Safe Screening of Non-Support Vectors in Pathwise SVM Computation

Kohei Ogawa OGAWA.MLLAB.NIT@GMAIL.COM
Yoshiki Suzuki SUZUKI.MLLAB.NIT@GMAIL.COM
Ichiro Takeuchi TAKEUCHI.ICHIRO@NITECH.AC.JP
Nagoya Institute of Technology, Nagoya, Japan

Abstract

In this paper, we claim that some of the non-support vectors (non-SVs) that have no influence on the SVM classifier can be screened out *prior to* the training phase in pathwise SVM computation scenario, in which one is asked to train a sequence (or path) of SVM classifiers for different regularization parameters. Based on a recently proposed framework so-called *safe screening rule*, we derive a rule for screening out non-SVs in advance, and discuss how we can exploit the advantage of the rule in pathwise SVM computation scenario. Experiments indicate that our approach often substantially reduce the total pathwise computation cost.

1. Introduction

The support vector machine (SVM) is one of the most successful classification algorithms. An advantage of the SVM is that, once we identify non-support vectors (non-SVs) that do not have any influence on the classifier, they can be thrown away in the future test phase. In this paper, we show that, in a certain scenario, some of the non-SVs can be screened out and they can be thrown away *prior to* the training phase, which often leads to substantial reduction in the training cost.

The main contribution of this paper is to introduce an idea of recently proposed *safe screening rule* for the purpose of identifying non-SVs in SVM prior to training phase. The safe screening rule was first introduced in Ghaoui et al. (2010) in the context of L_1 sparse regularization, and some extensions have been recently reported (Xiang et al., 2012; Xiang & Ramadge, 2012; Wang et al., 2012; Dai & Pelckmans, 2012). Those rules allow us to *safely* identify the features whose coefficients would be zero at the optimal solution before actually solving it[1]. Our contribution is in non-trivially adapting the idea of safe feature screening to non-SV screening in SVM. To the best of our knowledge, there are no other screening rules that can "safely" screen-out non-SVs of standard kernel-based SVM before actually training it.

In this paper, we argue that the advantage of our non-SV screening rule can be fully exploited in a *pathwise computation* scenario where a sequence (or path) of SVM classifiers for different regularization parameters must be trained. Since pathwise computation is indispensable for model selection, our approach, that can often substantially reduce the total cost, would be practically useful.

2. Problem Setup

Suppose we have a training set $D_{\mathbb{N}} := \{(x_i, y_i)\}_{i \in \mathbb{N}}$, where $x_i \in \mathcal{X} \subseteq \mathbb{R}^d$, $y_i \in \{-1, 1\}$ and $\mathbb{N} := \{1, \ldots, n\}$. We consider classifiers in the form[2]:

$$f(x) = w^\top x, \tag{1}$$

where w is a vector in a feature space that is often defined implicitly by a kernel function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. Our task is to train a support vector machine (SVM) by solving the following optimization problem

$$\min_{w \in \mathbb{R}^d} \frac{1}{2}||w||^2 + C \sum_{i \in \mathbb{N}} [1 - y_i f(x_i)]_+, \tag{2}$$

where $[z]_+$ indicates the positive part of z. Here, $C > 0$ is a regularization parameter that controls the balance

Proceedings of the 30^{th} International Conference on Machine Learning, Atlanta, Georgia, USA, 2013. JMLR: W&CP volume 28. Copyright 2013 by the author(s).

[1] Note that these safe rules are different from *non-safe* screening heuristics such as (Fan & Lv, 2008; Tibshirani et al., 2011; Vats, 2012).

[2] The bias term can be augmented to w and x as an additional dimension. See also the discussion in section 6.

between the regularization term and the loss term.

The dual of (2) is the following maximization problem

$$\max_{\alpha} \ -\frac{1}{2}\alpha^\top Q\alpha + \mathbf{1}^\top\alpha \text{ s.t. } \alpha \in [0, C]^n, \quad (3)$$

where α is an n-dimensional vector of Lagrange multipliers, and Q is an $n \times n$ matrix whose element is defined as $Q_{ij} := y_i y_j K(x_i, x_j)$. With the Lagrange multipliers, the classifier (1) is rewritten as

$$f(x) = \sum_{i \in \mathbb{N}} \alpha_i y_i K(x, x_i). \quad (4)$$

If we categorize the n training instances into

$$\begin{aligned} \mathcal{R} &:= \{i \in \mathbb{N} | y_i f(x_i) > 1\}, & (5a)\\ \mathcal{E} &:= \{i \in \mathbb{N} | y_i f(x_i) = 1\}, & (5b)\\ \mathcal{L} &:= \{i \in \mathbb{N} | y_i f(x_i) < 1\}, & (5c) \end{aligned}$$

the optimality conditions of the problems (2) or (3) are summarized as

$$\begin{aligned} i \in \mathcal{R} &\Rightarrow \alpha_i = 0, & (6a)\\ i \in \mathcal{E} &\Rightarrow \alpha_i \in [0, C], & (6b)\\ i \in \mathcal{L} &\Rightarrow \alpha_i = C, & (6c) \end{aligned}$$

where $\mathcal{E}$ stands for the points at the $\mathcal{E}$lbow of the hinge loss function, while $\mathcal{R}$ and $\mathcal{L}$ are for $\mathcal{R}$ight and $\mathcal{L}$eft of the elbow.

The optimality condition (6a) suggests that, if some of the training instances are known to be the members of $\mathcal{R}$ in advance, we can throw away those instances prior to the training stage. Similarly, if we know that some instances are the members of $\mathcal{L}$, we can fix the corresponding $\alpha_i = C$ at the following training phase. Namely, if some knowledge on these three index sets are known *a priori*, our training task would be extremely simple and easy. In fact, existing SVM solvers usually spend most of their computational resources for identifying these index sets (Joachims, 1999; Platt, 1999; Cauwenberghs & Poggio, 2001; Hastie et al., 2004; Fan et al., 2008; Chang & Lin, 2011).

