Support Vector Machine Classifier via $L_{0/1}$ Soft-Margin Loss

Huajun Wang, Yuanhai Shao, Shenglong Zhou, Ce Zhang and Naihua Xiu*

Abstract—Support vector machine (SVM) has attracted great attentions for the last two decades due to its extensive applications, and thus numerous optimization models have been proposed. To distinguish all of them, in this paper, we introduce a new model equipped with an $L_{0/1}$ soft-margin loss (dubbed as $L_{0/1}$-SVM) which well captures the nature of the binary classification. Many of the existing convex/non-convex soft-margin losses can be viewed as a surrogate of the $L_{0/1}$ soft-margin loss. Despite the discrete nature of $L_{0/1}$, we manage to establish the existence of global minimizer of the new model as well as revealing the relationship among its minimizers and KKT/P-stationary points. These theoretical properties allow us to take advantage of the alternating direction method of multipliers. In addition, the $L_{0/1}$-support vector operator is introduced as a filter to prevent outliers from being support vectors during the training process. Hence, the method is expected to be relatively robust. Finally, numerical experiments demonstrate that our proposed method generates better performance in terms of much shorter computational time with much fewer number of support vectors when against with some other leading methods in areas of SVM.

Index Terms—SVM, $L_{0/1}$ soft-margin loss, $L_{0/1}$-proximal operator, minimizers and KKT/P-stationary points, $L_{0/1}$-ADMM.

1 INTRODUCTION

SUPPORT vector machine (SVM) was first introduced by Vapnik and Cortes [1] and then has been widely applied into machine learning, statistic, pattern recognition and so forth. The basic idea is to find a hyperplane in the input space that separates the training data set. In the paper, we consider a binary classification problem that can be described as follows. Suppose we are given a training set $\{(x_i, y_i)\}_{i=1}^m$, where $x_i \in \mathbb{R}^n$ are the input vectors and $y_i \in \{-1, 1\}$ are the output labels. The purpose of SVM is to train a hyperplane $\langle w, x \rangle + b = w_1x_1 + \cdots + w_nx_n + b = 0$ with $w \in \mathbb{R}^n$ and $b \in \mathbb{R}$ by given training set. For any new input vector $x'$, we can predict the corresponding label $y'$ as $y' = 1$ for $\langle w, x' \rangle + b > 0$ and $y' = -1$ otherwise. In order to find optimal hyperplane, there are two possible cases: linearly separable and inseparable training data. If the training data is able to be linearly separated in the input space, then the unique optimal hyperplane can be obtained by solving a convex quadratic programming (QP) problem:

$$\min_{w \in \mathbb{R}^n, b \in \mathbb{R}} \frac{1}{2} ||w||^2 \\
\text{s.t.} \quad y_i(\langle w, x_i \rangle + b) \geq 1, i \in \{1, 2, \cdots, m\},$$

(1)

where $N_m := \{1, 2, \cdots, m\}$. Here, the $y_i(\langle w, x_i \rangle + b)$ provides the distance between the ith sample and the hyperplane. The above model is termed as hard-margin SVM because it requires correct classifications of all samples. When it comes to the training data that are linearly inseparable in the input space, the popular approach is to allow violations in the satisfaction of the constraints in problem (1) and penalize such violations in the objective function, namely,

$$\min_{w \in \mathbb{R}^n, b \in \mathbb{R}} \frac{1}{2} ||w||^2 + C \sum_{i=1}^m \ell(1 - y_i(\langle w, x_i \rangle + b)),$$

(2)

where $C > 0$ is a penalty parameter and $\ell$ is one of some loss functions that aim at penalizing some incorrectly classified samples and leaving the other ones. Therefore, the above model allows misclassified samples, and thus is known as soft-margin SVM. Clearly, different soft-margin loss functions yield different soft-margin SVM models. Generally speaking, soft-margin loss functions can be summarized as two categories based on the convexity of $\ell$.

1.1 Convex Soft-Margin Losses

Since there are large numbers of convex soft-margin loss function that have been proposed to deal with the soft-margin SVM problems, we only review some popular ones.

- **Hinge loss function**: $\ell_{\text{hinge}}(t) = \max\{0, t\}$ for all $t \in \mathbb{R}$. It is non-differentiable at $t = 0$ and unbounded. SVM with hinge loss ($\ell_{\text{hinge}}$-SVM) was first proposed by Vapnik and Cortes [1], aiming at only penalizing the samples with $t \geq 0$.

- **Pinball loss function**: $\ell_{\text{pinball}}(t) = \max\{t, -\tau t\}$, with $0 \leq \tau \leq 1$, which is still non-differentiable at $t = 0$ and unbounded. SVM with this loss function ($\ell_{\text{pinball}}$-SVM) was proposed in [2], [3] to pay penalty for all samples. There is a quadratic programming solver embedded in Matlab to solve the SVM with pinball loss function [3].
1.2 Non-Convex Soft-Margin Losses

Since those functions are convex, their corresponding SVM models are not difficult to be dealt with. However, the convexity often induces the unboundedness, which removes robustness of those loss functions to outliers from the training data. In order to overcome such drawback, authors in [11], [12] set an upper bound and enforce the loss to stop increasing after a certain extent. Doing so, the original convex loss functions become non-convex.

1.3 \( \ell_{0/1} \) Soft-Margin Loss

Taking above principles into consideration, we now introduce the 0-1 (\( \ell_{0/1} \)) soft-margin loss defined as

\[
\ell_{0/1}(t) = \begin{cases} 
1, & t > 0, \\
0, & t \leq 0.
\end{cases}
\]

The \( \ell_{0/1} \) soft-margin loss function is the most nature loss function for binary classification [20], [21]. Its properties are summarized as below.

(i) It is discontinuous at \( t = 0 \), which captures the discrete nature of the binary classification (correctness or incorrectness) [22].

(ii) It is lower semi-continuous and nonconvex by the definition in [23]. Since it is either 0 or 1, sparsity and robustness will be guaranteed. In fact, it does not count the number of samples with \( t < 0 \), which leads to sparsity, while returns 1 otherwise, which ensures robustness to outliers.

(iii) It is differentiable everywhere but at \( t = 0 \). However, it has subdifferential

\[
\partial \ell_{0/1}(0) = \mathbb{R}_+ := \{ r \in \mathbb{R} : r \geq 0 \}
\]

and zero gradients elsewhere, see Lemma 2.1, which makes the computation tractable.

