Sparse LSSVM in Primal using Cholesky Factorization for Large-scale Problems

Shuisheng Zhou

***Abstract*—For SVM learning, LSSVM, derived by duality (D-LSSVM), is a widely used model, because it has an explicit solution. One obvious limitation of the model is that the solution lacks sparseness, which limits it from training large-scale problems efficiently. In this paper, we derive an equivalent LSSVM model in primal space (P-LSSVM) by the representer theorem and prove that P-LSSVM can be solved exactly at some sparse solutions for problems with low-rank kernel matrices. Two algorithms are proposed for finding the sparse (approximate) solution of P-LSSVM by Cholesky factorization. One is based on the decomposition of the kernel matrix K as $PP^\top$ with the best low rank matrix P approximately by pivoting Cholesky factorization. The other is based on solving P-LSSVM by approximating the Cholesky factorization of the Hessian matrix with rank-one update scheme. For linear learning problems, theoretical analysis and experimental results support that P-LSSVM can give the sparsest solutions in all SVM learners. Experimental results on some large-scale nonlinear training problems show that our algorithms, based on P-LSSVM, can converge to acceptable test accuracies at very sparse solutions with a sparsity level less than 1%, and even as little as 0.01%. Hence our algorithms are a better choice for large-scale training problems.**

***Index Terms*—Representer theorem, D-LSSVM, P-LSSVM, Sparse solution, Sparsity level, Cholesky factorization.**

I. Introduction

In a learning problem, an input set $\mathbb{T} = \{(\boldsymbol{x}_1, y_1), \cdots, (\boldsymbol{x}_m, y_m)\}$ is given with samples $\boldsymbol{x}_i \in \mathcal{X} \subset \Re^n$ and corresponding targets $y_i \in \mathcal{Y} \subset \Re$, from which a deterministic function that best represents the relation between the samples and their targets is learned.

According to Vapnik and Chervonenkis' structural risk minimization principle [1], [2], support vector machine (SVM) is a computationally powerful and successful machine learning method. It is widely used in classification and regression problems, such as character identification, disease diagnosis, face recognition, time series prediction, etc. In SVM models for classification with $y_i \in \mathcal{Y} := \{-1, 1\}$, the following optimization problem

$$\begin{aligned} &\min_{\boldsymbol{w}\in\mathcal{H}, b\in\Re, \boldsymbol{\xi}\in\Re^m} \frac{\lambda}{2}\|\boldsymbol{w}\|_{\mathcal{H}}^2 + \sum_{i=1}^m L(\boldsymbol{\xi}_i), \qquad (1)\\ &s.t.\ y_i(\langle \boldsymbol{w}, \phi(\boldsymbol{x}_i)\rangle_{\mathcal{H}} + b) = 1 - \boldsymbol{\xi}_i,\ i = 1, 2, \cdots, m \end{aligned}$$

is solved to determine the optimal classification function $f(\boldsymbol{x}) = \langle \boldsymbol{w}, \phi(\boldsymbol{x})\rangle_{\mathcal{H}} + b$. In (1), $\lambda > 0$ is the regularization parameter. The loss function, $L : \Re \to \Re_+ \cup \{+\infty\}$, has some typical forms for different learning problems (see Table 1 of [3] for some losses and their conjugate functions to represent the dual), and the second item of the objective function is called the Risk of the model (see Chapters 2 and 3 of [4]). In addition, a feature map, $\phi : \Re^n \to \mathcal{H}$, maps input samples $\boldsymbol{x}_i$ to a high dimensional, even infinite dimensional, feature space $\mathcal{H}$ to deal with nonlinear problems, where ϕ is implicitly defined as $K(\boldsymbol{x}, \boldsymbol{z}) = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{z})\rangle_{\mathcal{H}}$ with a well-chosen positive kernel function $K : \Re^n \times \Re^n \to \Re$. Furthermore, $\mathcal{H}$ is a reproducing kernel Hilbert space (RKHS), corresponding to the kernel function $K(\cdot, \cdot)$, admitting the reproducing properties, such as $\langle K(\cdot, \boldsymbol{x}), K(\cdot, \boldsymbol{z})\rangle_{\mathcal{H}} = K(\boldsymbol{x}, \boldsymbol{z})$ for all $\boldsymbol{x}, \boldsymbol{z} \in \Re^n$.

Model (1) is not well-defined, bcause $\mathcal{H}$ is always high dimensional or infinite dimensional. Fortunately, it can be converted into a finite dimensional problem by the representer theorem [5]–[7] or duality [2], [8]–[14], which admits that the solution $\boldsymbol{w} \in \mathcal{H}$ to model (1) can be represented as

$$\boldsymbol{w} = \sum_{j=1}^m \boldsymbol{\alpha}_j \phi(\boldsymbol{x}_j). \qquad (2)$$

For simplicity, one can choose $\phi(\boldsymbol{x}) = K(\cdot, \boldsymbol{x})$ (See Chapter 2 of [13]). Then the output classification function is $f(\boldsymbol{x}) = \sum_{j=1}^m \boldsymbol{\alpha}_j K(\boldsymbol{x}_i, \boldsymbol{x}) + b$.

If the loss has the non-smooth form $L(u) = \max\{0, u\}$, called the hinge loss, the corresponding model (1) is just the standard SVM. Some smooth losses are adopted to improve the convergence properties of the standard SVM. For example, squared hinge loss $L(u) = \frac{1}{2}\max^2\{0, u\}$ is adopted for model (1) in [6], [7], [14]–[18], logistic loss $L_p(u) = \frac{1}{p}\log(1 + \exp(pu))$ in [3], [15]–[17], and Huber loss [19] $L(u) = \begin{cases} \max\{0, u\}, & |u| > \delta \\ \frac{(u+\delta)^2}{4\delta}, & |u| \le \delta \end{cases}$ in [3], [7].

In this paper, we focus on model (1) with the least squares loss $L(u) = \frac{1}{2}u^2$, which is called the **Least Squares Support Vector Machine** (LSSVM) proposed by Suykens et al. [10], [20]. Many researchers [10], [20]–[27] focused on the duality of LSSVM, which can be called as **D-LSSVM**. Experimental comparisons in [21] show that D-LSSVM performs well on many applications, but it has one obvious limitation of the solution lacking sparseness. Hence, it is not fit for training large-scale problems; besides, its test speed is slower than that of the other SVM learners.

Suykens et al. [22] proposed a pruning approach to improve the sparseness of D-LSSVM by iteratively removing some samples (say 5%) with the smallest support value spectrums, i.e., the absolute values of the current solution. Some other

This work was supported by the National Natural Science Foundation of China under Grants 61179040, 61173089 and 11101322, and the Fundamental Research Funds for the Central Universities under Grant JB140713.

The author is with the School of Mathematics and Statistics, Xidian University, Xi'an,710071 China (e-mail: sszhou@mail.xidian.edu.cn).

studies on pruning method can be found in [24]–[26]. All these pruning algorithms have to solve a system of linear equations (gradually decreasing in size) many times, which entails huge computational cost; yet, the final solution will only be an approximate one. Despite their improved test speed, they are still not suitable for training of large-scale problems.

To deal with sparseness, Jiao et al. [27] presented a fast sparse approximation scheme for D-LSSVM, called FSA-LSSVM, in which an approximated classification function was built iteratively by adding the basic function from a kernel-based dictionary, one by one, until the ε criterion was satisfied. Their FSA-LSSVM and PFSA-LSSVM [27] can handle large-scale problems. However, our experimental results, presented in this paper, reveal that those algorithms, if limited by very small sparsity level, will have drawbacks on some training data sets because they are based on D-LSSVM.

All the aforementioned works based on D-LSSVM are expected to solve a system of linear equations $\boldsymbol{A}\boldsymbol{\alpha} = \boldsymbol{h}$, (similar to (5) that follows), with a *nonsingular* symmetric dense matrix $\boldsymbol{A}$ and a dense vector $\boldsymbol{h}$. By solving this exactly, Suykens et al. [10], [28] obtained a unique dense solution. To obtain a sparse solution, Others [22]–[27] solved the linear equations approximately. However, they could obtain the non-sparse solution when the approximation error was set small. All the methods are not applicable to large-scale problems.

Fortunately, there is another equivalent LSSVM model induced by the representer theorem, where (2) is plugged into (1) with the least squares loss. On the lines of naming the model with the hinge loss as PSVM (Primal SVM) by the researches in [6], [7], we term the model (1) with the least squares loss as **P-LSSVM** (Primal LSSVM).

P-LSSVM still solves a system of linear equations $\boldsymbol{A}\boldsymbol{\alpha} = \boldsymbol{h}$, which will be seen in Eq. (8). However, the symmetrical coefficient matrix $\boldsymbol{A}$ is always *singular* if the related kernel matrix is of low rank or can be approximated with a low rank matrix. Hence P-LSSVM may have multiple solutions, which include some sparse solutions. For linear learning problems with $n \ll m$, we illustrate that P-LSSVM has sparse solutions at most n non-zero components, which are the sparsest solutions for all common SVM learners. For nonlinear kernel learning problems, whose kernel matrices are of low rank or can be approximated with low-rank matrices, our studies show that P-LSSVM can give a sparse solution with r non-zero components, where r is the rank of (approximated) kernel matrix. Our experiments demonstrate that P-LSSVM may achieve comparable performance at an approximated solution with a small number of non-zero components.

The fixed-size scheme for LSSVM (called FS-LSSVM in [20]) is a kind of approximated implementation of sparse P-LSSVM. In this scheme, a sample in the active set is randomly selected and replaced iteratively by a randomly selected sample from the training set, if the new sample improves the Rényi quadratic entropy criterion. However, it cannot ensure global optimization because the selection is random. This scheme was further studied in [29] for tuning the parameters of models, but further discussion on the details is beyond the scope of this paper.

In this paper, we propose two algorithms to achieve the sparse solutions of P-LSSVM. The first algorithm, similar but more efficient than the method in [30], is based on the pivoting Cholesky factorization [31], [32]. The pivoting Cholesky factorization, also called incomplete Cholesky factorization (ICF), was used to compute the kernel spectral clustering in [33], [34] and to decrease the SVM's complexity in [12] etc. Here, we efficiently implement ICF to decompose the kernel matrix for P-LSSVM in Algorithm 1. The second algorithm is derived from FSA-LSSVM in [27], but we solve P-LSSVM instead of D-LSSVM, thus achieving better results.

For solving LSSVM, CSI (Cholesky with Side Information) in [35] can be seen as another scheme to P-LSSVM, in which the side information and QR decompositions are equipped with ICF to obtain the low-rank approximation of the kernel matrix. With comparable performances, the proposed algorithms are more efficient than CSI in our experiments.

The remainder of this paper is organized as follows. We review D-LSSVM and derive P-LSSVM in Section II. In Section III, we propose two algorithms to accomplish the sparse solution of P-LSSVM. In Section IV we present some experimental results comparing the algorithms based on P-LSSVM and D-LSSVM, and in Section V the conclusions drawn form this study.

II. D-LSSVM AND P-LSSVM

In this section, besides reviewing D-LSSVM, we introduce P-LSSVM and its useful properties.

A. D-LSSVM—induced by duality

Many research studies (see [10], [20], [22]–[27] and the references therein) focused on D-LSSVM. They study the Lagrange dual of (1) with the least squares loss, and obtain

$$\min_{\boldsymbol{\beta}\in\Re^m} \frac{1}{2}\boldsymbol{\beta}^\top (\boldsymbol{K} + \lambda \boldsymbol{I})\,\boldsymbol{\beta} - \boldsymbol{y}^\top\boldsymbol{\beta},\ s.t.\ \boldsymbol{e}^\top\boldsymbol{\beta} = 0, \tag{3}$$

where the kernel matrix $\boldsymbol{K}$ satisfis $\boldsymbol{K}_{i,j} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and $\boldsymbol{e} := [1, \cdots, 1]^\top$ with a proper dimension. Or, including the offset b in the objective function of (3) as [27], it has an unconstrained form as

$$\min_{\boldsymbol{\beta}\in\Re^m, b\in\Re} \frac{1}{2}\boldsymbol{\beta}^\top (\boldsymbol{K} + \lambda \boldsymbol{I})\,\boldsymbol{\beta} - \boldsymbol{y}^\top\boldsymbol{\beta} + b\boldsymbol{e}^\top\boldsymbol{\beta}. \tag{4}$$

The solutions of (3) and (4) are obtained by solving the following system of linear equations:

$$\begin{bmatrix} \lambda\boldsymbol{I} + \boldsymbol{K} & \boldsymbol{e} \\ \boldsymbol{e}^\top & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\beta} \\ b \end{bmatrix} = \begin{bmatrix} \boldsymbol{y} \\ 0 \end{bmatrix}. \tag{5}$$

Its unique solution is

$$\begin{cases} b &= \frac{\boldsymbol{e}^\top(\lambda\boldsymbol{I}+\boldsymbol{K})^{-1}\boldsymbol{y}}{\boldsymbol{e}^\top(\lambda\boldsymbol{I}+\boldsymbol{K})^{-1}\boldsymbol{e}}, \\ \boldsymbol{\beta} &= (\lambda\boldsymbol{I} + \boldsymbol{K})^{-1}(\boldsymbol{y} - b\boldsymbol{e}). \end{cases} \tag{6}$$

Clearly, $\boldsymbol{\beta}$ in (6) is non-sparse, because $(\lambda\boldsymbol{I}+\boldsymbol{K})^{-1}$ and $(\boldsymbol{y} - b\boldsymbol{e})$ are all dense. So, D-LSSVM is non-sparse in nature.