The instances in $\mathcal{R}$ are often called *non-support vectors (non-SVs)* because they have no influence on the resulting classifier. In this paper, we show that, in a certain situation, some of the non-SVs and some of the instances in $\mathcal{L}$ can be screened out *prior to* the training stage. In what follows, we call the screening rule as *non-SV screening rule*. Note, however, that this terminology refers to the rule for screening out not only non-SVs (the instances in $\mathcal{R}$) but also those in $\mathcal{L}$.

3. Screening Non-Support Vectors

In order to derive a non-SV screening rule, let us write the primal SVM problem (2) as

$$\begin{aligned} \min_{w} \ J(w) &:= \frac{1}{2}||w||^2 & (7a)\\ \text{s.t. } H(w) &:= \sum_{i \in \mathbb{N}} [1 - y_i f(x_i)]_+ \le s, & (7b) \end{aligned}$$

where $s > 0$ is an upper bound of the sum of the hinge loss. The formulations in (2) and (7) are equivalent in the sense that there is always a correspondence between the regularization parameter C and the upper bound s (see section 4). We denote the optimal solution of (7) at s as $w^*(s)$ and the optimal classifier at s as $f^*(x|s) := w^*(s)^\top x$.

Properties of the optimal solutions Let us consider two upper bounds $s_a > s_b$. Then, it is easy to see that

$$J(w^*(s_a)) \le J(w^*(s_b)) \quad (8)$$

because the constraint in the latter problem is uniformly tighter than the former. The relation (8) can be rewritten with $\nabla J(w)|_{w=w^*(s_a)}$, the gradient of $J(w)$ at $w = w^*(s_a)$. Since (8) indicates that the vector $w^*(s_b) - w^*(s_a)$ must be an ascent direction of $J(w)$, we have

$$\begin{aligned} &(\nabla J(w)|_{w=w^*(s_a)})^\top (w^*(s_b) - w^*(s_a)) \ge 0\\ \Leftrightarrow \quad & w^*(s_a)^\top (w^*(s_b) - w^*(s_a)) \ge 0. \quad (9) \end{aligned}$$

Next, let $\hat{w}(s_b)$ be an arbitrary primal feasible solution to (7) at $s = s_b$. Then, it is clear that

$$J(w^*(s_b)) \le J(\hat{w}(s_b)) \ \Leftrightarrow \ ||w^*(s_b)||^2 \le ||\hat{w}(s_b)||^2 \quad (10)$$

because $w^*(s_b)$ minimizes $J(w)$ under the constraint $H(w) \le s_b$.

In fact, the relations (9) and (10) can be applied not only to $w^*(s_b)$, but also to the entire range of the optimal solutions for all $s \in [s_b, s_a]$. This simple fact is summarized in the following lemma:

Lemma 1 *Consider two positive scalars $s_a > s_b$, and assume that there exists at least a (primal) feasible solutions of (7) at $s = s_b$. Let $w^*(s_a)$ be the optimal solution of the problem (7) at $s = s_a$, and $\hat{w}(s_b)$ be an arbitrary feasible solution of (7) at $s = s_b$, i.e., $H(\hat{w}(s_b)) \le s_b$. Then,*

$$w^*(s_a)^\top (w^*(s) - w^*(s_a)) \ge 0 \quad (11)$$

and

$$||w^*(s)||^2 \le ||\hat{w}(s_b)||^2 \quad (12)$$

are satisfied for all $s \in [s_b, s_a]$.

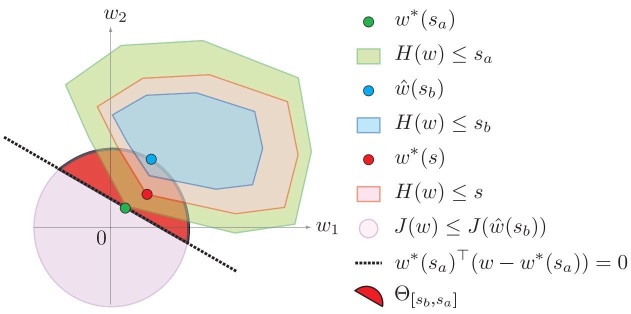

Figure 1. A geometric interpretation of the properties of the solutions of (7) in Lemma 1. For any $s_a > s_b$, when one has the optimal solution $w^*(s_a)$ at s_a, and a primal feasible solution $\hat{w}(s_b)$ at s_b, then the optimal solutions $w^*(s)$ at any $s \in [s_b, s_a]$ is guaranteed to be located in the red dome-shape region $\Theta_{[s_b,s_a]}$. It is because the vector $w^*(s_a) - w^*(s)$ must be an ascent direction of the objective function values $J(w)$, and because $J(w^*(s)) \le J(\hat{w}(s_b))$. The former suggests that the solution $w^*(s)$ must be in the upper right part of the dotted line, while the latter indicates that $w^*(s)$ must be inside the circle.

We omit the proof of this lemma because it is clear from the discussion in (9) and (10). Figure 1 provides a geometric interpretation of Lemma 1.