1.4 \( L_{0/1} \)-SVM

For the sake of easing the reading, we present some notations here. Let \( ||x|| \) and \( ||x||_0 \) be the Euclidean norm and the zero norm of \( x \) that counts the number of non-zero elements of \( x \). Denote \( A := \text{Diag}(y)X^\top \) with \( X = [x_1, x_2, \ldots, x_m] \in \mathbb{R}^{n \times m} \) and \( y = (y_1, y_2, \ldots, y_m) \in \mathbb{R}^m \), where \( \text{Diag}(y) \) is a diagonal matrix with diagonal elements being elements in \( y \). For a positive integer \( m \) and a vector \( u \in \mathbb{R}^m \), denote

\[
N_m := \{1, 2, \ldots, m\},
\]

\[
1 := (1, \ldots, 1)^\top \in \mathbb{R}^m,
\]

\[
\mathbb{R}_+^m := \{ u \in \mathbb{R}^m : u_i \geq 0, \; i \in N_m \},
\]

\[
|u| := (|u_1|, \ldots, |u_m|)^\top,
\]

\[
u := (\max\{u_1, 0\}, \ldots, \max\{u_m, 0\})^\top.
\]

These notations indicate

\[
L_{0/1}(u) := \|u\|_0 = \sum_{i=1}^m \ell_{0/1}(u_i),
\]

which returns the number of all positive elements in \( u \). We call (3) the \( L_{0/1} \) soft-margin loss. Now, replacing \( \ell \) by \( \ell_{0/1} \) in (2) and using above notations allow us to rewrite model (2) in a matrix form,

\[
\min_{w \in \mathbb{R}^n, b \in \mathbb{R}} f(w; b) := \frac{1}{2} \|w\|^2 + C\|(1 - (A w + b y))_+\|_0.
\]

We call this model \( L_{0/1} \)-SVM. The objective function \( f \) is lower semicontinuous, non-differentiable and non-convex.
It is difficult to be solved directly by most existing optimization algorithms. Despite that the discrete nature of zero norm makes above model NP-hard to be solved, the $L_{0/1}$-SVM model is an ideal SVM model because it guarantees as few misclassified as possible for binary classification. Therefore, we carry out this paper along with this model.

### 1.5 Contributions

In this paper, we start to study the theoretical properties of the $L_{0/1}$-SVM model and then design a new efficient and robust algorithm to solve the model. The main contributions of the paper can be summarized as follows.

(i) We prove the existence of a global minimizer of $L_{0/1}$-SVM, which has not been thoroughly studied in prior works. Based on the explicit expressions of subdifferential and proximal operator of the $L_{0/1}$ loss (3), we introduce two types of optimality conditions of the problem: KKT and P-stationary points. We then unravel the relationships among a global/local minimizer and the above two points. This result is essential to our algorithmic design later on.

(ii) We adopt the famous alternating direction method of multipliers (ADMM) to solve the $L_{0/1}$-SVM problem, and thus the method is dubbed as $L_{0/1}$-ADMM. We show that if the sequence generated by the proposed method converges, then it must converge to a P-stationary points. To the best of our knowledge, it is the first time that a method being created aims at solving (4) directly rather than its surrogate model (2). The novelty of the method is using the $L_{0/1}$-support vector operator as a filter to prevent the outliers from being support vectors during training process.

(iii) We compare $L_{0/1}$-ADMM with other four existing leading methods on solving SVM problems with synthetic and real data sets. Extensive numerical experiments demonstrate that our proposed method achieves better performance in terms of providing higher prediction accuracy, using a small number of support vectors and consuming shorter computational time.

This paper is organized as follows. In Section 2, we will give the explicit expressions of three subdifferentials of $L_{0/1}$ soft-margin loss and derive its proximal operator. Section 3 presents the main theoretical contributions. We will show the existence of a global minimizer to problem (4) as well as investigating the relationships among a global/local minimizer and the KKT/P-stationary points of $L_{0/1}$-SVM problem. In Section 4, we will introduce the $L_{0/1}$-support vector operator and design the algorithm based on the optimality conditions established in previous section. Numerical experiments including comparison with other solvers and concluding remarks are given in the last two sections.

## 2 Subdifferential and Proximal Operator

To well analyze the properties of the $L_{0/1}$ soft-margin loss, we need introduce the necessary background of the subdifferential and the proximal operator of the $\|u_+\|_0$.

### 2.1 $L_{0/1}$ Subdifferential

From [24, Definition 8.3], for a proper and lower semicontinuous function $f : \mathbb{R}^m \to \mathbb{R}$, the regular, limiting and horizon subdifferentials are defined respectively as

$\hat{\partial}f(u) = \left\{ v \in \mathbb{R}^m : \liminf_{z \to u} f(z) - f(u) - \langle v, z-u \rangle \geq 0 \right\},$

$\partial f(u) = \limsup_{z \to u} \hat{\partial}f(z) = \left\{ v \in \mathbb{R}^m : \exists z_j \to u, v_j \in \hat{\partial}f(z_j) \right\},$

$\partial^\infty f(u) = \limsup_{\sigma \to 0^+} \sigma \partial f(u) = \left\{ v \in \mathbb{R}^m : \exists z_j \to u, v_j \in \partial f(z_j) \right\},$

where $\sigma \downarrow 0$ means $\sigma > 0$ and $\sigma \to 0$, and $z_j \to u$ means both $z \to u$ and $f(z) \to f(u)$. If the function $f$ is convex, then the limiting subdifferential is also known to the subgradient.

**Lemma 2.1.** The regular, limiting and horizon subdifferentials of $\|u_+\|_0$ at $u$ enjoy following property,

$\Omega(u) := \hat{\partial}\|u_+\|_0 = \partial\|u_+\|_0 = \partial^\infty\|u_+\|_0 = \left\{ v \in \mathbb{R}^m : v_i \begin{cases} \geq 0, & u_i = 0, \\ = 0, & u_i \neq 0, \end{cases} \right\}.$

We use a simple example to illustrate the three subdifferentials of $\|u_+\|_0$. Consider one dimensional case $m = 1$. As shown in Figure 1, the red lines denote some elements in $\partial\|0_+\|_0 = \partial\ell_{0/1}(0)$. In fact, all right slashes crossing the origin comprise of the subdifferential $\partial\|0_+\|_0$.

![Fig. 1: The $\ell_{0/1}$ soft-margin loss function. The blue line (including the blue original) is the function value and the red lines are two of subdifferentials in $\partial\|0_+\|_0$.](image)

**Fig. 1: The $\ell_{0/1}$ soft-margin loss function. The blue line (including the blue original) is the function value and the red lines are two of subdifferentials in $\partial\|0_+\|_0$.**

Our next result is about $L_{0/1}$ Proximal operator, which will be very useful in designing the algorithm in Section 4.

### 2.2 $L_{0/1}$ Proximal Operator

By [25, Definition 12.23], the proximal operator of $f : \mathbb{R} \to \mathbb{R}$, associated with a parameter $\alpha > 0$, at point $s \in \mathbb{R}$, is defined by

$\text{Prox}_{\alpha f}(s) = \arg\min_{u \in \mathbb{R}} \alpha f(u) + \frac{1}{2}(u-s)^2.$

The following lemma states that the proximal operator admits a closed form solution when $f = \ell_{0/1}$.
Lemma 2.2 (One-dimensional case). For an \( \alpha > 0 \), the proximal operator of \( \ell_{0/1}(\cdot) \) at \( s \) is given by

\[
\operatorname{Prox}_{\alpha \ell_{0/1}}(s) := \begin{cases} 
0, & 0 \leq s < \sqrt{2\alpha}, \\
0 \text{ or } s, & s = \sqrt{2\alpha}, \\
1 / \alpha \cdot s, & s > \sqrt{2\alpha} \text{ or } s < 0.
\end{cases}
\] (7)

It is worth mentioning that the proximal operator may not be unique if \( s = \sqrt{2\alpha} \) in (7). However, to guarantee the uniqueness, hereafter, we always choose the proximal operator to be zero if it is not unique. Because of this, the proximal operator of \( \ell_{0/1} \) is rewritten as

\[
\operatorname{Prox}_{\alpha \ell_{0/1}}(s) := \begin{cases} 
0, & 0 \leq s \leq \sqrt{2\alpha}, \\
1 / \alpha \cdot s, & \text{otherwise}.
\end{cases}
\] (8)

The proximal operator of \( \ell_{0/1} \) is shown in Figure 2, where the red line denotes the proximal operator.