B. P-LSSVM— induced by Representer Theorem

By the representer theorem [5], the solution $\boldsymbol{w}$ of (1) admits (2). Plugging (2) in (1) with $L(u) = \frac{1}{2}u^2$ and eliminating ξ_i by the equality constraints, we get P-LSSVM as follows:

$$\min_{\boldsymbol{\alpha},b} \frac{1}{2}\boldsymbol{\alpha}^\top(\lambda\boldsymbol{K}+\boldsymbol{K}\boldsymbol{K}^\top)\boldsymbol{\alpha}+\boldsymbol{\alpha}^\top\boldsymbol{K}(\boldsymbol{e}b-\boldsymbol{y})-\boldsymbol{y}^\top\boldsymbol{e}b+\frac{mb^2}{2}. \quad (7)$$

Its solution is obtained by solving the linear equations:

$$\begin{bmatrix} \lambda\boldsymbol{K}+\boldsymbol{K}\boldsymbol{K}^\top & \boldsymbol{K}\boldsymbol{e} \\ \boldsymbol{e}^\top\boldsymbol{K}^\top & m \end{bmatrix}\begin{bmatrix} \boldsymbol{\alpha} \\ b \end{bmatrix} = \begin{bmatrix} \boldsymbol{K} \\ \boldsymbol{e}^\top \end{bmatrix}\boldsymbol{y}, \quad (8)$$

and the label of the new input $\boldsymbol{x}$ is predicted as $\mathtt{sgn}f(\boldsymbol{x})$ with $f(\boldsymbol{x}) = \sum_{j=1}^m \boldsymbol{\alpha}_j K(\boldsymbol{x}_j,\boldsymbol{x}) + b$.

C. Properties of P-LSSVM

The distinction between D-LSSVM and P-LSSVM is that $\lambda\boldsymbol{I}+\boldsymbol{K}$ in the former is replaced by $\lambda\boldsymbol{K}+\boldsymbol{K}\boldsymbol{K}^\top$ in the later. If $\boldsymbol{K}$ is full rank[1], the equivalence between them is simple.

By Gaussian elimination procedure, we have

Proposition 2.1: **i)** No matter whether the kernel matrix is full rank or not, the solutions of D-LSSVM models are unique and dense; **ii)** If the kernel matrix is not full rank, P-LSSVM models have sparse solutions $\boldsymbol{\alpha}$ with $rank(\boldsymbol{K})$ non-zeros.

Now we prove that two related models are equivalent on rebuilding the classification function for any $\boldsymbol{K}$.

Theorem 2.2: P-LSSVM is equivalent to D-LSSVM on classification, that is, **i)** the unique solution of (5) is the solution of (8); **ii)** the output classification function of D-LSSVM (5) meets the same classification function corresponding to all the solutions of P-LSSVM (8).

Proof: **i)** If $(\bar{\boldsymbol{\beta}},\bar{b})$ solves (5), we have $(\lambda\boldsymbol{I}+\boldsymbol{K})\bar{\boldsymbol{\beta}}+\bar{b}\boldsymbol{e} = \boldsymbol{y}$ and $\boldsymbol{e}^\top\bar{\boldsymbol{\beta}} = 0$. It is to be noted that $\boldsymbol{K}^\top = \boldsymbol{K}$, and hence

$$(\lambda\boldsymbol{K}+\boldsymbol{K}\boldsymbol{K}^\top)\bar{\boldsymbol{\beta}}+\boldsymbol{K}\boldsymbol{e}\bar{b} = \boldsymbol{K}\left((\lambda\boldsymbol{I}+\boldsymbol{K})\bar{\boldsymbol{\beta}}+\bar{b}\boldsymbol{e}\right) = \boldsymbol{K}\boldsymbol{y}$$

and $\boldsymbol{e}^\top\boldsymbol{K}^\top\bar{\boldsymbol{\beta}}+m\bar{b} = \lambda\boldsymbol{e}^\top\bar{\boldsymbol{\beta}}+\boldsymbol{e}^\top\boldsymbol{K}\bar{\boldsymbol{\beta}}+\boldsymbol{e}^\top\boldsymbol{e}\bar{b} = \boldsymbol{e}^\top((\lambda\boldsymbol{I}+\boldsymbol{K})\bar{\boldsymbol{\beta}}+\boldsymbol{e}\bar{b}) = \boldsymbol{e}^\top\boldsymbol{y}$. In the other words,

$$\begin{bmatrix} \lambda\boldsymbol{K}+\boldsymbol{K}\boldsymbol{K}^\top & \boldsymbol{K}\boldsymbol{e} \\ \boldsymbol{e}^\top\boldsymbol{K}^\top & m \end{bmatrix}\begin{bmatrix} \bar{\boldsymbol{\beta}} \\ \bar{b} \end{bmatrix} = \begin{bmatrix} \boldsymbol{K} \\ \boldsymbol{e}^\top \end{bmatrix}\boldsymbol{y}.$$

So $(\bar{\boldsymbol{\beta}},\bar{b})$ solves (8).

ii) This is true because all multiple solutions to P-LSSVM correspond to the same classification function. ■

So, P-LSSVM meet the same classification function as those of D-LSSVM, and P-LSSVM may possess sparse solution, which partly overcomes the limitation of D-LSSVM.

D. The sparse solution of P-LSSVM

By eliminating the offset b with $b = \frac{1}{m}(\boldsymbol{e}^\top\boldsymbol{y}-\boldsymbol{e}^\top\boldsymbol{K}\boldsymbol{\alpha})$, P-LSSVM (8) can be simplified to the following system of linear equations:

$$(\lambda\boldsymbol{K}+\boldsymbol{K}\boldsymbol{K}-\tfrac{1}{m}\boldsymbol{K}\boldsymbol{e}\boldsymbol{e}^\top\boldsymbol{K})\boldsymbol{\alpha} = \boldsymbol{K}(\boldsymbol{y}-\tfrac{\boldsymbol{e}^\top\boldsymbol{y}}{m}\boldsymbol{e}). \quad (9)$$

Let $\mathbb{M} = \{1,2,...,m\}$ be the index set of the input samples, and $\boldsymbol{K}$ with $rank(\boldsymbol{K}) = r$ be the positive semi-definite kernel matrix. Since any $r-$rank matrix can be rebuilt by its r linear independent columns, there exists a basic subset $\mathbb{B}\subset\mathbb{M}$ corresponding to the kernel columns of $r(=|\mathbb{B}|)$ input samples, such that the kernel matrix $\boldsymbol{K}$ can be represented as $\boldsymbol{K} := [\boldsymbol{K}_{\mathbb{M}\mathbb{B}} \;\; \boldsymbol{K}_{\mathbb{M}\mathbb{N}}] = [\boldsymbol{K}_{\mathbb{M}\mathbb{B}} \;\; \boldsymbol{K}_{\mathbb{M}\mathbb{B}}\boldsymbol{T}]$ with $\boldsymbol{T}\in\Re^{r\times(m-r)}$. Here $\boldsymbol{K}_{\mathbb{M}\mathbb{B}}\in\Re^{m\times r}$ is the sub-matrix of $\boldsymbol{K}$, whose elements are $K(\boldsymbol{x}_i,\boldsymbol{x}_j)$ for $i\in\mathbb{M}$ and $j\in\mathbb{B}$, and $\boldsymbol{K}_{\mathbb{M}\mathbb{N}}\in\Re^{m\times(m-r)}$, $\boldsymbol{K}_{\mathbb{B}\mathbb{B}}\in\Re^{r\times r}$, $\boldsymbol{K}_{\mathbb{N}\mathbb{N}}\in\Re^{(m-r)\times(m-r)}$ and $\boldsymbol{K}_{\mathbb{N}\mathbb{B}}\in\Re^{(m-r)\times r}$ have similar meanings, where $\mathbb{N} := \mathbb{M}\setminus\mathbb{B}$ is the non-basic subset, including the indexes for the remaining $m-r$ inputs. By the symmetry of $\boldsymbol{K}$, we have $\boldsymbol{T} = \boldsymbol{K}_{\mathbb{B}\mathbb{B}}^{-1}\boldsymbol{K}_{\mathbb{N}\mathbb{B}}^\top$; hence,

$$\boldsymbol{K} = \boldsymbol{K}_{\mathbb{M}\mathbb{B}}\boldsymbol{K}_{\mathbb{B}\mathbb{B}}^{-1}\boldsymbol{K}_{\mathbb{M}\mathbb{B}}^\top. \quad (10)$$

If $rank(\boldsymbol{K}) > r$, $\boldsymbol{K}_{\mathbb{M}\mathbb{B}}\boldsymbol{K}_{\mathbb{B}\mathbb{B}}^{-1}\boldsymbol{K}_{\mathbb{M}\mathbb{B}}^\top$ is only an approximation of $\boldsymbol{K}$, called the Nyström approximation; the approximation error is dominated by $\boldsymbol{K}_{\mathbb{N}\mathbb{N}}-\boldsymbol{K}_{\mathbb{N}\mathbb{B}}\boldsymbol{K}_{\mathbb{B}\mathbb{B}}^{-1}\boldsymbol{K}_{\mathbb{N}\mathbb{B}}^\top$, which is called Schur complement of $\boldsymbol{K}_{\mathbb{B}\mathbb{B}}$ with respect to $\boldsymbol{K}$ [36].

For a given basic set $\mathbb{B}$, there are two methods to obtain the nonzero part $\boldsymbol{\alpha}_{\mathbb{B}}$ of the sparse solution to P-LSSVM (9):

- With a low-rank representation $\boldsymbol{K}_{\mathbb{M}\mathbb{B}}\boldsymbol{K}_{\mathbb{B}\mathbb{B}}^{-1}\boldsymbol{K}_{\mathbb{M}\mathbb{B}}^\top$ for $\boldsymbol{K}$, (9) is solved at
$$\boldsymbol{\alpha}_{\mathbb{B}} = \boldsymbol{H}^{-1}\boldsymbol{K}_{\mathbb{M}\mathbb{B}}^\top\left(\boldsymbol{y}-\tfrac{\boldsymbol{e}^\top\boldsymbol{y}}{m}\boldsymbol{e}\right) \quad (11)$$
with $\boldsymbol{H} = \lambda\boldsymbol{K}_{\mathbb{B}\mathbb{B}}+\boldsymbol{K}_{\mathbb{M}\mathbb{B}}^\top(\boldsymbol{I}_m-\frac{1}{m}\boldsymbol{e}\boldsymbol{e}^\top)\boldsymbol{K}_{\mathbb{M}\mathbb{B}}$.
- If a full-rank sub-matrix $\boldsymbol{A}_{\mathbb{B}\mathbb{B}}$ of the coefficient matrix $\boldsymbol{A} := \lambda\boldsymbol{K}+\boldsymbol{K}\boldsymbol{K}-\frac{1}{m}\boldsymbol{K}\boldsymbol{e}\boldsymbol{e}^\top\boldsymbol{K}$ in (9) is decomposed as $\boldsymbol{L}\boldsymbol{L}^\top$ with $\boldsymbol{L}\in\Re^{r\times r}$ and $\boldsymbol{h}_{\mathbb{B}}$ is the sub-vector of $\boldsymbol{h} := \boldsymbol{K}(\boldsymbol{y}-\frac{\boldsymbol{e}^\top\boldsymbol{y}}{m}\boldsymbol{e})$, then (9) can be solvable at
$$\boldsymbol{\alpha}_{\mathbb{B}} = (\boldsymbol{L}\boldsymbol{L}^\top)^{-1}\boldsymbol{h}_{\mathbb{B}}. \quad (12)$$

So P-LSSVM models can meet the sparse solution, if the kernel matrix $\boldsymbol{K}$ has a low-rank representation, and the cost to achieve the solution with the given basic set $\mathbb{B}$ is $\mathcal{O}(mr^2)$. In Section III, we propose two algorithms to achieve the sparse solution by finding the basic set $\mathbb{B}$ within the cost $\mathcal{O}(mr^2)$.