Basic idea According to the optimality conditions (5) and (6), the ith training instance is categorized into either of $\mathcal{R}$, $\mathcal{E}$ or $\mathcal{L}$ based on the value of $y_i f(x_i)$. The basic idea for constructing our non-SV screening rule is to bound $y_i f(x_i)$ by using the properties discussed in Lemma 1. Namely, we consider a subset of the solution space $\Theta_{[s_b,s_a]} \subset \mathbb{R}^d$ in which the optimal solution $w^*(s)$ for any $s \in [s_b, s_a]$ is guaranteed to exist, and compute the minimum and the maximum values of $y_i f(x_i)$ within the set $\Theta_{[s_b,s_a]}$. From Lemma 1,

$$\Theta_{[s_b,s_a]} := \left\{ w \in \mathbb{R}^d \;\middle|\; \begin{array}{l} w^*(s_a)^\top (w - w^*(s_a)) \ge 0, \\ \|w\|^2 \le \|\hat{w}(s_b)\|^2 \end{array} \right\}.$$

We can ensure that ith training instance would be in $\mathcal{R}$ (i.e, non-SV) if

$$\min_{w \in \Theta_{[s_b,s_a]}} y_i f(x_i) > 1, \tag{13}$$

and, it would be in $\mathcal{L}$ if

$$\max_{w \in \Theta_{[s_b,s_a]}} y_i f(x_i) < 1. \tag{14}$$

for the entire range optimal solutions at $s \in [s_b, s_a]$. It is simply because (13) and (14) suggests $y_i f^*(x_i|s) > 1$ and $y_i f^*(x_i|s) < 1$, respectively. Figure 2 illustrates the idea behind this.

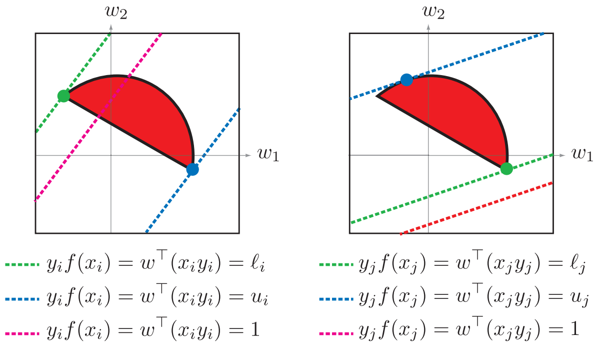
w_2
w_1
w_2
w_1
$y_i f(x_i) = w^\top (x_i y_i) = \ell_i$
$y_i f(x_i) = w^\top (x_i y_i) = u_i$
$y_i f(x_i) = w^\top (x_i y_i) = 1$
$y_j f(x_j) = w^\top (x_j y_j) = \ell_j$
$y_j f(x_j) = w^\top (x_j y_j) = u_j$
$y_j f(x_j) = w^\top (x_j y_j) = 1$

Figure 2. Schematic illustrations of how the lower and the upper bounds of $y_i f(x_i)$ can be used for screening out non-SVs. In the left, the ith instance would not be screened out because the pink line $y_i f(x_i) = w^\top (x_i y_i) = 1$ intersects with the red region $\Theta_{[s_b,s_a]}$, which means that $y_i f(x_i)$ can be either less than 1, equal to 1, or greater than 1 at the optimal solution. On the other hand, the jth instance in the right plot can be safely screened out as a non-SV because the lower bound of $y_j f(x_j)$ is greater than 1.

Computing the bounds A nice thing about (13) and (14) is that the solutions of these minimization and maximization problems can be explicitly computed. We obtain the following screening rule:

Theorem 2 *Consider two scalars $s_a < s_b$, and let $w^*(s_a)$ and $\hat{w}(s_b)$ be the optimal solution at $s = s_a$ and a feasible solution at $s = s_b$, respectively, as Lemma 1. Also, let us write*

$$\gamma_a := J(w^*(s_a)), \;\; \gamma_b := J(\hat{w}(s_b))$$

for notational simplicity. Then, for all $s \in [s_b, s_a]$,

$$y_i f^*(x_i|s_a) > 1 \text{ and } \ell_i > 1 \;\Rightarrow\; i \in \mathcal{R} \;\Leftrightarrow\; \alpha_i = 0, \tag{15}$$

where

$$\ell_i \;:=\; y_i f^*(x_i|s_a) - \sqrt{\frac{\gamma_b - \gamma_a}{\gamma_a}(2\gamma_a\|x_i\|^2 - f^*(x_i|s_a)^2)}. \tag{16}$$

Similarly,

$$u_i < 1 \;\Rightarrow\; i \in \mathcal{L} \;\Leftrightarrow\; \alpha_i = C, \tag{17}$$

where

$$u_i := \begin{cases} \sqrt{2\gamma_b}\|x_i\|, \\ \qquad \text{if } \frac{y_i f^*(x_i|s_a)}{\|x_i\|} \ge \frac{\sqrt{2\gamma_a}}{\sqrt{\gamma_b}}, \\ y_i f^*(x_i|s_a) \\ \quad + \sqrt{\frac{\gamma_b - \gamma_a}{\gamma_a}(2\gamma_a\|x_i\|^2 - f^*(x_i|s_a)^2)}, \\ \qquad \text{otherwise.} \end{cases} \tag{18}$$

Algorithm 1 SafeScreening

```
Input: $D_{\mathbb{N}} = \{(x_i, y_i)\}_{i \in \mathbb{N}}$, $w^*(s_a)$, $\hat{w}(s_b)$;
Output: $\tilde{\mathcal{L}}$, $\tilde{\mathcal{R}}$, $\tilde{\mathcal{Z}}$;
Initialize: $\tilde{\mathcal{L}} \leftarrow \emptyset$, $\tilde{\mathcal{R}} \leftarrow \emptyset$, $\tilde{\mathcal{Z}} \leftarrow \mathbb{N}$ and $i \leftarrow 1$;
while $i \le n$ do
    if $(x_i, y_i)$ satisfies the conditions in (15) then
        $\tilde{\mathcal{R}} \leftarrow \tilde{\mathcal{R}} \cup \{i\}$, $\tilde{\mathcal{Z}} \leftarrow \tilde{\mathcal{Z}} \setminus \{i\}$;
    end if
    if $(x_i, y_i)$ satisfies the condition in (17) then
        $\tilde{\mathcal{L}} \leftarrow \tilde{\mathcal{L}} \cup \{i\}$, $\tilde{\mathcal{Z}} \leftarrow \tilde{\mathcal{Z}} \setminus \{i\}$;
    end if
    $i \leftarrow i + 1$;
end while
```

The proof of Theorem 2 is given in the supplementary, where we just solved the minimization and the maximization problems (13) and (14) by using standard Lagrange multiplier method. The main cost for evaluating the rule is in the computation of $f^*(x_i|s_a) = w^*(s_a)^\top x_i$, which takes $\mathcal{O}(d)$ cost if x_i is dense. However, in pathwise computation scenario that we discuss in section 4, it is likely that this value would have been already computed in earlier steps and stored in cache. In this case, the rule can be evaluated with $\mathcal{O}(1)$ cost.

