Hindawi Publishing Corporation
Computational Intelligence and Neuroscience
Volume 2013, Article ID 968438, 7 pages
http://dx.doi.org/10.1155/2013/968438

*Research Article*

# Single Directional SMO Algorithm for Least Squares Support Vector Machines

**Xigao Shao,[1,2] Kun Wu,[1] and Bifeng Liao[3]**

[1] *School of Mathematics and Statistics, Central South University, Changsha, Hunan 41007, China*
[2] *Wengjing College, Yantai University, Yantai, Shandong 264005, China*
[3] *School of Mathematics and Information Science, Yantai University, Yantai, Shandong 264005, China*

Correspondence should be addressed to Kun Wu; wukuncs@qq.com

Received 1 October 2012; Revised 20 December 2012; Accepted 4 January 2013

Academic Editor: Daoqiang Zhang



Working set selection is a major step in decomposition methods for training least squares support vector machines (LS-SVMs). In this paper, a new technique for the selection of working set in sequential minimal optimization- (SMO-) type decomposition methods is proposed. By the new method, we can select a single direction to achieve the convergence of the optimality condition. A simple asymptotic convergence proof for the new algorithm is given. Experimental comparisons demonstrate that the classification accuracy of the new method is not largely different from the existing methods, but the training speed is faster than existing ones.

## 1. Introduction

In a classification problem, we consider a set of training samples, that is, the input vectors $\{\mathbf{x}_k\}_{k=1}^N$ along with corresponding class labels $\{y_k\}_{k=1}^N$. Our task is to find a deterministic function that best represents the relation between input vectors and class labels. For classification or forecasting problems in machine learning, support vector machine (SVM) has been adopted in many applications because of its high precision [1–4]. SVMs require the solution of a quadratic programming problem. Another successful method for machine learning is least squares support vector machine (LS-SVM) [5]. Instead of solving a quadratic programming problem as in SVMs, the solutions of a set of linear equations are obtained in LS-SVMS. There are many proposed algorithms for training LS-SVMs: Suykens et al. proposed an iterative algorithm based on conjugate gradient (CG) algorithms [6]; Ferreira et al. presented a gradient system which can train the LS-SVM model [7] effectively; Chua introduced efficient computations for large least square support vector machine classifiers [8]; Chu et al. improved the efficiency of the CG algorithm by using one reduced system of linear equations [9]; Keerthi and Shevade extended the sequential minimal optimization (SMO) algorithms to solve the linear equations in LS-SVMs where the maximum violating pair (MVP) was selected as the working set [10]; based on the idea of SMO algorithm, Lifeng Bo et al. presented an improved method for working set selection by using functional gain (FG) [11]; Jian et al. designed a multiple kernel learning algorithm for LS-SVMs by convex programming [12]; and so on. These numerical algorithms are computationally attractive. Empirical comparisons show that SMO algorithm is more efficient than CG one for the large scale datasets.

Fast SVM training speed with SMO algorithm is an important goal for practitioners and many other proposals have been given for this in the literature. Initially, Platt presented two heuristics that resulted in a bit cumbersome selection [13]. Later, Keerthi et al. introduced the concept of a violating pair to denote two coefficients which cause a violation in the KKT optimality conditions of the dual, and the authors suggested to select always the pair that violated them the most, that is, the maximum violating pair (MVP) [14]. Finally, Fan et al. proposed a second order selection that usually results in faster training than the MVP rule [15]. By the above improvement, we can decrease the computational expense of SMO algorithm, while there are repeated selections of some concrete updating patterns in sequential minimal optimization. They are called training cycles. Barbero et al.

studied the presence of them from a geometrical point of view [16]. They pointed out that the training cycles can be partially collapsed in a single updating vector that gave better optimal directions. The idea for training cycles can reduce the number of iterations and kernel operations for SMO algorithm.

Inspired by Barbero et al. [16], we present a single directional SMO algorithm for LS-SVMs, abbreviated as SD-SMO algorithm. In optimization procedure, an adaptive objective function is selected, and the single directional steps are given for the lagrangian multipliers, which can lessen the number of training cycles and further reduce iterations and kernel operations for SMO algorithm. Experiments show that the training time for LS-SVMs by SD-SMO algorithm can be reduced significantly, and it has a testing accuracy which is not largely different from traditional SMO algorithm.

The rest of this paper has the following structure. In the next section, LS-SVMs are briefly reviewed. In Section 3, SD-SMO algorithm for LS-SVMs is provided and the convergence of the improved algorithm is proved theoretically. Based on standard datasets, computational experiments describing the effectiveness of the improved algorithm are presented in Section 4. Finally, Section 5 is devoted to concluding remarks.

