# Optimal Feasible Step-size Based Working Set Selection for Large Scale SVMs Training

Shili Peng[a], Qinghua Hu[b,*], Jianwu Dang[b], Wengwu Wang[c]

[a] *Guangdong University of Finance, Guangzhou, 510521, China*
[b] *School of Computer Science and Technology, Tianjin University, Tianjin, 300072, China*
[c] *Centre for Vision, Speech and Signal Processing, University of Surrey, GU2 7XH, UK*

**Abstract**

Efficient training of support vector machines (SVMs) with large-scale samples is of crucial importance in the era of big data. Sequential minimal optimization (SMO) is considered as an effective solution to this challenging task, and the working set selection is one of the key steps in SMO. Various strategies have been developed and implemented for working set selection in LibSVM and Shark. In this work we point out that the algorithm used in LibSVM does not maintain the box-constraints which, nevertheless, are very important for evaluating the final gain of the selection operation. Here, we propose a new algorithm to address this challenge. The proposed algorithm maintains the box-constraints within a selection procedure using a feasible optional step-size. We systematically study and compare several related algorithms, and derive new theoretical results. Experiments on benchmark data sets show that our algorithm effectively improves the training speed without loss of accuracy.

*Keywords:* support vector machines, decomposition algorithm, working set selection, sequential minimal optimization, feasible step-size

## 1. Introduction

Support vector machines (SVMs) are widely used for classification and regression tasks in machine learning [1], [2], [3], [4]. A number of methods in-

*Corresponding author
*Email address:* huqinghua@tju.edu.cn (Qinghua Hu)

cluding the kernel method have been developed and successfully used in various applications [5], [6], [7], [8]. However, the training of SVMs becomes time-consuming for large-scale datasets [9], [10]. To address this limitation, a number of algorithms have recently been developed to accelerate the SVM training procedure [7], [11], [12], [13], [14], [15], [16].

The main computational effort in SVM training is to solve a constrained quadratic optimization problem in which the kernel matrix may be too large to store [17]. To improve computation, a sequence of sub-problems, instead of the entire problem, are often addressed using a decomposition algorithm at each iteration. Sequential minimal optimization (SMO) is one such algorithm [18], [19], [20], that solves a minimal sub-problem with two samples [17]. It can achieve an analytical solution, eliminating the need for using another iterative quadratic programming optimizer at each iteration. SMO has been widely used for training SVMs, and implemented in open-source machine learning libraries such as LibSVM[1] [21], and Shark[2] [22].

The convergence of SMO algorithm is highly dependent on the selection of a working set [23]. Platt et al. developed an alogrithm that uses a heuristics to select the two points [19]. The first point $\alpha_i$ is selected from the samples that violate the Karush-Kuhn-Tucker (KKT) conditions, and the second point $\alpha_j$ is selected to maximize $|E_i - E_j|$, where $E_i$ (or $E_j$) is the difference between the function value and the label of the point. Using the KKT conditions, Keerthi et al. proposed the most violating pair (MVP) selection method, which is related to the first-order approximation of the objective function [24]. Instead of using the first-order approximation, Fan et al. considered second-order information and proposed a working set selection technique, that uses the second-order infromation to select the second point [24]. Additionally, Glasmachers et al. proposed the hybrid maximum gain (HMG) working set selection algorithm for large-scale SVM [25]. HMG reuses one variable from the previous iteration

[1] `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`

[2] `http://shark-project.sourceforge.net/`

to reduce the number of kernel evaluations. LibSVM is a widely used solver that employs SMO to solve the quadratic programming in SVM [21]. To simplify the selection procedure, it does not maintain the box-constraints (that is, $0 \leq \alpha_i \leq C$ where $\alpha_i$ is a dual variable bounded by the factor $C$) in the working set selection.

The motivation of this work is that in the optimization problem with constraints, the constraints have a great influence on the optimization process. Therefore, the influence of box-constraints needs to be considered in the working set selection, especially in cases where no complicated calculations are required. Based on theoretical analysis and numerical experiments, we observe that the box-constraints are very important for working set selection, because box-constraints have a strong impact on the function gain. In this article, we propose an optimal feasible step-size (OFS) selection strategy which considers the box-constraints and can avoid time-consuming computations in the selection procedure. Experiments on extensive benchmark datasets show that the proposed selection strategy clearly improves the training speed without loss of accuracy. The major differences between the proposed OFS selection strategy and LibSVM are listed as follows:

1. OFS maintains the box-constraints in the working set selection, while LibSVM does not maintain the box-constraints in the selection procedure.
2. OFS selects the first point by examining $I_{up}$ and $I_{low}$ sets, but LibSVM selects the first point from $I_{up}$ using the MVP algorithm where $I_{up}$ and $I_{low}$ are feasible point sets.
3. OFS uses a different clipping method to satisfy the box-constraints to avoid time-consuming computations. OFS always keeps a feasible step-size for each point, and uses this step-size to evaluate the objective fcuntion in the selection procedure.

The main contribution of this work is that the OFS algorithm takes into account the feasible step size of the dual variable, which further improves the efficiency of the SVM. In addition, the working set selection method that simply

relies on gradients can only use gradients as stopping criteria. However, the OFS working set selection algorithm can use gradients or function gain as stopping criteria.

The remainder of this paper is organized as follows. Studies related to working set selection are discussed in Section 2. We describe our proposed OFS strategy in Section 3. Theoretical analysis and discussion are presented in Section 4. Extensive experiments on benchmark datasets are described in Section 5. Finally, we summarize the outcomes of this study in Section 6.

## 2. Related Work

In this section, we present the SVMs and decomposition algorithm, and review some existing algorithms for working set selection, such as MVP, LibSVM and HMG.

### *2.1. Support Vector Machines*

Using a function $\Phi(\cdot)$, an SVM maps an input vector $x$ to a high dimensional feature space, and then constructs an optimal separating hyperplane in the feature space [26], [1]. Given $n$ samples $\{(x_1, y_1), ...(x_i, y_i), ..., (x_n, y_n)\}$, the SVM seeks a decision function $f(x) = w^T\Phi(x) + b$ with the maximum margin between different classes in the feature space, where $x_i$ is an input vector and $y_i$ is its class label. Given the mapping function $\Phi(\cdot)$, the parameters $w$ and $b$ are obtained by solving the following convex quadratic problem:

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}\xi_i, \\ s.t. \quad & y_i(w^T\Phi(x) + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad \forall i = 1, ...n, \end{aligned} \tag{1}$$

where $\xi_i$ is a slack variable, and $C$ is a penalty factor.

This problem is often addressed as a dual description to avoid complex feature mapping. The dual problem of the SVM is obtained by introducing the Lagrange multipliers $\alpha$,

$$\begin{aligned} \min_{\alpha} \quad & f(\alpha) = \frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j K(x_i, x_j) - \sum_{i=1}^{n}\alpha_i \\ s.t. \quad & \sum_{i=1}^{n}\alpha_i y_i = 0, \quad 0 \leq \alpha_i \leq C, \quad \forall i = 1, .., n, \end{aligned} \tag{2}$$

where $\alpha_i$ is a dual variable that is bounded by the penalty factor $C$. The inner product can be efficiently computed by a given kernel function $K(x_i, x_j) = \Phi(x_i)^T \Phi(x_j)$ without an explicit feature mapping. Once the optimal solution $\alpha^*$ of the dual problem has been obtained, the optimal primal variables $w^*$, $b^*$ and the discriminant function $f(x)$ can be easily determined according to KKT conditions.

### *2.2. Decomposition Method and Sequential Minimal Optimization*

SVM algorithm is a convex quadratic optimization problem which has been studied extensively. When the problem scale is small, standard optimal toolboxes can be directly used for its training. For large-scale SVM, however, this becomes computationally prohibitive as the memory required for implementing the full kernel matrix grows quadratically with the problem scale [27]. To address this challenge, several algorithms have been developed, one of which is the decomposition method [17], where the idea is to solve a sequence of smaller quadratic programming problems [27]. Instead of updating all of the variables, the decomposition method optimizes a subset of variables $\alpha_i, i \in B$ and leaves the remaining variables $\alpha_j, j \notin B$ unchanged at each iteration, where $B$ is a working set selected with some strategy.

