value spectrum (SVS), i.e., the absolute value of the solution of LS-SVM. Kruif and Vries [14] presented a more sophisticated pruning mechanism that omits the sample bearing the least error after it is omitted. Zeng and Chen [15] proposed a SMO-based pruning method. However, these algorithms require solving a set of linear equations (slowly decreasing in size) many times, which incurs a large computational cost. Hence, the first limitation still exists. Furthermore, Suykens *et al.* [16] proposed fixed-size LS-SVM for fast finding the sparse approximate solution of LS-SVM. Unlike the algorithms mentioned previously, fixed-size LS-SVM (FSLS-SVM) solves the least squares problem in the primal space instead of in the dual space. Recently, Hoegaerts *et al.* [17] introduced the similar idea to solve kernel partial least squares regression.

In this paper, we present a fast sparse approximation scheme for LS-SVM (FSALS-SVM) to deal with the previous two limitations simultaneously. FSALS-SVM iteratively builds the decision function by adding one basis function from a kernel-based dictionary at one time until the ϵ-insensitive criterion is satisfied. FSALS-SVM consists of two key components: the selection of basis function and the solution of subproblem. By integrating some efficient schemes into these two components, FSALS-SVM is endowed with two compelling features: low complexity and sparse solution. It is possible to use a probabilistic speedup scheme to further improve the speed of FSALS-SVM and the resulting classifier is named PFSALS-SVM. Experiments on the benchmark data sets confirm the feasibility and validity of FSALS-SVM and PFSALS-SVM.

The rest of the paper is organized as follows. In Section II, we briefly introduce LS-SVM. In Section III, we interpret why the Wolfe dual of LS-SVM can be regarded as a regularized loss function consisting of the loss function induced by reproducing Kernel–Hilbert space (RKHS) norm and of the regularization term. FSALS-SVM is proposed based on a backfitting scheme in Section IV. The convergence, probabilistic speedup and computational complexity of FSALS-SVM is detail-analyzed in Section V. Related works are discussed in Section VI. Experimental results are reported in Section VII. Finally, in Section VIII, we give some concluding remarks.

For convenience, in Table I, we present some important notations used in the paper.

Manuscript received September 16, 2005; revised June 5, 2006; accepted October 8, 2006. This work was supported by the National Natural Science Foundation of China under Grant 60372050 and the National Defense Preresearch Foundation of China under Grant A1420060172.

The authors are with the Institute of Intelligent Information Processing, Xidian University, Xi'an 710071, China (e-mail: blf0218@163.com).

Digital Object Identifier 10.1109/TNN.2006.889500

TABLE I
SOME IMPORTANT NOTATIONS USED IN THE PAPER

Notation	Description	Notation	Description
l	size of the training samples	n	number of support vectors
d	dimension of samples	γ	regularization parameter
$\mathbf{w}$	weights vector	$\boldsymbol{\alpha}$	Lagrange multipliers vector
b	bias term	ϵ	insensitive constant
$\mathbf{K}$	kernel matrix	P	index set of support vectors
P_i	i-th element of P	Q	index set of non-support vectors
$\boldsymbol{\alpha}_P$	sub-vector of $\boldsymbol{\alpha}$ corresponding to P	$\mathbf{K}_{PP}$	sub-matrix of $\mathbf{K}$ corresponding to P
M	random subset of Q	m	size of random subset M

II. LS-SVM

In this section, we concisely review the basic principles of LS-SVM for classification. For more details, the interested reader can refer to [1] and [2]. In the feature space, LS-SVM takes the form

$$y = \operatorname{sgn}(\mathbf{w}^T\varphi(\mathbf{x}) + b) \tag{1}$$

where the nonlinear mapping $\varphi(\mathbf{x})$ maps the input data into a high-dimensional feature space whose dimension can be infinite. To obtain a classifier, LS-SVM solves the following optimization problem:

$$\begin{aligned} &\min\left\{\frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{\gamma}{2}\sum_{i=1}^{l} e_i^2\right\} \\ &\text{s.t.}\ \ y_i = \mathbf{w}^T\varphi(\mathbf{x}_i) + b + e_i, \qquad i = 1,\ldots,l. \end{aligned} \tag{2}$$

Its Wolfe dual problem is

$$\begin{aligned} &\min\left\{\frac{1}{2}\sum_{i,j=1}^{l}\alpha_i\alpha_j\varphi(\mathbf{x}_i)^T\varphi(\mathbf{x}_j) + \sum_{i=1}^{l}\frac{\alpha_i^2}{2\gamma} - \sum_{i=1}^{l}\alpha_i y_i\right\} \\ &\text{s.t.}\ \ \sum_{i=1}^{l}\alpha_i = 0. \end{aligned} \tag{3}$$

Including in (3) the bias b [18], we can eliminate the equality constraint and obtain

$$\min\left\{\frac{1}{2}\sum_{i,j=1}^{l}\alpha_i\alpha_j\varphi(\mathbf{x}_i)^T\varphi(\mathbf{x}_j) - \sum_{i=1}^{l}\alpha_i y_i + \sum_{i=1}^{l}\frac{\alpha_i^2}{2\gamma} + b\sum_{i=1}^{l}\alpha_i\right\}. \tag{4}$$

The form $\varphi(\mathbf{x}_i)^T\varphi(\mathbf{x}_j)$ in (4) is often replaced with a so-called positive–definite kernel function $k(\mathbf{x}_i,\mathbf{x}_j) = \varphi(\mathbf{x}_i)^T\varphi(\mathbf{x}_j)$. According to Mercer's theory [19], any positive–definite kernel function can be expressed as the inner product of two vectors in some feature space and, therefore, can be used in LS-SVM. Among all the kernel functions, Gaussian kernel $k(\mathbf{x}_i,\mathbf{x}_j) = \exp(-\theta\|\mathbf{x}_i - \mathbf{x}_j\|_2^2)$ is the most popular choice. For a new sample $\mathbf{x}$, we can predict its label by

$$f(\mathbf{x}) = \operatorname{sgn}(\mathbf{w}^T\varphi(\mathbf{x}) + b) = \operatorname{sgn}\left(\sum_{i=1}^{l}\alpha_i k(\mathbf{x},\mathbf{x}_i) + b\right) \tag{5}$$

where $\boldsymbol{\alpha}$ and b is the solution of (4).

III. RKHS NORM VIEW FOR LS-SVM

The key conclusion in this section is that the Wolfe dual of LS-SVM can be regarded as a regularized loss function consisting of the loss function induced by RKHS norm and of the regularization term, which is the basis of developing greedy approximation algorithms. Similar conclusion about support vector regression is reported by Girosi [20].

Theorem 1 [19]: Let $X \subset \mathbb{R}^d$; a real symmetric function $k(\mathbf{x},\mathbf{y}), \mathbf{x},\mathbf{y} \in X$ is positive–definite if and only if, for every set of real numbers $\{\alpha_l,\alpha_2,\ldots,\alpha_l\}$ and every set of vectors $\{\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_l\}$, we have $\sum_{i,j=1}^{l}\alpha_i\alpha_j k(\mathbf{x}_i,\mathbf{x}_j) \geq 0$.

Let H_k RKHS be induced by $k(\mathbf{x},\mathbf{y})$. The kernel function satisfies the following three properties:

1) $k(\cdot,\mathbf{x}) \in H_k$, where $\mathbf{x} \in X$;
2) $\sum_{i=1}^{l}\alpha_i k(\cdot,\mathbf{x}_i) \in H_k$, where α_i and l are finite;
3) for $f(\cdot) \in H_k$, $\mathbf{x} \in X$, $\langle f(\cdot), k(\cdot,\mathbf{x})\rangle_H = f(\mathbf{x})$, where $\langle\cdot,\cdot\rangle_H$ is the inner product of RKHS. In particular, $\langle k(\cdot,\mathbf{x}_i), k(\cdot,\mathbf{x}_j)\rangle_H = k(\mathbf{x}_i,\mathbf{x}_j)$.

We call $\|f\|_H = \sqrt{\langle f,f\rangle_H}$, where $f \in H$, reproducing norm, which can be derived from the inner product $\langle\cdot,\cdot\rangle_H$ of RKHS.

According to the property 2), we can derive that $\sum_{i=1}^{l}\alpha_i k(\mathbf{x},\mathbf{x}_i)$ belongs to RKHS H_k. Measuring the distance by RKHS norm between the target function $f^*(\mathbf{x})$ and $\sum_{i=1}^{l}\alpha_i k(\mathbf{x},\mathbf{x}_i)$, we have the following loss function:

$$L = \frac{1}{2}\left\| f^*(\mathbf{x}) - \sum_{i=1}^{l}\alpha_i k(\mathbf{x},\mathbf{x}_i)\right\|_H^2 \tag{6}$$

where $\|\cdot\|_H$ is RKSH norm. Equation (6) can be expanded as

$$\begin{aligned} L = \frac{1}{2}\left\|f^*(\mathbf{x})\right\|_H^2 &- \sum_{i=1}^{l}\alpha_i\langle f^*(\mathbf{x}), k(\mathbf{x},\mathbf{x}_i)\rangle_H \\ &+ \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i\alpha_j\langle k(\mathbf{x},\mathbf{x}_i), k(\mathbf{x},\mathbf{x}_j)\rangle_H. \end{aligned} \tag{7}$$

Using the reproducing property 3) of kernel function, we can transform (7) into

$$L = \frac{1}{2}\left\|f^*(\mathbf{x})\right\|_H^2 - \sum_{i=1}^{l}\alpha_i f^*(\mathbf{x}_i) + \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i\alpha_j k(\mathbf{x}_i,\mathbf{x}_j). \tag{8}$$

Since $f^*(\mathbf{x}_i)$ is the output of target function on the point $\mathbf{x}$, it is reasonable to estimate it by y_i (for noiseless data, $f^*(\mathbf{x}_i) = y_i$)

$$L = \frac{1}{2}\left\|f^*(\mathbf{x})\right\|_H^2 - \sum_{i=1}^{l}\alpha_i y_i + \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i\alpha_j k(\mathbf{x}_i,\mathbf{x}_j). \tag{9}$$

Adding the regularization term and the constraint term, we can estimate $\boldsymbol{\alpha}$ and b by (10) shown at the bottom of the page. Dropping the constant term, we get

$$\min\left\{\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i\alpha_j k(\mathbf{x}_i,\mathbf{x}_j)-\sum_{i=1}^{l}\alpha_i y_i+\sum_{i=1}^{l}\frac{\alpha_i^2}{2\gamma}\right\}$$
$$\text{s.t.}\quad \sum_{i=1}^{l}\alpha_i=0. \tag{11}$$

It is easy to check that (11) amounts completely to (4), which enlightens us to look the Wolfe dual of LS-SVM as a regularized loss function consisting of the loss function induced by RKHS norm and of the regularization term. Therefore, it is reasonable to approximate the Wolfe dual.