## 2. LS-SVM

In this section, we concisely review the basic principles of LS-SVMs. Given a training dataset of $N$ points $\{\mathbf{x}_k, y_k\}_{k=1}^N$ with input data $\mathbf{x}_k \in \mathsf{R}^n$ and output data $y_k \in \mathsf{R}$, we consider the following optimization problem in primal weight space:

$$\min_{w,b,e} J(w,e) = \frac{1}{2}w^T w + \frac{1}{2}\gamma\sum_{k=1}^{N} e_k^2, \tag{1}$$

such that

$$y_k - \left(w^T\varphi(\mathbf{x}_k) + b\right) = e_k, \quad k = 1, 2, \ldots, N, \tag{2}$$

where $\gamma$ is a regularization factor, $e_k$ is the difference between the desired output $y_k$ and the actual output, and $\varphi(\cdot)$ is a nonlinear function mapping the data points into a high-dimensional Hibert space; in addition, the dot product in the high-dimensional space is equivalent to a positive-definite kernel function $\mathbf{k}(\mathbf{x}_i, \mathbf{x}_j) = \varphi(\mathbf{x}_i)^T\varphi(\mathbf{x}_j)$.

In primal weight space, a linear classifier in the new space takes the following form:

$$y(\mathbf{x}) = \operatorname{sign}\left(w \cdot \varphi(\mathbf{x}) + b\right). \tag{3}$$

The weight vector $w$ may be infinite dimensional; hence, using (1) to find the solutions is impossible in general. In order to solve this problem, we would compute the model in the dual space instead of the primal space. Let $b = 0$, and the simple problem without a bias term is considered in this paper as in the paper by Keerthi and Shevade [10]. The Lagrangian for the simple problem is

$$L(w, e; \alpha) = J(w, e) - \sum_{k=1}^{N}\alpha_k\left\{w^T\varphi(\mathbf{x}_k) + e_k - y_k\right\}, \tag{4}$$

where $\alpha_i$ are Lagrangian multipliers and are called support values. The Karush-Kuhn-Tucker (KKT) conditions for optimality are

$$\begin{aligned}
&\frac{\partial L}{\partial w} = 0 \longrightarrow w = \sum_{k=1}^{k=N}\alpha_k y_k\varphi(\mathbf{x}_k),\\
&\frac{\partial L}{\partial e_k} = 0 \longrightarrow \alpha_k = \gamma e_k, \quad k = 1, \ldots, N,\\
&\frac{\partial L}{\partial \alpha_k} = 0 \longrightarrow w^T\varphi(\mathbf{x}_k) + e_k - y_k = 0, \quad k = 1, \ldots, N.
\end{aligned} \tag{5}$$

After elimination of $w$ and $e$, we could obtain the following linear system:

$$\left(K + \frac{I}{\gamma}\right)\alpha = y, \tag{6}$$

where $y = [y_1, y_2, \ldots, y_N]^T$, $\alpha = [\alpha_1, \alpha_2, \ldots \alpha_N]^T$, and $K \in R^{N\times N}$ is the kernel matrix. By solving the linear system (6), $\alpha_i's$ are obtained; hence, LS-SVM greatly simplifies the problem. The resulting LS-SVM model for function estimation is

$$y(\mathbf{x}) = \sum_{k=1}^{N}\alpha_k\mathbf{k}(\mathbf{x}, \mathbf{x}_k). \tag{7}$$

For the choice of the kernel function $\mathbf{k}(\cdot,\cdot)$, there are several possibilities: $\mathbf{k}(\mathbf{x}, \mathbf{x}_k) = \mathbf{x}_k^T\mathbf{x}$ (linear LS-SVM); $\mathbf{k}(\mathbf{x}, \mathbf{x}_k) = (\mathbf{x}_k^T\mathbf{x} + 1)^d$ (polynomial LS-SVM of degree $d$); $\mathbf{k}(\mathbf{x}, \mathbf{x}_k) = \exp\{-\|\mathbf{x} - \mathbf{x}_k\|_2^2/\sigma^2\}$ (RBF LS-SVM); $\mathbf{k}(\mathbf{x}, \mathbf{x}_k) = \tanh(k\mathbf{x}_k^T\mathbf{x}+\theta)$ (MLP LS-SVM). In this case, we focus on the choice of an RBF LS-SVM for the sequel. When solving large linear systems, we should apply iterative methods to (6), which was introduced by Jiao et al. [17]. The speed of convergence depends on the condition number of the matrix in (6). It is influenced by the choice of $(\gamma, \sigma)$ in the case of RBF LS-SVM. In the following section, we will discuss the algorithm of SMO versions and give the proof of convergence for SD-SMO algorithm.