From (16), we see that ℓ_i is not larger than $y_i f^*(x_i|s_a)$. It suggests that the screening rule (15) should be applied only to the instances with $y_i f^*(x_i|s_a) > 1$. On the other hand, we might get a chance to have $y_i f^*(x_i|s) < 1$ for some i and $s \in [s_b, s_a]$ even if $y_i f^*(x_i|s_a) \ge 1$ at s_a. Thus, the rule (17) is also applied to such an instance.

Algorithm 1 describes the non-SV screening rule. Here, $\{\tilde{\mathcal{L}}, \tilde{\mathcal{R}}, \tilde{\mathcal{Z}}\}$ are the output of the rule. Each of them represents the subset of the training instances guaranteed to be in $\mathcal{L}$ (i.e., $\alpha_i = C$), those guaranteed to be in $\mathcal{R}$ (i.e., $\alpha_i = 0$), and those we are not sure where it belongs to, respectively. When we compute the solution at an $s \in [s_b, s_a]$, we can only work with the subset of the training instances $D_{\tilde{\mathcal{Z}}}$ (note that, the variable $\alpha_i, i \in \tilde{\mathcal{L}}$ can be fixed to C during the optimization process).

Kernelization It is easy to note that the lower bound ℓ_i and the upper bound u_i can be computed only through kernels. The dual of (7) has similar form as (3) (see (20) later), and $f(x)$ is written as in (4) using the Lagrange multipliers $\alpha \in \mathbb{R}^n$. Noting that $||x_i||^2 = K(x_i, x_i)$, all the quantities in (16) and (18) can be evaluated by using the kernel K and the dual variables α. In this case, the computation of $y_i f^*(x_i|s_a) = (Q\alpha^*(s_a))_i$ takes $\mathcal{O}(n)$ cost, where $(v)_i$ for a vector v represents the ith element, and $\alpha^*(s_a)$ is the optimal Lagrange multipliers at s_a. In pathwise computation scenario, this value has been often computed in earlier steps. Then, the rule can be evaluated with $\mathcal{O}(1)$ cost for each instance.

Interpretation The non-SV screening rule can be interpreted in the following way. The lower bound ℓ_i in (16) indicates that

$$y_i f^*(x_i|s_a) - y_i f^*(x_i|s) \le \sqrt{\underbrace{(\gamma_b - \gamma_a)/\gamma_a}_{(\spadesuit)} \cdot \underbrace{(||x_i||^2 ||w^*(s_a)||^2 - (x_i^\top w^*(s_a))^2)}_{(\clubsuit)}},$$

i.e., the difference of $y_i f(x_i)$ values between s_a and s are bounded by the quantity in the r.h.s. Roughly speaking, ($\spadesuit$) is the relative difference of the objective function values between s_a and s_b, while ($\clubsuit$) represents how close the ith instance is to the decision boundary at s_a (note that, the quantity in ($\clubsuit$) is maximized when $\cos(x_i, w^*(s_a)) = 0$ and minimized when $\cos(x_i, w^*(s_a)) = \pm 1$ if the sizes of the two vectors are fixed). It suggests that, if there is no much difference between two objective function values and the ith instance is away from the decision boundary at s_a then $y_i f^*(x_i|s)$ would not be so different from $y_i f^*(x_i|s_a)$. Then, if $y_i f^*(x_i|s_a)$ is also reasonably away from 1, the ith instance is likely to be screened out as a non-SV.

4. Pathwise SVM Computation

For evaluating a non-SV screening rule, we need to know two things: the optimal solution at a smaller C, and a feasible solution at a larger C (note that a small s in (7) corresponds to a large C in (2), and vice-versa). If we do not have these two, they must be computed prior to the rule evaluation. Then, the computational gain brought by the screening rule might be small or even vanish with these additional costs.

However, in a situation that a series of pathwise optimal solutions w.r.t. the regularization parameter C must be computed, we can exploit the advantage of the screening rule. In this section, we describe how we can efficiently compute pathwise solutions by using the non-SV screening rule we have developed in the previous section. Computing pathwise solutions is practically important because model selection (e.g., by cross-validation) is indispensable for selecting a classifier that generalizes well.

Regularization Path These pathwise solutions are often referred to as *regularization paths*. In the SVM, the regularization path was first studied in (Hastie et al., 2004), where parametric programming (Allgower & George, 1993; Gal, 1995; Best, 1996; Takeuchi et al., 2009; Karasuyama et al., 2012) is introduced for computing the *exact* path of the SVM solutions. This algorithm is quite efficient for small or medium-size problems. For larger problems, *approximate* path obtained by computing a fine grid sequence of pathwise solutions can be used as an alternative (Friedman et al., 2007; 2010; Yuan et al., 2010; 2012).

In what follows, we present an algorithm for computing a sequence of the optimal solutions of (3) at T different Cs: $0 < C^{(1)} < \ldots < C^{(T)}$. By the reason that will be explained later, we first solve two optimization problems at the smallest $C^{(1)}$ (the largest $s^{(1)}$) and at the largest $C^{(T)}$ (the smallest $s^{(T)}$). Although these costs might be rather expensive, once we obtain these two optimal solutions, the rest of the pathwise solutions at $C^{(2)}, \ldots, C^{(T-1)}$ can be fairly cheaply computed by exploiting the advantage of the non-SV screening rule. In the algorithm, many non-SV screening rules are made, and each rule is used for computing a sub-sequence of the pathwise solutions. Let $M \geq 2$ be the length of each sub-sequence. Then, roughly speaking, each non-SV screening rule covers the sub-sequence of the solutions in the following way:

$$\begin{array}{lcllll} \texttt{1st rule} & : & \{C^{(2)} & \ldots & C^{(M+1)}\}, \\ \texttt{2nd rule} & : & \{C^{(M+2)} & \ldots & C^{(2M+1)}\}, \\ \vdots & & & \vdots & \\ \frac{T}{M}\texttt{th rule} & : & \{C^{(T-M)} & \ldots & C^{(T-1)}\}. \end{array}$$

The details of the algorithm are described in Algorithms 2 at the end of this section.