The SMO algorithm is one of the decomposition method that solves a subproblem with the smallest working set size (just two points) at each iteration [19]. The convergence of SMO depends strongly on the working set selection.

The best working set $B^* = \{i, j\}$ should be chosen to minimize the dual objective function at each iteration. Although with the best working set, the number of iterations required can be reduced, the complexity in each iteration becomes higher [24].

In general, the working set is obtained by compromising between the number of iterations and the complexity of each iteration [28]. Platt proposed a selection algorithm that uses two separate heuristics to choose the first and second points [19]. The first point $\alpha_i$ is selected from the samples which violate the KKT conditions, and the second point $\alpha_2$ is selected to maximize the absolute value

of $|E_i - E_j|$, where $E_i$ (or $E_j$) is the difference between the function value and the label of the training sample $x_i$ (or $x_j$).

### *2.3. Most Violating Pair and LibSVM Method*

Keerthi et al. identified a source of inefficiency in Platt's algorithm that uses only one point to maximize $|E_i - E_j|$ [29], and proposed the MVP selection algorithm using the KKT conditions. Considering two different cases of $\alpha_i$, the KKT conditions can be simplified to: 1) $\alpha_i > 0,\ \nabla f(\alpha)_i + \zeta y_i \leq 0$, and 2) $\alpha_i < C,\ \nabla f(\alpha)_j + \zeta y_i \geq 0$. Because $y_i = \pm 1$, the KKT conditions can be rewritten as

$$-y_i \nabla f(\alpha)_i \leq \zeta \leq -y_j \nabla f(\alpha)_j, \quad \forall\, i \in I_{up}(\alpha), j \in I_{low}(\alpha), \tag{3}$$

where $I_{up}$ and $I_{low}$ are defined as

$$\begin{aligned} I_{up}(\alpha) &\equiv \{t | \alpha_t < C, y_t = 1 \ \vee \ \alpha_t > 0, y_t = -1\}, \\ I_{low}(\alpha) &\equiv \{t | \alpha_t < C, y_t = -1 \ \vee \ \alpha_t > 0, y_t = 1\}. \end{aligned} \tag{4}$$

According to the KKT conditions (3), the MVP $\{i, j\}$ can be obtained as

$$\begin{aligned} i &= \arg\max_t \{-y_t \nabla(\alpha^k)_t \ \Big| \ t \in I_{up}(\alpha^k)\}, \\ j &= \arg\min_t \{-y_t \nabla(\alpha^k)_t \ \Big| \ t \in I_{low}(\alpha^k)\}, \end{aligned}$$

and the complexity is $O(n)$ instead of $O(n^2)$.

The MVP method is related to the first-order approximation of the dual objective function $f(\alpha)$. Instead of using the first-order approximation, Fan et al. considered the second-order information to select the second point $\alpha_j$ [24]. LibSVM selects $\alpha_i$ using the same strategy as MVP, and then checks $n$ possible pairs to select $\alpha_j$ according to the function gain by using second-order information.

However, in the working set selection procedure, LibSVM does not consider the box-constraints. After the selection procedure, $\alpha_i$ and $\alpha_j$ are clipped to satisfy $0 \leq \alpha_i \leq C$ and $0 \leq \alpha_j \leq C$[24].

### *2.4. Hybrid Maximum Gain Method*

Glasmachers and Igel proposed the HMG algorithm for large scale SVM [25]. HMG reduces the number of kernel evaluations in each iteration by reusing one variable from the previous iteration. The HMG algorithm evaluates the gain of the objective function $g(\alpha)$ to select the working set at each iteration. In the first iteration, HMG randomly selects a working set $B^0 = \{i_0, j_0\}$ that satisfies $y_{i_0} \neq y_{j_0}$. In the following iterations, given the previous working set $B^{k-1} = \{i_{k-1}, j_{k-1}\}$, HMG searches for the maximum function gain on the working sets $B_k = \{i_{k-1}, j_k\}$ and $B_k = \{i_k, j_{k-1}\}$, where $i_{k-1}, j_{k-1} \in B^{k-1}$ and $i_k, j_k \in \{1, ..., n\}$.

A limitation with maximum gain algorithm is that it may stop without obtaining the optimal solution. To solve this problem, HMG inherits the convergence property of the MVP algorithm by introducing MVP into the maximum gain algorithm [25]. When $\alpha_i$ or $\alpha_j$ is close to the boundary, the MVP algorithm is used to select the working set. The HMG algorithm is implemented in the Shark library [22].

## 3. New Strategies for Working Set Selection

SVMs involve a convex quadratic problem with an equality constraint $\sum_{i=1}^{n} y_i \alpha_i = 0$ and box-constraints $0 \leq \alpha_i \leq C$. In this section, we describe our proposed OFS working set selection. OFS maintains the box-constraints and equality-constraint in the selection procedure to maximize the function gain.

### *3.1. Optimal Feasible Step-size*

Considering the box-constraints, the feasible step-size of $\alpha_i$ is defined as

$$\lambda_{u_i} \equiv C - \alpha_i, \ \lambda_{d_i} \equiv \alpha_i,$$

where $\lambda_{u_i}$ and $\lambda_{d_i}$ indicate the up-direction and down-direction ranges of $\alpha_i$ respectively. The variable $\alpha_i$ is varied within this range, and does not violate the box constraint $0 \leq \alpha_i \leq C$.

In a working set consisting of $\alpha_i$ and $\alpha_j$, these points must change simultaneously to satisfy the equality-constraint. For the convenience of description, we define the feasible step-size of $\alpha_i$ and $\alpha_j$ as $\lambda_{fs_{ij}}$.

**Definition 3.1** (Feasible Step-size)**.** The feasible step-size $\lambda_{fs_{ij}}$ refers to the feasible range of the Lagrange multipliers $\alpha_i$ and $\alpha_j$.

Simultaneous changes to the two variables $\alpha_i$ and $\alpha_j$ within this range do not violate the box-constraints $0 \leq \alpha_i \leq C$ and $0 \leq \alpha_j \leq C$. This is expressed as follows:

$$\lambda_{fs_{ij}} \equiv \begin{cases} \min\{\lambda_{d_i}, \lambda_{d_j}\}, & if\ y_i \neq y_j\ and\ \nabla f(\alpha)_i + \nabla f(\alpha)_j > 0; \\ \min\{\lambda_{u_i}, \lambda_{u_j}\}, & if\ y_i \neq y_j\ and\ \nabla f(\alpha)_i + \nabla f(\alpha)_j < 0; \\ \min\{\lambda_{d_i}, \lambda_{u_j}\}, & if\ y_i = y_j\ and\ \nabla f(\alpha)_i - \nabla f(\alpha)_j > 0; \\ \min\{\lambda_{u_i}, \lambda_{d_j}\}, & if\ y_i = y_j\ and\ \nabla f(\alpha)_i - \nabla f(\alpha)_j < 0; \end{cases} . \quad (5)$$

We first discuss the situation $y_i \neq y_j$ and $\nabla f(\alpha)_i + \nabla f(\alpha)_j > 0$. When $y_i \neq y_j$, according to the equality-constraint, $\alpha_i$ and $\alpha_j$ must increase or decrease simultaneously and with the same step-size $\lambda$. In addition, if $\nabla f(\alpha)_i + \nabla f(\alpha)_j > 0$, $\alpha_i$ and $\alpha_j$ must decrease simultaneously to reduce the objective function value. The corresponding feasible range is then

$$\lambda_{fs_{ij}} = min\{\lambda_{d_i}, \lambda_{d_j}\}.$$

The other formulations in (5) can be derived in a similar manner.

The feasible step-size may not be the optimal step-size. If we only consider the equality-constraint and $Q_{ii} + Q_{jj} - 2Q_{ij} > 0$, the optimal step-size $\lambda_{opt_{ij}}$ is given by the second-order information. The optimal step-size $\lambda_{opt_{ij}}$ is

$$\lambda_{opt_{ij}} = \frac{|y_i \nabla f(\alpha)_i - y_j \nabla f(\alpha)_j|}{Q_{ii} + Q_{ij} - 2Q_{ij}},$$

where the working set $B = \{i, j\}$ is a violating pair. If the optimal solution violates the box-constraints, it needs to be restricted. For non-positive definite kernel matrices, if $Q_{ii} + Q_{jj} - 2Q_{ij} \leq 0$, it is replaced by a small positive number.