## 3. SMO and SD-SMO Algorithms for LS-SVM

For solving the LS-SVM problem, the matrix in (6) is usually fully dense and may be too large to be stored. Decomposition methods are designed to handle the difficulties, see Jiao et al. [17]. Unlike other optimization algorithms which update the whole Lagrangian multipliers vector $\alpha$ in each iterative process, the decomposition algorithm modifies only a subset of $\alpha$ per iteration. We denote the subset as the working set $B$. The SMO algorithm was developed in [10] as a decomposition method to solve the dual problems arising in LS-SVM formulations. In each iteration, SMO algorithm restricts $B$ to have only two elements. Because of the problem (4) without the bias term $b$, SMO can be simplified to optimize $B$ with only one element at an iteration. By substituting the KKT conditions (5) into the Lagrangian (4), the dual problem is to maximize the following objective function:

$$\max(L(\alpha)) = -\frac{1}{2}\sum_j\sum_i\alpha_i\alpha_j Q(\mathbf{x}_i, \mathbf{x}_j) + \sum_i\alpha_i y_i, \tag{8}$$

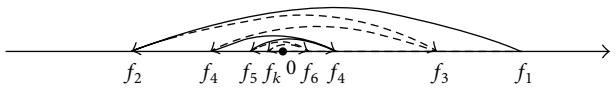


Figure 1: SMO sketch map, where $f_k$ represents the $k$th iteration for $f_i$, for all $i$.

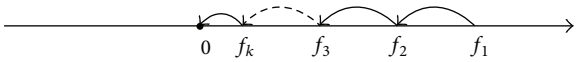


Figure 2: SD-SMO sketch map, where $f_k$ represents the $k$th iteration for $f_j$, for all $j$.

where $Q(\mathbf{x}_i, \mathbf{x}_j) = K(\mathbf{x}_i, \mathbf{x}_j) + \sigma_{ij}/\gamma$, and $\sigma_{ij} = 1$ if $i = j$ and 0 otherwise.

The SMO algorithm for (8) is sketched in the following.

*Algorithm 1.* SMO algorithm for (8) is as follows.

(1) Set $k = 1$ and find $\alpha^k = 0$ as the initial feasible solution.

(2) If the stop criterion is satisfied, stop. If not, find a one-element working set $B = \{i\} \subset \{1, \ldots, N\}$. Define $D \equiv \{1, \ldots, N\} \setminus B$ and $\alpha_B^k$ and $\alpha_D^k$ to be subvectors of $\alpha_k$ corresponding to $B$ and $D$, respectively.

(3) Solve the following subproblem with the variable $\alpha_B$:

$$\max_{\alpha_B} \left\{ -\frac{1}{2} \begin{bmatrix} \alpha_B^T & \left(\alpha_D^k\right)^T \end{bmatrix} \begin{bmatrix} Q_{BB} & Q_{BD} \\ Q_{DB} & Q_{DD} \end{bmatrix} \begin{bmatrix} \alpha_B \\ \alpha_D^k \end{bmatrix} + \begin{bmatrix} y_B^T & y_D^T \end{bmatrix} \begin{bmatrix} \alpha_B \\ \alpha_D^k \end{bmatrix} \right\}, \tag{9}$$

where $\begin{bmatrix} Q_{BB} & Q_{BD} \\ Q_{DB} & Q_{DD} \end{bmatrix}$ is a permutation of the matrix $Q$.

(4) Set $\alpha_B^{k+1}$ to be the optimal solution of (9) and $\alpha_D^{k+1} \equiv \alpha_D^k$. Set $k \leftarrow k + 1$ and go back to step (2).

In order to find working set $B$, we usually consider whether the KKT conditions is violated or not. The KKT conditions for the dual problem (8) are $\partial L/\partial \alpha_i = 0$, which lead to $y_i - \sum_j \alpha_j Q(\mathbf{x}_i, \mathbf{x}_j) = 0$, $i = 1, 2, \ldots, N$. If we define

$$f_i = \frac{\partial L}{\partial \alpha_i} = y_i - \sum_j \alpha_j Q\left(\mathbf{x}_i, \mathbf{x}_j\right), \tag{10}$$

then the KKT optimality condition is violated if there exists any index point $i$ such that $f_i \neq 0$. SMO algorithm for (8) achieves the convergence of optimal process when $f_i \to 0$, for all $i$.

A simple illustration of this is shown in Figure 1.

Since only one component is updated per iteration, the decomposition method can be quite costly and suffers from slow convergence. For this reason, many researchers improved SMO algorithm. For example, Chen et al. improved SMO algorithm by using the shrinking and caching techniques [18]; Barbero et al. presented a cycle-breaking acceleration of SVM training [16]; and Lin et al. provided three-parameter sequential minimal optimization for support vector machines [19].

As mentioned by Barbero et al. in [16], SMO algorithm is not free of cycle-related problems. For all $i$ in working set $B$, if $\alpha_i$ is optimized with step $t$ ($t > 0$ or $t < 0$) in a single direction per iteration, the number of cycles in SD-SMO algorithm will be reduced. We now detail SD-SMO formulation in the LS-SVM training process.

Define

$$F_i = \left(f_i\right)^2, \quad \forall i. \tag{11}$$

Then, the KKT optimality condition is violated if there exists any index point $i$ such that $F_i \neq 0$.