Our algorithm is a *meta*-algorithm in the sense that any SVM solvers that can solve the dual optimization problem (3) can be used as a component.

Relation between C-form and s-form Before presenting how to construct the rule for each sub-sequence, we need to clarify the relation between the two formulations (2) and (7) because the algorithm uses both formulations interchangeably. We call the former as C-form, and the latter as s-form.

First, consider a situation that we have the optimal solution of C-form at a certain C. Then, the corresponding s in the s-form is simply computed as

$$s = \sum_{i \in \mathbb{N}} [1 - y_i f^*(x_i|C)]_+, \qquad (19)$$

where $f^*(\cdot|C)$ is the optimal classifier at C. On the other hand, when we have an s-form problem at a certain s, the dual of (7) is written as

$$\max_{\alpha, C} \; -\frac{1}{2}\alpha^\top Q \alpha + \mathbf{1}^\top \alpha - Cs \text{ s.t. } \alpha \in [0, C]^n, \qquad (20)$$

and the corresponding C in C-form is obtained by solving this dual problem.

When we interchangeably use these two formulations in the algorithm, we use the following simple fact:

Lemma 3 *For $C_a < C_b$ in C-form, the corresponding s_a and s_b in s-form satisfy $s_a \geq s_b$. Conversely, for $s_a > s_b$ in s-form, the corresponding C_a and C_b satisfy $C_a \leq C_b$.*

The proof is in the supplementary.

Using the correspondence between C and s, we denote the sequence of s corresponding to $C^{(1)} < C^{(2)} < \ldots < C^{(T)}$ as $s^{(1)} \geq s^{(2)} \geq \ldots \geq s^{(T)}$, respectively. In addition, with a slight abuse of notation, we write $w^*(C)$ and $w^*(s)$ to represent the optimal solution at C in C-form, and at s in s-form, respectively.

4.1. Constructing a non-SV screening rule

Consider a situation that the pathwise solutions at $C \leq C^{(t)}$ have already been computed. Our next task is to construct a non-SV screening rule that is used for computing the solutions at $C^{(t+1)}, \ldots, C^{(t+M)}$ (equivalently, at $s^{(t+1)}, \ldots, s^{(t+M)}$). To this end, we need to know the optimal solution at an $s_a \geq s^{(t+1)}$ and a feasible solution at an $s_b \leq s^{(t+M)}$. For the former, we can just set $s_a = s^{(t)}$ and use the optimal solution at $s^{(t)}$ because it has already been computed.

For the later, we can set $s_b = s^{(T)}$ since $s^{(T)} \leq s^{(t+M)}$ (see Lemma 4). However, if $s^{(T)}$ is much smaller than $s^{(t+M)}$, then the rule would not be able to screen out many non-SVs. In order to make a better rule, we need to find a tighter $s_b \geq s^{(t+M)}$. Unfortunately, this is a non-trivial challenging task from the following two aspects. First, we do not know $s^{(t+M)}$ since it is available only after finding the optimal solution at $C^{(t+M)}$. Second, even if we could have an $s_b \leq s^{(t+M)}$, finding a feasible solution at the s_b is often as difficult as finding the optimal solution. We address the first issue in a trial-and-error manner, and its detail is discussed in section 4.2 and supplementary. In the sequel, we describe how to handle the second issue.

Computing a feasible solution To address the difficulty in finding a feasible solution at s_b, we first compute the optimal solution $w^*(s^{(T)})$ at the largest $C = C^{(T)}$ (corresponding to the smallest $s = s^{(T)}$) using any algorithms that suits to this task. Although this cost would be rather expensive, once we have $w^*(s^{(T)})$, we can utilize it for finding all the other feasible solutions in later steps in a trivial amount of computation.

Consider a case that we make a rule defined on $[s_b, s_a]$. Given $w^*(s_a)$ and $w^*(s^{(T)})$, we can easily find a feasible solution at s_b by solving the following one-dimensional convex optimization problem:

$$\min_{v\in[0,1]} \frac{1}{2}||w(v)||^2 \quad \text{s.t.} \sum_{i\in\mathbb{N}}[1-y_if(x_i)]_+ \le s_b \quad w(v) = vw^*(s_a) + (1-v)w^*(s^{(T)}). \tag{21}$$

This one-dimensional convex problem can be easily solved by a line-search algorithm such as bisection method. Note that we can always find a feasible $v \in [0,1]$ for any $s_b \in [s^{(T)}, s_a]$ because the constraint set $H(w) \le s_b$ is convex. This line-search can be kernelized since both the objective function values and the constraints can be evaluated only through kernel computations. Figure 3 depicts how to find a primal feasible solution.

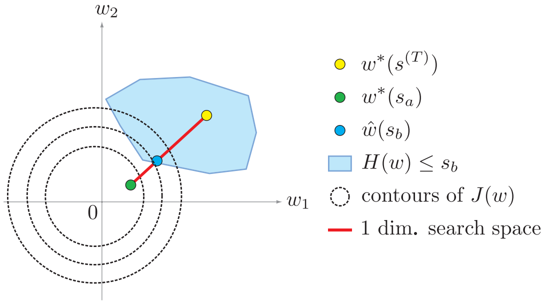

Figure 3. Geometric interpretation of the one-dimensional line-search problem in (21). For $s_a > s_b > s^{(T)}$, when one has the optimal solution $w^*(s_a)$ at s_a, and the optimal solution $w^*(s^{(T)})$ at the smallest $s = s^{(T)}$ (corresponding to the largest $C = C^{(T)}$), then a primal feasible solution $\hat{w}(s_b)$ at s_b can be easily found by solving this one-dimensional convex problem with any line-search solver.