**Definition 3.2** (Optimal Feasible Step-size)**.** The optimal feasible step-size $\lambda_{ofs_{ij}}$ is the optimal step-size of $\alpha_i$ and $\alpha_j$ that satisfies the equality-constraint and box-constraints.

We define $\lambda_{ofs_{ij}}$ as:

$$\lambda_{ofs_{ij}} = \min\{\lambda_{fs_{ij}}, \lambda_{opt_{ij}}\}. \tag{6}$$

When the step-size is $\lambda_{ofs_{ij}}$, we obtain the optimal solution that satisfies all constraints. In this study, we use the optimal feasible step-size to select the working set. In order to avoid the step-size from being too small, we restrict $\lambda_{ofs_{ij}}$ to be larger than a small positive number $\tau = 10^{-10}$.

### *3.2. Working Set Selection Using First-Order Information*

In this section, we propose a new selection algorithm called OFS1, which uses the first-order information to approximate the objective function. Using the first-order information, we have the following formulation

$$f(\alpha + \lambda d) - f(\alpha) \approx \lambda \nabla f(\alpha)^T d,$$

where $\lambda$ is the step-size in the feasible direction $d$. Using the optimal feasible step-size $\lambda_{ofs_{ij}}$, we have the following formulation:

$$\max_{\{i,j\}} \lambda \nabla f(\alpha)^T d \approx \max_{\{i,j\}} \{\lambda_{ofs_{ij}}(|\nabla f(\alpha)_i - y_i y_j \nabla f(\alpha)_j|)\}. \tag{7}$$

First, $\alpha_i$ is selected to maximize the violating gradient in sets $I_{up}$ and $I_{low}$. Second, $\alpha_j$ is selected to maximize the approximation function according to (7). The corresponding working set selection algorithm is given in Algorithm 1.

The feasible step-sizes $\{\lambda_{d_i},\ \lambda_{u_i},\ \lambda_{d_j},\ \lambda_{u_j}\}$ and $\{\alpha_i, \alpha_j\}$ are updated after the selection procedure for the next iteration.

### *3.3. Working Set Selection Using the Second-order Information*

Using the second-order information, we can write the following more accurate formulation for the objective value:

$$f(\alpha + \lambda d) - f(\alpha) = \lambda \nabla f(\alpha)^T d + \frac{1}{2}\lambda^2 d^T \nabla^2 f(\alpha) d,$$

**Algorithm 1:** OFS1 Working Set Selection

**Input**: training set $\{(x_i, y_i)\}_{i=1}^n$, gradient $\{\nabla f(\alpha)_i\}_{i=1}^n$, and feasible step-size $\{(\lambda_{d_i}, \lambda_{u_i})\}_{i=1}^n$;

**Output**: working set $B = \{i, j\}$;

1 Select $i$:

$$i = \arg\max_t \begin{cases} -\nabla f(\alpha)_t, & if\ \lambda_{u_t} > 0 \\ \nabla f(\alpha)_t, & if\ \lambda_{d_t} > 0 \end{cases}$$

2 Select $j$:

$$j = \arg\max_t \{\lambda_{ofs_{it}}(|\nabla f(\alpha)_i - y_i y_j \nabla f(\alpha)_j|) \Big\}$$

3 Return $B = \{i, j\}$.

where $\lambda$ is a step-size and $d = [1 \quad -y_i y_j]^T$ satisfies the equality-constraint.

First, we consider the relationship between the feasible step-size $\lambda_{fs_{ij}}$ and the optimal step-size $\lambda_{opt_{ij}}$. If $\lambda_{fs_{ij}} < \lambda_{opt_{ij}}$, the optimal step-size should be restricted using the box-constraints and $\lambda_{fs_{ij}}$ is the optimal solution. Then, we have:

$$\begin{aligned} f(\alpha + \lambda d) - f(\alpha) &= \lambda_{fs_{ij}} \nabla f(\alpha)^T d + \tfrac{1}{2} \lambda_{fs_{ij}}^2 d^T \nabla^2 f(\alpha) d \\ &= -\lambda_{fs_{ij}} |y_i \nabla f(\alpha)_i - y_j \nabla f(\alpha)_j| + \tfrac{1}{2} \lambda_{fs_{ij}}^2 (Q_{ii} + Q_{jj} - 2Q_{ij}) \end{aligned} \tag{8}$$

If $\lambda_{fs_{ij}} \geq \lambda_{opt_{ij}}$, the optimal solution satisfying all of the constraints is $\lambda_{opt_{ij}}$. This gives:

$$\begin{aligned} f(\alpha + \lambda d) - f(\alpha) &= \lambda_{opt_{ij}} \nabla f(\alpha)^T d + \tfrac{1}{2} \lambda_{opt_{ij}}^2 d^T \nabla^2 f(\alpha) d \\ &= -\tfrac{1}{2} \lambda_{opt_{ij}} (|y_i \nabla f(\alpha)_i - y_j \nabla f(\alpha)_j|) \end{aligned} \tag{9}$$

First, $\alpha_i$ is selected using OFS1, and then $\alpha_j$ is selected to maximize the function gain according to (8) and (9). The working set selection algorithm

using the second-order information is summarized in Algrothm 2.

**Algorithm 2:** OFS2 Working Set Selection

**Input**: training set $\{(x_i, y_i)\}_{i=1}^n$, gradient $\{\nabla f(\alpha)_i\}_{i=1}^n$, feasible step-size $\{(\lambda_{d_i}, \lambda_{u_i})\}_{i=1}^n$, and kernel matrix $Q$;

**Output**: working set $B = \{i, j\}$;

1 Select $i$:

$$i = \arg\max_t \begin{cases} -\nabla f(\alpha)_t, & if\ \lambda_{u_t} > 0 \\ \nabla f(\alpha)_t, & if\ \lambda_{d_t} > 0 \end{cases}$$

2 Select $j$:

$$j = \arg\max_t \begin{cases} \lambda_{fs_{it}} |y_i \nabla f(\alpha)_i - y_t \nabla f(\alpha)_t| - \frac{1}{2}\lambda_{fs_{it}}^2 (Q_{ii} + Q_{tt} - 2Q_{it}), & if\ \lambda_{fs_{it}} < \lambda_{opt_{it}} \\ \frac{1}{2}\lambda_{opt_{it}} |y_i \nabla f(\alpha)_i - y_t \nabla f(\alpha)_t|, & if\ \lambda_{fs_{it}} \geq \lambda_{opt_{it}} \end{cases}$$

3 Return $B = \{i, j\}$.

The feasible step-sizes $\{\lambda_{d_i},\ \lambda_{u_i},\ \lambda_{d_j},\ \lambda_{u_j}\}$ and $\{\alpha_i, \alpha_j\}$ are updated after the selection procedure. In SVMs such as LibSVM, the stopping criterion is checked using the information provided in the working set selection. The LibSVM algorithm stops if the sum of the violations of the working set is less than a predefined constant $\epsilon$. The propsed algorithms OFS1 and OFS2 also use the same stopping criterion as in LibSVM.

## 4. Analysis and Discussion

In this section, we discuss the convergence properties of OFS1 and OFS2, and their relation with other selection methods such as the original SMO, MVP and LibSVM algorithms.

### *4.1. Asymptotic Convergence of OFS1 and OFS2*

The decomposition algorithm may not converge to the optimal solution if the working set selection method is inappropriate. Using the results in [30], we

prove that our proposed selection algorithm leads to asymptotic convergence.

The general working set selection procedure defined in [30] consists of the following steps:

1) *Consider a fixed* $0 < \sigma < 1$ *for all iterations.*

2) *Select any* $i \in I_{up}(\alpha),\ j\ \in I_{low}(\alpha)$ *satisfying* :

$$-y_i \nabla f(\alpha)_i + y_j \nabla f(\alpha)_j \geq \sigma(m(\alpha) - M(\alpha)) > 0.$$

3) *Return* $B = \{i, j\}$.