SD-SMO algorithm works by optimizing only one $\alpha_i$ at each iteration and keeping the others fixed, that is, $\alpha$ is adjusted by a sign-invariable step $t$ ($t > 0$ or $t < 0$) per iteration as follows:

$$\alpha_i^{k+1} = \alpha_i^k + t; \quad \alpha_j^{k+1} = \alpha_j^k, \quad \forall j \neq i. \tag{12}$$

The update of $\alpha_i$ causes the change of all the $f_j$ as

$$f_j^{k+1} = f_j^k - \left(\alpha_i^{k+1} - \alpha_i^k\right) Q\left(\mathbf{x}_i, \mathbf{x}_j\right), \quad \forall j \tag{13}$$

and; therefore, the function value of $F_j$ will change. At each iteration we need to be sure that the sign of $f_j^k$ is not variable, that is, if $f_j^k \geq$ (or $\leq$) 0, then $f_j^{k+1} \geq$ (or $\leq$) 0. As $k$ increases, $f_j^k \to 0^+$ (or $0^-$) with the sign keeping invariable.

A simple illustration of this is shown in Figure 2.

To derive the optimal step $t$ and the termination conditions of iteration, we define $F_j$ as

$$F_j(t) = \left[f\left(\alpha^{\text{new}}(t)\right)\right]^2, \qquad F_j(0) = \left[f(\alpha)\right]^2. \tag{14}$$

Because $f_j^k \to 0$ as $k \to \infty$, $F_j(t) \leq F_j(0)$. Therefore, let $\Delta F_j = -(F_j(t) - F_j(0))$ and it can be written as

$$\Delta F_j = -\left(F_j(t) - F_j(0)\right) = 2t f_j Q\left(\mathbf{x}_j, \mathbf{x}_j\right) - t^2 Q^2\left(\mathbf{x}_j, \mathbf{x}_j\right). \tag{15}$$

The optimal step is obtained by maximizing $\Delta F_j$ as

$$t^{\text{opt}} = \frac{f_j}{Q\left(\mathbf{x}_j, \mathbf{x}_j\right)}, \tag{16}$$

and the optimal step $t^{\text{opt}}$ can induce the change of $F_j$ as

$$\Delta F_j = \frac{F_j}{2Q\left(\mathbf{x}_i, \mathbf{x}_j\right)}. \tag{17}$$

Hence we can choose an index point $j$ which has the maximum value of $F_j/2Q(\mathbf{x}_j, \mathbf{x}_j)$ and update $\alpha$ by (12) and (16). Suppose $F(\alpha) = (f_1, f_2, \ldots, f_j, \ldots, f_N)$ and $\|F(\alpha^k)\|_2^2 = \sum_j F_j^k$, then $\{\|F(\alpha^k)\|_2^2\}$ is a decreasing sequence. In fact, as

$k \to \infty$, $\|F(\alpha^k)\|_2^2 \to 0$. Therefore $\|F(\alpha^k)\|_2^2$ can be used as a termination criterion for the iterative algorithm as

$$\left\|F\left(\alpha^k\right)\right\|_2^2 \le \varepsilon^2 N, \tag{18}$$

where $\varepsilon$ is a positive constant. The flowchart of SD-SMO algorithm is shown in Algorithm 2.

*Algorithm 2.* SD-SMO algorithm for (8) is as follows.

(1) Set $k = 1$ and choose $\alpha^k$ such that $f_j \ge 0$ (or $f_j \le 0$) for all $j = 1, 2, \ldots, N$.

(2) If $\alpha^k$ satisfies (18), stop. If not, select $p_1 = \arg\max_j (F_j / 2Q(\mathbf{x}_j, \mathbf{x}_j))$

(3) Update $\alpha^k$ using $t^{\text{opt}} = f_{p_1} / Q(\mathbf{x}_{p_1}, \mathbf{x}_{p_1})$ and (12).

(4) While $f_{p_1} \ge 0$ ($f_{p_1} \le 0$), $k = k + 1$, go back to step (2).

One theoretical property of SD-SMO algorithm is presented in the following.

**Theorem 3.** *The sequence $\alpha^k$ generated by SD-SMO algorithm converges to the global optimal solution of* (8).

*Proof.* According to the definition of $\|F(\alpha^k)\|_2^2$ and combining (16) and (17), the following equation holds:

$$\begin{aligned}\left\|F(\alpha^{k+1})\right\|_2^2 - \left\|F\left(\alpha^k\right)\right\|_2^2 &= -\frac{\left(t^{\text{opt}}\right)^2 Q\left(\mathbf{x}_j, \mathbf{x}_j\right)}{2} \\ &= -\frac{\left(t^{\text{opt}}\right)^2 \left(K\left(\mathbf{x}_j, \mathbf{x}_j\right) + 1/\gamma\right)}{2}.\end{aligned} \tag{19}$$