Remark 2.3: **Linear application.** For linear classifications, let it is be assumed that $\boldsymbol{X} = [\boldsymbol{x}_1 \;\; \boldsymbol{x}_2 \;\; \cdots \;\; \boldsymbol{x}_m]^\top$ and that $\boldsymbol{X}_{\mathbb{B}}$ is constituted by any $r = rank(\boldsymbol{X})(\le n)$ linear independent rows of $\boldsymbol{X}$. Then the output classification function of P-LSSVM model (7) is $f(\boldsymbol{x}) = \boldsymbol{x}^\top\boldsymbol{X}_{\mathbb{B}}^\top\boldsymbol{\alpha}_{\mathbb{B}}^* + b^*$, where $\boldsymbol{\alpha}_{\mathbb{B}}^* = \boldsymbol{H}^{-1}\boldsymbol{X}_{\mathbb{B}}\boldsymbol{X}^\top(\boldsymbol{y}-\frac{\boldsymbol{e}^\top\boldsymbol{y}}{m}\boldsymbol{e})$ and $b^* = \frac{\boldsymbol{e}^\top(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{X}_{\mathbb{B}}^\top\boldsymbol{\alpha}_{\mathbb{B}}^*)}{m}$ with $\boldsymbol{H} = \boldsymbol{X}_{\mathbb{B}}\left[\lambda\boldsymbol{I}_n+\boldsymbol{X}^\top(\boldsymbol{I}_m-\frac{1}{m}\boldsymbol{e}\boldsymbol{e}^\top)\boldsymbol{X}\right]\boldsymbol{X}_{\mathbb{B}}^\top$. Compared with D-LSSVM (5), whose output $f(\boldsymbol{x}) = \boldsymbol{x}^\top\boldsymbol{X}^\top\boldsymbol{\beta}^* + b^*$ with $(\boldsymbol{\beta}^*,b^*)$ has been solved by (6) for $\boldsymbol{K} = \boldsymbol{X}\boldsymbol{X}^\top$, it is clear that the output of D-LSSVM is represented by all m inputs and that of P-LSSVM by any r linear independent inputs only.

The solution of P-LSSVM (7) has, at the most, n non-zeros in the linear case, which are the **least** number of inputs required to present a typical hyperplane in $\Re^n$. Compared with other SVM models (such as C-SVM [2], ν-SVM [37] etc.), we can call this model as the **sparsest LSSVM** of linear machine learners. Toy experiments in Subsection IV-A will further elaborate on this with illustration.

E. Approximation error of sparse P-LSSVM and D-LSSVM

In Section II-D, it is shown that P-LSSVM may have a sparse solution and D-LSSVM the unique non-sparse solution.

[1]This is rare, because the kernel function is a kind of similarity measurement and many such similar samples exist in the training set; so, the kernel matrix may have many (nearly) linear dependent kernel columns.

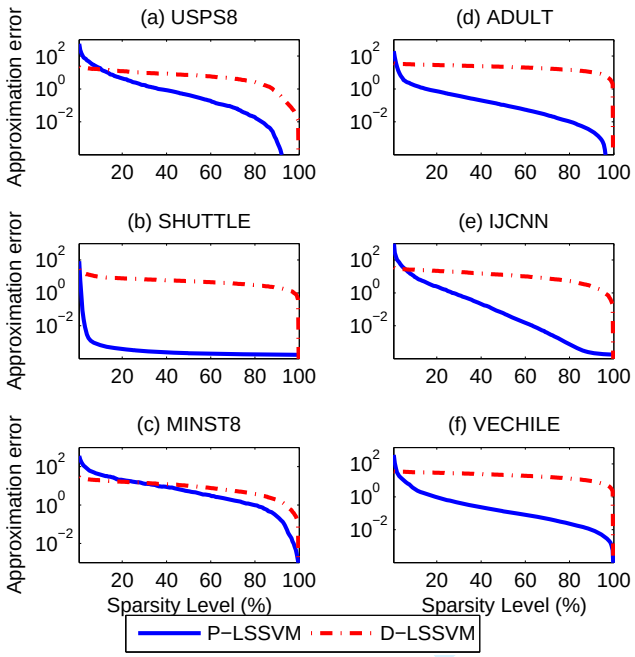

Fig. 1. Comparison of the approximation errors in terms of sparsity levels for P-LSSVM and D-LSSVM on six data sets with 2000 training samples. The approximation errors of P-LSSVM decrease sharply and those of D-LSSVM decrease only when the size of basic set nearly equals the training size.

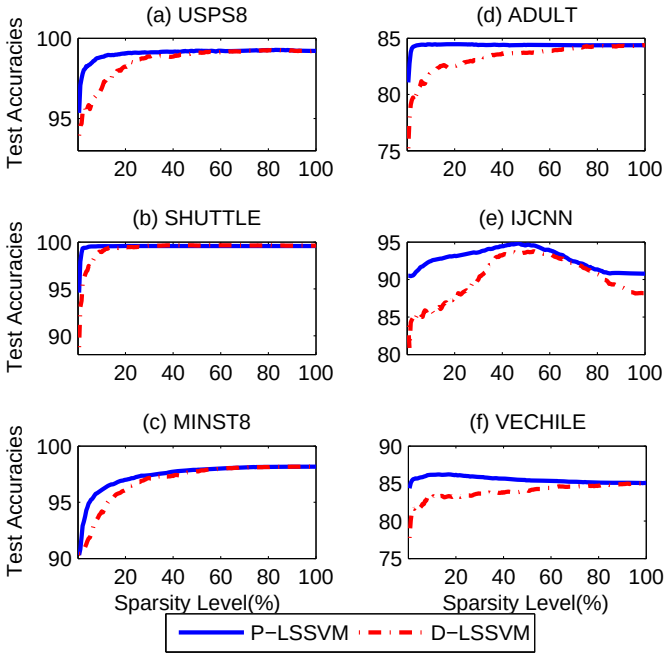

Fig. 2. Comparison of the test accuracies with sparse solutions for P-LSSVM and D-LSSVM on six data sets with 2000 training samples. The test accuracies of P-LSSVM are higher than those of of D-LSSVM, especially P-LSSVM always achieves higher values at lower sparsity levels.

Now we analyze the difference between P-LSSVM and D-LSSVM assuming that the kernel matrix $\boldsymbol{K}$ is approximated by low rank matrix $\boldsymbol{K}_{\mathbb{MB}}\boldsymbol{K}_{\mathbb{BB}}^{-1}\boldsymbol{K}_{\mathbb{MB}}^{\top}$ with a well-chosen $\mathbb{B}$.

Given the basic set $\mathbb{B}$, the sparse solution of D-LSSVM in [27] was obtained at $\hat{\boldsymbol{\beta}}$ with $\hat{\boldsymbol{\beta}}_{\mathbb{N}} = 0$ and $\hat{\boldsymbol{\beta}}_{\mathbb{B}}$ by solving

$$\begin{bmatrix} \lambda \boldsymbol{I} + \boldsymbol{K}_{\mathbb{BB}} & \boldsymbol{e} \\ \boldsymbol{e}^{\top} & 0 \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{\beta}}_{\mathbb{B}} \\ \hat{b} \end{bmatrix} = \begin{bmatrix} \boldsymbol{y}_{\mathbb{B}} \\ 0 \end{bmatrix}. \tag{13}$$

However, the sparse solution of P-LSSVM as obtained at $\bar{\boldsymbol{\alpha}}$ with $\bar{\boldsymbol{\alpha}}_{\mathbb{N}} = 0$, and $\bar{\boldsymbol{\alpha}}_{\mathbb{B}}$ by solving the same size of linear equations (11). The approximation errors of D-LSSVM and P-LSSVM are denoted as ε_D and ε_P respectively:

$$\varepsilon_D = \left\| \begin{bmatrix} \lambda \boldsymbol{I} + \boldsymbol{K} & \boldsymbol{e} \\ \boldsymbol{e}^{\top} & 0 \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{\beta}} \\ \hat{b} \end{bmatrix} - \begin{bmatrix} \boldsymbol{y} \\ 0 \end{bmatrix} \right\|_2, \tag{14}$$

$$\varepsilon_P = \left\| \begin{bmatrix} \lambda \boldsymbol{K} + \boldsymbol{K}\boldsymbol{K}^{\top} & \boldsymbol{K}\boldsymbol{e} \\ \boldsymbol{e}^{\top}\boldsymbol{K}^{\top} & m \end{bmatrix} \begin{bmatrix} \bar{\boldsymbol{\alpha}} \\ \bar{b} \end{bmatrix} - \begin{bmatrix} \boldsymbol{K}\boldsymbol{y} \\ \boldsymbol{e}^{\top}\boldsymbol{y} \end{bmatrix} \right\|_2. \tag{15}$$

Here, we compare them experimentally. We randomly chose 2000 training samples from six large benchmark data sets as inputs (See Section IV for detailed information and parameter settings). The plots of ε_D and ε_P, according to sparsity levels, are shown in Fig. 1, and the corresponding variations of test accuracies with sparsity levels in Fig. 2. All the plots were averaged on 10 random trials. In the experiments, the basic set $\mathbb{B}$ was chosen by the following proposed Algorithm 1 and its size varied from 10 to 2000 in steps of 10. The *Sparsity Level* is defined as

$$Sparsity\ Level := \frac{r(Nonzeros\ of\ solution)}{m(Training\ size)} \times 100\%. \tag{16}$$

In Fig.1, it is clear that the approximation errors of P-LSSVM decrease sharply for most data sets with small sparsity levels, and those of D-LSSVM decrease sharply only when sparsity levels nearly equal 100%, which means the least sparsity. Hence the sparse solution of P-LSSVM model may have smaller approximation error than that of D-LSSVM. At the same time, it can be seen in Fig.2 that the test accuracies of P-LSSVM are always higher than those of D-LSSVM, especially in the case of lower sparsity levels. All the experiments illustrate that P-LSSVM might be a better choice for studying sparseness of LSSVM.

III. Algorithms Based on P-LSSVM for Large-Scale Nonlinear Problems

In this section, we discuss the methods for obtaining the sparse solution of P-LSSVM for large-scale problems, where the full kernel matrix may not be available to fit the memory. In such a situation, P-LSSVM model (9) is unsolvable directly, because it needs $\boldsymbol{K}\boldsymbol{K}^{\top}$ with the cost $O(m^3)$. Besides, some pruning methods based on D-LSSVM, such as [22]–[26], do not work, because they initially need to train the model on the whole input set. Also, methods like [27] will prove to be of poor efficiency if a tiny approximation error is set. Next we discuss the methods to obtain the sparse solution of P-LSSVM efficiently, without $\boldsymbol{K}\boldsymbol{K}^{\top}$.

As has been mentioned above, the problem can be solved by Eq. (11) or (12) with the given basic set $\mathbb{B}$. So, the main task is to determine the basic set $\mathbb{B}$. We started with $\mathbb{B}$ being an empty set and chose one candidate sample to join $\mathbb{B}$ iteratively until some stop criteria were met. There are two different approaches. The first one is to minimize $\|\boldsymbol{K} - \boldsymbol{K}_{\mathbb{MB}}\boldsymbol{K}_{\mathbb{BB}}^{-1}\boldsymbol{K}_{\mathbb{MB}}^{\top}\|$ iteratively with a proper norm $\|\cdot\|$ by enlarging the basic set $\mathbb{B}$ and then getting the solution by (11). The second approach is to solve equation (9) by minimizing

P-LSSVM (7) with an optimized sub-matrix of the coefficient matrix and acquiring the solution by (12).

For the first approach, Smola et al. [30] proposed an algorithm with the cost $\mathcal{O}(\kappa mr^2)$ plus many extra kernel column evaluations to get the *suboptimal* basic set $\mathbb{B}$, where $\kappa = 59$ was set for their *probabilistic speedup trick*, which involves randomly choosing a subset $\widetilde{\mathbb{N}} \subset \mathbb{N}$ with fixed size κ and picking the best one of $\widetilde{\mathbb{N}}$ into the basic set $\mathbb{B}$. In [13], [30], they prove that this trick could suffice to obtain an estimate that is better than 95% of all other estimates with $1 - 0.05$ probability if $\kappa = 59 \approx \frac{\log(0.05)}{\log(0.95)}$. In this study, based on the idea of pivoting Cholesky factorization in [31], [32] for dense positive semi-definite matrices, we propose an algorithm which can obtain the *optimal* basic set $\mathbb{B}$ by minimizing $\mathrm{tr}(\boldsymbol{K} - \boldsymbol{K}_{\mathbb{MB}}\boldsymbol{K}_{\mathbb{BB}}^{-1}\boldsymbol{K}_{\mathbb{MB}}^{\top})$. In the new algorithm, only the diagonal entries and the selected rows corresponding to $\mathbb{B}$ of the kernel matrix $\boldsymbol{K}$ are evaluated, and the total cost is just $\mathcal{O}(mr^2)$.

For the second approach, Jiao et al. [27] minimized D-LSSVM (4) iteratively to obtain an approximated sparse solution that can take the place of the unique dense solution within the cost $\mathcal{O}(mr^2)$, and to reduce the cost to $\mathcal{O}(r^3)$ by the *probabilistic speedup trick* with $\kappa = 146 \approx \frac{\log(0.025)}{\log(0.975)}$. Experimental results show that the methods work well or fail depending on whether the input training data sets are "easy" (test errors probably lower than 5%) or "hard" (test errors probably higher than 5%). However we design an algorithm to minimize P-LSSVM (7) with similar motivation, and the new algorithm works very well for any problem.

A. PCP-LSSVM

In this subsection, we design an iterative method, based on pivoting Cholesky factorization, to simultaneously determine the basic set $\mathbb{B}$ and the factorization of $\boldsymbol{K}$.