(10)

**Lemma 4.1.** *Let* $\{\alpha^k\}$ *be the infinite sequence generated by the SMO-type method satisfying the above general working set selection. Then, any limit point of* $\{\alpha^k\}$ *is a stationary point of SVM [30].*

The decomposition algorithm converges asymptotically to the optimum, and terminates after satisfying a stopping condition [24].

**Theorem 4.2.** *Let* $\{\alpha^k\}$ *be the infinite sequence generated by the OFS1. Then, any limit point of* $\{\alpha^k\}$ *is a stationary point of SVM. Moreover, if the kernel matrix is positive definite,* $\{\alpha^k\}$ *is globally convergent to the unique minimum of SVM.*

**Proof:** OFS1 selects $\alpha_i$ from $I_{up}$ or $I_{low}$. In order to simplify the proof procedure, we assume that $\alpha_i$ is selected from $I_{up}$, and $y_i = 1$. According to the definition of $I_{up}$ and OFS1, if $y_i = 1$ and $\alpha_i < C$, the second point needs to satisfy $y_j = 1$ and $\alpha_j > 0$ to ensure $\lambda_{ij} > 0$. That is, $\alpha_j$ belongs to $I_{low}$.

Then, OFS1 selects $i = m(\alpha)$ and $j = \arg\max_t \{\lambda_{ofs_{it}}(-\nabla f(\alpha)_i + \nabla f(\alpha)_j) | \lambda_{ofs_{it}} > 0)\}$. And, we ssume that $\bar{j}$ is selected by $\bar{j} = \min_{j \in I_{low}(\alpha)} -y_j \nabla f(\alpha)_j$ (that is, $\bar{j} = M(\alpha)$). As a consequence, the following inequality is satisfied with $\lambda_{ofs_{ij}} > 0$ and $\lambda_{ofs_{i\bar{j}}} > 0$,

$$\lambda_{ofs_{ij}}(-\nabla f(\alpha)_i + \nabla f(\alpha)_j) \geq \lambda_{ofs_{i\bar{j}}}(m(\alpha) - M(\alpha)).$$

Thus,

$$\begin{aligned} -\nabla f(\alpha)_i + \nabla f(\alpha)_j &\geq \frac{\lambda_{ofs_{i\bar{j}}}}{\lambda_{ofs_{ij}}}(m(\alpha) - M(\alpha)) \\ &\geq \frac{\min_{k,l}\{\lambda_{ofs_{kl}}\}}{\max_{k,l}\{\lambda_{ofs_{kl}}\}}(m(\alpha) - M(\alpha)) \\ &= \frac{\tau}{C}(m(\alpha) - M(\alpha)), \end{aligned} \tag{11}$$

where $\tau$ is the minimum step-size, and $C$ is the maximum step-size. Then, the OFS1 algorithm satisfies the inequality (10) for $\sigma = \frac{\tau}{C}$.

Similarly, the other situations can also be proved. If OFS1 selects $\alpha_i$ from the set $I_{low}$, the proof is the same as the above procedure, because we can exchange the class label for binary classification. Then, the OFS1 algorithm is a special case of (10), and all theoretical properties of (10) proved by Lin [30] hold here.

□

For the OFS2 algorithm, if $\lambda_{fs_{ij}} \geq \lambda_{opt_{ij}}$, it selects

$$j = \arg\max\{\frac{1}{2}\lambda_{opt_{ij}}|y_i\nabla f(\alpha)_i - y_j\nabla f(\alpha)_j|\}.$$

The proof process is the same as for the OFS1 algorithm using $\lambda_{opt_{ij}}$ instead of $\lambda_{ofs_{ij}}$. If $\lambda_{fs_{ij}} < \lambda_{opt_{ij}}$, OFS2 selects

$$j = \arg\max_t\{\lambda_{fs_{it}}|y_i\nabla f(\alpha)_i - y_t\nabla f(\alpha)_t| - \frac{1}{2}\lambda^2_{fs_{it}}(Q_{ii} + Q_{tt} - 2Q_{it})\}.$$

We can also find a fixed $\sigma$ to satisfy the inequality (10). Moreover, we can pre-define a small tolerance $\epsilon > 0$ and check whether the maximal violation is sufficiently small, that is, $m(\alpha) - M(\alpha) < \epsilon$. Or more simply, we may check whether the selected working set satisfies $-y_i\nabla f(\alpha)_i + y_j\nabla f(\alpha)_j \leq \epsilon$, because this implies $m(\alpha) - M(\alpha) < \epsilon/\sigma$ [24].

The time complexity of OFS is the same as that of LibSVM. The difference is that the efficiency of OFS is further improved under the same complexity. According to the analysis of Lin et al., the convergence rate of the decomposition algorithm is linear [21]. Therefore, in the worst case, the convergence speed of OFS is also linear. Its time complexity is between $O(D * n^2)$ and $O(N_{sv}^2 + N_{sv} * D * n)$, where $n$ is the number of training samples, $D$ is the original dimension of the sample, and $N_{sv}$ is the number of support vectors [31].

*4.2. Relation of OFS1 to SMO and MVP*

As mentioned above, the SMO algorithm uses two criteria to select the two working variables: $\alpha_i$ is chosen from among those samples that violate the KKT conditions, and $\alpha_j$ is selected to maximize $|E_i - E_j|$, where $E_i$ and $E_j$ are defined in (**??**).

We present an interesting result that $|E_i - E_j|$ is equal to $|y_i \nabla f(\alpha)_i - y_j \nabla f(\alpha)_j|$. According to the definition (**??**), we have

$$\begin{aligned} |E_i - E_j| &= \left| \left( \sum_{t=1}^{n} \alpha_t y_t K(x_t, x_i) + b - y_i \right) - \left( \sum_{t=1}^{n} \alpha_t y_t K(x_t, x_j) + b - y_j \right) \right| \\ &= \left| y_i \left( \sum_{t=1}^{n} \alpha_t y_i y_t K(x_t, x_i) - 1 \right) - y_j \left( \sum_{t=1}^{n} \alpha_t y_j y_t K(x_t, x_j) - 1 \right) \right| \\ &= \left| y_i \nabla f(\alpha)_i - y_j \nabla f(\alpha)_j \right|, \end{aligned}$$

where $\nabla f(\alpha)_i$ and $\nabla f(\alpha)_j$ are the gradients of the dual objective function (2) with respect to $\alpha_i$ and $\alpha_j$,

$$\nabla f(\alpha)_i = \left( \sum_{t=1}^{n} \alpha_t y_i y_t K(x_t, x_i) \right) - 1, \quad \nabla f(\alpha)_j = \left( \sum_{t=1}^{n} \alpha_t y_j y_t K(x_t, x_j) \right) - 1.$$

Therefore, the original SMO algorithm selects the first point $\alpha_i$ from those samples that violate the KKT conditions, and selects the second point $\alpha_j$ just as in the MVP algorithm.

The second interesting result is that the MVP algorithm can be considered as a special case of the OFS1 algorithm which uses the same step-size for all variables. The MVP algorithm selects the working set $B = \{i, j\}$ by solving the optimization problem $\max_{i \in I_{up}, j \in I_{low}} \{-y_i \nabla f(\alpha)_i + y_j \nabla f(\alpha)_j\}$. It may be rewritten as:

$$\begin{aligned} \max_{i \in I_{up}, j \in I_{low}} \{-y_i \nabla f(\alpha)_i + y_j \nabla f(\alpha)_j\} &= \max_{\lambda_{u_i} > 0, \lambda_{d_j} > 0} \{-\nabla f(\alpha)_i + \nabla f(\alpha)_j\} \\ &= \max_{i,j} \{-\nabla f(\alpha)_i + \nabla f(\alpha)_j \;\Big|\; \lambda_{ij} > 0\}. \end{aligned}$$

Without loss of generality, we assume that the step-size $\lambda_{ofs_{ij}}$ is fixed to 1.