The positive-definite kernel function implies $K(\mathbf{x}_j, \mathbf{x}_j) \ge 0$, furthermore $\|\alpha^{k+1} - \alpha^k\|_2^2 = (t^{\text{opt}})^2$, and the following equation is obtained:

$$\left\|F(\alpha^{k+1})\right\|_2^2 - \left\|F(\alpha^k)\right\|_2^2 = -\frac{\left\|\alpha^{k+1} - \alpha^k\right\|_2^2 \left(K\left(\mathbf{x}_j, \mathbf{x}_j\right) + 1/\gamma\right)}{2}. \tag{20}$$

Equality (20) yields that $\{\|F(\alpha^k)\|_2^2\}$ is a decreasing sequence. Together with $\|F(\alpha^k)\|_2^2 \ge 0$, we have that $\{\|F(\alpha^k)\|_2^2\}$ converges. Applying (20) again, we get that $\{\alpha^{k+1} - \alpha^k\}$ converges to 0 as $k \to \infty$.

Since $F_j$ $(\forall j)$ is a positive-definite quadratic form, $\{\|F(\alpha)\|_2^2\} = \sum_j F_j$ is a positive-definite quadratic form too. Therefore, the set $\{\alpha \mid \|F(\alpha)\|_2^2 \le \|F(\alpha^0)\|_2^2\}$ is a compact set. $\{\alpha^k\}$ lies in this set, so it is a bounded sequence. Let $\widehat{\alpha}$ be the limit point of any convergent subsequence $\{\alpha^k\}$, $k \in \Gamma$. For all $j$, $F_j(\widehat{\alpha}) = \lim_{k \to \infty} F_j(\alpha^k)$. According to the definition of $F(\alpha^k)$, $0 \le F_j(\alpha^k) \le F(\alpha^k)$. Inequality (18) yields $\lim_{k \to \infty}\{\|F(\alpha^k)\|_2^2\} = 0$; furthermore, for all $j$, $F_j(\widehat{\alpha}) = \lim_{k \to \infty} F_j(\alpha^k) = 0$. While $F_j(\widehat{\alpha}) = (f_j(\widehat{\alpha}))^2$, so $f_1(\widehat{\alpha}) = f_2(\widehat{\alpha}), \ldots, = f_N(\widehat{\alpha}) = 0$. From the KKT conditions, $\widehat{\alpha}$ is the global optimal solution of (8). Since $L(\alpha)$ is strictly convex, (8) has a unique global solution and we denote it as $\alpha^*$. Assume that $\{\alpha^k\}$ does not converge to $\alpha^*$. Then, for all $\epsilon > 0$, there exists an infinite subset $\widetilde{\Gamma}$ such that for all $k \in \widetilde{\Gamma}$, $\|\alpha^k - \alpha^*\| > \epsilon$. Because $\{\alpha^k\}$, for all $k \in \widetilde{\Gamma}$ is a compact set, there is a convergent subsequence. Without loss of generality, we assume its limit to be $\widehat{\alpha}$. Thus, $\|\widehat{\alpha} - \alpha^*\| > \epsilon$. Since $\widehat{\alpha}$ is the global optimal solution of (8), this contradicts that $\widetilde{\Gamma}$ is the unique global optimal solution. The proof of Theorem is completed. □

## 4. Numerical Experiments

Under the framework Algorithm 2, we conduct experiments to check whether using SD-SMO is really faster than using SMO or not in this section. There have been two techniques for working set selection in SMO-type decomposition methods. The former is first order SMO (FO-SMO) algorithm and the latter is second order SMO (SO-SMO) algorithm for LS-SVM classifiers [20]; that is, the former uses first order information to achieve fast convergence and the latter uses second order information. Two groups of experiment have been done in order to compare SD-SMO with the above two algorithms. All methods are implemented in MATLAB and executed on a personal computer with Intel(R) Core(TM) i3 2.53 GHz processors, 2.00-GB memory, and Windows 7 operation systems. For all algorithms, the optimization process is terminated when the maximal violation of the KKT conditions is within $\varepsilon = 0.001$. For simplicity, we consider only Gaussian kernel $k(\mathbf{x}, \mathbf{x}_k) = \exp\{-\|\mathbf{x} - \mathbf{x}_k\|_2^2 / 2\sigma^2\}$ to construct LS-SVM.

*4.1. The Comparison of SD-SMO with First Order SMO.* In this section, we compare SD-SMO with first order SMO on four benchmark datasets for evaluating the performance of the proposed method. We compare the two methods in terms of computational cost, which is measured by the number of iteration. The examples introduced by Keerthi and Shevade [10] are used. Datasets used for this purpose are Banana, Image, Waveform, and Splice. For each dataset, the value of $\sigma^2$ is determined by the five-fold cross validation on a small random subset.

In the first experiment, we vary $\gamma$ over a small range because the extremely small and large $\gamma$ values are usually of little interest. We try the following nine $\gamma$ values: $2^i$, $i = -4, -3, \ldots, 3, 4$. In Table 1, the computational costs associated with the four datasets as functions of $\gamma$ are given when the optimization process is terminated.