![Diagram of Prox_{alpha}](image)

Fig. 2: Demonstration of \( \operatorname{Prox}_{\alpha \ell_{0/1}}(s) \).

Based on the one dimensional case, we could derive the proximal operator of \( L_{0/1}(\cdot) = \| \cdot \|_0 \). The proof is similar to that of Lemma 2.2 and thus is omitted.

Lemma 2.3 (Multi-dimensional case). For an \( \alpha > 0 \), the proximal operator of \( L_{0/1} \) at \( s \in \mathbb{R}^m \) is given by

\[
\operatorname{Prox}_{\alpha L_{0/1}}(s) := \begin{bmatrix} 
\operatorname{Prox}_{\alpha \ell_{0/1}}(s_1) \\
\vdots \\
\operatorname{Prox}_{\alpha \ell_{0/1}}(s_m)
\end{bmatrix}.
\] (9)

To proceed further, we consider the following problem

\[
\min_{\mathbf{u} \in \mathbb{R}^m} f_C(\mathbf{u}) := h(\mathbf{u}) + C\|\mathbf{u}_+\|_0, \tag{10}
\]

where \( h : \mathbf{u} \mapsto \mathbb{R} \) is a smooth convex function and gradient Lipschitz continuous with a Lipschitz constant \( \tau_h > 0 \) and \( C > 0 \) is given. To see the global solution of above problem, same as [26], we introduce an auxiliary problem

\[
\min_{\mathbf{u}, \mathbf{z} \in \mathbb{R}^m} f_{\gamma}(\mathbf{u}, \mathbf{z}) := C\|\mathbf{u}_+\|_0 + h(\mathbf{z}) + \langle \nabla h(\mathbf{z}), \mathbf{u} - \mathbf{z} \rangle + \frac{1}{2\gamma}\|\mathbf{u} - \mathbf{z}\|^2, \tag{11}
\]

for some \( \gamma > 0 \) and fixed \( \mathbf{z} \in \mathbb{R}^m \), where \( \nabla h \) is the gradient of \( h \). This problem allows us to acquire the result related to the proximal operator of \( L_{0/1} \).

Lemma 2.4. For any given \( C > 0 \), we have following results.

(i) If \( \mathbf{u}^* \) is the global optimal solution to (11) for any fixed \( \gamma > 0 \) and \( \mathbf{z} \in \mathbb{R}^m \), then it holds

\[
\mathbf{u}^* = \operatorname{prox}_{\gamma CL_{0/1}}(\mathbf{z} - \gamma \nabla h(\mathbf{z})).
\]

(ii) If \( \mathbf{u}^* \) is a global optimal solution to (10), then it is also a global optimal solution to (11) with \( \mathbf{z} = \mathbf{u}^* \) and \( 0 < \gamma \leq 1/\tau_h \), namely,

\[
f_C(\mathbf{u}^*) = f_{\gamma}(\mathbf{u}^*, \mathbf{u}^*) \leq f_{\gamma}(\mathbf{u}, \mathbf{u}^*), \quad \forall \mathbf{u} \in \mathbb{R}^m.
\]

This lemma suffices to show that a global optimal solution \( \mathbf{u}^* \) to (10) must satisfy a fixed point equation, which is well established by following theorem whose proof is easy and is omitted here.

Theorem 2.1. If \( \mathbf{u}^* \) is a global optimal solution to (10), then for any given \( 0 < \gamma \leq 1/\tau_h \) it satisfies

\[
\mathbf{u}^* = \operatorname{prox}_{\gamma CL_{0/1}}(\mathbf{u}^* - \gamma \nabla h(\mathbf{u}^*)). \tag{12}
\]

3 Optimality Conditions of \( L_{0/1} \)-SVM

This section provides the existence of optimal solutions of \( L_{0/1} \)-SVM and establishes two types of first-order optimality conditions: KKT points and P-stationary points.

3.1 Existence of \( L_{0/1} \)-SVM Minimizer

Theorem 3.1. Assume \( \mathbf{b} \) is finite-valued. Then the solution set of (4) is bounded and its global minimizer exists.

We observe that \( (\mathbf{w} ; \mathbf{b}) = (0; \mathbf{b}) \) may be an optimal solution (trivial solution) to (4), which possibly incorrectly predict the corresponding label \( y' \) for some new input vector \( \mathbf{x}' \) because \( \langle \mathbf{w}, \mathbf{x}' \rangle + b = b \). However, for any \( \mathbf{b} \in \mathbb{R} \), it follows from \( y_i \in \{1, -1\} \) that

\[
f(\mathbf{0}; \mathbf{b}) = C\|(1 - \mathbf{b} y)_+\|_0 = C \min\{m_+, m_-\},
\]

where \( m_+ \) and \( m_- \) denote the number of the positive and the negative labels in \( y \). Based on above equation, this means that any optimal solution \( (\mathbf{w}; \mathbf{b}) \) satisfying

\[
f(\mathbf{w}; \mathbf{b}) < C \min\{m_+, m_-\}
\]

is a non-trivial optimal solution to (4).

3.2 First-Order Optimality Condition

In this subsection, we discuss the first-order optimality conditions for the problem (4). To proceed this, we introduce a variable \( \mathbf{u} \in \mathbb{R}^m \) to equivalently reformulate (4) as

\[
\min_{\mathbf{w} \in \mathbb{R}^m, \mathbf{b} \in \mathbb{R}, \mathbf{u} \in \mathbb{R}^m} \frac{1}{2}\|\mathbf{w}\|^2 + C\|\mathbf{u}_+\|_0 \quad \text{s.t.} \quad \mathbf{u} + A\mathbf{w} + \mathbf{b}y = \mathbf{1}. \tag{13}
\]

The Lagrangian function of above problem is

\[
L(\mathbf{w}, \mathbf{b}, \mathbf{u}, \lambda) = \frac{1}{2}\|\mathbf{w}\|^2 + C\|\mathbf{u}_+\|_0 + \langle \lambda, \mathbf{u} + A\mathbf{w} + \mathbf{b}y - \mathbf{1} \rangle, \tag{14}
\]

where \( \lambda \in \mathbb{R}^m \) is the Lagrange multiplier, based on which we introduce the well known Karush-Kuhn-Tucker (KKT) point of problem (13).
Definition 3.2 (P-stationary point of (13)). For a given $C > 0$, we say that $(w^*: b^*; u^*)$ is a P-stationary point of problem (13) if there is a multiplier vector $\lambda^* \in \mathbb{R}^m$ such that
\[
\begin{align*}
\begin{cases}
w^* + A^T \lambda^* &= 0, \\
\langle y, \lambda^* \rangle &= 0, \\
u^* + Aw^* + b^* y &= 1,
\end{cases}
\end{align*}
\] (15)
Theorem 3.2. For a given $C > 0$, then $(w^*: b^*; u^*)$ is a local minimizer of (13) if and only if it is a P-stationary point.