The working set $\{i, j\}$ of OFS1 is selected as follows:

$$\begin{aligned} i &= \arg\max_t \{-\nabla f(\alpha)_t \;\Big|\; \lambda_{u_t} > 0,\ \nabla f(\alpha)_t \;\Big|\; \lambda_{d_t} > 0\}, \\ j &= \arg\max_t \{(|\nabla f(\alpha)_i - y_i y_j \nabla f(\alpha)_j|) \Big| \lambda_{ij} > 0\}. \end{aligned} \tag{12}$$

The result can be proved under four conditions according to the class labels $y_i$ and $y_j$. We assume that $y_i = 1$ and $y_j = 1$. According to the definitions of $I_{up}$ and $I_{low}$, we have

$$y_i = 1 \wedge y_j = 1 \Rightarrow \alpha_i < C \wedge \alpha_j > 0 \Rightarrow \lambda_{u_i} > 0 \wedge \lambda_{d_j} > 0.$$

If OFS1 selects $\alpha_i$ according to $i = \arg\max_t \{\nabla f(\alpha)_t \mid \lambda_{u_t} > 0\}$, that is, $i \in I_{up}$, then OFS1 needs to select $j$ from the set $I_{low}$ to make $\lambda_{ij} > 0$, and this implies $-\nabla f(\alpha)_i \geq \nabla f(\alpha)_j$. Hence, we can obtain

$$\max\{| - \nabla f(\alpha)_i + \nabla f(\alpha)_j| \Big| \lambda_{ij}\} = \max\{-\nabla f(\alpha)_i + \nabla f(\alpha)_j \Big| \lambda_{ij}\}.$$

This mean that MVP is the same as the OFS1 algorithm using the same step-size. The other situations can be easily proved in a similar manner.

### *4.3. Relation of OFS2 to LibSVM*

In the working set selection algorithm for LibSVM, $\alpha_i$ is selected (as in MVP) from $I_{up}(\alpha)$, and $\alpha_j$ is selected from $I_{low}(\alpha)$ using the second-order information that disregards the box-constraints. The values of $\alpha_i$ and $\alpha_j$ are bounded after the working set selection to satisfy the box-constraints.

OFS2 uses the second-order information that is similar to LibSVM, but with two important differences. The first is that OFS2 selects $\alpha_i$ from both $I_{up}(\alpha)$ and $I_{low}(\alpha)$ sets, instead of only $I_{up}(\alpha)$. The second difference is that OFS2 considers the box-constraints in selecting $\alpha_j$.

Two points should be stated regarding the effect of maintaining the box-constraints when selecting the working set. First, box-constraints are an important factor affecting the function gain, and second, the non-degenerate points that can ignore the box-constraints constitute only a small part of the support vectors (SVs) in some data.

**1)** Box-constraints have a strong impact on the function gain. When the optimal solution is located in the feasible region, the box-constraints are non-active and do not influence the optimal solution. However, if the optimal point is outside the feasible region, the box-constraints are active and affect the optimal solution. Using the second-order information, we have the following equation:

$$f(\alpha + \lambda d) - f(\alpha) = \lambda \nabla f(\alpha)^T d + \frac{1}{2}\lambda^2 d^T \nabla^2 f(\alpha) d,$$

where $\lambda$ is the step-size, $\nabla f(\alpha) = [\nabla f(\alpha)_i \;\; \nabla f(\alpha)_j]^T$, and $d = [1, -y_i y_j]^T$ is a feasible direction. Solving this sub-problem, we obtain the optimal step-size $\lambda^*$ without referring to the box-constraints as follows

$$\lambda^* = -\frac{(\nabla f(\alpha)_i) - y_i y_j \nabla f(\alpha)_j)}{(Q_{ii} + Q_{ii} - 2y_i y_j Q_{ij})}.$$

The optimal function gain without the box-constraints is

$$g(\lambda^*) = f(\alpha^*) - f(\alpha) = -\frac{1}{2}\lambda^{*2}(Q_{ii} + Q_{ii} - 2y_i y_j Q_{ij}).$$

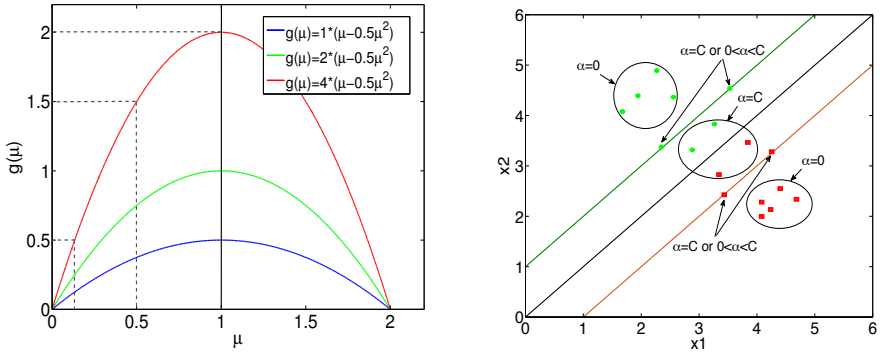
Considering the box-constraints $0 \leq \alpha \leq C$, we obtain the feasible optimal step-size $\lambda'$. That is, $\lambda^*$ is clipped by the box-constraints and denoted by $\lambda' = \mu\lambda^*$, where $0 \leq \mu \leq 1$. We can then rewrite the objective function gain $g(\mu)$ with respect to $\mu$ as:

$$\begin{aligned} g(\mu) &= f(\alpha + \mu\lambda^* d) - f(\alpha) \\ &= \mu\lambda^* \nabla f(\alpha)^T d + \tfrac{1}{2}(\mu\lambda^*)^2 d^T \nabla^2 f(\alpha) d \\ &= \mu\lambda^* [\nabla f(\alpha)_i - y_i y_j \nabla f(\alpha)_j] + \tfrac{1}{2}(\mu\lambda^*)^2 ((Q_{ii} + Q_{ii} - 2y_i y_j Q_{ij})) \\ &= -\tfrac{1}{2}\lambda^{*2}(Q_{ii} + Q_{ii} - 2y_i y_j Q_{ij})(2\mu - \mu^2) \\ &= g(\lambda^*)(2\mu - \mu^2) \end{aligned} \tag{13}$$

The above equation shows that the function gain is a quadratic function with respect to $\mu$. When $0 < \mu \ll 1$, the objective gain sharply decreases because of the box-constraints. More importantly, the coefficient of the gain function is $g(\lambda^*)$. The larger the value of $g(\lambda^*)$, the greater is the impact of the box-constraints. This phenomenon is shown in Figure 1(a). The function gain becomes smaller as $\mu$ decreases, and it is proportional to the coefficient $g(\lambda^*)$.

**2)** For noisy data typically there are a large number of bounded support vectors, and the number of non-degenerate points (i.e. $0 < \alpha_i < C$) is a small part of SVs. Fan et al. showed that the box-constraints do not affect the working set selection under the assumptions that the kernel matrix $Q$ is positive definite and the optimal solution satisfies the non-degeneracy condition [24]. If all the SVs are non-degenerate points, the box-constraints can be ignored. However, the non-degenerate SVs only constitute a small portion of the SVs for some data.

To clarify this problem, we consider a linear SVM classifier. For a non-degenerate point $0 < \alpha_i < C$, we have $y_i(w^T x_i + b) = 1$. This means that it lies on the margin. That is, all of the samples that satisfy the non-degenerate condition lie on the margin. However, the samples that lie on the margin do not necessarily satisfy the non-degenerate condition. For clarity, this situation is shown in Figure 1(b).




(a) Function gain becomes smaller as $\mu$ decreases, and it is proportional to the coefficient $g(\lambda^*)$.

(b) The non-degenerate points lie on the margin and are only a part of the SVs.

Figure 1: Box-constraints have a strong impact on the function gain, and the non-degenerate SVs that can ignore the box-constraints only constitute a small portion of the SVs for some data.

## 5. Experiments

Our experiments focus on the number of iterations and running-time for OFS1, OFS2 and LibSVM-3.21.