As a basis for the comparisons, Table 1 shows the computational costs of first order SMO and SD-SMO algorithms at different values of parameter $\gamma$. For first order SMO algorithm, the computational cost increases with the increase of $\gamma$. While for SD-SMO algorithm, it is not so. For instance, see the computational cost of SD-SMO for the Banana and Waveform datasets. From Table 1, we can see that the number of iterations of SD-SMO algorithm is much smaller than that of first order SMO one, especially for Image dataset.

Table 1: Computational costs for first order SMO (FO-SMO) and SD-SMO algorithms.

| $\log_2^\gamma$ | Banana $\sigma^2 = 1.8221$ | | Image $\sigma^2 = 2.7183$ | | Waveform $\sigma^2 = 24.5325$ | | Splice $\sigma^2 = 29.9612$ | |
|---|---|---|---|---|---|---|---|---|
| | FO-SMO | SD-SMO | FO-SMO | SD-SMO | FO-SMO | SD-SMO | FO-SMO | SD-SMO |
| −4 | 0.4460 | 0.3548 | 0.4838 | 0.1104 | 0.5375 | 0.2234 | 0.4375 | 0.3166 |
| −3 | 0.5023 | 0.3542 | 0.5150 | 0.1191 | 0.5854 | 0.2499 | 0.4683 | 0.3152 |
| −2 | 0.6379 | 0.3381 | 0.5844 | 0.1217 | 0.6109 | 0.2343 | 0.5066 | 0.3029 |
| −1 | 0.8733 | 0.2632 | 0.7413 | 0.1248 | 0.6682 | 0.2245 | 0.6060 | 0.2662 |
| 0 | 1.3545 | 0.2231 | 0.9816 | 0.1283 | 0.7440 | 0.1879 | 0.7738 | 0.2105 |
| 1 | 2.3782 | 0.1607 | 1.4816 | 0.1326 | 0.8512 | 0.1672 | 1.3078 | 0.1775 |
| 2 | 2.4793 | 0.0679 | 1.8371 | 0.2927 | 0.9569 | 0.1490 | 1.3537 | 0.1675 |
| 3 | 2.6521 | 0.0486 | 2.3751 | 0.2136 | 1.0829 | 0.1369 | 1.7175 | 0.1481 |
| 4 | 2.8906 | 0.0231 | 2.9305 | 0.2205 | 1.2195 | 0.1344 | 2.1520 | 0.1402 |

Note: each unit corresponds to $10^4$ iterations.

Table 2: Training time (in seconds) and classification accuracy in parentheses for first order SMO (FO-SMO) and SD-SMO algorithms.

| $\log_2^\gamma$ | Banana $\sigma^2 = 1.8221$ | | Image $\sigma^2 = 2.7183$ | |
|---|---|---|---|---|
| | FO-SMO | SD-SMO | FO-SMO | SD-SMO |
| −4 | 43.6589 (0.8675) | **35.947 (0.895)** | 7.90140 (0.9012) | 2.47260 (0.9214) |
| −3 | 47.3385 (0.8753) | 35.3045 (0.8712) | 8.41620 (0.9156) | 2.50380 (0.9324) |
| −2 | 59.8110 (0.8832) | 34.3882 (0.8653) | 9.76570 (0.9223) | 2.59740 (0.9348) |
| −1 | 88.6335 (0.8889) | 28.9070 (0.8377) | 11.7874 (0.9382) | 2.57400 (0.9358) |
| 0 | 129.505 (0.8877) | 22.5036 (0.8667) | 15.3895 (0.9430) | 2.58180 (0.9410) |
| 1 | 220.437 (0.8900) | 16.1617 (0.8502) | 23.4157 (0.9521) | 2.60520 (0.9511) |
| 2 | 229.891 (0.8943) | 8.42400 (0.7853) | 31.1026 (0.9588) | 3.93120 (0.9602) |
| 3 | 238.068 (0.8977) | 3.47140 (0.7032) | **41.611 (0.967)** | **4.2979 (0.963)** |
| 4 | **259.36 (0.898)** | 2.02800 (0.6126) | 50.6560 (0.9616) | 4.50900 (0.9578) |

Table 3: Training time (in seconds) and classification accuracy in parentheses for first order SMO (FO-SMO) and SD-SMO algorithms.