Now let us define some notation
\[
B := [A \ y] \in \mathbb{R}^{m \times (n+1)}, \quad H := \begin{bmatrix}I_{n \times n} & 0 \\ 0 & 0 \end{bmatrix} B^+,
\] (16)
where $B^+$ is the generalized inverse of $B$. These notations could equivalently rewrite (13) as
\[
\min_{u \in \mathbb{R}^m} \frac{1}{2} ||H(u - 1)||^2 + C||u_+||_0,
\] (17)
which is an unconstrained non-convex optimization problem. Based on (17), we will derive the proximal stationary point of (13), and this point is useful as a stop criteria of our algorithm proposed later.

Definition 3.3 (KKT point of (13)). For a given $C > 0$, we say $(w^*: b^*; u^*)$ is a KKT point of (13) if there is a multiplier vector $\lambda^* \in \mathbb{R}^m$ and constant $\gamma > 0$ such that
\[
\begin{align*}
\begin{cases}
w^* + A^T \lambda^* &= 0, \\
\langle y, \lambda^* \rangle &= 0, \\
u^* + Aw^* + b^* y &= 1,
\end{cases}
\end{align*}
\] (18)
Theorem 3.3. Assume $B$ has a full column rank. For a given $C > 0$, if $(w^*: b^*; u^*)$ is a global minimizer of (13) then it is a P-stationary point with $0 < \gamma \leq \gamma_H$.

Note that $B$ having a full column rank means $m \geq n$. However, numerical experiments will demonstrate that our proposed algorithm also works for the cases of $m \leq n$ in terms of finding a P-stationary point. To end this section, we also unravel the relationship between a P-stationary point and a KKT point of (13).

Theorem 3.4. For a given $C > 0$, if $(w^*: b^*; u^*)$ is a P-stationary point with $0 < \gamma \leq \gamma_H$ of (13), then it is also a KKT point.

4 ALGORITHMIC DESIGN
In this section, we introduce the concept of $L_{0/1}$-support vector operator and describe how ADMM can be applied into solving the $L_{0/1}$-SVM problem (13).

4.1 $L_{0/1}$-Support Vector Operator
In SVMs, the optimal hyperplane is actually only determined by a small portion of training samples. These samples are called support vectors. It is well known that soft-margin loss functions at non-support vectors have zero subdifferentials [13], [14], [28], [29].

In other words, to select support vectors, one could find samples at which the loss function has nonzero subdifferentials. However, this approach is not suitable for $L_{0/1}$ soft-margin loss, since $\partial \ell_{0/1}(0) = \mathbb{R}^+$ and $\partial \ell_{0/1}(t) = \{0\}$ elsewhere. This indicates samples with $u_i = 1 - y_i(\langle w, x_i \rangle + b) \neq 0$ always have zero subdifferentials, and samples with $u_i = 0$ also have zero subdifferentials due to $0 \in \mathbb{R}_+$, which probably leads to empty set of support vectors. To overcome such drawback, we introduce a novel selection scheme, $L_{0/1}$-support vectors operator, to choose samples to be support vectors.

Definition 4.1 ($L_{0/1}$-support vector operator). For a given $\alpha > 0$, the $L_{0/1}$-support vector operator is defined by
\[
T^\alpha(z) := \{i \in \mathbb{N}_m : \text{prox}_{\alpha \ell_{0/1}}(z)_i = 0\}. \tag{19}
\]
Hereafter, we let $z_T$ (resp. $A_T$) be the sub-vector (resp. sub-matrix) contains elements of $z$ (resp. rows of $A$) indexed on $T$. Let $T := T^\alpha(z)$ and its complementarity set be $\bar{T} := \mathbb{N}_m \setminus T$. It follows from Definition 4.1 and (8) that
\[
\begin{bmatrix}
(\text{prox}_{\alpha \ell_{0/1}}(z))_T \\
(\text{prox}_{\alpha \ell_{0/1}}(z))_{\bar{T}}
\end{bmatrix} = \begin{bmatrix}0 \\ z_T\end{bmatrix}.
\]
This leads to
\[
u = \text{prox}_{\alpha \ell_{0/1}}(z) \iff \begin{bmatrix}u_T \\ u_{\bar{T}} - z_T\end{bmatrix} = 0. \tag{20}
\]
The above equivalence will help us to design the algorithm that we are ready to outline as below.

4.2 Framework of ADMM
The augmented Lagrangian function associated with the model (13) can be written as
\[
\begin{align*}
L_\sigma(w, b, u, \lambda) &= \frac{1}{2} ||w||^2 + C||u_+||_0 + \langle \lambda, w \rangle + \frac{\sigma}{2} ||w||^2, \tag{21}
\end{align*}
\]
where $\lambda$ is Lagrangian multiplier, $\sigma > 0$ is a given parameter and $w := u + Aw + by - 1$.

We take advantage of the ADMM to solve the augmented Lagrangian function. Given the $k$th iteration $(w^k, b^k, u^k, \lambda^k)$, its framework takes the following form
\[
\begin{align*}
u^{k+1} &= \arg\min_{u \in \mathbb{R}^m} L_\sigma(w^k, b^k, u, \lambda^k), \tag{22}
\end{align*}
\]
\[
\begin{align*}
w^{k+1} &= \arg\min_{w \in \mathbb{R}^m} L_\sigma(w, b^k, u^{k+1}, \lambda^k) + \frac{\sigma}{2} ||w - w^k||_2^2, \\
b^{k+1} &= \arg\min_{b \in \mathbb{R}} L_\sigma(w^{k+1}, b, u^{k+1}, \lambda^k), \tag{23}
\end{align*}
\]
\[
\begin{align*}
\lambda^{k+1} &= \lambda^k + \eta \sigma w^{k+1},
\end{align*}
\]
where $\eta > 0$ is referred as the dual step size and $\varpi^{k+1} := u^{k+1} + A w^{k+1} + b^{k+1} y - 1$. Here,

$$\|w - w^k\|_{D_k}^2 = \langle w - w^k, D_k (w - w^k) \rangle$$

is the so-called proximal term and $D_k \in \mathbb{R}^{n \times n}$ is symmetric. Note that if $D_k$ is positive semidefinite, then the above framework is the standard semi-proximal ADMM [30]. However, authors in papers [31]–[33] have also investigated ADMM with the indefinite proximal terms, namely $D_k$ is indefinite. The basic principle of choosing $D_k$ is to guarantee the convexity of $w$-subproblem of (22). Since $L_\sigma(w, b, u^{k+1}, \lambda^k)$ here is strongly convex with respect to $w$, $D_k$ is able to be chosen as a negative semidefinite matrix. The flexibility of selecting $D_k$ allows us to design a very efficient algorithm when support vectors are used.