IV. Fast Sparse Approximation Scheme by Backfitting

In this section, we will describe a fast sparse approximation scheme for LS-SVM, named FSALS-SVM. Given a kernel function $k(\mathbf{x},\mathbf{y})$, the function $k(\mathbf{x},\mathbf{x}_i)$ defines one basis function for each sample $\mathbf{x}_i$ in the training set. A set of basis functions, $D=\{k(\mathbf{x},\mathbf{x}_i)|i=1,2,\ldots,l\}$, is called a kernel-based dictionary. FSALS-SVM is a greedy algorithm, which iteratively builds the decision function by adding one basis function from the kernel-based dictionary at one time. FSALS-SVM consists of two key components: the selection of basis function and the solution of subproblem. Starting with an empty index set $P=\emptyset$ and a full index set $Q=\{1,2,\ldots,l\}$, FSALS-SVM first selects a new basis function $k(\mathbf{x},\mathbf{x}_s)$ from the set $\{k(\mathbf{x},\mathbf{x}_i)|i\in Q\}$ according to some criterion. Then, the index s is removed from Q and added to P. After determining the basis function to be included, FSALS-SVM solves the subproblem containing the new basis function and all previously picked basis functions. This procedure is repeated until some stopping criterion is satisfied.

We consider two schemes for the selection of basis function: prefitting and backfitting [21]. Prefitting solves the subproblem containing the ith basis function and all previously picked basis functions for each $i\in Q$ and selects the basis function resulting in the largest decrease in the objective function. Backfitting solves the same subproblem as prefitting while fixing the Lagrange multipliers corresponding to all previously picked basis functions and selects the basis function in the same manner.

As for the solution of subproblem, the inversion of the kernel matrix is a computational bottleneck. If we compute it from scratch at each iteration, the computational cost is $O(n^3)$. In Section IV-A, we develop an iterative computation of the inverse kernel matrix to drop this cost.

For convenience, we reformulate (4) into

$$\min\left\{f=\frac{1}{2}[b\quad \boldsymbol{\alpha}^T]\begin{bmatrix}0 & \mathbf{1}^T\\ \mathbf{1} & \bar{\mathbf{K}}\end{bmatrix}\begin{bmatrix}b\\ \boldsymbol{\alpha}\end{bmatrix}-[b\quad \boldsymbol{\alpha}^T]\begin{bmatrix}0\\ \mathbf{y}\end{bmatrix}\right\} \tag{12}$$

where $\bar{\mathbf{K}}=\mathbf{K}+(1/2\gamma)\mathbf{I}$ with $\mathbf{K}_{ij}=k(\mathbf{x}_i,\mathbf{x}_j)$ and $\mathbf{I}$ denotes the identity matrix and $\mathbf{1}=[1,1,\ldots,1]^T$.

A. Iterative Computation of the Inverse Kernel Matrix

If the sth basis function is chosen at the $(n+1)$th iteration, we have

$$\mathbf{R}^{n+1}=\begin{bmatrix}0 & \mathbf{1}^T & 1\\ \mathbf{1} & \bar{\mathbf{K}}_{PP} & \bar{\mathbf{k}}_s\\ 1 & \bar{\mathbf{k}}_s^T & \bar{\mathbf{K}}_{ss}\end{bmatrix}^{-1} \tag{13}$$

where $\bar{\mathbf{k}}_s=\left[\bar{\mathbf{K}}_{P_1 s},\bar{\mathbf{K}}_{P_2 s},\ldots,\bar{\mathbf{K}}_{P_n s}\right]^T$. Given that $\mathbf{R}^n=\begin{bmatrix}0 & \mathbf{1}^T\\ \mathbf{1} & \bar{\mathbf{K}}_{PP}\end{bmatrix}^{-1}$ has already been computed at the nth iteration, the following matrix inversion Lemma 1 allows $\mathbf{R}^{n+1}$ to be found at a cost of $O(n^2)$.

Lemma 1 [22]: Given an invertible matrix $\mathbf{A}$ and matrices $\mathbf{U}$, $\mathbf{V}$, and $\mathbf{D}$, (14), shown at the bottom of the page, holds.

Applying the formula (14) to the matrix inversion problem given in (13), we have an updating formula

$$\mathbf{R}^{n+1}=\begin{bmatrix}\mathbf{R}^n & \mathbf{0}\\ \mathbf{0}^T & 0\end{bmatrix}+\lambda\begin{bmatrix}\boldsymbol{\beta}\\ -1\end{bmatrix}[\boldsymbol{\beta}^T\quad -1] \tag{15}$$

where $\boldsymbol{\beta}=\mathbf{R}^n\begin{bmatrix}1\\ \bar{\mathbf{k}}_s\end{bmatrix}$ and $\lambda=\left(\bar{\mathbf{K}}_{ss}-[1\quad \bar{\mathbf{k}}_s^T]\boldsymbol{\beta}\right)^{-1}$. According to (15), the formula of computing $\boldsymbol{\alpha}$ and b at the $(n+1)$th iteration can be expressed as

$$\begin{bmatrix}b^{n+1}\\ \boldsymbol{\alpha}_P^{n+1}\\ \alpha_s^{n+1}\end{bmatrix}=\mathbf{R}^{n+1}\begin{bmatrix}0\\ \mathbf{y}_P\\ y_s\end{bmatrix}$$
$$=\begin{bmatrix}\mathbf{R}^n\begin{bmatrix}0\\ \mathbf{y}_P\end{bmatrix}\\ 0\end{bmatrix}+\lambda\left(\boldsymbol{\beta}^T\begin{bmatrix}0\\ \mathbf{y}_P\end{bmatrix}-y_s\right)\begin{bmatrix}\boldsymbol{\beta}\\ -1\end{bmatrix}. \tag{16}$$

Given that $\boldsymbol{\alpha}$ and b at the nth iteration have been computed with the equation $\begin{bmatrix}b^n\\ \boldsymbol{\alpha}_P^n\end{bmatrix}=\mathbf{R}^n\begin{bmatrix}0\\ \mathbf{y}_P\end{bmatrix}$, we have

$$\begin{bmatrix}b^{n+1}\\ \boldsymbol{\alpha}_P^{n+1}\\ \alpha_s^{n+1}\end{bmatrix}=\begin{bmatrix}b^n\\ \boldsymbol{\alpha}_P^n\\ 0\end{bmatrix}+\lambda\left(\boldsymbol{\beta}^T\begin{bmatrix}0\\ \mathbf{y}_P\end{bmatrix}-y_s\right)\begin{bmatrix}\boldsymbol{\beta}\\ -1\end{bmatrix}. \tag{17}$$

$$\min\left\{\frac{1}{2}\left\|f^*(\mathbf{x})\right\|_H^2+\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i\alpha_j k(\mathbf{x}_i,\mathbf{x}_j)-\sum_{i=1}^{l}\alpha_i y_i+\sum_{i=1}^{l}\frac{\alpha_i^2}{2\gamma}\right\}\qquad \text{s.t.}\quad \sum_{i=1}^{l}\alpha_i=0. \tag{10}$$

$$\begin{bmatrix}\mathbf{A} & \mathbf{U}\\ \mathbf{V} & \mathbf{D}\end{bmatrix}^{-1}=\begin{bmatrix}\mathbf{A}^{-1}+\mathbf{A}^{-1}\mathbf{U}(\mathbf{D}-\mathbf{V}\mathbf{A}^{-1}\mathbf{U})^{-1}\mathbf{V}\mathbf{A}^{-1} & -\mathbf{A}^{-1}\mathbf{U}(\mathbf{D}-\mathbf{V}\mathbf{A}^{-1}\mathbf{U})^{-1}\\ -(\mathbf{D}-\mathbf{V}\mathbf{A}^{-1}\mathbf{U})^{-1}\mathbf{V}\mathbf{A}^{-1} & (\mathbf{D}-\mathbf{V}\mathbf{A}^{-1}\mathbf{U})^{-1}\end{bmatrix}. \tag{14}$$

Equations (15) and (17) indicate that we can efficiently update $\mathbf{R}$, $\boldsymbol{\alpha}$, and b at a cost of $O(n^2)$ without explicitly computing the inverse matrix. The updating formula (15) is also used in online SVM [18], [23] and sparse online Gaussian processes [24]. It is worth emphasizing that this updating formula is numerical stable since the regularization term $(1/2\gamma)\mathbf{I}$ greatly improves the condition numbers of the matrix $\mathbf{K}$. Of course, one can further improve the numerical stability by considering the Cholesky decomposition [25]. Using the lower triangular matrix $\mathbf{L}$ and the identity $\mathbf{R} = \mathbf{L}^T\mathbf{L}$, we have the updating formula for the Cholesky decomposition

$$\mathbf{L}^{n+1} = \begin{bmatrix} \mathbf{L}^n & \mathbf{0} \\ -\lambda^{1/2}\boldsymbol{\beta}^T & \lambda^{1/2} \end{bmatrix}. \tag{18}$$

B. Prefitting

Prefitting solves the subproblem containing the ith basis function and all previously picked basis functions for each $i \in Q$ and selects the basis function resulting in the largest decrease in the objective function. For a fixed $i \in Q$, the subproblem confronted by prefitting can be expressed as

$$f_i^{n+1} = \begin{cases} \min f(\alpha_i, b, \alpha_{P_1}, \alpha_{P_2}, \ldots, \alpha_{P_n}) \\ s.t. \quad \boldsymbol{\alpha}_{Q-\{i\}} = \mathbf{0}, \qquad i \in Q \end{cases}. \tag{19}$$

With (12), we can translate (19) into (20), as shown at the bottom of the page, where $\bar{\mathbf{k}}_i = [\bar{\mathbf{K}}_{P_1 i}, \bar{\mathbf{K}}_{P_2 i}, \ldots, \bar{\mathbf{K}}_{P_n i}]^T$.