With the current basic set $\mathbb{B}_i$ and its corresponding kernel matrix $\boldsymbol{K}_{\mathbb{MB}_i}$, $\widetilde{\boldsymbol{K}}^i := \boldsymbol{K}_{\mathbb{MB}_i}\boldsymbol{K}_{\mathbb{B}_i\mathbb{B}_i}^{-1}\boldsymbol{K}_{\mathbb{MB}_i}^{\top}$ is the low-rank approximation of kernel matrix $\boldsymbol{K}$ based on $\mathbb{B}_i$. Note $\mathbb{N}_i = \mathbb{M} \setminus \mathbb{B}_i$. Let $\boldsymbol{E}^i = \boldsymbol{K} - \widetilde{\boldsymbol{K}}^i$ be the error matrix of the approximation, which is also a positive semi-definite matrix satisfying $\boldsymbol{E}^i_{j,k} = 0$ if $j \in \mathbb{B}_i$ or $k \in \mathbb{B}_i$. Noticing that $\max_{j,k} \boldsymbol{E}^i_{jk} = \max_j \boldsymbol{E}^i_{jj}$ and $\boldsymbol{E}^i = \boldsymbol{0}$ iif $\max_j \boldsymbol{E}^i_{jj} = 0$ (Theorem 4.2.6 in [31]) and that

$$\|\boldsymbol{E}^i\|_2 \le \|\boldsymbol{E}^i\|_F \le \mathrm{tr}(\boldsymbol{E}^i) \tag{17}$$

holds for any positive (semi-)definite matrix $\boldsymbol{E}^i$, we selected the sample corresponding to the maximum diagonal of $\boldsymbol{E}^i$ into the basic set $\mathbb{B}_i$ to minimize $\mathrm{tr}(\boldsymbol{E}^i)$, which will also minimize the upper bounds of $\|\boldsymbol{E}^i\|_2$ and $\|\boldsymbol{E}^i\|_F$. Our idea is very similar to that of the pivoting Cholesky factorization in [31] and the recent realization in [32]. One remarkable advantage of this method is that only the diagonal elements of $\boldsymbol{E}^i$, $\boldsymbol{E}^i_{jj}$, are required, not the whole $\boldsymbol{E}^i$.

To reduce computational complexity, we discuss here how to compute $\boldsymbol{E}^{i+1}_{jj}$ from $\boldsymbol{E}^i_{jj}$ iteratively. Let $t = \arg\max_{j\in\mathbb{N}_i} \boldsymbol{E}^i_{jj}$ be the selected index into the basic set $\mathbb{B}_i$ and the corresponding kernel column be $\boldsymbol{K}_{\mathbb{M}t} := [K(\boldsymbol{x}_1, \boldsymbol{x}_t), \cdots, K(\boldsymbol{x}_m, \boldsymbol{x}_t)]^{\top}$. Set $\mathbb{B}_{i+1} := \mathbb{B}_i \cup \{t\}$ and $\mathbb{N}_{i+1} := \mathbb{N}_i \backslash \{t\}$. We have:

Algorithm 1 ***PCP-LSSVM*—Pivoted Cholesky factorization of P-LSSVM**

Input: Training set $\mathbb{T}$ and kernel function $K(\boldsymbol{x}, \boldsymbol{z})$, bound on residuals ϵ, r_{max}.

Output: Basic set $\mathbb{B}$ and $\boldsymbol{P}$ with $\mathrm{tr}(\boldsymbol{K} - \boldsymbol{P}\boldsymbol{P}^{\top}) \le \epsilon$ or $|\mathbb{B}| = r_{max}$.

$\mathbb{B}_0 = \emptyset$, $\mathbb{N}_0 = \mathbb{M}$, $\boldsymbol{d}^0 = [K(\boldsymbol{x}_1, \boldsymbol{x}_1), \cdots, K(\boldsymbol{x}_m, \boldsymbol{x}_m)]^{\top}$, $\varepsilon_0 = \|\boldsymbol{d}^0\|_1$.
Set $i \leftarrow 0$;
while $\varepsilon_i > \epsilon$ and $i \le r_{max}$ **do**
  $t = \arg\max_{j\in\mathbb{N}_i} \boldsymbol{d}^i_j$;
  Set $\mathbb{B}_{i+1} := \mathbb{B}_i \cup \{t\}$, $\mathbb{N}_{i+1} := \mathbb{N}_i \setminus \{t\}$;
  Calculate $\boldsymbol{K}_{\mathbb{M}t} := [K(\boldsymbol{x}_1, \boldsymbol{x}_t), \cdots, K(\boldsymbol{x}_m, \boldsymbol{x}_t)]^{\top}$;
  if $i = 0$ **then**
    $\boldsymbol{P}^{i+1} := \boldsymbol{p}$ with $\boldsymbol{p} = \boldsymbol{K}_{\mathbb{M}t}/\sqrt{\boldsymbol{K}_{tt}}$;
  else
    $\boldsymbol{P}^{i+1} := [\boldsymbol{P}^i \quad \boldsymbol{p}]$ with $\boldsymbol{p} = \nu^{-1}(\boldsymbol{K}_{\mathbb{M}t} - \boldsymbol{P}^i\boldsymbol{u})$, $\boldsymbol{u} = \boldsymbol{P}^{i\,\top}_{t,\cdot}$ and $\nu = \sqrt{\boldsymbol{K}_{tt} - \boldsymbol{u}^{\top}\boldsymbol{u}}$;
  end if
  $\boldsymbol{d}^{i+1}_j := \boldsymbol{d}^i_j - \boldsymbol{p}^2_j$, $j \in \mathbb{N}_{i+1}$;
  $\varepsilon_{i+1} := \sum_{j\in\mathbb{N}_{i+1}} \boldsymbol{d}^{i+1}_j$;
  $i := i + 1$;
end while
return $\mathbb{B} \leftarrow \mathbb{B}_i$, $\boldsymbol{P} \leftarrow \boldsymbol{P}^i$; $\boldsymbol{\alpha}_B$ and b are solved by (11) with $\boldsymbol{K}_{\mathbb{MB}} = \boldsymbol{P}\boldsymbol{P}_{\mathbb{B}}^{\top}$.

Theorem 3.1: If $\widetilde{\boldsymbol{K}}^i = \boldsymbol{K}_{\mathbb{MB}_i}\boldsymbol{K}^{-1}_{\mathbb{B}_i\mathbb{B}_i}\boldsymbol{K}^{\top}_{\mathbb{MB}_i} := \boldsymbol{P}^i\boldsymbol{P}^{i\,\top}$, then

$$\widetilde{\boldsymbol{K}}^{i+1} := \boldsymbol{K}_{\mathbb{MB}_{i+1}}\boldsymbol{K}^{-1}_{\mathbb{B}_{i+1}\mathbb{B}_{i+1}}\boldsymbol{K}^{\top}_{\mathbb{MB}_{i+1}} = \boldsymbol{P}^{i+1}\boldsymbol{P}^{i+1\,\top}, \tag{18}$$

with $\boldsymbol{P}^{i+1} = [\boldsymbol{P}^i \quad \boldsymbol{p}]$, where $\boldsymbol{p} = \nu^{-1}(\boldsymbol{K}_{\mathbb{M}t} - \boldsymbol{P}^i\boldsymbol{u})$, $\boldsymbol{u}^{\top} = \boldsymbol{P}^i_{t,\cdot}$ being the t-th row of $\boldsymbol{P}^i$ and $\nu = \sqrt{\boldsymbol{K}_{tt} - \boldsymbol{u}^{\top}\boldsymbol{u}}$. Furthermore,

$$\boldsymbol{E}^{i+1}_{jj} := (\boldsymbol{K} - \widetilde{\boldsymbol{K}}^{i+1})_{jj} = \boldsymbol{E}^i_{jj} - \boldsymbol{p}^2_j,\ j \in \mathbb{N}_{i+1}. \tag{19}$$

Proof: If $\boldsymbol{L}^i\boldsymbol{L}^{i\,\top}$ is the Cholesky factorization of $\boldsymbol{K}_{\mathbb{B}_i\mathbb{B}_i}$, then $\boldsymbol{P}^i = \boldsymbol{K}_{\mathbb{MB}_i}(\boldsymbol{L}^i)^{-\top}$. Let $\boldsymbol{K}_{\mathbb{B}_{i+1}\mathbb{B}i+1} = \begin{bmatrix} \boldsymbol{K}_{\mathbb{B}_i\mathbb{B}_i} & \boldsymbol{K}_{\mathbb{B}_i t} \\ \boldsymbol{K}^{\top}_{\mathbb{B}_i t} & \boldsymbol{K}_{tt} \end{bmatrix} := \boldsymbol{L}_{i+1}\boldsymbol{L}^{\top}_{i+1}$. We have

$$\boldsymbol{L}^{i+1} := \begin{bmatrix} \boldsymbol{L}^i & 0 \\ \boldsymbol{u}^{\top} & \nu \end{bmatrix},\ \boldsymbol{u} = (\boldsymbol{L}^i)^{-1}\boldsymbol{K}_{\mathbb{B}_i t},\ \nu = \sqrt{\boldsymbol{K}_{tt} - \boldsymbol{u}^{\top}\boldsymbol{u}}.$$

Since $\boldsymbol{u}^{\top} = \boldsymbol{K}^{\top}_{\mathbb{B}_i t}(\boldsymbol{L}^i)^{-\top}$ is just the t-th row of $\boldsymbol{P}^i$, by

$$(\boldsymbol{L}^{i+1})^{-\top} = \begin{bmatrix} (\boldsymbol{L}^i)^{-\top} & -\nu^{-1}(\boldsymbol{L}^i)^{-\top}\boldsymbol{u} \\ 0 & \nu^{-1} \end{bmatrix},$$

we have

$$\boldsymbol{P}^{i+1} = \boldsymbol{K}_{\mathbb{MB}_{i+1}}(\boldsymbol{L}^{i+1})^{-\top} = [\boldsymbol{P}^i \quad \boldsymbol{p}],$$

$$\boldsymbol{E}^{i+1}_{jj} = \boldsymbol{K}_{jj} - (\boldsymbol{P}^{i+1}\boldsymbol{P}^{i+1\,\top})_{jj} = \boldsymbol{E}^i_{jj} - \boldsymbol{p}^2_j, j \in \mathbb{N}_{i+1}.$$

■

If $\boldsymbol{P}^i_{\mathbb{B}_i}$ is the sub-matrix of $\boldsymbol{P}^i$, composed of rows corresponding to $\mathbb{B}_i$, then $\boldsymbol{L}^i = \boldsymbol{P}^i_{\mathbb{B}_i}$ and $\boldsymbol{K}_{\mathbb{MB}_i} = \boldsymbol{P}^i\boldsymbol{P}^{i\,\top}_{\mathbb{B}_i}$.

The overall algorithm is listed as Algorithm 1, where the pivoting Cholesky factorization is used implicitly. We call the algorithm **PCP-LSSVM**, meaning **P**ivoted **C**holeskian of **P-LSSVM**. From the foregoing analysis, it follows that the cost of updating $\mathbb{B}_{i+1}$ from $\mathbb{B}_i$ is only about $\mathcal{O}(mr_i)$ (most of the cost goes for calculating $\boldsymbol{p}$ to update $\boldsymbol{P}^{i+1}$ in Step 7 of Algorithm 1), where $r_i = |\mathbb{B}_i|$. The total cost is $\mathcal{O}(\sum_{i=1}^{r_{max}} mr_i) = \mathcal{O}(mr_{max}^2)$.

B. ICP-LSSVM

In this part, we propose an algorithm by Jiao et al.'s motivation [27] to solve P-LSSVM, but not D-LSSVM; hence, many better experimental results could be obtained.