### 4.3 $L_0/1$ADMM

We mainly describe how each subproblem of (22) can be addressed efficiently as well as how the support vectors can be applied into reducing the computational cost.

(i) Updating $u^{k+1}$. By (19), we denote

$$z^k := 1 - A w^k - b^k y - \lambda^k / \sigma, \quad T_k := T^{C/\sigma}(z^k).$$

Then the $u$-subproblem of (22) is reformulated as

$$\arg \min_{u \in \mathbb{R}^m} C\|u_+\|_0 + \sigma/2 \|u - z^k\|^2.$$  

which combining (20) results in

$$u^{k+1}_T := 0, \quad u^{k+1} = z^k_T. \tag{24}$$

(ii) Updating $w^{k+1}$. We always choose

$$D_k = -A^T_{T_k} A_{T_k}, \tag{25}$$

which enables us to derive the $w$-subproblem of (22) as

$$w^{k+1} = \arg \min_{w \in \mathbb{R}^n} \frac{1}{2} \|w\|^2 + \sigma/2 \|A w - v^k\|^2 + \sigma/2 \|w - w^k\|^2_{-A^T_{T_k} A_{T_k}}$$

$$= \arg \min_{w \in \mathbb{R}^n} \frac{1}{2} \|w\|^2 + \sigma/2 \|A w - v^k\|^2 - \sigma/2 \|A_{T_k} w - A_{T_k} w^k\|^2$$

where $v^k := -(u^{k+1} + b^k y - 1 + \lambda^k / \sigma)$. Moreover,

$$v^k_{T_k} = -(u^{k+1}_T + b^k y_{T_k} - 1 + \lambda^k_{T_k} / \sigma) = -(z^k_{T_k} + b^k y_{T_k} - 1 + \lambda^k_{T_k} / \sigma) = A_{T_k} w^k,$$

where the second and third equation are from (24) and (23). Now we rewrite (26) as

$$w^{k+1} = \arg \min_{w \in \mathbb{R}^n} \frac{1}{2} \|w\|^2 + \sigma/2 \|A w - v^k\|^2 - \frac{\sigma}{2} \|A_{T_k} w - v^k_{T_k}\|^2$$

$$= \arg \min_{w \in \mathbb{R}^n} \frac{1}{2} \|w\|^2 + \sigma/2 \|A_{T_k} w - v^k_{T_k}\|^2.$$  

To solve (27), we need find the solution to the equation

$$(I + \sigma A_{T_k} A_{T_k}) w = \sigma A_{T_k} v^k_{T_k}. \tag{28}$$

Note that $A_{T_k} \in \mathbb{R}^{[T_k] \times n}$, where $[T_k]$ is the cardinality of $T_k$. Then (28) can be addressed efficiently by following rules:

- If $n \leq |T_k|$, one could just solve (28) through

$$w^{k+1} = \sigma P_k^{-1} A_{T_k} v^k_{T_k}.$$  

- If $n > |T_k|$, the matrix inverse lemma enables us to calculate the inverse as

$$P_k^{-1} = I - \sigma A_{T_k} (I + \sigma A_{T_k} A_{T_k})^{-1} A_{T_k}. \tag{30}$$

Then we update $w^{k+1}$ as

$$w^{k+1} = \sigma A_{T_k} v^k_{T_k} - \sigma A_{T_k} Q_k^{-1} \sigma A_{T_k} A_{T_k} v^k_{T_k}$$

$$= \sigma A_{T_k} v^k_{T_k} - \sigma A_{T_k} Q_k^{-1} (Q_k - I) v^k_{T_k}$$

$$= \sigma A_{T_k} Q_k^{-1} w^k_{T_k}. \tag{31}$$

(iii) Updating $b^{k+1}$. By letting $r^k := -(A w^{k+1} - 1 + u^{k+1} + \lambda^k / \sigma)$, it follows from $b$-subproblem in (22) that

$$b^{k+1} = \arg \min_{b \in \mathbb{R}} \frac{\sigma}{2} \|b u^{k+1} + A w^{k+1} + b y\|^2 + \langle \lambda^k, by \rangle$$

$$= \arg \min_{b \in \mathbb{R}} \frac{\sigma}{2} \|b y - r^k\|^2$$

$$= \langle y, r^k \rangle / \|y\|^2 = (y, r^k) / m. \tag{32}$$

(iv) Updating $\lambda^{k+1}$. According to (15) and Lemma 2.1, $\lambda$ and $u$ have the relation $-\lambda \in C \|u\|_0$, namely $\lambda_i = 0$ if $u_i \neq 0$. Based on this, we update the Lagrangian multiplier $\lambda^{k+1}$ in the following way:

$$\lambda^{k+1} = \lambda^k + \eta \varpi^{k+1}, \quad \lambda^{k+1} = 0. \tag{33}$$

We now summarize the framework of the algorithm in Algorithm 1. We call the method $L_0/1$ADMM, an abbreviation for $L_0/1$-SVM solved by ADMM.

**Algorithm 1:** $L_0/1$ADMM for solving problem (4)

Initialize $(w^0, b^0, u^0, \lambda^0)$. Choose parameters $C, \sigma, K > 0$ and set $k = 0$.

**while** The halting condition does not hold and $k \leq K$ **do**

- Update $T_k := T^{C/\sigma}(z^k)$ as in (23).
- Update $u^{k+1}$ by (24).
- Update $w^{k+1}$ by (29) if $n \leq |T_k|$ and by (31) otherwise.
- Update $b^{k+1}$ by (32).
- Update $\lambda^{k+1}$ by (33).
- Set $k := k + 1$.

**end while**

**return** the final solution $(w^k, b^k)$ to (4).

**Remark 4.1.** We have some comments on Algorithm 1 regarding to the computational complexity. Note that in each step, updating $w^{k+1}$ dominates the whole computation, which needs solve a linear equation system (28) through (29) or (31). If $n \leq |T_k|$, then the computational complexities of calculating $A_{T_k} A_{T_k}$ and $P_k^{-1}$ are
Remark 4.2. Let \( \text{stationary point of (13)} \) by \( \text{four leading methods both on synthetic data and real data.} \)

In this part, we will conduct extensive numerical experiments, to show sparsity, robustness and effectiveness of our

establishment of convergence property of ADMM. There are a few publications that aim at studying ADMM to solve non-convex optimization problems while the established convergence results always rely on heavy assumptions. Importantly, Theorem 4.1 allows us to take advantage of the P-stationary point as a stopping criteria. In fact, we will terminate the algorithm if the point \( (w^k, b^k, u^k, \lambda^k) \) closely satisfies the conditions in (18), namely,

\[
\max \{ \theta_1^k, \theta_2^k, \theta_3^k, \theta_4^k \} < \text{tol},
\]

where \text{tol} is the tolerance level and

\[
\begin{align*}
\theta_1^k & := \frac{||w^k + A_T^T \lambda_T^k||}{1 + ||w^k||}, \quad \theta_2^k := \frac{|\langle y_{T_k}, \lambda_{T_k}^k \rangle|}{1 + |T_k|}, \\
\theta_3^k & := \frac{||u^k - 1 + Aw^k + b^k y||}{\sqrt{m}}, \\
\theta_4^k & := \frac{||u^k - \text{prox}_{C/\sigma L_{0/1}}(u^k - \lambda^k/\sigma)||}{1 + ||u^k||}.
\end{align*}
\]

5 Numerical experiments

In this part, we will conduct extensive numerical experiments to show sparsity, robustness and effectiveness of our algorithm \( L_{0/1 \text{ADMM}} \) by using MATLAB (2017a) on a laptop of 32GB of memory and Inter Core i7 2.7GHz CPU, against four leading methods both on synthetic data and real data.