It is easily checked that the optimal value of (20) is

$$f_i^{n+1} = -\frac{1}{2}[0 \quad \mathbf{y}_P^T \quad y_i] \begin{bmatrix} 0 & \mathbf{1}^T & 1 \\ \mathbf{1} & \bar{\mathbf{K}}_{PP} & \bar{\mathbf{k}}_i \\ 1 & \bar{\mathbf{k}}_i^T & \bar{\mathbf{K}}_{ii} \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ \mathbf{y}_P \\ y_i \end{bmatrix}, \quad i \in Q. \tag{21}$$

Substituting (15) into (21), we obtain

$$f_i^{n+1} = -\frac{1}{2}[0 \quad \mathbf{y}_P^T]\mathbf{R}^n \begin{bmatrix} 0 \\ \mathbf{y}_P \end{bmatrix} - \frac{1}{2}[0 \quad \mathbf{y}_P^T \quad y_i] \left(\lambda \begin{bmatrix} \boldsymbol{\beta} \\ -1 \end{bmatrix} [\boldsymbol{\beta}^T \quad -1]\right) \begin{bmatrix} 0 \\ \mathbf{y}_P \\ y_i \end{bmatrix}, \quad i \in Q. \tag{22}$$

Together with $\boldsymbol{\beta} = \mathbf{R}^n \begin{bmatrix} 1 \\ \bar{\mathbf{k}}_i \end{bmatrix}$ and $\lambda = \left(\bar{\mathbf{K}}_{ii} - [1 \quad \bar{\mathbf{k}}_i^T]\boldsymbol{\beta}\right)^{-1}$, we can further simplify (22) into

$$f_i^{n+1} = -\frac{1}{2}[0 \quad \mathbf{y}_P^T]\mathbf{R}^n \begin{bmatrix} 0 \\ \mathbf{y}_P \end{bmatrix} - \frac{\left([0 \quad \mathbf{y}_P^T]\mathbf{R}^n \begin{bmatrix} 1 \\ \bar{\mathbf{k}}_i \end{bmatrix} - y_i\right)^2}{2\left(\bar{\mathbf{K}}_{ii} - [1 \quad \bar{\mathbf{k}}_i^T]\mathbf{R}^n \begin{bmatrix} 1 \\ \bar{\mathbf{k}}_i \end{bmatrix}\right)}, \quad i \in Q. \tag{23}$$

Thus, we can find the index of the basis function to be included by

$$s = \arg\min_{i \in Q} \left(f_i^{n+1}\right). \tag{24}$$

Equation (23) suggests the cost of doing one basis function inclusion is $O(n^2)$. To find the basis function resulting in the largest decrease in the objective function, we need to do the basis function inclusion for all the $(l-n)$ basis functions, which incurs a computational cost of $O(n^2 l)$. This cost is much higher than that of solving the subproblem. Thus, we would like to go for a cheaper method.

C. Backfitting

Backfitting solves the subproblem containing the ith basis function and all previously picked basis functions while fixing the Lagrange multipliers corresponding to all previously picked basis functions and selects the basis function resulting in the largest decrease in the objective function. For a fixed $i \in Q$, the subproblem confronted by backfitting can be expressed as

$$f_i^{n+1} = \begin{cases} \min f(\alpha_i) \\ \text{s.t.} \quad \boldsymbol{\alpha}_{Q-\{i\}} = \mathbf{0}, \qquad i \in Q. \\ \qquad \boldsymbol{\alpha}_P = \boldsymbol{\alpha}_P^n \end{cases} \tag{25}$$

With (12), we can translate (25) into (26), shown at the bottom of the page.

It is easily checked that (26) can be simplified into

$$f_i^{n+1} = \min_{\alpha_i} \left\{\frac{1}{2}\left(k(\mathbf{x}_i, \mathbf{x}_i) + \frac{1}{2\gamma}\right)\alpha_i^2 + r_i^n \alpha_i\right\}, \qquad i \in Q. \tag{27}$$

$$f_i^{n+1} = \min_{\alpha_i, b, \boldsymbol{\alpha}_P} \left\{\frac{1}{2}[b \quad \boldsymbol{\alpha}_P^T \quad \alpha_i] \begin{bmatrix} 0 & \mathbf{1}^T & 1 \\ \mathbf{1} & \bar{\mathbf{K}}_{PP} & \bar{\mathbf{k}}_i \\ 1 & \bar{\mathbf{k}}_i^T & \bar{\mathbf{K}}_{ii} \end{bmatrix} \begin{bmatrix} b \\ \boldsymbol{\alpha}_P \\ \alpha_i \end{bmatrix} - [b \quad \boldsymbol{\alpha}_P^T \quad \alpha_i] \begin{bmatrix} 0 \\ \mathbf{y}_P \\ y_i \end{bmatrix}\right\}, \qquad i \in Q \tag{20}$$

$$f_i^{n+1} = \min_{\alpha_i} \left\{\frac{1}{2}\left[b^n \quad (\boldsymbol{\alpha}_P^n)^T \quad \alpha_i\right] \begin{bmatrix} 0 & \mathbf{1}^T & 1 \\ \mathbf{1} & \bar{\mathbf{K}}_{PP} & \bar{\mathbf{k}}_i \\ 1 & \bar{\mathbf{k}}_i^T & \bar{\mathbf{K}}_{ii} \end{bmatrix} \begin{bmatrix} b^n \\ \boldsymbol{\alpha}_P^n \\ \alpha_i \end{bmatrix} - \left[b^n \quad (\boldsymbol{\alpha}_P^n)^T \quad \alpha_i\right] \begin{bmatrix} 0 \\ \mathbf{y}_P \\ y_i \end{bmatrix}\right\}, \qquad i \in Q. \tag{26}$$

where

$$r_i^n = \begin{cases} -y_i, & n = 0 \\ \sum_{k=1}^{n} \alpha_{P_k} k(\mathbf{x}_i, \mathbf{x}_{P_k}) + b^n - y_i, & n > 0. \end{cases} \tag{28}$$

The optimal value of (27) is

$$f_i^{n+1} = -\frac{(r_i^n)^2}{\left(2\left(k(\mathbf{x}_i, \mathbf{x}_i) + \frac{1}{2\gamma}\right)\right)}, \qquad i \in Q. \tag{29}$$

Thus, we can find the index of the basis function to be included by

$$s = \arg\min_{i \in Q} \left\{ \frac{-(r_i^n)^2}{\left(2\left(k(\mathbf{x}_i, \mathbf{x}_i) + \frac{1}{2\gamma}\right)\right)} \right\}. \tag{30}$$

In backfitting, the most time-consuming operation is computing r_i^n for each $i \in Q$, which incurs a computational cost of $O(nl)$, greatly smaller than $O(n^2 l)$ of prefitting. Consequently, we will adopt backfitting as our final scheme.

For a greedy algorithm, it is necessary to choose an appropriate stopping criterion. In this paper, we will terminate FSALS-SVM if the ϵ-insensitive criterion

$$\max\left(\left|r_Q^n\right|\right) < \epsilon \tag{31}$$

is satisfied, where ϵ is a small positive constant. Note that $\epsilon = 1$ is the smallest value that guarantees the unselected training samples being correctly classified. The reason for choosing the previous criterion is the following. From (28) and (31), we can derive that if inequality (31) is satisfied, FSALS-SVM will predict the unselected training samples with an error smaller than ϵ. For an ϵ small enough, the generalization performance of FSALS-SVM will not be greatly improved by adding a new basis function. This stopping criterion is similar to the early stopping [26] in NNs, an effective mechanism for avoiding the overfitting.

Combining the iterative computation of the inverse kernel matrix, backfitting, and stopping criterion (31), we have an efficient sparse approximation scheme for LS-SVM, as shown in Algorithm 1. Following SVM, we call the samples corresponding to nonzero Lagrange multipliers as support vectors.

Algorithm 1: Fast sparse approximation for LS-SVM with backfitting

```
1) Let $\boldsymbol{\alpha}^0 = \mathbf{0}^T$, $b = 0$, $\mathbf{r}^0 = -\mathbf{y}$, $Q = \{1, 2, 3, \ldots, l\}$,
   $P = \emptyset$, and $n = 0$.
2) If $Q = \emptyset$ or $\max(|\mathbf{r}_Q^n|) < \epsilon$, stop.
3) $s = \arg\min_{i \in Q}(-(r_i^n)^2/(2(k(\mathbf{x}_i, \mathbf{x}_i) + 1/2\gamma)))$.
4) If $n = 0$
       $\mathbf{R}^{n+1} = \begin{bmatrix} 0 & 1 \\ 1 & k(\mathbf{x}_s, \mathbf{x}_s) + 1/2\gamma \end{bmatrix}^{-1}$
   and
       $\begin{bmatrix} b^{n+1} \\ \alpha_s^{n+1} \end{bmatrix} = \mathbf{R}^{n+1} \begin{bmatrix} 0 \\ y_s \end{bmatrix}$.
5) If $n \neq 0$, compute $\mathbf{R}^{n+1}$, $\boldsymbol{\alpha}_P^{n+1}$, $\alpha_s^{n+1}$, and $b^{n+1}$
   according to (15) and (17).
6) $Q = Q - \{s\}$ and $P = P + \{s\}$.
7) $\mathbf{r}_Q^{n+1} = \mathbf{K}_{QP}\boldsymbol{\alpha}_P^{n+1} + b^{n+1}$.
8) $n = n + 1$, go to 2).
```

V. Convergence, Probabilistic Speedup, and Computational Complexity

A. Convergence

Theorem 2: The objective function f monotonously decreases with an increasing number of basis functions. With ϵ set to be 0, FSALS-SVM is equivalent to the original LS-SVM.