It is obvious that (9) is equivalent to the following optimization:

$$\min_{\boldsymbol{\alpha}} f(\boldsymbol{\alpha}) := \frac{1}{2}\boldsymbol{\alpha}^\top \boldsymbol{A}\boldsymbol{\alpha} - \boldsymbol{\alpha}^\top \boldsymbol{h} \tag{20}$$

with $\boldsymbol{A} = \lambda \boldsymbol{K} + \boldsymbol{K}(\boldsymbol{I} - \frac{1}{m}\boldsymbol{e}\boldsymbol{e}^\top)\boldsymbol{K}^\top$, $\boldsymbol{h} = \boldsymbol{K}^\top(\boldsymbol{y} - \frac{\boldsymbol{e}^\top \boldsymbol{y}}{m}\boldsymbol{e})$. Given the basic set $\mathbb{B}_i$ and a candidate sample $\boldsymbol{x}_j$ with $j \in \mathbb{N}_i$, as long as its kernel column $\boldsymbol{K}_{\mathbb{M}j} = [K(\boldsymbol{x}_1, \boldsymbol{x}_j), \cdots, K(\boldsymbol{x}_m, \boldsymbol{x}_j)]^\top$, let

$$\boldsymbol{A}_{\mathbb{B}_i\mathbb{B}_i} = \lambda \boldsymbol{K}_{\mathbb{B}_i\mathbb{B}_i} + \boldsymbol{K}_{\mathbb{M}\mathbb{B}_i}^\top \boldsymbol{K}_{\mathbb{M}\mathbb{B}_i} - \tfrac{1}{m}\boldsymbol{K}_{\mathbb{M}\mathbb{B}_i}^\top \boldsymbol{e}\boldsymbol{e}^\top \boldsymbol{K}_{\mathbb{M}\mathbb{B}_i},$$

$$\boldsymbol{u}_j = \lambda \boldsymbol{K}_{\mathbb{B}_i j} + \boldsymbol{K}_{\mathbb{M}\mathbb{B}_i}^\top \boldsymbol{K}_{\mathbb{M}j} - \frac{\boldsymbol{K}_{\mathbb{M}j}^\top \boldsymbol{e}}{m}\boldsymbol{K}_{\mathbb{M}\mathbb{B}_i}^\top \boldsymbol{e}, \tag{21}$$

$$\mu_j = \lambda \boldsymbol{K}_{jj} + \boldsymbol{K}_{\mathbb{M}j}^\top \boldsymbol{K}_{\mathbb{M}j} - \frac{(\boldsymbol{K}_{\mathbb{M}j}^\top \boldsymbol{e})^2}{m}. \tag{22}$$

Noting

$$f_{\mathbb{B}_i}^* := \min_{\boldsymbol{\alpha}_{\mathbb{B}_i}} \frac{1}{2}\boldsymbol{\alpha}_{\mathbb{B}_i}^\top \boldsymbol{A}_{\mathbb{B}_i\mathbb{B}_i}\boldsymbol{\alpha}_{\mathbb{B}_i} - \boldsymbol{h}_{\mathbb{B}_i}^\top \boldsymbol{\alpha}_{\mathbb{B}_i} \tag{23}$$

with $\boldsymbol{h}_{\mathbb{B}_i} = \boldsymbol{K}_{\mathbb{M}\mathbb{B}_i}^\top(\boldsymbol{y} - \frac{\boldsymbol{e}^\top \boldsymbol{y}}{m}\boldsymbol{e})$, we define

$$f_{\mathbb{B}_i\cup\{j\}}^* := \min_{\boldsymbol{\alpha}_j} \phi(\boldsymbol{\alpha}_j), j \in \mathbb{N}_i, \tag{24}$$

where $\phi(\boldsymbol{\alpha}_j) := \frac{1}{2}\begin{bmatrix} \boldsymbol{\alpha}_{\mathbb{B}_i}^* \\ \boldsymbol{\alpha}_j \end{bmatrix}^\top \begin{bmatrix} \boldsymbol{A}_{\mathbb{B}_i\mathbb{B}_i} & \boldsymbol{u}_j \\ \boldsymbol{u}_j^\top & \mu_j \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha}_{\mathbb{B}_i}^* \\ \boldsymbol{\alpha}_j \end{bmatrix} - \begin{bmatrix} \boldsymbol{h}_{\mathbb{B}_i} \\ \boldsymbol{h}_j \end{bmatrix}^\top \begin{bmatrix} \boldsymbol{\alpha}_{\mathbb{B}_i}^* \\ \boldsymbol{\alpha}_j \end{bmatrix}$ with $\boldsymbol{\alpha}_{\mathbb{B}_i}^* = \boldsymbol{A}_{\mathbb{B}_i\mathbb{B}_i}^{-1}\boldsymbol{h}_{\mathbb{B}_i}$ being the solution of (23) and

$$\boldsymbol{h}_j = \boldsymbol{K}_{\mathbb{M}j}^\top(\boldsymbol{y} - \tfrac{\boldsymbol{e}^\top \boldsymbol{y}}{m}\boldsymbol{e}). \tag{25}$$

Then, we choose the basic sample $\boldsymbol{x}_t$ such that

$$t = \arg\min_{j\in\mathbb{N}_i} f_{\mathbb{B}_i\cup\{j\}}^*. \tag{26}$$

This is similar to the "Backfitting" in [27]. Problem (24) can be simplified to $f_{\mathbb{B}_i\cup\{j\}}^* = f_{\mathbb{B}_i}^* - \delta_j$ with

$$\delta_j = \frac{(\boldsymbol{u}_j^\top \boldsymbol{\alpha}_{\mathbb{B}_i} - \boldsymbol{h}_j)^2}{2\mu_j}. \tag{27}$$

Then (26) is equivalent to

$$t = \arg\max_{j\in\mathbb{N}_i} \delta_j. \tag{28}$$

With the chosen index t, set $\mathbb{B}_{i+1} = \mathbb{B}_i \cup \{t\}$ and $\mathbb{N}_{i+1} = \mathbb{N}_i \setminus \{t\}$. To update $\boldsymbol{\alpha}_{\mathbb{B}_{i+1}}^*$ from $\boldsymbol{\alpha}_{\mathbb{B}_i}^*$, let $\boldsymbol{A}_{\mathbb{B}_i\mathbb{B}_i} = \boldsymbol{L}^i\boldsymbol{L}^{i\top}$ be the Cholesky factorization of $\boldsymbol{A}_{\mathbb{B}_i\mathbb{B}_i}$ and set $\boldsymbol{A}_{\mathbb{B}_{i+1}\mathbb{B}_{i+1}} = \begin{bmatrix} \boldsymbol{A}_{\mathbb{B}_i\mathbb{B}_i} & \boldsymbol{u}_t \\ \boldsymbol{u}_t^\top & \mu_t \end{bmatrix} := \boldsymbol{L}^{i+1}\boldsymbol{L}^{i+1\top}$. Then we have

$$\boldsymbol{L}^{i+1} := \begin{bmatrix} \boldsymbol{L}^i & 0 \\ \boldsymbol{v}_t^\top & \nu_t \end{bmatrix}, \tag{29}$$

$$\boldsymbol{\alpha}_{\mathbb{B}_{i+1}}^* := \begin{bmatrix} \boldsymbol{\alpha}_{\mathbb{B}_i}^* \\ 0 \end{bmatrix} + \frac{\boldsymbol{u}_t^\top \boldsymbol{\alpha}_{\mathbb{B}_i}^* - \boldsymbol{h}_t}{\nu_t^2}\begin{bmatrix} (\boldsymbol{L}^i)^{-\top}\boldsymbol{v}_t \\ -1 \end{bmatrix}, \tag{30}$$

with $\boldsymbol{v}_t = (\boldsymbol{L}^i)^{-1}\boldsymbol{u}_t$, $\nu_t = \sqrt{\mu_t - \boldsymbol{v}_t^\top \boldsymbol{v}_t}$.

Remark 3.2: **Probabilistic speedup trick.** From the foregoing analysis, it follows that most of the cost goes to compute $\boldsymbol{u}_j$ with the computational complexity $\mathcal{O}(mr_i)$. If we want to select the best sample from the candidate set $\mathbb{N}_i$, the total cost would go up to $\mathcal{O}(m^2 r_i)$, which is more than what we can afford. The *probabilistic speedup trick* proposed in [30], used in [6], [27], [38], can be adopted to reduce the cost with very little performance loss. For our situation, $\kappa = 59$ is enough as the default setting and $\kappa = 22 \approx \frac{\log(0.1)}{\log(0.9)}$ is also good enough.

The algorithm is listed in Algorithm 2, where we call **ICP-LSSVM**(**I**terative **C**holeskian **P-LSSVM**). The total cost is $O(\kappa m r_{max}^2)$ plus some extra kernel column evaluations.

Algorithm 2 ICP-LSSVM—Iterative Choleskian P-LSSVM

Input: Input set $\mathbb{T}$ and kernel function $K(\boldsymbol{x}, \boldsymbol{z})$; parameters κ, ϵ and r_{max}.

Output: Basic set $\mathbb{B}$ and the sparse solution $\boldsymbol{\alpha}_{\mathbb{B}}$.

1: Randomly select $\widetilde{\mathbb{N}} \subset \mathbb{N}_0 := \mathbb{M}$ with $|\widetilde{\mathbb{N}}| = \kappa$, compute μ_j by (22) and $\boldsymbol{h}_j$ by (25) for $j \in \widetilde{\mathbb{N}}$;
2: $t = \arg\max_{j\in\widetilde{\mathbb{N}}} \frac{\boldsymbol{h}_j^2}{2\mu_j}$;
Set $\mathbb{B}_1 := \{t\}$ and $\mathbb{N}_1 := \mathbb{N}_0 \setminus \{t\}$;
Set $\boldsymbol{L}^1 := \sqrt{\mu_t}$, $\boldsymbol{\alpha}_{\mathbb{B}_1} := \frac{\boldsymbol{h}_t}{\mu_t}$;
$i := 1$;
3: **while** $i \le r_{max}$ **do**
4: Randomly select $\widetilde{\mathbb{N}} \subset \mathbb{N}_i$ with $|\widetilde{\mathbb{N}}| = \kappa$, compute $\boldsymbol{u}_j$, $\boldsymbol{h}_j$, μ_j and δ_j by (21), (22), (25) and (27) for $j \in \widetilde{\mathbb{N}}$;
$\delta_t := \max_{j\in\widetilde{\mathbb{N}}} \delta_j$;
5: **if** $\delta_t \le \epsilon$ **then**
6: **return** $\mathbb{B} := \mathbb{B}_i$ and $\boldsymbol{\alpha}_{\mathbb{B}}$.
7: **else**
8: Set $\mathbb{B}_{i+1} := \mathbb{B}_i \cup \{t\}$ and $\mathbb{N}_{i+1} := \mathbb{N}_i \setminus \{t\}$;
Update $\boldsymbol{L}^{i+1}$ and $\boldsymbol{\alpha}_{\mathbb{B}_{i+1}}$ by (29) and (30);
9: **end if**
10: $i := i + 1$;
11: **end while**
12: **return** $\mathbb{B} := \mathbb{B}_i$, and $\boldsymbol{\alpha}_{\mathbb{B}}$.

C. Comments on the two algorithms

By PCP-LSSVM, we accomplish the optimal rank-r approximation $\widetilde{\boldsymbol{K}}$ of $\boldsymbol{K}$ with respect to trace norm. This algorithm may be independent of P-LSSVM and can be used for other SVM models with different loss functions, such as finding the better reduced set for large-scale RSVM [6], [12], [16], [17], [39]. It can also be used for some other kernel learning

problems, such as predictive low-rank decomposition [35], Gaussian Process [38], kernel matching pursuit [40] etc. As this algorithm is also independent of the regularizer parameter λ and targets $\boldsymbol{y}$, it can be chosen to quickly tune the parameter λ for LSSVM or other SVM models, and generalized to learn the multiple classes or multiple targets problems efficiently.

According to ICP-LSSVM, its output can provide the smaller structural risk for (1) as well as the smaller generalization errors with the given targets $\boldsymbol{y}$ and regularizer parameter λ.

If the size of the basic set is large enough to be nearly the same as the rank of kernel matrix $\boldsymbol{K}$, the performances of PCP-LSSVM and ICP-LSSVM will be equal. However, with smaller basic set size, PCP-LSSVM obtains lower error bound between $\boldsymbol{K}$ and $\widetilde{\boldsymbol{K}}$, and ICP-LSSVM gives smaller generalization error.

IV. Experimental Results

In this section, we discuss some experiments performed to evaluate the sparseness of P-LSSVM and D-LSSVM. All the experiments were run on a PC operating on Windows 7 with Matlab 7.10 and an Intel Core i3 2100 CPU having a maximum of 4Gbytes of memory.

The first set of experiments in Section IV-A, performed on a very small toy linear problem, shows that P-LSSVM can get the sparsest solution in all related SVM learners for linear classifications with low input dimension.

The second set of experiments in Section IV-B were carried out on some large-scale nonlinear problems. In this case, only those LSSVM algorithms that are suitable for training large-scale problems, such as PCP-LSSVM and ICP-LSSVM[2], FSA-LSSVM and PFSALS-SVM in [27][3], and CSI [35][4], were considered. Other pruning algorithms, such as those in [22]–[26], which are unsuitable for large-scale training, were not considered. The results obtained by some popular SVM learners are also referenced for comparison.

In Section IV-C, the third set of experiments, on two very large data sets, are provided to compare P-LSSVM and D-LSSVM with very low sparsity levels.

A. Linear toy experiments

An inseparable data set for linear classification was randomly sampled, which comprised 400 training samples and 1000 test samples. We compared P-LSSVM model with traditional D-LSSVM [10], [20], Suykens et al.'s pruning LS-SVM [22], Jiao et al.'s [27] FAS-LSSVM, and the standard SVM ([2], [8], [9], [37]). The training samples were plotted as squares (□) and triangles (▽), and the "support vectors" as bullets (•) and diamonds (⧫) respectively (see Fig. 3). The test samples are not shown in figures. In Figs. 3(b), 3(c) and 3(d), the sizes of bullets and diamonds are proportional to their weights. The test errors are 5.4%, 5.4%, 5.6%, 6.1% and 5.6% for the algorithms corresponding respectively to Figs. 3(a)-3(e) on 1000 test samples.