(a) Parameters setting. In our algorithm, the parameters \( C \) and \( \sigma \) control the number of support vectors, see (23), so choosing a good value of these two parameters is crucial.

The standard 10-fold cross validation is employed in training set to choose optimal parameters, where the parameters \( C \) and \( \sigma \) are both selected from the candidate values \( \{a^{-7}, a^{-6}, \cdots, a^{-7}\} \) with \( a = \sqrt{2} \). The parameters with highest cross validation accuracy are picked out. In addition, we set \( \eta = 1.618 \). For the initial points, \( w^0 = 0.01 \times 1, b^0 = 0 \) and \( u^0 = \lambda^0 = 0 \). Finally, the maximum iteration number is \( K = 10^4 \) and the tolerance level is set as \text{tol} = 10^{-5} \) on synthetic data and \text{tol} = 10^{-3} \) on real data.

(b) Benchmark methods. Four leading methods are introduced to make comparisons. All their parameters are optimized to maximize the accuracy by 10-fold cross validation in each training set.

- **SVM** with hinge soft-margin loss is implemented by LibSVM [34], where the parameter \( C \) is selected from the set \( \{2^{-7}, 2^{-6}, \cdots, 2^7\} =: \Omega \).
- **SSVM** with square soft-margin loss [7] is implemented by LibSSVM [35], where the parameter \( C \) is picked from the range \( \Omega \).
- **PSVM** with pinball soft-margin loss can be achieved by using the traversal algorithm [36], where \( C \) and \( \tau \) are turned from the candidate values \( \{0, 0.1, 0.5, 1, 5, 10\} \cup \Omega \) and \( \{-1, -0.99, \cdots, 0.99\} \), respectively [36]. In order to improve computational efficiency of the traversal algorithm, authors in [36] suggested \( \tau = 0 \) (i.e., HSVM) when the number of training data is large.
- **R SVM** with ramp soft-margin loss can be achieved by employing the CCCP [37], where the parameters \( C \) and \( \mu \) are selected from \( \Omega \) and \( \{0, 0.1, 0.2, \cdots, 1\} \).

(c) Evaluation criterions. For the evaluation of classification performances, we report three evaluation criterions of five methods, that is, accuracy (ACC), number of support vectors (NSV) and CPU time (CPU). Let \( \{x_{test}^1, y_{test}^1, \cdots, x_{test}^{m_t}, y_{test}^{m_t}\} \) be \( m_t \) test samples data. The testing accuracy is defined by

\[
\text{ACC} := 1 - \frac{1}{2m_t} \sum_{j=1}^{m_t} \text{sign}(\langle w, x_{test}^j \rangle + b) - y_{test}^j, \]

where \( \text{sign}(\pi) = 1 \) if \( \pi > 0 \) and \( \text{sign}(\pi) = -1 \) otherwise, \( (w, b) \) is obtained by each method. The accuracy measures the ability of a model/method to correctly predict the class labels of any new input vectors. The higher the value of ACC is, the better the model/method is. The NSV and CPU are two comprehensive measures for classification models. The smaller their values are, the better the model is.

5.1 Comparisons with Synthetic Data

In this subsection, we first show that \( L_{0/1 \text{ADMM}} \) has the ability of support vector selection. For visualization, we consider a two-dimensional example where the features come from Gaussian distributions used in [3], [36].

Example 5.1 (Synthetic data in \( \mathbb{R}^2 \) without outliers). In this example, samples \( x_i \) with positive labels \( y_i = +1 \) are drawn from \( N(\mu_1, \Sigma_1) \) and samples \( x_i \) with negative labels \( y_i = -1 \) are drawn from \( N(\mu_2, \Sigma_2) \), where \( \mu_1 = [0.5, -3]^\top, \mu_2 = [-0.5, 3]^\top \) and \( \Sigma_1 = \Sigma_2 = \text{Diag}(0.2, 3) \). We generate \( m \) samples with two classes having equal numbers, and then evenly split all samples into a training set and a testing set.

Data generated in this way has centralized features of each class. For this experiment, the corresponding Bayes classifier is \( x_2 = 2.5x_1 \). We display Bayes classifier and 100 training data for each class in Figure 3 (a), where samples are able to be linearly separated and no extra noises contaminate the samples. We then add outliers on data generated in Example 5.1 as follows.

Example 5.2 (Synthetic data in \( \mathbb{R}^2 \) with outliers). Firstly, \( m \) samples with two classes having equal numbers are generated as in Example 5.1. Then in each class, we
randomly flip $r$ percentage of labels. For instance, in $m/2$ samples with positive labels +1, we change $mr/2$ labels of them to –1. This means $r$ percentage of $m$ samples are flipped their labels, namely $rm$ outliers are generated. Finally, we again evenly split those samples into a training set and a testing set. In Figure 3 (b), one training set with $r=10\%$ outliers are produced.

Fig. 3: Blue stars: sampling points in class –1. Red crosses: sampling points in class +1. Red dashed lines: the Bayes classifier. (a) A two dimensions training set with $m=200$ samples. (b) Data in (a) but with $r=10\%$ outliers.

To solve these two examples, five methods are applied to calculate the classification boundary $x_2 = w_1 x_1 + b$. Since data are generated randomly, we repeat above process 10 times to avoid randomness and report average results of ACC, NSV and CPU.

(d) Synthetic data without outliers. We first compare five methods for solving Example 5.1, where $m \in \{4000, 8000, \cdots, 20000\}$. Average results are reported in Table 1. It can be clearly seen that all methods achieved desirable ACC and $L_{0/1}\text{ADMM}$ got slightly better ones. When it comes to NSV, the picture is significant different. $L_{0/1}\text{ADMM}$ used a very small portion of samples as the support vectors, while SSVM and PSVM used all samples. Therefore, the phenomenon manifests that our constructed $L_{0/1}$ support vector operator is very effective to choose informative samples as the support vectors. As we mentioned in Remark 4.1, a small portion of samples used will greatly speed up the computation. This is testified by very short CPU time taken by $L_{0/1}\text{ADMM}$. Apparently, PSVM and RSVM consumed much longer time, which indicates these two methods would suffer from computational slowness in large scale date settings.