Proof: Assume that FSALS-SVM is at the nth iteration. Before the sth basis function is included, the value of the objective function is

$$f^n = \begin{cases} \min\left\{ f = \frac{1}{2}\begin{bmatrix} b & \boldsymbol{\alpha}^T \end{bmatrix} \begin{bmatrix} 0 & \mathbf{1}^T \\ \mathbf{1} & \bar{\mathbf{K}} \end{bmatrix} \begin{bmatrix} b \\ \boldsymbol{\alpha} \end{bmatrix} - \begin{bmatrix} b & \boldsymbol{\alpha}^T \end{bmatrix} \begin{bmatrix} 0 \\ \mathbf{y} \end{bmatrix} \right\} \\ \text{s.t.} \quad \boldsymbol{\alpha}_Q = \mathbf{0} \end{cases}. \tag{32}$$

After the sth basis function is included, the value of the objective function becomes

$$f^{n+1} = \begin{cases} \min\left\{ f = \frac{1}{2}\begin{bmatrix} b & \boldsymbol{\alpha}^T \end{bmatrix} \begin{bmatrix} 0 & \mathbf{1}^T \\ \mathbf{1} & \bar{\mathbf{K}} \end{bmatrix} \begin{bmatrix} b \\ \boldsymbol{\alpha} \end{bmatrix} - \begin{bmatrix} b & \boldsymbol{\alpha}^T \end{bmatrix} \begin{bmatrix} 0 \\ \mathbf{y} \end{bmatrix} \right\} \\ \text{s.t.} \quad \boldsymbol{\alpha}_{Q-\{s\}} = \mathbf{0}, \qquad s \in Q \end{cases}. \tag{33}$$

Comparing (32) with (33), we can see that the constraint $\alpha_s = 0$ in (32) does not appear in (33), which means that the feasible set of (32) is the subset of that of (33). Consequently, we have

$$f^{n+1} \leq f^n. \tag{34}$$

This completes the proof of the first part of Theorem 2.

Because $\max(|\mathbf{r}_Q^n|)$ is greater than or equal to 0, the stopping criterion $\max\left(|\mathbf{r}_Q^n|\right) < \epsilon$ does not hold true when ϵ is set to be 0. Hence, FSALS-SVM will select all the basis functions, exactly resulting in the original LS-SVM. This completes the proof of the second part of Theorem 2.

B. Probabilistic Speedup

The computational bottleneck of FSALS-SVM is to update $\mathbf{r}_Q^n$, which can be further reduced by only considering the random subset of Q with size m. In other words, we select the basis function only from a random subset rather than perform an exhaustive search in the full set Q. The resulting classifier is named probabilistic FSALS-SVM (PFSALS-SVM), which is described in Algorithm 2. PFSALS-SVM is a feasible scheme due to Lemma 2.

TABLE II
COMPARISONS OF FSALS-SVM, PFSALS-SVM, CG, AND SMO IN TERMS OF MEMORY REQUIREMENT, TRAINING COST, AND PREDICTION COST

	FSALS-SVM	PFSALS-SVM	CG	SMO
Memory	$O(nl)$	$O(n^2)$	$O(l)$	$O(l)$
Training	$O(n^2 l)$	$O(n^3)$	$O(\#it \times l^2)$	$O(\#it \times l)$
Prediction	$O(n)$	$O(n)$	$O(l)$	$O(l)$

Algorithm 2: Probabilistic fast sparse approximation for LS-SVM

```
1) Let $\boldsymbol{\alpha}^0 = \mathbf{0}^T, b = 0, \mathbf{r}^0 = -\mathbf{y}, M = Q = \{1, 2, 3, \dots, l\}$,
   $P = \emptyset$, and $n = 0$.
2) If $Q = \emptyset$ or $\max(|\mathbf{r}_M^n|) < \epsilon$, stop.
3) $s = \arg\min_{i \in M}(-(r_i^n)^2 / (2(k(\mathbf{x}_i, \mathbf{x}_i) + 1/2\gamma)))$.
4) If $n = 0$
       $$\mathbf{R}^{n+1} = \begin{bmatrix} 0 & 1 \\ 1 & k(\mathbf{x}_s, \mathbf{x}_s) + 1/2\gamma \end{bmatrix}^{-1}$$
   and
       $$\begin{bmatrix} b^{n+1} \\ \alpha_s^{n+1} \end{bmatrix} = \mathbf{R}^{n+1} \begin{bmatrix} 0 \\ y_s \end{bmatrix}.$$
5) If $n \neq 0$, compute $\mathbf{R}^{n+1}$, $\boldsymbol{\alpha}_P^{n+1}$, $\alpha_s^{n+1}$, and $b^{n+1}$
   according to (15) and (17).
6) $Q = Q - \{s\}$ and $P = P + \{s\}$.
7) Randomly choose a subset $M$ of size 146 from $Q$.
8) $\mathbf{r}_M^{n+1} = \mathbf{K}_{MP} \boldsymbol{\alpha}_P^{n+1} + b^{n+1}$.
9) $n = n + 1$, go to 2).
```

Lemma 2 [27]: Denote by $\xi_1, \xi_2, \dots, \xi_m$ identical distributed independent random variables with the common cumulative distribution function $F(\xi_i)$. Then, the cumulative distribution function of $\xi = \max_{i \le m}(\xi_i)$ is $(F(\xi))^m$.

For the uniform distribution [0, 1], the distribution of $\xi = \max_{i \le m}(\xi_i)$ is ξ^m. Thus, in order to obtain an estimate with probability 0.975 among the best 0.025 of all estimates, we only need a random subset of size

$$\left\lceil \frac{\log(0.025)}{\log(0.975)} \right\rceil = 146. \tag{35}$$

C. Computational Complexity

At each iteration of FSALS-SVM, we need to update $\mathbf{R}^n$, which is $O(n^2)$, and compute $\mathbf{r}_Q^n$, which is $O(nl)$. Adding up these costs till n basis functions are selected, we have the computational cost of $O(n^2 l)$. The prediction cost of FSALS-SVM is $O(n)$. The memory requirement of FSALS-SVM is $O(nl)$. For PFSALS-SVM, the cost of the selection of basis function is dropped to $O(n)$, so updating $\mathbf{R}^n$ becomes a computational bottleneck. Successive n such updates incur a computational cost of $O(n^3)$. The memory requirement of PFSALS-SVM is $O(n^2)$. The time and space complexity of FSALS-SVM, PFSALS-SVM, conjugate gradient (CG), and SMO are summarized in Table II.

VI. RELATED WORK

There are many efforts for kernel methods in greedy styles. Our discussion will focus on a few key algorithms close related to the least square loss function. The typical examples include kernel match pursuit (KMP) [21] and sparse greedy Gaussian process (SGGP) [27].

Kernel matching pursuit is an extension to the kernel-based dictionary of matching pursuit [28] primarily developed from the signal processing domain. Given a kernel function $k(\mathbf{x}, \mathbf{y})$, KMP uses a set of basis functions centered on the training samples: $D = \{k(\mathbf{x}, \mathbf{x}_i) | i = 1, 2, \dots, l\}$ as a dictionary. During training, one only considers the values of the basis functions at the training samples, so that it amounts to doing matching in a vector-space of l dimensions. According to Vincent and Bengio's suggestion, the error estimated on an independent validation set is a good stop criterion. In using the least square loss function, the computational cost of KMP is $O(nl^2)$ if one uses all the training samples as the candidate support vectors. It is possible to use a small random set of the training samples as the candidate support vectors, which will drop the computational cost of KMP to $O(n^2 l)$.

Sparse greedy Gaussian process is another greedy algorithm. Like KMP and FSALS-SVM, SGGP incrementally builds the decision function by adding one basis function from a kernel-based dictionary at one time until an appropriate criterion is satisfied. The key contribution is that the accuracy of the approximate solution can be estimated by a simple approach. The computational cost of SGGP is $O(n^2 l^2)$ if one uses all the training samples as the candidate support vectors. In using a small random set of the training samples as the candidate support vectors, the cost is dropped to $O(mn^2 l)$, with m the size of random set.

Although FSALS-SVM and PFSALS-SVM are similar with KMP and SGGP in many aspects, they have some novel behaviors due to the difference among the loss functions used. In details, the loss function of KMP is

$$f_1 = \frac{1}{2}\boldsymbol{\alpha}^T \mathbf{K}^T \mathbf{K} \boldsymbol{\alpha} - \mathbf{y}^T \mathbf{K} \boldsymbol{\alpha} + \frac{1}{2\gamma} \boldsymbol{\alpha}^T \boldsymbol{\alpha} \tag{36}$$

where $\mathbf{K}$ is the gram matrix and γ is a regularization parameter, and that of SGGP is

$$f_2 = \frac{1}{2}\boldsymbol{\alpha}^T \mathbf{K}^T \mathbf{K} \boldsymbol{\alpha} - \mathbf{y}^T \mathbf{K} \boldsymbol{\alpha} + \frac{1}{2\gamma} \boldsymbol{\alpha}^T \mathbf{K} \boldsymbol{\alpha} \tag{37}$$

and that of our algorithms is

$$f = \frac{1}{2}\boldsymbol{\alpha}^T \mathbf{K} \boldsymbol{\alpha} - \mathbf{y}^T \boldsymbol{\alpha} + \frac{1}{2\gamma} \boldsymbol{\alpha}^T \boldsymbol{\alpha} + b \sum_{i=1}^{l} \alpha_i. \tag{38}$$

Due to the concise loss function, the selection of basis function and the updating formula in FSALS-SVM and PFSALS-SVM are much cheaper than those in KMP and SGGP. In Table III, we show the memory requirement, training cost, and prediction cost of FSALS-SVM, KMP, and SGGP.