| $\log_2^\gamma$ | Waveform $\sigma^2 = 24.5325$ | | Splice $\sigma^2 = 29.9612$ | |
|---|---|---|---|---|
| | FO-SMO | SD-SMO | FO-SMO | SD-SMO |
| −4 | 43.4541 (0.9094) | 35.4434 (0.8404) | 31.9303 (0.8649) | 44.4478 (0.6507) |
| −3 | 46.5039 (0.9108) | 36.1884 (0.8918) | 33.2688 (0.8736) | 44.0110 (0.7061) |
| −2 | 48.8049 (0.9114) | **37.635 (0.908)** | 36.0175 (0.8910) | 41.9830 (0.8944) |
| −1 | **52.907 (0.912)** | 35.3499 (0.8948) | 43.6085 (0.8963) | **37.730 (0.911)** |
| 0 | 58.9295 (0.9096) | 29.9522 (0.8974) | 55.2503 (0.9037) | 33.4865 (0.8866) |
| 1 | 67.2830 (0.9071) | 26.6060 (0.8955) | **72.543 (0.911)** | 26.1801 (0.8826) |
| 2 | 79.3185 (0.9068) | 24.5008 (0.8859) | 94.8392 (0.9060) | 23.3596 (0.8769) |
| 3 | 86.3930 (0.9004) | 22.9251 (0.8876) | 121.219 (0.9054) | 21.7434 (0.8750) |
| 4 | 95.7465 (0.9100) | 22.4251 (0.8860) | 153.243 (0.9032) | 21.0508 (0.8746) |

In order to further show the performance of SD-SMO algorithm, Tables 2 and 3 are given. The tables report the training time and the generalization performance of first order SMO and SD-SMO algorithms for four benchmark datasets. The generalization performance is illustrated by the classification accuracy of an independent test set for each dataset.

From Tables 2 and 3, we can see that the generalization capabilities of both methods are comparable, but the training time of SD-SMO algorithm is shorter than first order SMO algorithm. For instance, in the case of Image dataset, the training time for first order SMO algorithm with the best generalization performance is 41.6108 s. It represents the equivalent of ten times the cost of SD-SMO algorithm.

Table 4: Number of iterations (in thousands), execution times (in seconds), and average misclassification rates for second order SMO (SO-SMO) and SD-SMO algorithms.

| Dataset | Iterations | | Executiontimes | | Misclassification rate | |
|---|---|---|---|---|---|---|
| | SO-SMO | SD-SMO | SO-SMO | SD-SMO | SO-SMO | SD-SMO |
| Titanic | 277.1512 | 59.7346 | 1129.2009 | 80.9348 | 23.5723 | 23.5612 |
| Heart | 5.8993 | 2.2315 | 10.3623 | 4.4652 | 16.1117 | 17.1092 |
| Cancer | 10.1908 | 4.1127 | 21.6765 | 9.0972 | 27.6643 | 27.8764 |
| Thyroid | 30.1537 | 17.7325 | 77.3341 | 52.5521 | 5.5123 | 5.6725 |
| Pima | 60.6751 | 30.7366 | 104.9616 | 69.8546 | 25.0155 | 25.7761 |

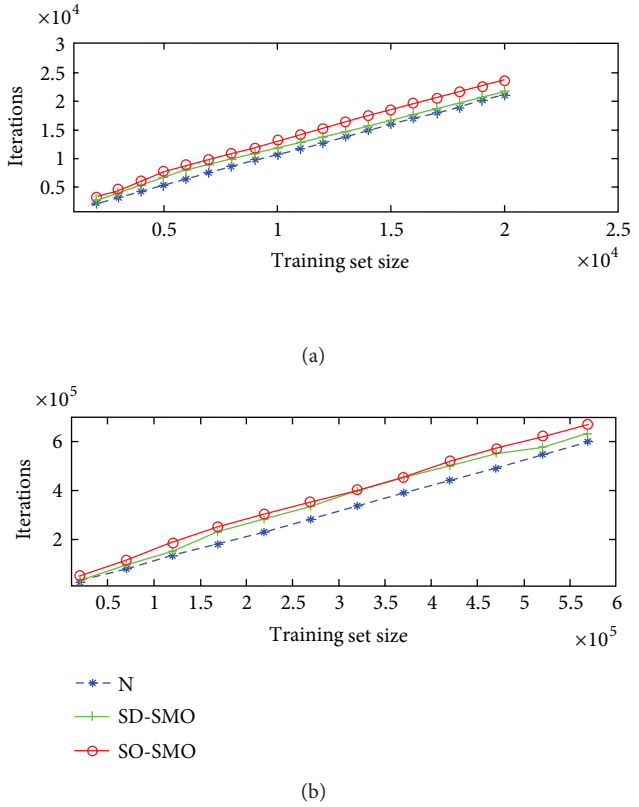


Figure 3: Variation of the number of iterations with training set size for a8a (a) and covtype (b).

The classification accuracy for Image dataset with SD-SMO algorithm is 0.963, and it is almost equal to the one with first order SMO algorithm. In consequence, the efficacy and feasibility of the proposed SD-SMO algorithm is superior to that of first order SMO one for LS-SVMs.

*4.2. The Comparison of SD-SMO with Second Order SMO.* To further explore the performance of the proposed method, we compare SD-SMO with second order SMO by a second set of experiments on the datasets Titanic, Heart, Breast Cancer, Thyroid, and Pima (available in [21]). We use the datasets provided in [21] to certify the good generalization properties of the proposed method. In Table 4, the number of iterations and execution times per experiment is reported. The misclassification rates are also reported in Table 4.