4.2. Pathwise computation algorithm

Algorithm 2 describes how we compute pathwise SVM solutions with the help of non-SV screening rules. In this algorithm, we first compute two SVM solutions $w^*(C^{(1)})$ and $w^*(C^{(T)})$ using the entire training set. Once we obtain these two optimal solutions, the rest of the pathwise solutions at $C^{(2)}, \ldots, C^{(T-1)}$ can be computed efficiently because non-SV screening rules can reduce the training set size. The algorithm is a bit complicated because we need to use C-form and s-form interchangeably. Due to space limitation, FindSb and ComputeSubPath functions in Algorithm 2 and some additional information about the algorithm is provided in the supplementary.

Algorithm 2 SafeScreeningPath (see section 4.2)

```
Input: $D_{\mathbb{N}} := \{(x_i, y_i)\}_{i\in\mathbb{N}}$, $\{C^{(t)}\}_{t=1}^T$, $M$;
Output: $\{w^*(C^{(t)})\}_{t=1}^T$;
Initialization:
  $w^*(C^{(1)}) \leftarrow$ SVMSolver$(D_{\mathbb{N}}, C^{(1)})$;
  Compute $s^{(1)}$ by (19);
  $w^*(C^{(T)}) \leftarrow$ SVMSolver$(D_{\mathbb{N}}, C^{(T)})$;
  Compute $s^{(T)}$ by (19);
  $t \leftarrow 1$;
while $t < T - M$ do
    $s_a \leftarrow s^{(t)}$;
    $\{s_b, w^*(s_b)\} \leftarrow$ FindSb$(D_{\mathbb{N}}, C^{(t+M)}, w^*(s_a))$;
    $\{\tilde{\mathcal{L}}, \tilde{\mathcal{R}}, \tilde{\mathcal{Z}}\} \leftarrow$ Screen$(D_{\mathbb{N}}, w^*(s_a), w^*(s_b))$;
    $\{w^*(C^{(u)})\}_{u=t+1}^{t+M-1}$
        $\leftarrow$ ComputeSubPath$(D_{\tilde{\mathcal{Z}}}, \{C^{(u)}\}_{u=t+1}^{t+M-1})$
    $t \leftarrow t + M$;
end while
$\{\tilde{\mathcal{L}}, \tilde{\mathcal{R}}, \tilde{\mathcal{Z}}\} \leftarrow$ Screen$(D_{\mathbb{N}}, w^*(s^{(t)}), w^*(s^{(T)}))$;
$\{w^*(C^{(u)})\}_{u=t+1}^{T-1}$
    $\leftarrow$ ComputeSubPath$(D_{\tilde{\mathcal{Z}}}, \{C^{(u)}\}_{u=t+1}^{T-1})$
```

5. Numerical Experiments

In this section, we report experimental results.

Toy example First, we applied our non-SV screening to a simple 2-dimensional toy data set plotted in Figure 4. In each class, 300 data points were generated from $N(\{1.5, 1.5\}^\top, 0.75^2 I)$ and $N(\{-1.5, -1.5\}^\top, 0.75^2 I)$, respectively, where I is an identity matrix. We constructed a non-SV screening rule for the range of $C \in [1, 10]$. Among 600 data points, the rule screened out 530 points. Among them, only one instance was screened out as a member of $\tilde{\mathcal{L}}$, and the rest were screened out as $\tilde{\mathcal{R}}$ (non-SVs). It means that, if we want to train an SVM for any $C \in [1, 10]$, we have only to solve the optimization problem with the remaining 70 data points. Figure 4 shows which data points are screened out. Interestingly, some data points close to the boundaries were also screened out, while some points distant away from the boundaries are kept remained.

Pathwise computation cost We investigate the performance of our non-SV screening in the pathwise computation scenario discussed in section 4. Here, we only report the experiments with linear kernel. As the plug-in SVM solver, we used a state-of-the-art linear SVM solver LIBLINEAR (Hsieh et al., 2008).

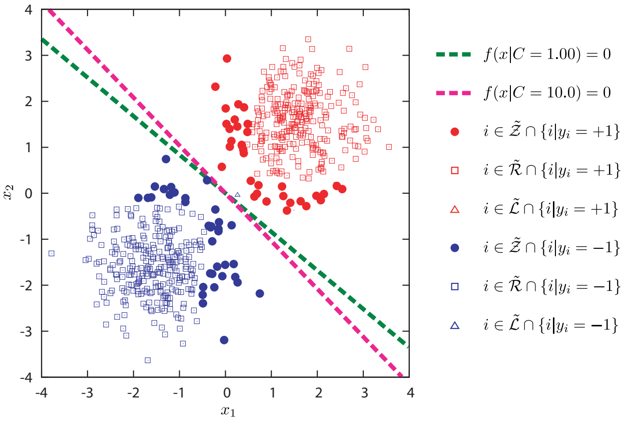

Figure 4. The non-SV screening rule defined in $C \in [1, 10]$ for linear classification of 2D toy data set. Green and pink dotted lines are the optimal decision function at $C = 1$ and $C = 10$, respectively. The points represented by open squares and triangles are those screened out by the rule. The points represented by filled circle are not screened out and kept remained. Any SVM for $C \in [1, 10]$ can be trained only with those data points.

We applied the pathwise computation algorithm to four data sets as described in Table 1. For each data set, our task is to compute the SVMs for 100 different $C \in [10^{-2}, 10^2]$ evenly distributed in logarithmic scale. As explained in the previous section, we set $M = 5$, where M is the number of solutions each screening rule covers. It means that, we constructed 20 screening rules for each data set.