Table III does not imply that the computational cost of FSALS-SVM is necessarily lower than that of KMP or SGGP because the number of support vectors of FSALS-SVM, n, may be much larger than that of KMP or SGGP for some

TABLE III
COMPARISONS OF FSALS-SVM, KMP, AND SGGP IN TERMS OF MEMORY REQUIREMENT, TRAINING COST, AND PREDICTION COST

	Full Search			Probabilistic Speedup		
	FSALS-SVM	KMP	SGGP	PFSALS-SVM	KMP	SGGP
Memory	$O(nl)$	$O(l^2)$	$O(l^2)$	$O(n^2)$	$O(nl)$	$O(nl)$
Training	$O(n^2 l)$	$O(nl^2)$	$O(n^2 l^2)$	$O(n^3)$	$O(n^2 l)$	$O(mn^2 l)$
Prediction	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$

TABLE IV
INFORMATION ON BENCHMARK DATA SETS

Problem	θ	Training	Test	Attributes	Classes
Ringnorm	2^{-3}	3000	4400	20	2
Satimage	2^{-2}	4435	2000	36	6
USPS	2^{-7}	7291	2007	256	10
MNIST1	2^{-6}	20000	10000	784	10
MNIST	2^{-6}	60000	10000	784	10

classification problems. In other words, the computational cost of the above three algorithms depends on specific problems. However, we still can obtain from Table III the valuable information: when the three algorithms are comparable in terms of the number of support vectors, FSALS-SVM will give the lowest computational cost.

VII. EMPIRICAL STUDY

In order to investigate the behavior of our algorithms, we perform three kinds of experiments. In Section VII-A, we compare our algorithms with LS-SVM, SVS, and SVM on Ringnorm, Satimage, USPS, and MNIST data sets in terms of the classification accuracy, the training time, and the number of support vectors. The information of benchmark data sets is described in Table IV. In Section VII-B, we show the variation of classification accuracy, training time, and the number of support vectors with the insensitive parameter and give some related remarks. In Section VII-C, we compare our algorithms with KMP and SGGP on Ringnorm and USPS data sets.

One-against-all is used to extend FSALS-SVM to multiclass classification, which has been independently devised by different researchers. Rifkin and Klautau [29] carefully compared one-against-all with some other popular multiclass strategies and concluded that one-agianst-all is as accurate as any other approaches if the underlying binary classifiers are well-tuned regularized classifiers.

A. *Comparisons With CG, SMO, SVS, and SVM*

Ringnorm data set comes from [30]. This is 20 dimensions,two-class data. Class 1 is multivariate normal with mean zero and covariance matrix four times the identity. Class 2 has unit covariance matrix and mean $\left[2/\sqrt{20},\ldots,2/\sqrt{20}\right]$. The 3000 randomly generated samples are used for training the classifiers and the 4400 randomly generated samples for testing the generalization performance.

Satimage data set comes from Statlog collection [31]. This data set consists of 4435 training samples and 2000 test samples with 36 dimensions for each sample. Before training, we scale all the training samples with zero mean and unit variance, and then adjust the test samples using the same linear transformation.

USPS data set [32] is well known and has been extensively used for testing the performance of diversified kinds of classifiers. This data set consists of 7291 training samples and 2007 test samples. Each sample is a 16×16 image represented as a 256-dimentional vector with entries between 0 and 1.

MNIST data set [33] consists of 60 000 training samples and 10 000 test samples. Each sample is a 28×28 image represented as a 784-dimentional vector with entries between 0 and 1. We perform two experiments on MNIST data set. In the first one, only first 20 000 training samples are used for training classifiers and, in the second one, all the training samples are used.

We implement our own C code for FSALS-SVM and PSALS-SVM, which is available online at http://see.xidian.edu.cn/graduate/lfbo/. LS-SVM is constructed by two methods, i.e., CG implemented by LS-SVMlab 1.5 [16] and SMO implemented by our own C code. The stopping criterion used in CG and SMO is the same and based on the value of the mean-square error (mse), i.e.

$$\frac{1}{l}\left(\bar{\mathbf{K}}\boldsymbol{\alpha}+b\times\mathbf{1}-\mathbf{y}\right)^T\left(\bar{\mathbf{K}}\boldsymbol{\alpha}+b\times\mathbf{1}-\mathbf{y}\right)\le\tau \qquad (39)$$

where $\tau = 10^{-6}$. SVM is constructed using library of SVM (LIBSVM) [34] that implements the improved SMO for SVM. The elements of Gram matrix $\mathbf{K}$ are computed using Gaussian kernel function of form $k(\mathbf{x}_i,\mathbf{x}_j)=\exp(-\theta\left\|\mathbf{x}_i-\mathbf{x}_j\right\|_2^2)$. All the experiments are run in a personal computer with 2.4-GHz processor, 1-G memory, and Windows XP operation system. In the most experiments, ϵ is set to be 0.5. We have found that it works well by using this setting for various data sets. We also try some other values on USPS and MNIST data sets.

For each data set, we fix θ at a suitable value which gives good generalization performance and vary C over a wide range. In all the experiments, we try the following 12 C values: 2^i, $i=-1,0,\ldots,9,10$.

We compare FSALS-SVM, PFSALS-SVM, LS-SVM, SVS, and SVM in terms of the classification accuracy, the training time, and the number of support vectors in Tables V–XVII. For fair comparison, the number of support vectors of SVS and FSALS-SVM is kept equal for each binary classifier. The results of some classifiers on USPS, MNIST1, and MNIST data sets are not reported because of not enough memory or too long training time.

Note that the number of unique support vectors is reported for multiclass problems. We say "unique" support vectors because a training sample may be support vector in different binary classifiers. Here, we only report the number of the training samples that are support vectors of at least one binary classifier, because it is a dominant factor that affects the test time.

TABLE V
ACCURACIES OBTAINED BY FSALS-SVM, PFSALS-SVM, LS-SVM, SVS, AND SVM ON THE TEST SET OF RINGNORM DATA SET

$\log 2(\gamma)$	FSALS-SVM	PFSALS-SVM	LS-SVM	SVS	SVM
-1	0.9859	0.9852	0.9864	0.9668	0.9868
0	0.9836	0.9834	0.9848	0.9652	0.9870
1	0.9825	0.9830	0.9841	0.9586	0.9861
2	0.9809	0.9832	0.9825	0.9545	0.9852
3	0.9827	0.9815	0.9814	0.9525	0.9852
4	0.9827	0.9811	0.9811	0.9518	0.9852
5	0.9816	0.9809	0.9811	0.9509	0.9852
6	0.9807	0.9800	0.9814	0.9495	0.9852
7	0.9809	0.9795	0.9814	0.9498	0.9852
8	0.9816	0.9797	0.9811	0.9495	0.9852
9	0.9802	0.9814	0.9811	0.9493	0.9852
10	0.9818	0.9811	0.9811	0.9491	0.9852

TABLE VI
NUMBER OF SUPPORT VECTORS OBTAINED BY FSALS-SVM, PFSALS-SVM, LS-SVM, SVS, AND SVM ON THE TRAINING SET OF RINGNORM DATA SET

$\log 2(\gamma)$	FSALS-SVM	PFSALS-SVM	LS-SVM	SVS	SVM
-1	666	661	3000	666	538
0	622	625	3000	622	497
1	580	608	3000	580	510
2	552	588	3000	552	538
3	548	601	3000	548	541
4	554	579	3000	554	541
5	549	604	3000	549	541
6	555	573	3000	555	541
7	561	592	3000	561	541
8	558	595	3000	558	541
9	529	599	3000	529	541
10	549	599	3000	549	541

TABLE VII
TRAINING TIME OBTAINED BY FSALS-SVM, PFSALS-SVM, LS-SVM, SVS, AND SVM ON THE TRAINING SET OF RINGNORM DATA SET. TIME DENOTES THE CPU TIME IN SECONDS

$\log 2(\gamma)$	FSALS-SVM	PFSALS-SVM	LS-SVM		SVS(CG)	SVM
			CG	SMO		
-1	9.45	7.12	23.56	7.00	111.21	0.47
0	8.16	6.29	27.50	7.61	132.93	0.45
1	7.09	5.77	30.16	8.58	179.75	0.48
2	6.58	5.61	43.39	10.40	192.17	0.52
3	7.08	5.23	49.11	11.57	239.20	0.52
4	6.42	5.36	54.16	13.62	313.38	0.50
5	7.01	5.56	58.70	14.89	361.15	0.52
6	6.70	4.64	66.14	21.69	362.30	0.54
7	6.63	5.02	66.53	16.33	365.69	0.51
8	6.56	5.00	70.91	17.18	368.48	0.52
9	6.02	5.08	68.36	18.36	371.13	0.51
10	6.39	5.12	71.51	18.21	377.22	0.62