In Fig. 3, it can be seen that P-LSSVM has only two support vectors, whereas the original D-LSSVM model has all the 400 inputs as support vectors. However, they meet in the same classification hyperplane in $\Re^2$. With similar test accuracies, Suykens et al.'s pruning LS-SVM [22] has 93 support vectors, Jiao et al.'s FSA-LSSVM [27] 152 support vectors, and the standard SVM 55 support vectors. It is thus clear that P-LSSVM gives the sparsest results in all the SVM learners. Some other experiments corroborate this conclusion.

Actually, any two linear independent training samples in Fig. 3(a) can be the "support vectors" for P-LSSVM, no matter to which class they belong. That is to say, for P-LSSVM, the sample with non-zero weight has already lost the meaning as the "support vector". Yet, for consistency, we call them "support vector".

B. Comparisons on large-scale nonlinear problems

Six large benchmark data sets from the site of [41] were adopted, and some of them can be found in [6] and [42] also. Gaussian kernel function $K(\boldsymbol{x}, \boldsymbol{z}) = \exp(-\gamma\|\boldsymbol{x} - \boldsymbol{z}\|^2)$, with different spread parameters γ, was used for all data sets. The kernel spread parameters γ and the tradeoff parameters λ listed in Table I were chosen roughly by cross-validation with $\gamma \in \{2^{-9}, \cdots, 2^3\}$ and $\lambda \in \{10^{-6}, \cdots, 10^0\}$. The details of the data sets are listed below:

- **USPS8**: This is a multi-class data set with 10 classes comprising 7,291 training samples and 2,007 test samples. Each sample has 256 features. Here, the task of classifying the digit 8 versus the rest classes was trained.
- **SHUTTLE**: This is a data set with seven classes comprising 43,500 training examples and 14,500 test examples. Each example has 9 features. Here, a binary classification problem is trained to separate class 1 from the rest.
- **MNIST8**: This is a multi-class data set to classify the handwritten digits 0 to 9. It has 60,000 training examples and 10,000 test examples. Each example is described by 28×28 features. Here, the task of classifying digit 8 versus other classes was solved.
- **ADULT**: This is the version given by Platt [9], which has 32,561 training examples and 16,281 test examples. Each example has 123 binary features.
- **IJCNN**: This has 49,990 training examples and 91,701 test examples. Each example is described by 22 features.
- **VECHILE**: This is the combined SensIT Vechile in [41]. It has 78,823 training examples and 19,705 test examples in three classes. Each example has 100 features. Here, the task of classifying class 3 from the rest was trained.

The objective of this study is to limit the size of basic set to smaller than $1\%m$ or the sparsity level to less than 1% for large-scale training problems. The only exception is on USPS8 with sparsity level 2.7%, because it has a smaller training size. We performed some experiments comparing P-LSSVM and D-LSSVM to illustrate their sparseness solutions to those problems, based on a very small basic set (very low sparsity level). The proposed algorithms are compared with

[2]Example Matlab codes are available in http://web.xidian.edu.cn/sszhou/en/paper.html.

[3]The algorithms were reprogrammed for early stopping according to authors' code in http://homes.cs.washington.edu/~lfb/software/FSALS-SVM.htm.

[4]Codes are available http://www.di.ens.fr/~fbach/csi/index.html.

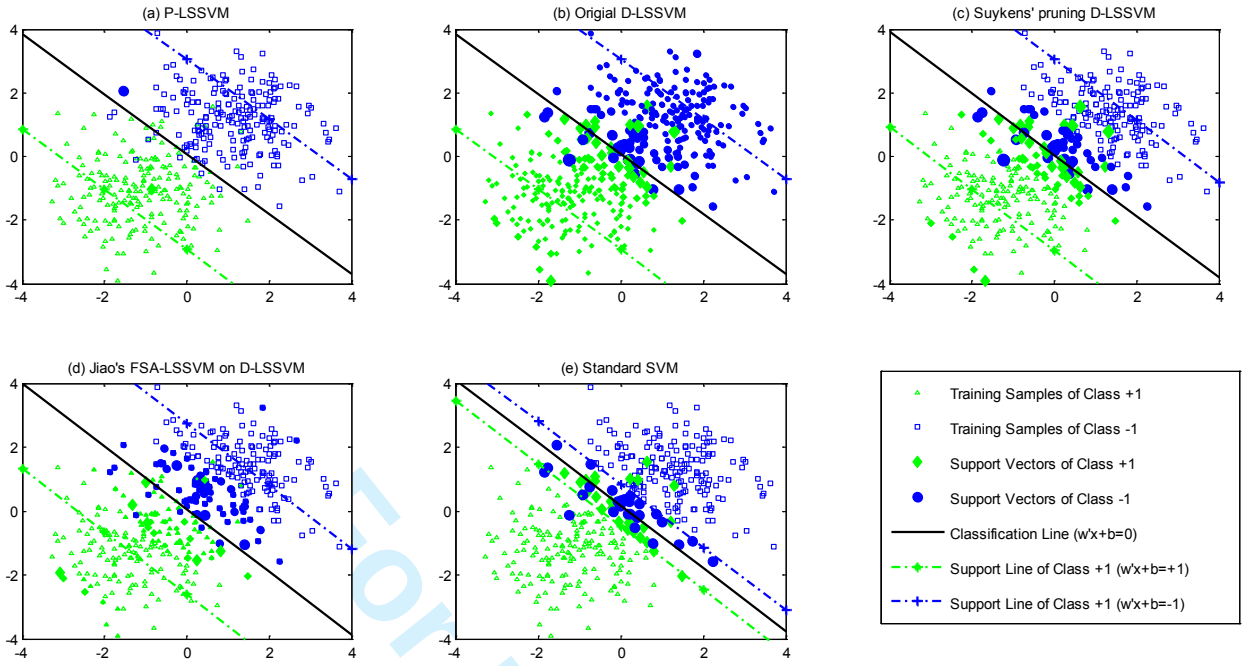

Fig. 3. Linear classification experiments. The training samples are plotted as squares(□) and triangles (▽), and the "support vectors" are plotted as bullets(●) and diamonds (♦)(In (b), (c), (d), their sizes are proportional to their weights). The numbers of the support vector are respectively 2, 400, 93, 152 and 55 for (a)-(e), and the test errors for the corresponding algorithms are 5.4%, 5.4%, 5.6%, 6.1% and 5.6% on 1000 test samples.

FSA-LSSVM and PFSA-LSSVM [27], as well as CSI [35]. All the algorithms were *early stopped* with the very small size of basic set.

We also included in the simplest algorithms, RRP-LSSVM (Random Reduced P-LSSVM) and RRD-LSSVM (Random Reduced D-LSSVM), where P-LSSVM was solved by (11) and D-LSSVM by (13) respectively with the random-chosen basic sets. ICP_{22}-LSSVM is the simple version of ICP-LSSVM with $\kappa = 22 \approx \frac{\log(0.1)}{\log(0.9)}$.

The plots in Fig. 4 show the variations in test accuracy with sparsity level on six data sets; information on the data sets and the parameters of the experiments are given briefly in Table I.

All the plots were averaged over 10 random trials. The curves were plotted as solid lines with different markers for the algorithms based on P-LSSVM, and as dash or dotted lines with different markers for the algorithms based on D-LSSVM. We also plotted the horizontal line, called the Majority Level, which is the test accuracy obtained by a very weaker classifier where all the test samples were categorized into the majority.

The chosen data sets can be classified into two classes: the "hard" data sets with test accuracy of less than 95% and the "easy" data sets with test accuracy more than 95%.

The results in Fig.4 show that the algorithms based on P-LSSVM work well on all the data sets, whereas those based on D-LSSVM work well only on "easy" data sets (USPS8, SHUTTLE and MINIST8), but fail on "hard" data sets (ADULT, IJCNN and VECHILE) within the limitation of sparsity levels. They are even much worse than RRP-LSSVM. Especially, FSA-LSSVM and PFSA-LSSVM are worse than RRD-LSSVM and their test accuracies are lower than Majority Levels in Figs. 4(d) and 4(e).

Next, we performed some experiments to understand why the algorithms based on D-LSSVM are much worse than the algorithms of P-LSSVM on "hard" data sets. With fixed maximum sizes of the basic sets, we trained related algorithms on "hard" data sets (ADULT, IJCNN and VECHILE) with small and medium sizes of training samples and tested on the whole test sets. The results are plotted in Fig. 5.

From a comparison of Fig. 5 with the second row of Fig. 4, we conclude that the algorithms on P-LSSVM are always stable and maintain a high level on "hard" data sets. The differences between the small cases (the first row of Fig. 5), the medium case (the second row of Fig. 5) and the full case (the second row of Fig. 4) are almost negligible. However, the algorithms on D-LSSVM are not stable under all conditions. The basic set being of fixed size, the gap between P-LSSVM and D-LSSVM is small for the small training data sets and becomes larger when the training set enlarges. This is because the sparseness of small data sets is weaker than that of the big data sets in our settings. So, D-LSSVM is not suitable to obtain very sparse solutions for "hard" data sets.

The details test accuracies and the training times on all the six data sets, for the given basic set sizes, are presented in Table I together with brief information on the data sets and the parameters of the experiments. All the results were obtained on an average over 20 random trials. Recently reported results [42] on part data sets by some state-of-the-art SVM solvers are also presented as Table's footnotes for reference.

From Figs. 4 & 5 and Table I, we draw the following conclusions:

- The best results obtained by the sparse LSSVM, with very small basic sets, are comparable to the results obtained recently by the state-of-the-art methods [17], [42]. The

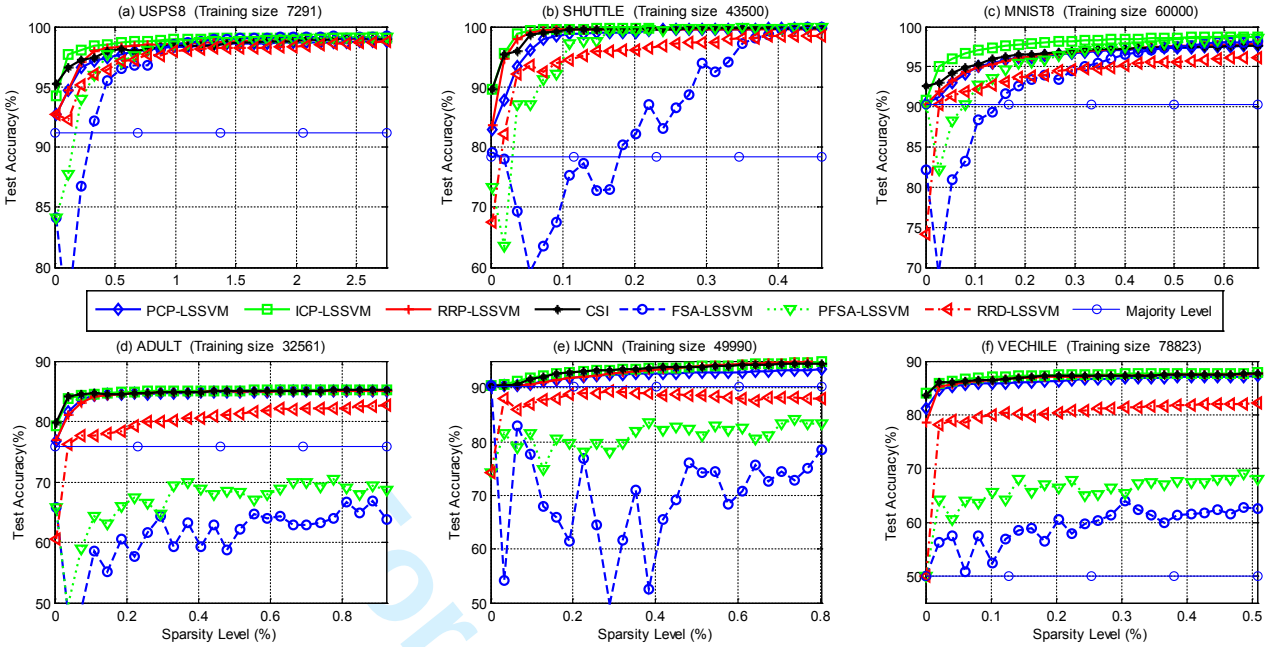

Fig. 4. Plots for comparing the convergence of P-LSSVM and D-LSSVM on related algorithms with respect to the sparsity levels. The plots show that all the algorithms based on P-LSSVM (PCP-LSSVM, ICP-LSSVM, RRP-LSSVM and CSI) converge well to the ideal results, while these algorithms based on D-LSSVM (FSA-LSSVM, PFSA-LSSVM and RRD-LSSVM) converge to the acceptable results only in respect of the "easy" data sets (a), (b), (c) and not "hard" data sets (d), (e), (f).