The implementations of OFS1 and OFS2 were based on the LibSVM-3.21 library, and the time required for each iteration is very similar. Therefore, the performance improvement of the new algorithms mainly depends on reducing the number of iterations. That is, the actual running time of the OFS algorithm is reduced by reducing the number of iterations. We used a grid search technique using 5-fold cross-validation to determine the optimal hyper-parameter values of various kernel functions for different data sets. The iterations and runtimes presented in this paper are the sum over the 5-fold cross-validation.

The runtimes given in the experimental results were achieved on a Linux operating system with an Intel Core i7-6820 CPU (2.7 GHz) and DDR4 RAM (32 GB). The stopping criterion was the same for all the compared algorithms: $|y_i \nabla f(\alpha)_i - y_j \nabla f(\alpha)_j| < \epsilon$, where $\epsilon = 10^{-3}$.

### *5.1. Datasets and Experimental Settings*

Experiments were conducted on nine datasets. Most of these were obtained from the National Taiwan University[3] [21], with the others taken from UCI [32], Statlog [33], and OpenML[4]. OpenML is an online platform in which scientists can automatically log and share machine learning datasets, code, and experiments, allowing them to build directly on the work of others [34]. Table 1 presents detailed information about all of the experimental datasets. For multi-class classification, we used the one-against-one strategy as LibSVM does.

We examined the performance of OFS1 and OFS2 algorithms with different kernel functions: radial basis function (RBF) kernel $K(x_i, x_j) = exp(-\gamma \parallel x_i - x_j \parallel^2)$, polynomial kernel $K(x_i, x_j) = (\gamma(x_i^T x_j + r))^d$, and sigmoid kernel $K(x_i, x_j) = tanh(\gamma(x_i^T x_j + r))$.

[3] `http://ntucsu.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`

[4] `http://www.openml.org/`

Table 1: Dataset information

| Dataset | Samples | Features | Classes |
|---|---|---|---|
| mushroom | 8,124 | 112 | 2 |
| ijcnn1 | 49,990 | 22 | 2 |
| covtype | 100,000 | 54 | 2 |
| skin | 245,057 | 3 | 2 |
| svmguide3 | 1,243 | 21 | 2 |
| kddcup99 | 494,020 | 41 | 23 |
| mnist | 8,000 | 780 | 10 |
| w7a | 24,692 | 300 | 2 |
| cod-rna | 59,535 | 8 | 2 |

The radial basis function kernel is also called Gaussian kernel function, which is one of the most commonly used kernel functions in SVMs. The polynomial kernel is well-suited to problems in which the training data are normalized. Its adjustable parameters are the slope $\gamma$, the constant term $r$ and the polynomial degree $d$. The sigmoid kernel is also known as the hyperbolic tangent Kernel or multilayer perceptron kernel. It comes from neural networks, where the bipolar sigmoid function is often used as an activation function [23]. The sigmoid kernel matrix $Q$ may not be positive semi-definite.

We use a grid search technique with 5-fold cross-validation to find out the optimal parameter values of various kernels. They are listed in Table 2 to facilitate the reproduction of experimental results. The parameters of the sigmoid kernel for datasets *mnist* is not given in the table, as they have very low precision and too long running-time.

### *5.2. Comparison of Working Set Selection*

There are several different algorithms for working set selection, such as MVP, LibSVM and HMG. In our experiments, we found that LibSVM is one of the fastest working set selection algorithms currently available. We conducted a comparative analysis for HMG and LibSVM (both used in the shark library).

Table 2: Parameters used for the various kernels.

| Dataset | RBF kernel | | Polynomial kernel | | | | Sigmoid kernel | | |
|---|---|---|---|---|---|---|---|---|---|
| | $C$ | $\gamma$ | $C$ | $\gamma$ | $r$ | $d$ | $C$ | $\gamma$ | $r$ |
| mushroom | 0.5 | 0.25 | 0.03125 | 1 | 0.25 | 3 | 16 | 0.03125 | 0.03125 |
| ijcnn1 | 64 | 1.0 | 1 | 1 | 4 | 3 | 1 | 0.125 | 0.03125 |
| covtype | 64 | 0.5 | 1 | 1 | 0.25 | 3 | 16 | 0.0625 | 0.0 |
| skin | 8 | 256 | 1 | 4 | 0.5 | 3 | 1 | 0.125 | 0.25 |
| KDD99 | 64 | 0.5 | 1 | 0.5 | 4 | 3 | 16 | 0.0625 | 0.03125 |
| svmguide3 | 64 | 0.125 | 4 | 1 | 4 | 3 | 1 | 0.0625 | 0.03125 |
| mnist | 2 | 0.0625 | 0.03125 | 0.125 | 4 | 3 | - | - | - |
| w7a | 8 | 0.0625 | 8 | 0.25 | 0 | 3 | 8 | 0.03125 | 0.0 |
| cod-rna | 32 | 1.0 | 1.0 | 4.0 | 2.0 | 3 | 32 | 0.03125 | 0.0 |

On our test datasets, we found that the average running-time of LibSVM is 52.66% of HMG. Therefore, we chose LibSVM as the reference for the new algorithm. The experiments used the same stopping criteria for all algorithms, and results are presented in Tables 3, 4 and 5.

In experiments, we found that the optimal objective function values of different algorithms are almost the same, and their prediction accuracy is also very similar. The accuracy of each algorithm is in the right column of the table. Both OFS1 and OFS2 use the second-order information when calculating the feasible step size and save it in memory. Their difference is that OFS1 only considers first-order information in the working set selection process to simplify the selection. In the OFS2 algorithm, the second-order information is also considered in the working set selection process, and the function gain is accurately calculated. Therefore, each iteration of OFS2 is longer than OFS1, but OFS2 has fewer iterations. Generally, the overall running-time of the OFS2 algorithm is shorter than that of OFS1, and the OFS2 is recommended. However, on some special data sets, if the number of iterations of OFS1 is very close to OFS2, the running time of OFS1 is shorter.

Table 3: Comparison of the number of iterations and runtime of LibSVM, OFS1, and OFS2 using the RBF kernel. The iterations and runtime are the average of 5-fold cross-validation. The ratio is defined as the ratio of the corresponding value and LibSVM. In each case, the best value is highlighted.

| Dataset | Iterations | | | Runtime (s) | | | Accuracy (%) | | |
|---|---|---|---|---|---|---|---|---|---|
| | LibSVM | OFS1 | OFS2 | LibSVM | OFS1 | OFS2 | LibSVM | OFS1 | OFS2 |
| mushroom | 80345 | **60600** | 61477 | 40 | **33** | 34 | 100.00 | 100.00 | 100.00 |
| ijcnn1 | 169300 | 95145 | **95790** | 175 | 131 | **130** | 98.50 | 98.51 | 98.50 |
| covtype | 1430549 | 702645 | **649107** | 4986 | 3707 | **3537** | 83.78 | 83.57 | 83.79 |
| skin | 35317 | **19451** | 19560 | 358 | **222** | 223 | 99.97 | 99.97 | 99.97 |
| kddcup99 | 129882 | **92349** | 93058 | 355 | **309** | 310 | 99.94 | 99.94 | 99.94 |
| svmguide3 | 22268 | 16414 | **14790** | 0.64 | 0.48 | **0.45** | 84.47 | 84.47 | 84.47 |
| mnist | 308363 | **304975** | 305266 | 290 | 284 | **281** | 94.91 | 94.90 | 94.90 |
| cod-rna | 172775 | **125466** | 113327 | 280 | 237 | **230** | 95.42 | 95.41 | 95.41 |
| w7a | 37277 | **35401** | 32790 | 40 | 38 | **37** | 98.71 | 98.71 | 98.71 |
| avgerage | 265120 | 161382 | **153907** | 752 | 551 | **531** | 95.08 | 95.05 | 95.08 |
| (ratio) | (1.000) | (0.609) | **(0.581)** | (1.000) | (0.760) | **(0.733)** | (1.000) | (0.9997) | (1.000) |

Table 4: Comparison of the number of iterations and runtime of LibSVM, OFS1, and OFS2 for the polynomial kernel. The iterations and runtime are the average of 5-fold cross-validation. The ratio is defined as the ratio of the corresponding value and LibSVM. In each case, the best value is highlighted.