TABLE VIII
ACCURACIES OBTAINED BY FSALS-SVM, PFSALS-SVM, LS-SVM, SVS, AND SVM ON THE TEST SET OF SATIMAGE DATA SET

$\log 2(\gamma)$	FSALS-SVM	PFSALS-SVM	LS-SVM	SVS	SVM
-1	0.9210	0.9190	0.9180	0.9200	0.9090
0	0.9220	0.9225	0.9180	0.9200	0.9160
1	0.9195	0.9195	0.9200	0.9190	0.9235
2	0.9195	0.9200	0.9220	0.8980	0.9140
3	0.9170	0.9180	0.9210	0.8985	0.9155
4	0.9175	0.9175	0.9200	0.8955	0.9165
5	0.9170	0.9190	0.9175	0.8990	0.9160
6	0.9155	0.9116	0.9160	0.8815	0.9155
7	0.9175	0.9145	0.9145	0.8475	0.9155
8	0.9150	0.9110	0.9140	0.8545	0.9155
9	0.9145	0.9130	0.9140	0.8415	0.9155
10	0.9140	0.9120	0.9140	0.8560	0.9155

TABLE IX
NUMBER OF SUPPORT VECTORS OBTAINED BY FSALS-SVM, PFSALS-SVM, LS-SVM, SVS, AND SVM ON THE TRAINING SET OF SATIMAGE DATA SET

$\log 2(\gamma)$	FSALS-SVM	PFSALS-SVM	LS-SVM	SVS	SVM
-1	1809	1844	4435	1881	2139
0	1675	1726	4435	1746	2071
1	1626	1642	4435	1749	2042
2	1620	1640	4435	1788	2053
3	1637	1662	4435	1808	2057
4	1695	1695	4435	1916	2051
5	1736	1718	4435	1952	2042
6	1797	1768	4435	1980	2041
7	1799	1717	4435	2018	2041
8	1828	1783	4435	2038	2041
9	1842	1825	4435	2039	2041
10	1837	1800	4435	2036	2041

TABLE X
TRAINING TIME OBTAINED BY FSALS-SVM, PFSALS-SVM, LS-SVM, SVS, AND SVM ON THE TRAINING SET OF SATIMAGE DATA SET. TIME DENOTES THE CPU TIME IN SECONDS

$\log 2(\gamma)$	FSALS-SVM	PFSALS-SVM	LS-SVM		SVS(SMO)	SVM
			CG	SMO		
-1	70.81	62.09	403.41	72.33	598.44	14.78
0	56.63	50.03	504.70	92.97	712.31	14.50
1	51.03	42.42	636.19	123.89	954.22	14.45
2	49.38	40.55	808.40	171.31	1316.28	14.19
3	49.92	41.78	1002.27	234.55	1814.80	13.59
4	53.45	43.92	1194.61	309.36	2429.94	13.98
5	55.80	44.84	1447.80	386.67	3067.22	14.02
6	60.83	47.80	1670.61	452.50	3643.58	13.75
7	62.16	43.05	1861.71	505.75	4080.16	14.31
8	63.77	49.28	1992.14	538.97	4356.97	15.55
9	63.35	51.08	2018.46	559.64	4537.84	15.03
10	64.19	53.19	2072.29	570.02	4631.53	14.50

TABLE XI
ACCURACIES OBTAINED BY FSALS-SVM, PFSALS-SVM, LS-SVM, AND SVM ON THE TEST SET OF USPS DATA SET

$\log 2(\gamma)$	FSALS-SVM	PFSALS-SVM		LS-SVM	SVM
		$\epsilon = 0.5$	$\epsilon = 0.4$		
-1	0.9482	0.9450	0.9477	0.9481	0.9467
0	0.9502	0.9517	0.9522	0.9507	0.9482
1	0.9517	0.9497	0.9547	0.9522	0.9522
2	0.9581	0.9502	0.9562	0.9542	0.9576
3	0.9581	0.9567	0.9567	0.9557	0.9567
4	0.9567	0.9547	0.9561	0.9562	0.9561
5	0.9586	0.9557	0.9581	0.9562	0.9557
6	0.9586	0.9567	0.9606	0.9552	0.9552
7	0.9547	0.9522	0.9562	0.9552	0.9562
8	0.9567	0.9537	0.9527	0.9552	0.9552
9	0.9557	0.9532	0.9517	0.9552	0.9542
10	0.9571	0.9512	0.9562	0.9552	0.9542

From Tables V–X, we can see that the classification accuracy of FSALS-SVM and PFSALS-SVM is comparable with that of SVS for the small γ values and higher than that of SVS for the large γ values. The training time of FSALS-SVM and PFSALS-SVM is significantly shorter than that of SVS.

From Tables V–XIII, we can see that the classification accuracy of FSALS-SVM and PFSALS-SVM is comparable with that of LS-SVM for all the γ values. The number of unique support vectors of FSALS-SVM and PFSALS-SVM is $0.3 \sim 0.5$ times that of LS-SVM, so we can expect that the test time of FSALS-SVM and PFSALS-SVM is $0.3 \sim 0.5$ times that of LS-SVM. The training time of FSALS-SVM and PFSALS-SVM is significantly shorter than that of SMO and CG on these data sets. We do not report the results of LS-SVM on MNIST1 and MNIST data sets because of a too long training time.

From Tables V –XVII, we can see that the classification accuracy of FSALS-SVM and PFSALS-SVM is comparable with that of SVM for all the γ values. The best classification accuracy of FSALS-SVM and PFSALS-SVM is slightly better than that of SVM on USPS and MNIST data sets and slightly worse than that of SVM on Ringnorm and Satimage data sets. The number of unique support vectors of FSALS-SVM and PFSALS-SVM is nearly as many as that of SVM, so we can expect that the test time of FSALS-SVM and PFSALS-SVM is comparable with that of SVM. The training time of FSALS-SVM and PFSALS-SVM is also comparable with that of SVM.

B. Insensitive Parameter

In this section, we will investigate the variation of the performance measure with the insensitive parameter ϵ. Two small

TABLE XII
NUMBER OF SUPPORT VECTORS OBTAINED BY FSALS-SVM, PFSALS-SVM, LS-SVM, AND SVM ON THE TRAINING SET OF USPS DATA SET

$\log 2(\gamma)$	FSALS-SVM	PFSALS-SVM		LS-SVM(CG)	SVM
		$\epsilon = 0.5$	$\epsilon = 0.4$		
-1	2341	2250	2612	7291	3050
0	2102	1926	2403	7291	2780
1	1992	1790	2245	7291	2676
2	1983	1741	2110	7291	2637
3	2002	1718	2235	7291	2636
4	2042	1711	2237	7291	2631
5	2100	1838	2320	7291	2634
6	2154	1811	2356	7291	2645
7	2204	1808	2461	7291	2642
8	2273	1874	2381	7291	2632
9	2296	1911	2506	7291	2623
10	2234	1848	2543	7291	2623

TABLE XIII
TRAINING TIME OBTAINED BY FSALS-SVM, PFSALS-SVM, LS-SVM, AND SVM ON THE TRAINING SET OF USPS DATA SET

$\log 2(\gamma)$	FSALS-SVM	PFSALS-SVM		LS-SVM		SVM
		$\epsilon = 0.5$	$\epsilon = 0.4$	CG	SMO	
-1	99.34	90.58	144.86	18305.49	1057.64	60.25
0	78.59	59.02	112.45	24341.42	1378.53	55.50
1	70.14	48.53	92.95	31887.59	1899.97	54.25
2	68.75	44.45	78.81	42013.13	1681.49	54.36
3	68.25	43.67	87.97	54523.11	3757.03	54.80
4	71.33	42.30	87.54	71790.98	5135.47	53.89
5	72.25	48.59	95.22	95263.73	6813.11	54.05
6	76.30	46.61	94.73	/	8675.82	54.09
7	79.11	47.27	105.97	/	10502.78	54.14
8	81.23	52.88	97.30	/	12244.23	54.98
9	83.27	52.98	106.60	/	13829.45	53.69
10	85.81	48.05	115.31	/	15729.13	53.73

TABLE XIV
ACCURACIES, NUMBER OF SUPPORT VECTORS, AND TRAINING TIME OBTAINED BY FSALS-SVM AND SVM ON MNIST1 DATA SET. ACC DENOTES THE CLASSIFICATION ACCURACY ON TEST SET, NSV DENOTES THE NUMBER OF SUPPORT VECTORS ON TRAINING SET, AND TIME DENOTES THE CPU TIME IN SECONDS ON TRAINING SET

$\log 2(\gamma)$	FSALS-SVM			SVM		
	ACC	NSV	Time	ACC	NSV	Time
-1	0.9741	6915	2335.78	0.9659	8374	746.20
0	0.9715	6382	1935.36	0.9712	7412	935.25
1	0.9782	6172	1761.40	0.9749	7105	1210.27
2	0.9802	6228	1758.37	0.9777	7126	1393.28
3	0.9809	6393	1807.10	0.9787	7192	1463.45
4	0.9810	6609	1924.87	0.9787	7191	1470.84
5	0.9807	6771	1992.53	0.9788	7192	1471.19
6	0.9808	6861	2075.25	0.9788	7192	1470.83
7	0.9802	6923	2104.97	0.9788	7192	1471.50
8	0.9815	6957	2141.50	0.9788	7192	1477.22
9	0.9808	6971	2139.09	0.9788	7192	1473.97
10	0.9814	6983	2150.45	0.9788	7192	1470.09

data sets, Ringnorm and Satimage, are used for this purpose. For each data set, γ is set to be a suitable value giving the best generalization performance. Plots of the classification accuracy, the number of support vectors, and the training time as functions of ϵ are shown in Figs. 1–3.

We can see that the number of support vectors and the training time of FSALS-SVM monotonously decrease with an increasing value of ϵ, which can be explicitly explained by the meaning of ϵ. However, the classification accuracy of FSALS-SVM possibly suffers from an ascending and descending process with an increasing value of ϵ, which means that LS-SVM, i.e., FSALS-SVM with $\epsilon = 0$, may not be the best choice if our purpose is to achieve the best generalization performance.

C. Comparisons With KMP and SGGP

The task of this section is to compare PSASLS-SVM with KMP and SGGP. All three algorithms use the probabilistic speedup scheme with the random set of size 146. All the free parameters are tuned using tenfold cross validation [35]. The results of the three algorithms on Ringnorm and USPS data sets are shown in Tables XVIII and XIX.