(e) Synthetic data with outliers. In the following experiment, we test five methods for solving Example 5.2, with fixing $m = 10000$, $n = 2$, $r \in \{0, 0.05, 0.1, 0.15, 0.2\}$. Average results are presented in Table 2. Again, there is no big difference of ACC generated by five methods. When more outliers were added, ACC became smaller. In addition, $L_{0/1}\text{ADMM}$ got slightly better ACC, which means it is more robust to outliers than other methods. As for NSV, SSVM and PSVM again took all samples. Compared with solving Example 5.1, HSVM this time used more support vectors and NSV increased when more outliers added, which means it is sensitive to the outliers. By contrast $L_{0/1}\text{ADMM}$ and RSVM seem to be more robust to the outliers since NSVs did not vary greatly with $r$ altering. Interestingly, being different with HSVM, these two methods needed fewer support vectors when more outliers added. Finally, $L_{0/1}\text{ADMM}$ always ran the fastest, with only taking less than 0.01 seconds, followed by HSVM and SSVM. Same as solving such data without outliers, PSVM consumed quite long CPU time. This implies that it may suffer from severe computational slowness for data with large size.

5.2 Comparisons with Real Data

We now focus on applying five methods into solving 13 real data sets. Table 3 presents the detailed information of them, where the last five ones have the training and testing data.

Example 5.3 (Real data without outliers). We perform 10-fold cross validation for the first six data sets, where each data is randomly split into ten parts, one of which is used for testing and the remaining nine parts is for training. We thus record average results to evaluate the performance. However, for the two large size samples: SUSY and HIGGS, the last 500,000 samples are used for testing, and the rest are for training. In our experiments, all features in each data set are scaled to $[-5, 5]$.

Example 5.4 (Real data with outliers). We still use these 13 real data sets in Example 5.3 but with adding outliers.
For each data set, we randomly pick $r$ percentage of training samples and then flip their labels. Same procedure is also applied into testing samples.

(f) Real data without outliers. Average results of five methods are recorded in Table 4. Note that some large size data sets make the other four methods run too much time, (e.g. over than one hour), so we do not report their results relating to those data sets. Clearly, $L_{0/1}$ADMM outperformed others in terms of biggest ACC, smallest NSV and shortest CPU for the most of data sets. More detailed, $L_{0/1}$ADMM and RSVM got better ACC than the other three methods. For instance, they predicted almost 90% samples correctly for col testing data whilst HSVM and PSVM only got less than 80% correct predictions. In terms of using support vectors, SSV and PSVM again took all samples into consideration. By contrast, $L_{0/1}$ADMM made use of a few number of support vectors, e.g. 113 v.s. 1247 by RSVM for adul data. As what we expected, $L_{0/1}$ADMM ran much faster than other methods for large size data sets because of small number of support vectors being used. For instance, 0.573 seconds v.s. 36.95 seconds by HSVM for icjc data. In addition, it only took 14.26 seconds to get the solution for hig data with more than ten million samples. This demonstrated that $L_{0/1}$ADMM is capable of dealing with data in extremely large scales.

(g) Real data with outliers. Finally, we would like to see the performance of each method on solving the real data sets with outliers, namely Example 5.4. We choose different ratios $r$ from $\{0.01, 0.02, \cdots , 0.1\}$. As reported in Table 4, the other four methods suffered from the computational slowness for data sets with large sizes, thus we only present results of six data sets with small sizes: col, aus, two, mus, lek and spl. In terms of the accuracy in Figure 4, ACC obtained by all methods dropped down with $r$ ascending, namely, more outliers being added. Generally speaking, $L_{0/1}$ADMM got the highest ACC except for spl, followed by RSVM. As for NSV in Figure 5, SSV and PSVM always took all samples. It can be seen that lines from $L_{0/1}$ADMM and RSVM did not go up when $r$ rose, which means they were quite robust to $r$, namely robust to the outliers. By contrast, more support vectors were needed by HSVM due to the rising of NSV when $r$ got increased. For each data set and each $r$, $L_{0/1}$ADMM always used the fewest support vectors, followed by RSVM and HSVM. When it comes to the CPU time in Figure 6, since col and lek have very small sizes, all methods got solutions quickly. While for other four data sets with moderate sizes, $L_{0/1}$ADMM ran fastest, and PSVM and RSVM came the last, such as, less than 0.1 second by $L_{0/1}$ADMM v.s. more than 100 seconds by PSVM and RSVM.

6 CONCLUSION

In this paper, we proposed a new soft-margin SVM model with the $L_{0/1}$ soft-margin loss function. It well captures the nature of the binary classification. The establishment of its optimality conditions made this NP-hard problem tractable. We then took advantage of the negative semidefinite proximal ADMM to solve this problem. The creation of $L_{0/1}$ Support vectors greatly reduced the computational complexity. Extensive numerical experiments demonstrated that our proposed method enjoys high order of accuracy and super fast computational speed. What is more, since it only took very small number of support vectors into consideration, the proposed method turns out to be very robust to the outliers. The idea of using $L_{0/1}$ soft-margin loss function might be able to extend to deal with the different types of SVM models, such as SVM [38]-[41], which severely suffers from outliers. It is also interesting to see how similar method and techniques can be designed to solve the kernel SVM problems. We leave this topic as a future research.

APPENDIX A

PROOFS OF ALL THEOREMS

ACKNOWLEDGEMENTS

This work is supported by the National Natural Science Foundation of China (11971052), the National Natural Science Foundation of China (61866010, 11871183), and the National Science Foundation of Hainan Province (118QN181).

REFERENCES

[1] C. Cortes and V. Vapnik, “Support vector networks,” Machine Learning, vol. 20, no. 3, pp. 273-297, 1995.
[2] V. Jumutc, X. Huang, and A. K. Suykens, “Fixed-size pegasos for hinge and pinball loss SVM,” International Joint Conference on Neural Networks, 2013.
[3] X. Huang, L. Shi, and A. K. Suykens, “Support vector machine classifier with pinball loss,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 36, no. 5, pp. 984-997, 2014.
[4] S. Rosset and J. Zhu, “Piecewise linear regularized solution paths,” The Annals of Statistics, vol. 35, no. 3, pp. 1012-1030, 2007.
[5] L. Wang, J. Zhu, and H. Zou, “Hybrid huberized support vector machines for microarray classification,” Bioinformatics, vol. 24, no. 3, pp. 412-419, 2008.
[6] Y. Xu, I. Akrotirianakis, and A. Chakraborty, “Proximal gradient method for huberized support vector machine,” Pattern Analysis and Applications, vol. 19, no. 4, pp. 989-1005, 2016.
[7] A. K. Suykens and J. Vandewalle, “Least squares support vector machine classifiers,” Neural Processing Letters, vol. 9, no. 3, pp. 293-300, 1999.
[8] Y. Yang, L. Tan, and L. F. He, “A robust least squares support vector machine for regression and classification with noise,” Neurocomputing, vol. 140, pp. 41-52, 2014.
[9] Y. Freund and R. E. Schapire, “A decision theoretic generalization of on-line learning and an application to boosting,” Journal of Computer and System Sciences, vol. 55, no. 1, pp. 119-139, 1997.
Fig. 4: ACC v.s. r of five methods for solving six data sets.