We compared our non-SV screening with "naive" and "heuristic screening" approaches. In the naive approach, we just computed each pathwise SVM solutions one by one. In the heuristic screening approach, we predicted the active sets $\mathcal{R}$ and $\mathcal{L}$ as those at the previously computed SVM solution with smaller C, and their corresponding α_i were fixed as 0 and C, respectively. If some of those predictions were found to be wrong (by cheking the optimality condition), we trained the SVM again after adding those violating instnaces to the training set, and this process was repeated until all the instances satisfy the optimality. Table 1 summarizes the computation time for obtaining the pathwise solutions on these four data sets.

Table 1. Computation Time in Pathwise Computation on four data sets. In "Naive approach", all the pathwise solutions were naively computed one by one. In "Heuristic screening", the active sets $\mathcal{R}$ and $\mathcal{L}$ were assumed to be same as the previous SVM with smaller C, and "Wasted" indicates the computation time taken for computing non-optimal solutions (by wrong prediction). In our Non-SV screening, "Line Search" and "Rule" indicates the computation time spent for finding feasible solutions, and for constructing the rules, respectively. " SVM Solver" is the total time spent to compute pathwise solutions in SVM solver.

DIGIT1 Data ($n = 1500, d = 241$)		
Naive Approach	Total	53.405 sec
Heuristic Screening	Wasted	14.400 sec
	Total	22.334 sec
Non-SV Screening	Line Search	0.0020 sec
	Rule	0.0010 sec
	SVM Solver	18.591 sec
	Total	**18.599** sec

PCMAC Data ($n = 1946, d = 7511$)		
Naive Approach	Total	1.9817 sec
Heuristic Screening	Wasted	1.3800 sec
	Total	3.0016 sec
Non-SV Screening	Line Search	0.0060 sec
	Rule	0.0040 sec
	SVM Solver	1.2588 sec
	Total	**1.2748** sec

Magic Gamma Telescope Data ($n = 19020, d = 10$)		
Naive Approach	Total	7255.5 sec
Heuristic Screening	Wasted	6407.5 sec
	Total	12139 sec
Non-SV Screening	Line Search	0.0210 sec
	Rule	0.0240 sec
	SVM Solver	4916.8 sec
	Total	**4916.8** sec

IJCNN1 Data ($n = 49990, d = 22$)		
Naive Approach	Total	27876 sec
Heuristic Screening	Wasted	830.33 sec
	Total	**8020.5** sec
Non-SV Screening	Line Search	0.0500 sec
	Rule	0.0360 sec
	SVM Solver	16495 sec
	Total	16495 sec

Figures 5 $\sim$ 8 show the screening results of the four data sets in Table 1. In each figure, the top plot indicates the rate of screened out instances at each C. Since we made 20 rules in this experiment, there are 20 segments in these plots. The bottom plot indicates the computation time spent for computing an SVM solution at each C. The blue lines and dots indicate the time when we solve an SVM with the entire training set in the naive approach. The green lines and dots indicate the total computation time of the heuristic screening. The red lines and points indicate SVM training time for reduced training set after applying our non-SV screening rule. In these, red circles are plotted at every five Cs (since $M = 5$). These points correspond to the cost at line 8 of Algorithm 3 (see supplementary). The costs at these points are usually

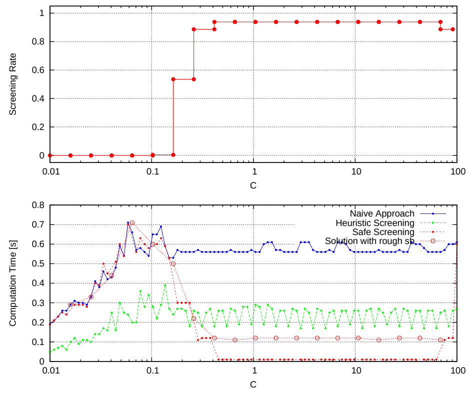

Figure 5. **DIGIT1** Screening Results

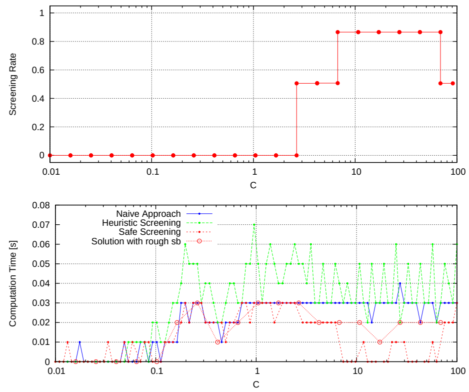

Figure 6. **PCMAC** Screening Results

higher than others because we used a rough estimate of s_b when making the rule for this solution.

From Table 1 and Figures 5 $\sim$ 8, we see that our non-SV screening rule could substantially reduce the total pathwise computation cost. Compared with the naive approach, our non-SV screening rule reduced the total cost by 65%, 36%, 32%, and 41% in each data set. Note that the cost for finding feasible solutions "Line Search" and that for constructing rules "Rule" are negligible. In three of the four data sets, the costs of the heuristic screening rule were larger than those of our non-SV screening rule. Note that, in PCMAC data set, it was even worse than the naive approach. This is because the heuristic screening rule is NOT safe, meaning that, when it makes false positive screenings, one needs to repeat the training after correcting those mistakes ("Wasted" in Table 1 indicates the training cost with false positive screenings). Overall, the experiments illustrate that our non-SV screening rule is useful in pathwise computation scenario.