For Ringnorm data set, the classification accuracy of PFSALS-SVM is slightly higher than KMP and SGGP. The

TABLE XV
ACCURACIES, NUMBER OF SUPPORT VECTORS, AND TRAINING TIME OBTAINED BY PFSALS-SVM ON MNIST1 DATA SET. ACC DENOTES THE CLASSIFICATION ACCURACY ON TEST SET, NSV DENOTES THE NUMBER OF SUPPORT VECTORS ON TRAINING SET, AND TIME DENOTES THE CPU TIME IN SECONDS ON TRAINING SET

$\log 2(\gamma)$	PFSALS-SVM($\epsilon = 0.5$)			PFSALS-SVM($\epsilon = 0.4$)		
	ACC	NSV	Time	ACC	NSV	Time
-1	0.9730	6395	1618.63	0.9738	7639	3075.59
0	0.9758	5563	1124.42	0.9756	6874	2280.46
1	0.9778	5116	893.70	0.9782	6708	2121.91
2	0.9786	5113	876.81	0.9796	6675	1916.72
3	0.9793	5199	884.75	0.9804	6865	2068.24
4	0.9788	5286	910.03	0.9803	6791	1981.55
5	0.9783	5236	902.98	0.9803	7008	2098.52
6	0.9786	5481	934.75	0.9802	7127	2268.69
7	0.9789	5264	838.34	0.9804	7331	2427.34
8	0.9779	5297	904.13	0.9801	7301	2431.62
9	0.9791	5539	1014.48	0.9813	7202	2297.97
10	0.9780	5418	932.72	0.9812	7445	2496.45

TABLE XVI
ACCURACIES, NUMBER OF SUPPORT VECTORS, AND TRAINING TIME OBTAINED BY PFSALS-SVM AND SVM ON MNIST DATA SET. ACC DENOTES THE CLASSIFICATION ACCURACY ON TEST SET, NSV DENOTES THE NUMBER OF SUPPORT VECTORS ON TRAINING SET, AND TIME DENOTES THE CPU TIME IN SECONDS ON TRAINING SET

$\log 2(\gamma)$	PFSALS-SVM($\epsilon = 0.5$)			SVM		
	ACC	NSV	Time	ACC	NSV	Time
-1	0.9826	12520	6994.84	0.9773	17758	5850.30
0	0.9852	10877	4700.19	0.9815	15577	6370.23
1	0.9859	10350	3940.33	0.9842	14576	7909.18
2	0.9846	10107	3812.30	0.9855	14487	9104.25
3	0.9850	10363	3942.86	0.9859	14611	9846.84
4	0.9850	10634	4176.33	0.9861	14616	9992.76
5	0.9848	10805	4194.77	0.9859	14631	9999.20
6	0.9867	11332	4752.20	0.9859	14622	9995.24
7	0.9851	11406	4659.92	0.9858	14622	9997.15
8	0.9842	10708	4288.67	0.9858	14622	9996.55
9	0.9843	11607	5053.11	0.9858	14622	10011.77
10	0.9861	11360	4752.05	0.9858	14622	10002.32

TABLE XVII
ACCURACIES, NUMBER OF SUPPORT VECTORS, AND TRAINING TIME OBTAINED BY PFSALS-SVM ON MNIST DATA SET. ACC DENOTES THE CLASSIFICATION ACCURACY ON TEST SET, NSV DENOTES THE NUMBER OF SUPPORT VECTORS ON TRAINING SET, AND TIME DENOTES THE CPU TIME IN SECONDS ON TRAINING SET

$\log 2(\gamma)$	PFSALS-SVM($\epsilon = 0.4$)			PFSALS-SVM($\epsilon = 0.6$)		
	ACC	NSV	Time	ACC	NSV	Time
-1	0.9845	16835	16558.61	0.9828	10918	4792.25
0	0.9833	16441	14212.95	0.9845	9250	3150.72
1	0.9852	14544	10098.65	0.9836	7965	2089.11
2	0.9855	14105	9245.75	0.9805	7094	1609.87
3	0.9864	14547	9450.79	0.9831	7242	1694.54
4	0.9876	14527	8996.06	0.9827	6751	1389.06
5	0.9870	15408	9864.97	0.9829	7510	1760.33
6	0.9868	15223	10290.26	0.9799	7173	1535.34
7	0.9870	15624	10224.36	0.9834	7865	1903.23
8	0.9862	16685	12688.10	0.9818	7461	1730.51
9	0.9823	7795	1873.22	0.9823	7795	1873.22
10	0.9805	7616	1880.09	0.9805	7616	1880.09

number of support vectors of PFSALS-SVM is more than that of KMP and SGGP, however, the training time of PF-SALS-SVM is comparable with that of KMP and significantly shorter than that of SGGP. For USPS data set, PFSALS-SVM outperforms KMP and SGGP in terms of the classification accuracy, the number of support vectors, and the training time. In particular, the training time of PFSALS-SVM is about 1/20 that of KMP and 1/200 that of SGGP.

VIII. CONCLUSION

LS-SVM is a successful approach to classification, but two obvious limitations still exist. First, its computational complexity usually scales cubically with the size of training samples. Second, the solution of LS-SVM lacks the sparseness and, hence, the test speed is very slow. This paper describes a fast greedy algorithm for LS-SVM, named FSALS-SVM,

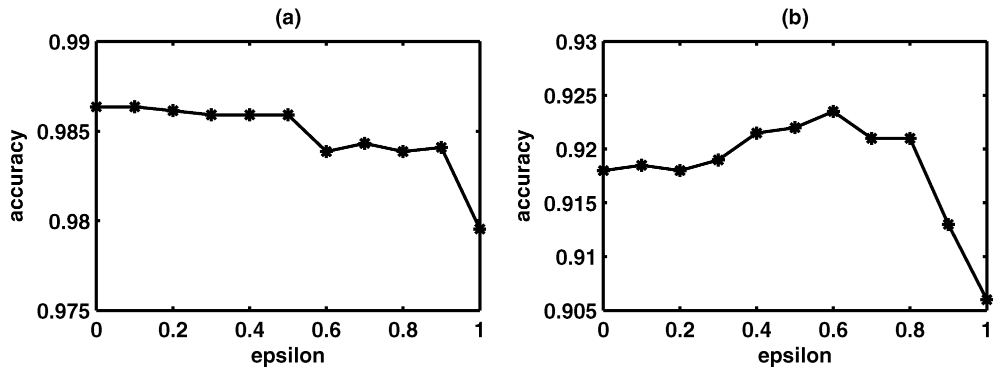

Fig. 1. Variation of the classification accuracy of FSALS-SVM with parameter ϵ. (a) Ringnorm data set with $\gamma = 2^{-1}$. (b) Statimage data set with $\gamma = 2^{0}$.

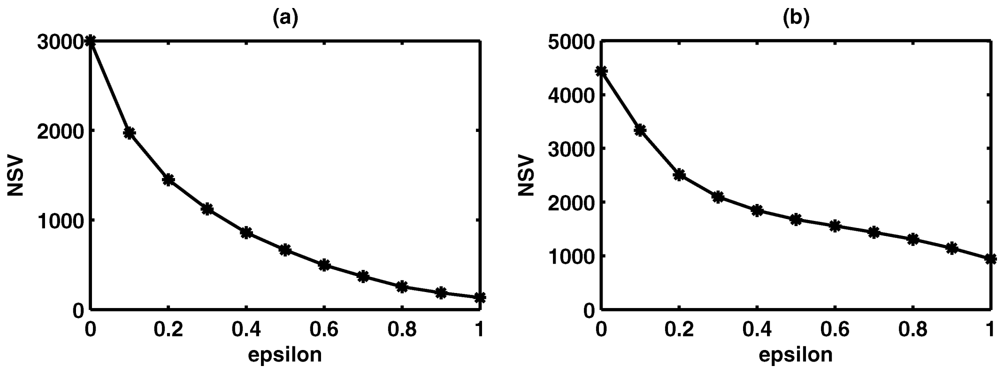

Fig. 2. Variation of the number of support vectors of FSALS-SVM with parameter ϵ. (a) Ringnorm data set with $\gamma = 2^{-1}$. (b) Statimage data set with $\gamma = 2^{0}$.

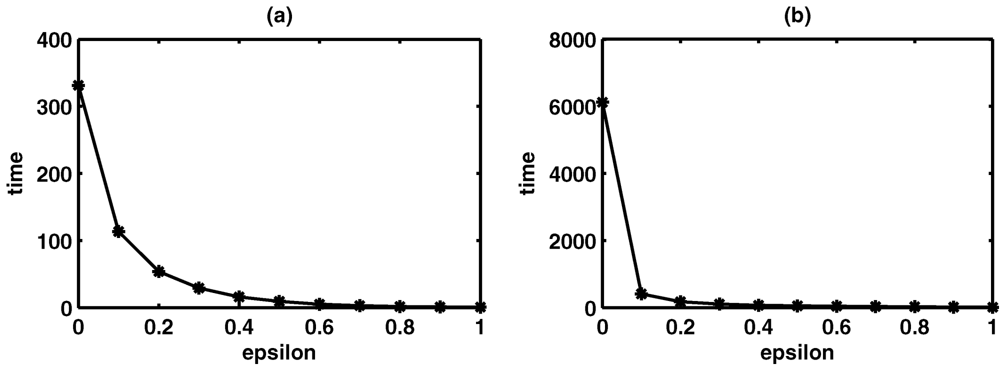

Fig. 3. Variation of the training time of FSALS-SVM with parameter ϵ. (a) Ringnorm data set with $\gamma = 2^{-1}$. (b) Statimage data set with $\gamma = 2^{0}$.