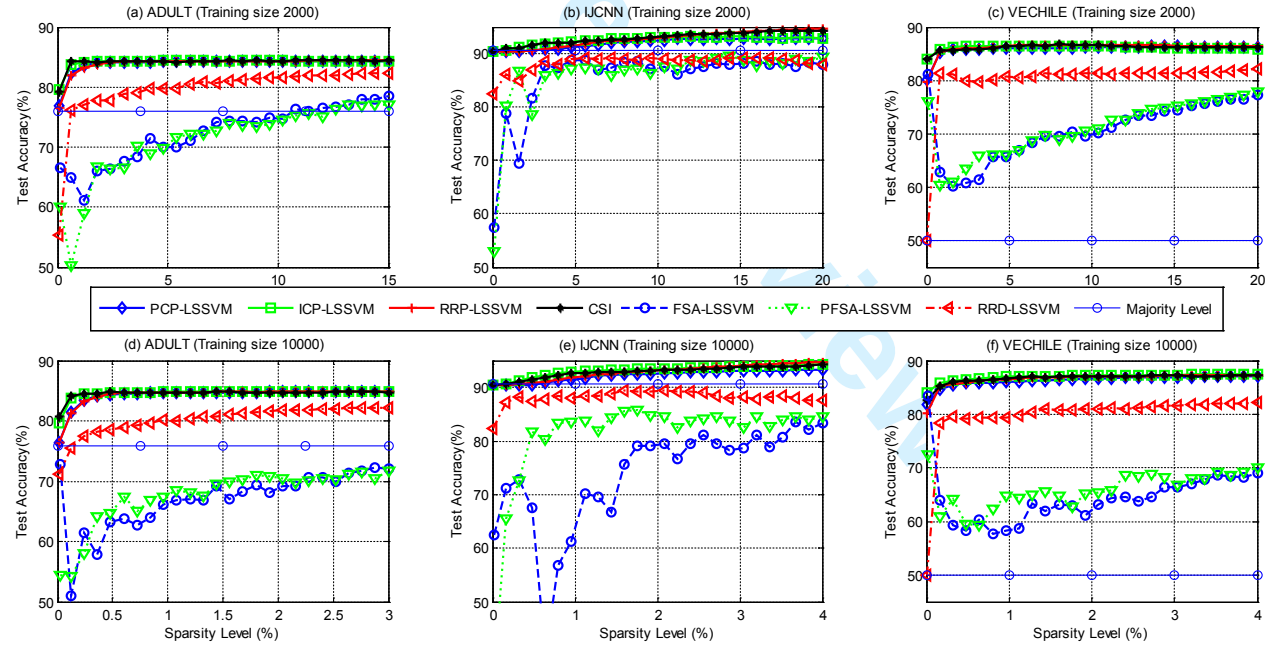

Fig. 5. Plots for comparing the convergence of P-LSSVM and D-LSSVM with related algorithms on "hard" data sets with different training sizes, in term of sparsity levels. From a comparison between the plots of the upper (2,000 training samples) and lower (10,000 training samples) rows, it is clear that the algorithms based on P-LSSVM are stable on any size of training data set, while those based on D-LSSVM become worse with increasing training set size.

differences between test accuracies are less than 1%.

- All the algorithms based on P-LSSVM lead to ideal results on all data sets. Considering the very sparse solution and the less training time, we find that the final test accuracies are comparable to the reported results [6], [17], [18], [27], [42].
- For "easy" data sets, all the algorithms converge well. For "hard" data sets, the algorithms based on P-LSSVM work well, but those based on D-LSSVM cannot reach positive results within the sparsity limitation. They are even worse than the random selection methods (RRP-LSSVM, RRD-LSSVM).
- ICP-LSSVM is the slower one (a little quicker than CSI). However, it always converges to better test accuracies

TABLE I
COMPARISON OF TEST ACCURACIES (%) AND TRAINING TIMES (SECONDS) OF RELATED ALGORITHMS ON SIX LARGER SIZE BENCHMARK DATA SETS. ALL THE RESULTS WERE OBTAINED ON AN AVERAGE OVER 20 RANDOM TRIALS. THE STANDARD DEVIATIONS WERE GIVEN IN BRACKETS. THE BEST VALUES ARE SHOWN IN BOLD AND THE NON-CONVERGENT VALUES (MUCH LESS THAN THE POSITIVE VALUE) IN ITALICS.

	USPS8[1)]	SHUTTLE	MNIST8[2)]	ADULT[3)]	IJCNN	VECHILE
Training Size	7291	43500	60000	32561	49990	78823
Test Size	2007	14500	10000	16281	91790	19705
Dimension	256	9	784	123	22	100
Max. Basic Size	200	200[4)]	400	300	400	400
Sparsity Level	2.7%	0.46%	0.67%	0.92%	0.8%	0.51%
λ	10^{-5}	10^{-5}	10^{-4}	10^{-1}	10^{-5}	10^{-2}
γ	2^{-7}	2	2^{-6}	2^{-6}	1	2^{-3}
	Test Accuracies(%)					
RRP-LSSVM	99.00(0.06)	99.55(0.06)	97.88(0.14)	85.09(0.06)	95.11(0.16)	87.57(0.07)
PCP-LSSVM	98.84(0.14)	99.71(0.02)	97.71(0.08)	85.08(0.10)	93.48(0.08)	87.24(0.03)
ICP-LSSVM	99.11(0.05)	99.79(0.02)	**98.77**(0.05)	**85.23**(0.08)	94.84(0.28)	**87.82**(0.05)
ICP_{22}-LSSVM	99.11(0.06)	99.79(0.02)	98.71(0.07)	**85.23**(0.05)	**95.12**(0.18)	87.80(0.04)
CSI	99.00(0.00)	99.79(0.00)	97.85(0.00)	85.22(0.00)	94.54(0.00)	87.59(0.00)
FSA-LSSVM	**99.22**(0.10)	**99.90**(0.07)	98.32(0.17)	*65.17*(3.93)	*78.29*(6.93)	*62.08*(4.24)
PFSA-LSSVM	99.15(0.18)	99.85(0.02)	98.55(0.18)	*70.02*(4.05)	*82.83*(4.02)	*68.03*(3.64)
RRD-LSSVM	98.56(0.27)	98.55(0.55)	95.91(0.42)	82.64(0.49)	87.45(1.88)	82.26(1.05)
	Training time(seconds)					
RRP-LSSVM	0.43(0.07)	0.39(0.02)	17.85(0.22)	1.38(0.04)	1.34(0.05)	4.80(0.07)
PCP-LSSVM	0.67(0.01)	1.72(0.03)	23.54(0.19)	3.24(0.05)	6.11(0.12)	13.62(0.17)
ICP-LSSVM	6.75(0.17)	13.43(0.83)	162.46(0.73)	34.32(0.30)	73.87(0.35)	122.38(0.37)
ICP_{22}-LSSVM	3.04(0.10)	4.83(0.28)	79.50(0.40)	15.53(0.21)	32.12(0.35)	54.43(0.22)
CSI	6.25(0.06)	16.49(0.15)	159.52(0.24)	37.05(0.17)	108.46(0.23)	192.03(0.11)
FSA-LSSVM	0.67(0.02)	1.58(0.04)	23.64(0.18)	3.12(0.04)	5.87(0.11)	13.20(0.13)
PFSA-LSSVM	0.58(0.01)	0.81(0.01)	6.31(0.30)	1.49(0.17)	2.36(0.16)	3.72(0.19)
RRD-LSSVM	0.18(0.01)	0.02(0.00)	2.11(0.03)	0.13(0.01)	0.07(0.00)	0.24(0.01)

[1)] Results in [42]: `Pegasos` (99.54%, 120s)–means achieve 99.54% test accuracy with 120s training time, `SDCA`(99.49%, 21s), `LASVM` 99.54%,1.8s), `SVMlight` (99.54%, 3.3s), `Optima SVM` (99.54%, not reported).
[2)] Results in [42]: `Pegasos` (99.40%, 4200s), `SDCA` (99.44%,1800s), `LASVM` (99.44%,280s), `SVMlight` (99.42%, 290s), `Optima SVM` (99.43%, not reported).
[3)] Results in [42]: `Pegasos` (84.5%, 30s), `SDCA` (84.5%,4.8s), `LASVM` (84.4%,1.5s) `SVMlight` (84.9%, 59s), `Optima SVM` (85.1%, not reported).
[4)] The resulting basic set is nearly 107 for ICP-LSSVM and 94 for ICP_{22}-LSSVM, much less than the limitation.

with smaller basic set; so, it may fit the very small sparsity level learning. For example, on SHUTTLE, ICP-LSSVM was stopped when the size of the basic set was nearly 107 and ICP_{22}-LSSVM was stopped when the size of the basic set was nearly 94, which is far less than 200.

- ICP_{22}-LSSVM decreases the training time to half of ICP-LSSVM, while the performance is comparable.
- An interesting point is that FSA-LSSVM is always worse than PFSA-LSSVM on the "hard" data sets. As FSA-LSSVM is selected as the best candidate sample from the whole non-basic set, and PFSA-LSSVM from the small subset of the non-basic set, we can conclude that this case may be considered a kind of over-fitting one for FSA-LSSVM. It also suggests that a very large κ is unnecessary for ICP-LSSVM.

The results reported in [27], based on D-LSSVM by FSA-LSSVM and PFSA-LSSVM, are better because the test accuracies of the selected training data sets were mostly higher than 90% (almost all of them belong to "easy" data sets) and the chosen basic set sizes were larger than $13\%m$ (most of them lager than $20\%m$). Those are very different from our settings here.

Next, we show that the proposed PCP-LSSVM and ICP-LSSVM can be used to find better reduced set for other RSVM learners [16], [17], [39]. For this, we selected the models with squared hinge loss, because many experiments show that SVM models with squared hinge loss can achieve better test accuracies by Newton algorithm with quadratic convergence rate (see [3] for details).

With well-chosen or random-chosen reduced set (basic set) $\mathbb{B}$, RSVM was to solve the following problem by (semi-smooth) Newton algorithms [3], [18]:

$$\min_{\boldsymbol{\alpha}_{\mathbb{B}}\in\Re^r, b\in\Re} \frac{\lambda}{2}\boldsymbol{\alpha}_{\mathbb{B}}^{\top}\boldsymbol{K}_{\mathbb{B}\mathbb{B}}\boldsymbol{\alpha}_{\mathbb{B}}+\sum_{i=1}^{m}\max\left\{0, 1-y_i(\boldsymbol{K}_{i\mathbb{B}}\boldsymbol{\alpha}_{\mathbb{B}}+b)\right\}^2.$$

In Table II, the reduced set was random-chosen for RRSVM, and well-chosen by PCP-LSSVM for PCP+RSVM, as also by ICP_{22}-LSSVM for ICP_{22}+RSVM. All the parameters were set to be the same as those in Table I. We did not consider ICP-LSSVM because it was slow.

The results in Table II show that the proposed methods improve the performance of RSVM with a little extra training time.

C. Challenge on very large data sets

In this subsection, we will discuss some experiments performed on two very large-scale data sets to compare P-LSSVM and D-LSSVM. The selected data sets are as follows:

- ***COVTYPE***: It is a binary class problem with 581,012 samples [41], and each example has 54 features. We

TABLE II
COMPARISON OF TEST ACCURACIES (%) AND TRAINING TIMES (SECONDS) OF THE RELATED RSVM ALGORITHMS ON SIX BENCHMARK DATA SETS. ALL THE RESULTS WERE OBTAINED ON AN AVERAGE OVER 20 RANDOM TRIALS; THE STANDARD DEVIATIONS ARE GIVEN IN BRACKETS.

	USPS8	SHUTTLE	MNIST8	ADULT	IJCNN	VECHILE
	Test Accuracies(%)					
RRSVM	99.07(0.17)	99.89(0.01)	98.66(0.06)	85.15(0.08)	98.05(0.17)	87.69(0.06)
PCP+RSVM	**99.10**(0.12)	99.95(0.00)	98.74(0.05)	**85.24**(0.06)	97.96(0.10)	87.37(0.03)
ICP_{22}+RSVM	99.03(0.14)	**99.90**(0.01)	**99.04**(0.08)	**85.24**(0.09)	**98.25**(0.12)	**87.90**(0.06)
	Training Tims(s)					
RRSVM	0.87(0.04)	1.83(0.15)	19.74(0.24)	2.03(0.05)	2.91(0.08)	7.44(1.24)
PCP+RSVM	1.50(0.04)	3.72(0.13)	43.29(0.32)	5.28(0.06)	9.07(0.15)	21.14(1.64)
ICP_{22}+RSVM	3.99(0.11)	5.75(0.35)	99.11(0.46)	17.60(0.22)	35.06(0.35)	61.61(0.31)

[1)] In [17], the results are (99.31%, 1.1s), (99.87%, 68.8s), (99.36%, 86.9s), (85.31%, 18.8s), (98.6%, 58.2s) and (88.34%, 162.3s) for the respective six data sets with squared hinge loss function and much larger reduced set.

randomly split it into 464,810 training samples (80%) and 116,202 test samples (20%).

- ***Checkerboard3M***: It is based on the noisy free version of Checkerboard data set (4×4-grid XOR problem), which was first given in [43] and later widely used to show the effectiveness of nonlinear kernel methods [15], [16], [18], [44], [45]. Here, we regenerated it as a very large data set: The data set was sampled by uniformly discretizing the regions $[0, 1] \times [0, 1]$ to $2000^2 = 4,000,000$ points and labeling two classes by 4×4-grid XOR problem; it was then split randomly into 3,000,000 training samples and 1,000,000 test samples.