It can be seen that for these datasets it is better to use SD-SMO in Cancer, Pima, and Titanic. The results in Table 4 shows that the biggest improvement with SD-SMO happens for Titanic. Therefore, this is further evidence on the previous observation that for large-scale problems SD-SMO outperforms second order SMO.

The final set of experiments aims to ascertaining how well the SMO algorithm scales for large-scale datasets when it uses the different working set selections. In order to test this, we use the datasets a8a and covtype.binary, available with several increasing numbers of patterns in [22].

In Figure 3, we plot the results for a8a with $C = 2, \sigma^2 = 10$ and covtype.binary with $C = 10$, $\sigma^2 = 10$, respectively. As it can be seen, the number of iterations scales linearly with the training set size. Note that SD-SMO needs less iterations to convergence, as expected. And the reduction is greater for covtype.binary because of its larger value of $C$. In any case, the scaling is linear in both cases.

## 5. Conclusion

In this paper, a new algorithm, that is, SD-SMO, is proposed. It can be used to select working set for LS-SVM classifier training, and its asymptotic convergence is proved theoretically. Based on SMO formulation, the path of one-side convergence is used effectively in our method. The number of iterations and kernel operations in SD-SMO algorithm is less than that of the traditional SMO algorithm, so the new algorithm provides faster convergence speed. Simulation experiments have been carried out on four benchmark datasets. The empirical comparisons demonstrate that SD-SMO algorithm is much more efficient in terms of computational time than first order and second order SMO, and at the same time there are no large differences in terms of accuracy.

## Acknowledgments

The authors would like to thank the Handling Editor and the anonymous reviewers for their constructive comments, which led to significant improvement of the paper. This work was partially supported by the National Natural Science Foundation of China under Grant no. 51174236.

## References

[1] C. H. Song, S. J. Yoo, C. S. Won, and H. G. Kim, "Svm based indoor/mixed/outdoor classification for digital photo

annotation in a ubiquitous computing environment," *Computing and Informatics*, vol. 27, no. 5, pp. 757–767, 2008.

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

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

[4] H. Esen, F. Ozgen, M. Esen, and A. Sengur, "Modelling of a new solar air heater through least-squares support vector machines," *Expert Systems with Applications*, vol. 36, no. 7, pp. 10673–10682, 2009.

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

[6] 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 *Proceedings of the European Conference on Circuit Theory and Design (ECCTD '99)*, pp. 839–842, Stresa, Italy, 1999.

[7] 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 Transactions on Neural Networks*, vol. 16, no. 2, pp. 501–505, 2005.

[8] K. S. Chua, "Efficient computations for large least square support vector machine classifiers," *Pattern Recognition Letters*, vol. 24, no. 1–3, pp. 75–80, 2003.

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

[10] S. S. Keerthi and S. K. Shevade, "SMO algorithm for least-squares SVM formulations," *Neural Computation*, vol. 15, no. 2, pp. 487–507, 2003.

[11] L. Bo, L. Jiao, and L. Wang, "Working set selection using functional gain for LS-SVM," *IEEE Transactions on Neural Networks*, vol. 18, no. 5, pp. 1541–1544, 2007.

[12] L. Jian, Z. Xia, X. Liang, and C. Gao, "Design of a multiple kernel learning algorithm for LS-SVM by convex programming," *Neural Networks*, vol. 24, no. 5, pp. 476–483, 2011.

[13] J. C. Platt, "Training of support vector machines using sequential minimal optimization," in *Advances in Kernel Methods: Support Vector Learning*, pp. 185–208, MIT Press, Cambridge, Mass, USA, 1999.

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

[15] R. E. Fan, P. H. Chen, and C. J. Lin, "Working set selection using second order information for training support vector machines," *Journal of Machine Learning Research*, vol. 6, pp. 1889–1918, 2005.

[16] Á. Barbero, J. López, and J. R. Dorronsoro, "Cycle-breaking acceleration of SVM training," *Neurocomputing*, vol. 72, no. 7–9, pp. 1398–1406, 2009.

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

[18] P. H. Chen, R. E. Fan, and C. J. Lin, "A study on SMO-type decomposition methods for support vector machines," *IEEE Transactions on Neural Networks*, vol. 17, no. 4, pp. 893–908, 2006.

[19] Y.-L. Lin, J.-G. Hsieh, H.-K. Wu, and J.-H. Jeng, "Three-parameter sequential minimal optimization for support vector machines," *Neurocomputing*, vol. 74, no. 17, pp. 3467–3475, 2011.

[20] J. López and J. A. K. Suykens, "First and second order SMO algorithms for LS-SVM classifiers," *Neural Processing Letters*, vol. 33, no. 1, pp. 31–44, 2011.

[21] G. R. Rätsch, "Benchmark Repository," *Intelligent Data Analysis Group, Fraunhofer-FIRST, Tech. Rep.*, 2005.

[22] C. C. Chang and C. J. Lin, "LIBSVM: a Library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, article 27, 2011.