| Dataset | Iterations | | | Runtime (s) | | | Accuracy (%) | | |
|---|---|---|---|---|---|---|---|---|---|
| | LibSVM | OFS1 | OFS2 | LibSVM | OFS1 | OFS2 | LibSVM | OFS1 | OFS2 |
| mushroom | 25241 | 15585 | **15274** | 6.69 | 5.52 | **5.25** | 100.00 | 100.00 | 100.00 |
| ijcnn1 | 536776 | 279071 | **264186** | 407 | 235 | **226** | 97.91 | 97.88 | 97.88 |
| covtype | 2767628 | 1344649 | **1247943** | 7702 | 5139 | **4842** | 81.84 | 81.76 | 81.85 |
| skin | 621478 | 452966 | **418787** | 1730 | 1447 | **1365** | 99.83 | 99.82 | 99.83 |
| kddcup99 | 248875 | 148022 | **133620** | 406 | 300 | **283** | 99.93 | 99.93 | 99.93 |
| svmguide3 | 316743 | 236144 | **228717** | 5.72 | 4.42 | **4.11** | 84.79 | 84.79 | 84.79 |
| mnist | **117334** | 121452 | 121288 | **523** | 545 | 538 | 95.30 | 95.30 | 95.29 |
| cod-rna | 6510558 | 3748627 | **3199943** | 4788 | 2787 | **2306** | 95.31 | 95.30 | 95.30 |
| w7a | 16272 | 13212 | **13118** | 22 | 21.7 | **21.2** | 98.50 | 98.52 | 98.52 |
| avgerage | 1240101 | 706636 | **626986** | 1732 | 1165 | **1065** | 94.82 | 94.81 | 94.82 |
| (ratio) | (1.000) | (0.570) | **(0.506)** | (1.000) | (0.673) | **(0.615)** | (1.000) | (0.9999) | (1.000) |

Table 4 indicates that using the polynomial kernel is similar to using the RBF kernel. The proposed algorithms can improve the training speed without loss of accuracy. On some datasets, the running-time of the polynomial kernel is significantly greater than the Gaussian kernel, such as the skin dataset. But on other data sets, the polynomial kernel takes less time than the Gaussian kernel, such as the mushroom data set. The main reason for this difference is that running-time is greatly affected by hyperparameters. For example, in the mnist dataset, the hyperparameter $C$ of the polynomial kernel is 0.03125, and the number of iterations and running-time of OFS is slightly larger than LibSVM. In general, the Gaussian kernel is usually faster than the polynomial kernel, and the performance improvement of the OFS2 algorithm is more obvious.

Table 5: Comparison of the number of iterations and runtime of LibSVM, OFS1, and OFS2 for the sigmoid kernel. The iterations and runtime are the average of 5-fold cross-validation. The ratio is defined as the ratio of the corresponding value and LibSVM. In each case, the best value is highlighted.

| Dataset | Iterations | | | Runtime (s) | | | Accuracy (%) | | |
|---|---|---|---|---|---|---|---|---|---|
| | LibSVM | OFS1 | OFS2 | LibSVM | OFS1 | OFS2 | LibSVM | OFS1 | OFS2 |
| mushroom | 10873 | **6491** | 7672 | 2.2 | **1.7** | 1.8 | 99.80 | 99.83 | 99.83 |
| ijcnn1 | 40841 | 28900 | **27436** | 329 | 243 | **237** | 90.61 | 90.63 | 90.62 |
| covtype | 137011 | 88837 | **87696** | 2725 | 1998 | **1994** | 63.05 | 63.02 | 63.02 |
| skin | 245909 | 152286 | **102447** | 10294 | 6060 | **4019** | 85.84 | 85.85 | 85.84 |
| kddcup99 | 95609 | 69930 | **65846** | 533 | 457 | **430** | 99.83 | 99.83 | 99.83 |
| svmguide3 | 2618 | **2099** | 2154 | 0.37 | **0.31** | 0.32 | 79.08 | 79.08 | 79.16 |
| cod-rna | 52188 | **38464** | 43000 | 281 | **239** | 248 | 93.40 | 93.37 | 93.39 |
| w7a | 7351 | 6307 | **7427** | 107 | **100** | 112 | 97.61 | 97.62 | 97.73 |
| arith. avg. | 74050 | 49164 | **42960** | 1771 | 1126 | **868** | 88.65 | 88.65 | 88.68 |
| (ratio) | (1.000) | (0.663) | **(0.580)** | (1.000) | (0.636) | **(0.490)** | (1.000) | (1.000) | (1.0003) |

From Table 5, we can see that the OFS algorithm obtains greater improvement when using the sigmoid kernel.

In addition, it should be noted that on some datasets such as kddcup99 and covtype, the performance improvement of OFS is very significant. However, on

other datasets such as mnist, the performance improvement of OFS is small. Its running-time is almost the same as LibSVM. The performance improvement of OFS is closely related to the characteristics of the datasets. When the data dimension is high, the running-time is mainly used for the calculation of the kernel matrix. If the OFS algorithm does not significantly reduce the number of iterations, its running-time is very close to LibSVM. For example, the mnist dataset has 780 features. With the Gaussian kernel function, the number of iterations of LibSVM is 308363, and the running-time is 290 seconds. The number of iterations of OFS2 is 305266 and the running-time is 281 seconds. However, in the experiments of kddcup99, covtype and skin, the number of iterations of the OFS algorithm is significantly less than that of LibSVM, and the corresponding running-time is also greatly reduced. Moreover, the performance improvement of OFS is closely related to the support vector (SV) proportion of the data set. For example, in the mnist dataset (with the Gaussian kernel), the percentage of SV is 48.125%, and the running-time of OFS is 97.17% of LibSVM, with almost no speed increase. However, in other data sets, SV accounts for a small proportion, and the OFS performance improvement is relatively clear. For example, in the skin, covtype, kddcup99, and ijcnn1 datasets, the SV proportions are 0.79%, 3.31%, 1.14% and 4.86%, respectively. The runtimes of OFS were 59.36%, 74.35%, 86.47% and 74.56% of those of LibSVM, respectively. Furthermore, we found that the performance improvement of OFS using polynomial kernels and sigmoid kernels is more significant than that of Gaussian kernels. Generally, for the same data set, the running-time of the Gaussian kernel is often less than that of the polynomial and sigmoid kernels.

### *5.3. Experiments Using Different Hyper-parameters*

This section analyzes the hyper-parameters that affect the number of iterations and runtime of the SVM using the *skin* data set. We examined the RBF kernel, as it has some nice properties. We analyzed these hyper-parameters independently to simplify the experiments. The experimental results are given in Figure 2 and 3.

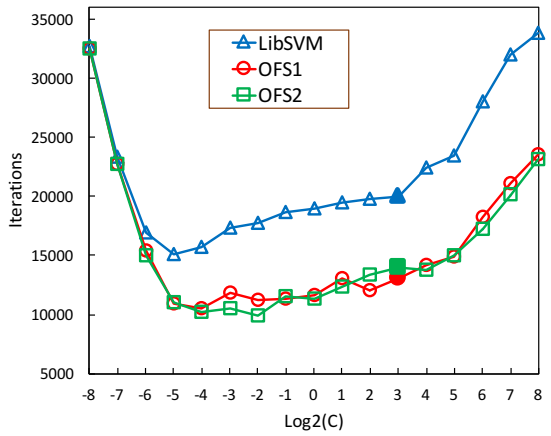



Figure 2: Comparison of the number of iterations required for different hyper-parameters with various $C$, where $\gamma = 256$. The solid point represents the optimal value of parameter.

Figure 2 presents the relation required between the number of iterations and the regularization parameter $C$. Note that the box-constraints cannot be ignored although $C$ is very large, because the constraint $0 \leq \alpha$ always exist.