TABLE XVIII
RESULTS OF PFSALS-SVM, KMP, AND SGGP ON RINGNORM DATA SET. NSV DENOTES THE NUMBER OF SUPPORT VECTORS, ACC DENOTES THE CLASSIFICATION ACCURACY, AND TIME DENOTES THE TRAINING TIME IN SECONDS

Problem	PFSALS-SVM	KMP	SGGP
NSV	661	77	74
ACC	0.9852	0.9811	0.9827
Time	7.12	7.53	50.13

TABLE XIX
RESULTS OF PFSALS-SVM, KMP, AND SGGP ON USPS DATA SET. NSV DENOTES THE NUMBER OF SUPPORT VECTORS, ACC DENOTES THE CLASSIFICATION ACCURACY, AND TIME DENOTES THE TRAINING TIME IN SECONDS

Problem	PFSALS-SVM	KMP	SGGP
NSV	1718	1583	1566
ACC	0.9567	0.9517	0.9532
Time	43.67	853.03	8525.08

which attempts to overcome the two limitations simultaneously. Using a probabilistic speedup scheme, we can further improve the speed of FSALS-SVM and the resulting classifier is named PFSALS-SVM. FSALS-SVM and PFSALS-SVM achieve both low complexity and sparseness due to the introduction of ϵ-insensitive criterion. Extensive empirical comparisons suggest that FSALS-SVM and PFSALS-SVM yield good generalization performance on various classification problems.

ACKNOWLEDGMENT

The authors would lie to thank the three reviewers for their helpful comments that greatly improved this paper.

REFERENCES

[1] J. A. K. Suykens and J. Vandewalle, “Least squares support vector machine classifiers,” *Neural Process. Lett.*, vol. 9, no. 3, pp. 293–300, 1999.

[2] T. Van Gestel, J. Suykens, G. Lanckriet, A. Lambrechts, B. De Moor, and J. Vandewalle, “Bayesian framework for least squares support vector machine classifiers, Gaussian processes and kernel fisher discriminant analysis,” *Neural Comput.*, vol. 15, no. 5, pp. 1115–1148, 2002.

[3] T. Van Gestel, J. Suykens, B. Baesens, S. Viaene, J. Vanthienen, G. Dedene, B. De Moor, and J. Vandewalle, “Benchmarking least squares support vector machine classifiers,” *Mach. Learn.*, vol. 54, no. 1, pp. 5–32, 2004.

[4] L. V. Ferreira, E. Kaszkurewicz, and A. Bhaya, “Solving systems of linear equations via gradient systems with discontinuous righthand sides: Application to LS-SVM,” *IEEE Trans. Neural Netw.*, vol. 16, no. 2, pp. 501–505, Mar. 2005.

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

[6] ——, *Statistical Learning Theory*. New York: Wiley, 1998.

[7] R. Neal, *Bayesian Learning for Neural Networks*. New York: Springer-Verlag, 1996.

[8] R. Ripley, *Pattern Recognition and Neural Networks*. Cambridge, U.K.: Cambridge Univ. Press, 1996.

[9] J. A. K. Suykens, L. Lukas, P. Van Dooren, B. De Moor, and J. Vandewalle, “Least squares support vector machine classifiers: A large scale algorithm,” in *Proc. Euro. Conf. Circuit Theory Design*, 1999, pp. 839–842.

[10] W. Chu, C. J. Ong, and S. S. Keerthy, “An improved conjugate gradient method scheme to the solution of least squares SVM,” *IEEE Trans. Neural Netw.*, vol. 16, no. 2, pp. 498–501, Mar. 2005.

[11] S. S. Keerthi and S. K. Shevade, “SMO for least squares SVM formulations,” *Neural Comput.*, vol. 15, pp. 487–507, 2003.

[12] J. A. K. Suykens, L. Lukas, and J. Vandewalle, “Sparse approximation using least square vector machines,” in *IEEE Proc. Int. Symp. Circuits Syst.*, Genvea, Switzerland, 2000, pp. 757–760.

[13] J. A. K. Suykens, J. D. Brabanter, L. Lukas, and J. Vandewalle, “Weighted least squares support vector machines: Robustness and sparse approximation,” *Neurocomput.*, vol. 48, pp. 85–105, 2002.

[14] B. J. de Kruif and T. J. A. de Vries, “Pruning error minimization in least squares support vector machines,” *IEEE Trans. Neural Netw.*, vol. 14, no. 3, pp. 696–702, May 2004.

[15] X. Y. Zeng and X. W. Chen, “SMO-based pruning methods for sparse least squares support vector machines,” *IEEE Trans. Neural Netw.*, vol. 16, no. 6, pp. 1541–1546, Nov. 2005.

[16] J. A. K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle, *Least Squares Support Vector Machines*. Singapore: World Scientific, 2002.

[17] L. Hoegaerts, J. Suykens, J. Vandewalle, and B. De Moor, “Primal space sparse kernel partial least squares regression for large scale problems,” in *IEEE Proc. Int. Joint Conf. Neural Networks*, 2004, pp. 561–566.

[18] G. Gauwenberghs and T. Poggio, “Incremental and decremental support vector machine learning,” in *Proc. Neural Inf. Process. Syst.*, 2000, vol. 13, pp. 668–674.

[19] N. Aronszajn, “Theory of reproducting kernels,” *Trans. Amer. Math. Soc.*, vol. 686, pp. 337–404, 1950.

[20] F. Girosi, “An equivalence between sparse approximation and support vector machine,” *Neural Comput.*, vol. 10, pp. 1455–1480, 1998.

[21] P. Vincent and Y. Bengio, “Kernel matching pursuit,” *Mach. Learn.*, vol. 48, pp. 165–187, 2001.

[22] J. Stoer and R. Bulirsch, *Introduction to Numerical Analysis*. New York: Springer-Verlag, 1993.

[23] J. H. Ma, J. Theiler, and S. Perkins, “Accurate on-line support vector regression,” *Neural Comput.*, vol. 15, pp. 2683–2703, 2003.

[24] L. Csató and M. Opper, “Sparse on-line Gaussian processes,” *Neural Comput.*, vol. 14, pp. 641–668, 2002.

[25] X. D. Zhang, *Matrix Analysis and Application*. Beijing, China: Tsinghua Univ. Press, 2004.

[26] R. Caruana, S. Lawrence, and L. Giles, “Overfitting in neural nets: Backpropagation, gradient, and early stopping,” in *Proc. Neural Inf. Process. Syst.*, 2001, vol. 13, pp. 402–408.

[27] A. J. Smola and P. L. Bartlett, “Sparse greedy Gaussian process regression,” in *Proc. Neural Inf. Process. Syst.*, 2001, vol. 13, pp. 619–625.

[28] S. Mallat and Z. Zhang, “Matching pursuit with time-frequency dictionaries,” *IEEE Trans. Signal Process.*, vol. 41, no. 12, pp. 3397–3415, Dec. 1993.

[29] R. Rifkin and A. Klautau, “In defense of one-vs-all classification,” *J. Mach. Learn. Res.*, vol. 5, pp. 101–141, 2004.

[30] L. Breiman, “Arcing classifiers,” *Ann. Statist.*, vol. 26, no. 3, pp. 801–809, 1998.

[31] D. Michie, D. J. Spiegelhalter, and C. C. Taylor, *Machine Learning, Neural and Statistical Classification*. Englewood Cliffs, NJ: Prentice Hall, 1994 [Online]. Available: ftp.ncc.up.pt/pub/statlog/

[32] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, “Backpropagation applied to handwritten zip code recognition,” *Neural Comput.*, vol. 1, pp. 541–551, 1989.

[33] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[34] C.-C. Chang and C.-J. Lin, “LIBSVM: A library for support vector machines” 2004 [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/libsvm

[35] L. F. Bo, L. Wang, and L. C. Jiao, “Feature scaling for kernel fisher discriminant analysis using leave-one-out cross validation,” *Neural Comput.*, vol. 18, pp. 961–978, 2006.

Licheng Jiao (SM’89) was born in Shaanxi, China, on October 15, 1959. He received the B.S. degree from Shanghai Jiaotong University, Shanghai, China, in 1982, and the M.S. and Ph.D. degrees from Xi’an Jiaotong University, Xi’an, China, in 1984 and 1990, respectively, all in electronical engineering.

From 1984 to 1986, he was an Assistant Professor in the Civil Aviation Institute of China, Tianjing, China. From 1990 to 1991, he was a Postdoctoral Fellow in the National Key Laboratory for Radar Signal Processing, Xidian University, Xi’an, China. Currently, he is the Dean of the Electronic Engineering School and the Director of the Institute of Intelligent Information Processing, Xidian University. He is the author of three books: *Theory of Neural Network Systems* (Xi’an, China: Xidian Univ. Press, 1990), *Theory and Application on Nonlinear Transformation Functions* (Xi’an, China: Xidian Univ. Press, 1992), and *Applications and Implementations of Neural Networks* (Xi’an, China: Xidian Univ. Press, 1996). He is the author or coauthor of more than 150 scientific papers. His current research interests include signal and image processing, nonlinear circuit and systems theory, learning theory and algorithms, optimization problems, wavelet theory, and data mining.

Liefeng Bo was born in Xi’an, China, on March 26, 1978. He received the B.S. degree from the School of Science, Xidian University, Xi’an, China, in 2002, and he is currently working towards the Ph.D. degree in circuits and systems from the Institute of Intelligent Information Processing, Xidian University.

He has published several papers in some leading journals such as *Neural Computation* and IEEE TRANSACTIONS ON NEURAL NETWORKS. His current research interests include kernel-based learning, manifold learning, NNs, and computer version.

Ling Wang was born in Xi’an, China, on November 10, 1978. She received the B.S. degree from the School of Science and the M.S. degree in computer science from Xidian University, Xi’an, China, in 2001 and 2005, respectively, where she is currently working towards the Ph.D. degree in circuits and systems at the Institute of Intelligent Information Processing.

Her current research interests include pattern recognition, statistical machine learning, and image processing.