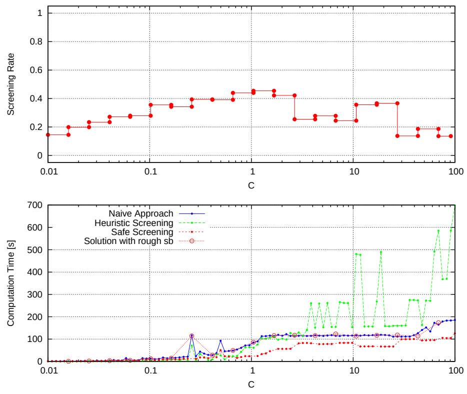

Figure 7. **MAGIC** Screening Results

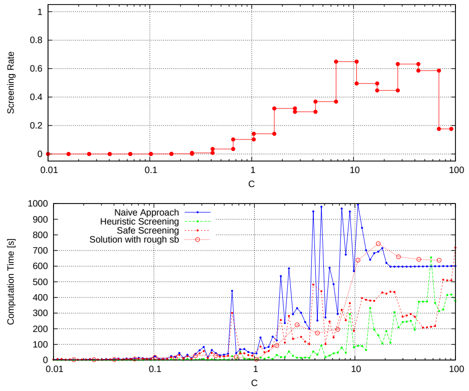

Figure 8. **IJCNN1** Screening Results

6. Conclusions and Discussions

In this paper, we introduced the idea of *safe screening rule* for the task of identifying non-SVs prior to SVM training. We show that the advantage of non-SV screening rule can be fully exploited in pathwise computation scenario. Some experiments illustrate advantages of our approach.

In our experiments, our non-SV screening rule could not screen-out many instances when we used Gaussian kernel. It would be important to clarify what types of kernel could be used with our non-SV screening rule. In addition, we would also need to consider how we can extend the current non-SV screening rules when the decision function has constant bias term.

Acknowledgments

We thank Kohei Hatano for his insightful comments and suggestions. IT was supported in part by MEXT KAKENHI 23700165 and the Hori sciences and arts foundation.

References

Allgower, E. L. and George, K. Continuation and path following. *Acta Numerica*, 2:1–63, 1993.

Best, M. J. An algorithm for the solution of the parametric quadratic programming problem. *Applied Mathemetics and Parallel Computing*, pp. 57–76, 1996.

Boyd, S. and Vandenberghe, L. *Convex Optimization*. Cambridge University Press, 2004.

Cauwenberghs, G. and Poggio, T. Incremental and decremental support vector machine learning. *Advances in Neural Information Processing Systems*, 13, 2001.

Chang, C. C. and Lin, C. J. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

Dai, L. and Pelckmans, K. An ellipsoid based two-stage sreening test for bpdn. In *Proceedings of the 20th European Signal Processing Conference*, 2012.

Fan, J. and Lv, J. Sure independence screening for ultrahigh dimensional feature space. *Journal of The Royal Statistical Society B*, 70:849–911, 2008.

Fan, R. R., Chang, K. W., Hsieh, C. J., Wang, X. R., and Lin, C. J. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

Friedman, J., Hastie, T., Ho"fling, H., and Tibshirani, R. Pathwise coordinate optimization. 1(2):302–332, 2007.

Friedman, J., Hastie, T., and Tibshirani, R. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33 (1):1–22, 2010.

Gal, T. *Postoptimal Analysis, Parametric Programming, and Related Topics*. Walter de Gruyter, 1995.

Ghaoui, L. E., Viallon, V., and Rabbani, T. Safe feature elimination in sparse supervised learning. *eprint arXiv:1009.3515*, 2010.

Hastie, T., Rosset, S., Tibshirani, R., and Zhu, J. The entire regularization path for the support vector machine. *Journal of Machine Learning Research*, 5: 1391–415, 2004.

Hsieh, C. J., Chang, K. W., and Lin, C. J. A dual coordinate descent method for large-scale linear svm. *Proceedings of the 25th International Conference on Machine Learning*, 2008.

Joachims, T. Making large-scale svm learning practical. In Scholkopf, B., Burges, C. J. C., and Smola, A. J. (eds.), *Advances in Kernel Methods - Support Vector Learning*, pp. 169–184. MIT Press, 1999.

Karasuyama, M., Harada, N., Sugiyama, M., and Takeuchi, I. Multi-parametric solution-path algorithm for instance-weighted support vector machines. *Machine Learning*, 88(3):297–330, 2012.

Platt, J. Fast training of support vector machines using sequential minimal optimization. In Scholkopf, B., Burges, C. J. C., and Smola, A. J. (eds.), *Advances in Kernel Methods - Support Vector Learning*, pp. 185–208. MIT Press, 1999.

Takeuchi, I., Nomura, K., and Kanamori, T. Nonparametric conditional density estimation using piecewise-linear solution path of kernel quantile regression. *Neural Computation*, 21(2):539–559, 2009.

Tibshirani, R., Bien, J., Friedman, J., Hastie, T., Simon, N., Taylor, J., and Tibshirani, R. J. Strong rules for discarding predictors in lasso-type problems. *Journal of the Royal Statistical Society, Series B*, 74:245–266, 2011.

Vats, D. Parameter-free high-dimensional screening using multiple grouping of variables. *eprint arXiv:1208.2043*, 2012.

Wang, J., Lin, B., Gong, P., Wonka, P., and Ye, J. Lasso screening rules via dual polytope projection. *eprint arXiv:1211.3966*, 2012.

Xiang, Z. J. and Ramadge, P. J. Fast lasso screening test based on correlatins. In *Proceedings of the 37th IEEE International Conference on Acoustics, Speech and Signal Processing*, 2012.

Xiang, Z. J., Xu, H., and Ramadge, P. J. Learning sparse representations of high dimensional data on large scale dictionaries. In *Advances in Neural Information Processing Systems 24*, pp. 900–908. 2012.

Yuan, G. X., Chang, K. W., Hsieh, C. J., and Lin, C. J. A comparison of optimization methods and software for large-scale l1-regularized linear classification. *Journal of Machine Learning Research*, 11: 3183–3234, 2010.

Yuan, G. X., Ho, C. H., and Lin, C. J. An improved glmnet for l1-regularized logistic regression. *Journal of Machine Learning Research*, 13:1999–2030, 2012.