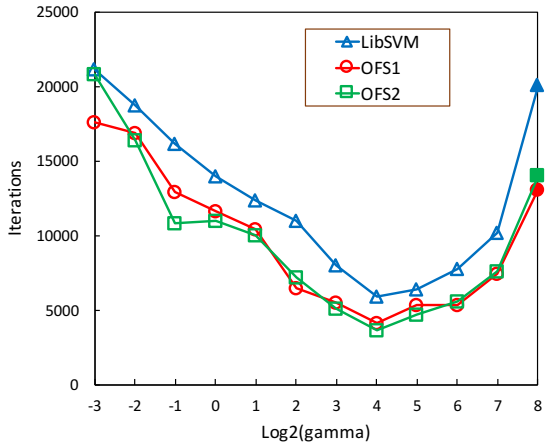



Figure 3: Comparison of the number of iterations for different hyper-parameters with various $\gamma$, where $C = 8$. The solid point represents the optimal value of parameter.

Figure 3 presents the relation between the number of iterations and hyper-

parameter $\gamma$. When the value of the hyperparameter is very small, the performance improvement of OFS is not obvious.

Moreover, in the mnist dataset, the optimal hyperparameter $C$ of the polynomial kernel is 0.03125, and the number of iterations and running-time of OFS are slightly larger than LibSVM.

## 6. Conclusions

Training SVMs involves solving constrained quadratic optimization problems. For large scale data, these are usually solved by decomposition algorithms such as SMO, and the performance of the SMO algorithm is highly dependent on the working set selection scheme. In this paper, we have presented the OFS strategy for working set selection. In addition, we provide a theoretical analysis for different selection methods. Finally, experiments were conducted to compare the performance of different selection algorithms. In smmary, we can state the following main conclusions:

1. Box-contraints have a great impact on the function gain and we have empirically shown that it is important for working set selection.
2. The OFS strategy considers box-constraints and can avoid time-consuming computations. Experiments show that this method efficiently improves the SVM training speed.
3. In general, working set selection using the second-order information is better than the first-order information. However, the OFS1 algorithm is effective for some data, even though it uses the first-order information to approximate the function gain.

The strategies for working set selection are crucial for SVMs training. The OFS strategy can be combined with other efficient solutions, such as parallel training. Moreover, the proposed idea can also be used in multi-kernel learning. In future work, we will study these methods using the optimal step-size method.

**Acknowledgement**

This work is supported by the National Natural Science Foundation of China (NSFC) under Grants 61732011, 61432011 and U1435212.

**References**

[1] C. Cortes, V. Vapnik, Support-vector networks, Machine learning 20 (3) (1995) 273–297.

[2] G. Loosli, S. Canu, C. S. Ong, Learning svm in krein spaces, IEEE Transactions on Pattern Analysis and Machine Intelligence 38 (2016) 1204–1216.

[3] M. Neumann, R. Garnett, C. Bauckhage, K. Kersting, Propagation kernels: efficient graph kernels from propagated information, Machine Learning 102 (2016) 209–245.

[4] H. Huang, X. Wei, Y. Zhou, Twin support vector machines: A survey, Neurocomputing 300 (2018) 34 – 43.

[5] E. Bron, M. Smits, J. van Swieten, W. Niessen, S. Klein, Feature selection based on SVM significance maps for classification of dementia, in: Machine Learning in Medical Imaging, Springer, 2014, pp. 272–279.

[6] M. Donini, F. Aiolli, Learning deep kernels in the space of dot product polynomials, in: Machine Learning, 2016, pp. 1–25.

[7] J. Baek, E. Kim, A new support vector machine with an optimal additive kernel, Neurocomputing 329 (2019) 279 – 299.

[8] W. Li, D. Dai, M. Tan, D. Xu, L. V. Gool, Fast algorithms for linear and kernel svm+, in: IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2258–2266.

[9] Z. Wang, K. Crammer, S. Vucetic, Breaking the curse of kernelization: Budgeted stochastic gradient descent for large-scale SVM training, The Journal of Machine Learning Research 13 (1) (2012) 3103–3131.

[10] F. Nie, W. Zhu, X. Li, Decision tree svm: An extension of linear svm for non-linear classification, Neurocomputing.

[11] K.-W. Chang, C.-J. Hsieh, C.-J. Lin, Coordinate descent method for large-scale l2-loss linear support vector machines, The Journal of Machine Learning Research 9 (2008) 1369–1398.

[12] Y.-W. Chang, C.-J. Hsieh, K.-W. Chang, M. Ringgaard, C.-J. Lin, Training and testing low-degree polynomial data mappings via linear SVM, The Journal of Machine Learning Research 11 (2010) 1471–1490.

[13] C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S. S. Keerthi, S. Sundararajan, A dual coordinate descent method for large-scale linear SVM, in: Proceedings of the 25th International Conference on Machine Learning, ACM, 2008, pp. 408–415.

[14] M. Nandan, P. P. Khargonekar, S. S. Talathi, Fast SVM training using approximate extreme points, The Journal of Machine Learning Research 15 (1) (2014) 59–98.

[15] M. Salhov, A. Bermanis, G. Wolf, A. Averbuch, Learning from patches by efficient spectral decomposition of a structured kernel, Machine Learning 103 (2015) 81–102.

[16] I. Steinwart, P. Thomann, liquidSVM: A fast and versatile svm package, ArXiv e-prints 1702.06899.
URL `http://www.isa.uni-stuttgart.de/software`

[17] N. Cristianini, J. Shawe-Taylor, An introduction to support vector machines and other kernel-based learning methods, Cambridge University Press, 2000.

[18] S. S. Bucak, R. Jin, A. K. Jain, Multiple kernel learning for visual object recognition: A review, IEEE Transactions on Pattern Analysis and Machine Intelligence 36 (2014) 1354–1369.

[19] J. C. Platt, Fast training of support vector machines using sequential minimal optimization, Advances in Kernel Methods (1999) 185–208.

[20] S. V. N. Vishwanathan, Z. sun, N. Ampornpunt, M. Varma, Multiple kernel learning and the smo algorithm, in: Advances in Neural Information Processing Systems, 2010, pp. 2361–2369.

[21] C.-C. Chang, C.-J. Lin, LIBSVM: a library for support vector machines, ACM Transactions on Intelligent Systems and Technology (TIST) 2 (3) (2011) 27.

[22] C. Igel, V. Heidrich-Meisner, T. Glasmachers, Shark, The Journal of Machine Learning Research 9 (2008) 993–996.

[23] H.-T. Lin, C.-J. Lin, A study on sigmoid kernels for SVM and the training of non-psd kernels by SMO-type methods, Neural Computation (2003) 1–32.

[24] R.-E. Fan, P.-H. Chen, C.-J. Lin, Working set selection using second order information for training support vector machines, The Journal of Machine Learning Research 6 (2005) 1889–1918.

[25] T. Glasmachers, C. Igel, Maximum-gain working set selection for SVMs, The Journal of Machine Learning Research 7 (2006) 1437–1466.

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

[27] T. Joachims, Making large-scale SVM learning practical, in: B. Schölkopf, C. Burges, A. Smola (Eds.), Advances in Kernel Methods - Support Vector Learning, MIT Press, Cambridge, MA, 1999, Ch. 11, pp. 169–184.

[28] N. List, H. U. Simon, General polynomial time decomposition algorithms, in: Learning Theory, Springer, 2005, pp. 308–322.

[29] S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, K. R. K. Murthy, Improvements to Platt's SMO algorithm for SVM classifier design, Neural Computation 13 (3) (2001) 637–649.

[30] C.-J. Lin, A study on SMO-type decomposition methods for support vector machines, IEEE Transactions on Neural Networks 17 (4) (2006) 893–908.

[31] C. Burges, A tutorial on support vector machines for pattern recognition, Data mining and knowledge discovery 2 (2) (1998) 121–167.

[32] M. Lichman, UCI machine learning repository (2013).
URL `http://archive.ics.uci.edu/ml`

[33] R. D. King, Statlog databases, Department of Statistics and Modelling Science, University of Strathclyde, Glasgow, UK.

[34] J. Vanschoren, J. N. van Rijn, B. Bischl, Taking machine learning research online with OpenML, The Journal of Machine Learning Research (2015) 1–4.

[35] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, C.-J. Lin, Liblinear: A library for large linear classification, The Journal of Machine Learning Research 9 (2008) 1871–1874.