The test accuracies and the training times of the related algorithms are listed in Table III, where different sparsity levels are considered. In this Table, RRP-LSSVM/PCP-LSSVM were chosen for P-LSSVM, and RRD-LSSVM/PFSA-LSSVM for D-LSSVM. Others were not chosen because either their training time was too long or their test accuracies were too low with very small sparsity level settings.

From Table III, it can be seen that the results on COVTYPE with PCP-LSSVM are better than those of [46] and are comparable with the results in [29] (81.8%). Apparently, in our settings, the algorithms based on D-LSSVM failed on this data set. For Checkerboard3M with the tiny sparsity level, it is clear that the algorithms based on P-LSSVM work much better than those of D-LSSVM. We noticed that the training time of PCP-LSSVM with $r_{max} = 300$ and that of RRP-LSSVM with $r_{max} = 450$ are very long. This is because the total memory needed is over 4GByte (the RAM of my computer) and virtual memory was used to save the variables on hard disk in Matlab environment with more time.

TABLE III
COMPARISON OF P-LSSVM AND D-LSSVM ON VERY LARGE DATA SETS. ALL THE RESULTS WERE OBTAINED ON AN AVERAGE OVER 5 RANDOM TRIALS AND THE STANDARD DEVIATIONS ARE GIVEN IN BRACKETS.

	COVTYPE[1)]		Checkerboard3M[2)]	
Training Size	464,180		3,000,000	
Test Size	116,202		1,000,000	
Dimension	54		2	
λ	10^{-7}		10^{-6}	
γ	2^{-8}		2^{6}	
Max. Basic Size	500	1000	300	450
Sparsity Level	0.11%	0.22%	0.01%	0.015%
	Test Accuracies(%)			
RRP-LSSVM	75.71(0.02)	75.95(0.07)	99.34(0.03)	**99.58**(0.01)
PCP-LSSVM	**79.66**(0.06)	**80.87**(0.06)	**99.41**(0.01)	—[3)]
RRD-LSSVM	68.43(1.18)	73.60(0.52)	76.46(2.24)	84.04(1.55)
PFSA-LSSVM	53.27(3.88)	50.24(1.55)	83.97(2.56)	95.70(0.25)
	Training time(seconds)			
RRP-LSSVM	43.42(0.29)	103.70(13.32)	57.01(7.44)	4790(279)
PCP-LSSVM	108.91(0.19)	314.24(7.23)	2466.3(227.5)	—[3)]
RRD-LSSVM	20.14(0.09)	57.59(15.70)	2.68(0.62)	9.14(1.16)
PFSA-LSSVM	67.51(1.72)	176.02(12.83)	109.25(3.46)	173.5(2.8)

[1)] Reported results in [46] (78.83%, not reported), in [29] (81.8%, very long training time reported).
[2)] The target test accuracy is 100%.
[3)] The values are missing because the training time is too long.

V. CONCLUSION

LSSVM gives good performance on many classification problems, but for the obvious limitation of the solution lacking sparseness. Many researchers tried to overcome this limitation by approximation methods, based on D-LSSVM. In this paper, we introduce P-LSSVM model, induced by the representer theorem, which is equivalent to D-LSSVM on classification. However, P-LSSVM will have sparse solution if the kernel matrix has low rank or can be approximated by a low rank matrix, which can partly overcome the limitation of D-LSSVM.

For linear classification problems, we reveal that P-LSSVM may have the **sparsest** solution among all SVM learners. For nonlinear problems, to achieve the sparse solutions of P-LSSVM, we propose two algorithms . Many experimental results illustrate that the algorithms based on P-LSSVM with very less sparsity level can keep convergence on all the data sets with acceptable training time. However, the algorithms based on D-LSSVM with very low sparsity level can converge only on some "easy" data sets, but not on many "hard" data sets. Especially for the very large data set Checkerboard3M, P-LSSVM with very low sparsity level (less than 0.015%) can quickly achieve very high test accuracies (better than 99%) .

The objective of this paper is to encourage the researchers working on sparseness of LSSVM on primal more, but not on dual.

ACKNOWLEDGMENT

The author would like to thank the anonymous reviewers and editors for their valuable suggestions that greatly improved

the quality of this presentation.

REFERENCES

[1] V. N. Vapnik, "An overview of statistical learning theory," *IEEE Trans. Neural Network*, vol. 10, pp. 988–999, Sep. 1999.

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

[3] S. Zhou, "Which is better? Regularization in RKHS vs R^m on Reduced SVMs," *Statistics, Optimization and Information Computing*, vol. 1, no. 1, pp. 82–106, 2013.

[4] I. Steinwart and A. Christmann, *Support Vector Machines*. New York, NY: Springer, 2008.

[5] B. Schölkopf, R. Herbrich, and A. J. Smola, "A generalized representer theorem," in *Proceedings of the Annual Conference on Computational Learning Theory*. Amsterdam, Netherlands, Jul. 2001, pp. 416–426.

[6] S. S. Keerthi, O. Chapelle, and D. Decoste, "Building Support Vector Machines with reduced classifier complexity," *Journal of Machine Learning Research*, vol. 7, pp. 1493–1515, 2006.

[7] O. Chapelle, "Training a Support Vector Machine in the primal," *Neural Computation*, vol. 19, no. 5, pp. 1155–1178, 2007.

[8] T. Joachims, *SVM^{light}–Support Vector Machine*, 1998. [Online]. Available: http://www.cs.cornell.edu/people/tj/svm_light/

[9] J. C. Platt, "Fast training of Support Vector Machines using Sequential Minimal Optimization," in *Advances in Kernel Method-Support Vector Learning*, B. Schölkopf, C. J. Burges, and A. J. Smola, Eds. Cambridge, MA: MIT Press, 1999, pp. 185–208.

[10] J. Suykens and J. Vandewalle, "Least square Support Vector Machine classifiers," *Neural Processing Letters*, vol. 9, no. 3, pp. 293–300, 1999.

[11] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines*. Cambridge, MA: Cambridge University Press, 2000.

[12] S. Fine and K. Scheinberg, "Efficient SVM training using low-rank kernel representations," *Journal of Machine Learning Research*, vol. 2, pp. 243–264, 2001.

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

[14] M. C. Ferris and T. S. Munson, "Semismooth Support Vector Machines," *Mathematical Programming*, vol. 101, no. 1, pp. 185–204, 2004.

[15] Y.-J. Lee and O. L. Mangasarian, "SSVM: A smooth Support Vector Machine for classification," *Computational Optimization and Applications*, vol. 20, no. 1, pp. 5–22, 2001.

[16] Y.-J. Lee and O. L. Mangasarian, "RSVM: Reduced Support Vector Machines," in *CD Proceedings of the SIAM International Conference on Data Mining*. Chicago, IL, Apr. 2001, pp. 1–17.

[17] S. Zhou, J. Cui, F. Ye, H. Liu, and Q. Zhu, "New smoothing SVM algorithm with tight error bound and efficient reduced techniques," *Computational Optimization and Applications*, vol. 56, no. 3, pp. 599–618, 2013.

[18] S. Zhou, H. Liu, L. Zhou, and F. Ye, "Semismooth Newton Support Vector Machine," *Pattern Recognition Letters*, vol. 28, no. 15, pp. 2054–2062, 2007.

[19] P. J. Huber, *Robust Statistics*. New York: John Wiley, 1981.

[20] J. A. K. Suykens, T. V. Gestel, J. D. Brabanter, B. D. Moor, and J. Vandewalle, *Least squares Support Vector Machines*. River Edge, NJ: World Scientific, 2002.

[21] 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," *Machine Learning*, vol. 54, no. 1, pp. 5–32, 2004.

[22] J. Suykens, L. Lukas, and J. Vandewalle, "Sparse approximation using least squares Support Vector Machines," in *IEEE International Symposium on Circuits and Systems (ISCAS 2000)*, Geneva, Switzerland, May, 2000, pp. 757–760.

[23] J. Suykens, J. De Brabanter, L. Lukas, and J. Vandewalle, "Weighted least squares Support Vector Machines: robustness and sparse approximation," *Neurocomputing*, vol. 48, no. 1-4, pp. 85–105, 2002.

[24] B. J. Kruif and T. J. Vries, "Pruning error minimization in least squares Support Vector Machines," *IEEE Trans. Neural Networks*, vol. 14, pp. 696–702, May 2003.

[25] X. Zeng and X. Chen, "SMO-based pruning methods for sparse least squares Support Vector Machines," *IEEE Trans. Neural Networks*, vol. 16, pp. 1541–1546, Nov. 2005.

[26] A. Kuh and P. D. Wilde, "Comments on "Pruning Error Minimization in Least Squares Support Vector Machines"," *IEEE Trans. Neural Networks*, vol. 18, pp. 606–609, Mar. 2007.

[27] L. Jiao, L. Bo, and L. Wang, "Fast sparse approximation for Least Squares Support Vector Machines," *IEEE Trans. Neural Networks*, vol. 18, pp. 685–697, May 2007.

[28] L. Bo, L. Jiao, and L. Wang, "Working set selection using functional gain for LS-SVM," *IEEE Trans. Neural Networks*, vol. 18, pp.1541–1544, Sep. 2007.

[29] K. D. Brabanter, J. D. Brabanter, J. Suykens, and B. D. Moor, "Optimized fixed-size kernel models for large data sets," *Computational Statistics & Data Analysis*, vol. 54, no. 6, pp. 1484–1504, 2010.

[30] A. J. Smola and B. Schökopf, "Sparse greedy matrix approximation for machine learning," in *Proceedings of the Seventeenth International Conference on Machine Learning (ICML'00)*, San Francisco, CA, USA, Jun. 2000, pp. 911–918.

[31] G. H. Golub and C. F. V. Loan, *Matrix Computations*. Baltimore, Maryland: John Hopkins University Press, 1996.

[32] H. Harbrecht, M. Peters, and R. Schneider, "On the low-rank approximation by the pivoted cholesky decomposition," *Applied Numerical Mathematics*, vol. 62, no. 4, pp. 428–440, 2012.

[33] C. Alzate and J. A. K. Suykens, "Sparse kernel models for spectral clustering using the incomplete cholesky decomposition," in *IEEE International Joint Conference on Neural Networks (IJCNN 2008)*, Hong Kong, China, Jun. 2008, pp. 3556–3563.

[34] C. Alzate and J. A. Suykens, "Sparse kernel spectral clustering models for large-scale data analysis," *Neurocomputing*, vol. 74, no. 9, pp. 1382–1390, 2011.

[35] F. R. Bach and M. I. Jordan, "Predictive low-rank decomposition for kernel methods," in *Proceedings of the 22nd International Conference on Machine Learning*, Bonn, Germany, Aug. 2005, pp.33–40.

[36] R. A. Horn and C. R. Johson, *Matrix Analysis*. Cambridge, MA: Cambridge University Press, 1990.

[37] B. Schölkopf, A. J. Smola, R. C. Williamson, and P. L. Bartlett, "New support vector algorithms," *Neural Computation*, vol. 12, no. 5, pp. 1207–1245, May 2000.

[38] A. J. Smola and P. Bartlett, "Sparse greedy gaussian process regression," in *Advances in Neural Information Processing Systems 13*. Cambridge, MA, 2001, pp. 619–625.

[39] Y.-J. Lee and S.-Y. Huang, "Reduced Support Vector Machines: A statistical theory," *IEEE Trans. Neural Networks*, vol. 18, pp. 1–13, Jan. 2007.

[40] P. Vincent and Y. Bengio, "Kernel matching pursuit," *Machine Learning*, vol. 48, no. 1-3, pp. 165–187, 2002.

[41] R.-E. Fan and C.-J. Lin. (2010) LIBSVM Data. [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

[42] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter, "Pegasos: Primal estimated sub-gradient solver for SVM," *Mathematic Programming*, vol. 127, no. 1, pp. 3–30, Mar. 2011.

[43] T. K. Ho and E. M. Kleinberg, "Building projectable classifiers of arbitrary complexity," in *Proceedings of the 13th International Conference on Pattern Recognition*, Vienna, Austria, Aug. 1996, pp. 880–885.

[44] O. L. Mangasarian and D. R. Musicant, "Successive overrelaxation for Support Vector Machines," *IEEE Trans. Neural Networks*, vol. 10, pp.1032–1037, Sep. 1999.

[45] O. L. Mangasarian and D. R. Musicant, "Lagrangian Support Vector Machines," *Journal of Machine Learning Research*, vol. 1, pp. 161–177, 2001.

[46] T. Jung and D. Polani, "Sequential learning with LS-SVM for large-scale data sets," in *International Conference on Artificial Neural Networks(ICANN'06)*. Berlin, Heidelberg, Sep. 2006, pp. 381–390.

Shuisheng Zhou was born in Shaanxi, China, on March 9, 1972. He received M.S. degree in applied mathematics and Ph.D. degree in Computer Science from Xidian University, China, in 1998 and 2005.

He is currently a Professor in School of Mathematics and Statistics at Xidian University, China. His current research interests include optimization algorithm and its application, machine learning, pattern recognition, kernel-based learning, and Support Vector Machines.