Fig. 5: NSV v.s. r of five methods for solving six data sets.
TABLE 4: Comparisons of five methods for solving Ex. 5.3, where \( L_{0/1} \) stands for \( L_{0/1}^{\text{ADMM}} \).

| Name | ACC(%) | CPU(seconds) |
|------|--------|--------------|
|      | \( L_{0/1} \) |         | \( L_{0/1} \) |         | \( L_{0/1} \) |         | \( L_{0/1} \) |         | \( L_{0/1} \) |         |
| col  |       |         |         |         |         |         |         |         |         |         |
| aus  |       |         |         |         |         |         |         |         |         |         |
| two  |       |         |         |         |         |         |         |         |         |         |
| mus  |       |         |         |         |         |         |         |         |         |         |
| adu  |       |         |         |         |         |         |         |         |         |         |
| lek  |       |         |         |         |         |         |         |         |         |         |
| spl  |       |         |         |         |         |         |         |         |         |         |
| wga |       |         |         |         |         |         |         |         |         |         |
| wij |      |         |         |         |         |         |         |         |         |         |
| env |       |         |         |         |         |         |         |         |         |         |
| hig |       |         |         |         |         |         |         |         |         |         |

**Fig. 6:** CPU v.s. \( r \) of five methods for solving six data sets.

[10] J. Friedman, T. Hastie, and R. Tibshirani, “Additive logistic regression: a statistical view of boosting,” *Annals of statistics*, vol. 28, no. 2, pp. 337-374, 2000.

[11] L. Mason, P. L. Bartlett, and J. Baxter, “Improved generalization through explicit optimization of margins,” *Machine Learning*, vol. 38, no. 3, pp. 243-255, 2000.

[12] F. Perez-Cruz, A. Navia-Vazquez, A. R. Figueiras-Vidal, and A. Artes-Rodriguez, “Empirical risk minimization for support vector classifiers,” *IEEE Transactions on Neural Networks*, vol. 14, no. 2, pp. 296-303, 2003.

[13] X. Huang, L. Shi, and J. A. K. Suykens, “Ramp loss linear programming support vector machine,” *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 2185-2211, 2014.

[14] X. Shen, L. F. Niu, Z. Qi, and Y. J. Tian, “Support vector machine classifier with truncated pinball loss,” *Pattern Recognition*, vol. 68, pp. 199-210, 2017.

[15] L. M. Yang and H. G. Dong, “Support vector machine with truncated pinball loss and its application in pattern recognition,” *Chemometrics and Intelligent Laboratory Systems*, vol. 177, pp. 89-99, 2018.

[16] F. Perez-Cruz, A. Navia-Vazquez, P. L. Alarcon-Diana, and A. Artes-Rodriguez, “Support vector classifier with hyperbolic tangent penalty function,” *International Conference on Acoustics, Speech, and Signal Processing*, 2000.

[17] L. Mason, J. Baxter, P. Bartlett, and M. Frean, “Boosting algorithms as gradient descent,” *International Conference on Neural Information Processing Systems*, 1999.

[18] P. L. Bartlett, M. I. Jordan, and J. D. McAllell, “Large margin classifiers: convex loss, low noise, and convergence rates,” *International Conference on Neural Information Processing Systems*, 2004.

[19] P. L. Bartlett, M. I. Jordan, and J. D. McAllell, “Convexity, classification, and risk bounds,” *Journal of the American Statistical Association*, vol. 101, no. 473, pp. 138-156, 2006.

[20] J. H. Friedman, “On bias, variance, 0/1-loss, and the curse-of-dimensionality,” *Data Mining and Knowledge Discovery*, vol. 1, no. 1, pp. 55-77, 1997.
[21] J. P. Brooks, “Support vector machines with the ramp loss and the hard margin loss,” *Operations Research*, vol. 59, no. 2, pp. 467-479, 2011.
[22] L. Li and H. T. Lin, “Optimizing 0/1 loss for perceptrons by random coordinate descent,” *International Joint Conference on Neural Networks*, 2007.
[23] B. Mordukhovich and N. Nam, “An easy path to convex analysis and applications,” *Morgan and Claypool Publishers*, 2014.
[24] R. T. Rockafellar and R. J. Wets, “Variational analysis,” *Springer Science and Business Media*, 1998.
[25] H. H. Bauschke and P. L. Combettes, “Convex analysis and monotone operator theory in hilbert space,” *New York: Springer*, 2011.
[26] Y. Q. Chen, N. H. Xiu, and D. T. Peng, “Global solutions of non-Lipschitz $S_2$-$S_p$ minimization over the positive semidefinite cone,” *Optimization Letters*, vol. 8, no. 7, pp. 2053-2064, 2014.
[27] A. Beck and Y. C. Eldar, “Sparsity constrained nonlinear optimization: optimality conditions and algorithms,” *SIAM Journal on Optimization*, vol. 23, no. 3, pp. 1480-1509, 2013.
[28] I. Steinwart and N. Christianini, “Sparseness of support vector machines,” *Journal of Machine Learning Research*, vol. 4, no. 6, pp. 1071-1105, 2004.
[29] S. Ertekin, L. Bottou, and C. L. Giles, “Nonconvex online support vector machines,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 4, pp. 368-381, 2010.
[30] M. Fazel, T. K. Pong, D. F. Sun, and P. Tseng, “Hankel matrix rank minimization with applications to system identification and realization,” *SIAM Journal on Matrix Analysis and Applications*, vol. 34, no. 3, pp. 946-977, 2013.
[31] M. Li, D. F. Sun and K. C. Toh, “A majorized ADMM with indefinite proximal terms for linearly constrained convex composite optimization,” *SIAM Journal on Optimization*, vol. 26, no. 2, pp. 922-950, 2016.
[32] B. S. He, F. Ma, and X. M. Yuan, “Optimal linearized alternating direction method of multipliers for convex programming,” Available on http://www.optimization-online.org/DB_FILE/2017/09/6228.pdf, 2017.
[33] X. Chang, S. Liu, P. Zhao, and D. Song, “A generalization of linearized alternating direction method of multipliers for solving two-block separable convex programming,” *Journal of Computational and Applied Mathematics*, vol. 357, no. 2, pp. 251-272, 2019.
[34] C. C. Chang and C. J. Lin, “LIBSVM: a library for support vector machines,” *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, pp. 27, 2011.
[35] K. Pelckmans, J. A. K. Suykens, T. V. Gestel, J. D. Brabanter, L. Lukas, B. Hamers, B. D. Moor, and J. Vandewalle, “SSVM lab: a matlab/c toolbox for least squares support vector machines,” Tutorial. KULeuven-ESAT. Leuven, Belgium, vol. 142, pp. 1-2, 2002.
[36] X. Huang, L. Shu, and J. A. K. Suykens, “Solution path for pin-SVM classifiers with positive and negative $\tau$ values,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 7, pp. 1584-1593, 2016.
[37] Y. Wu and Y. Liu, “Robust truncated hinge loss support vector machines,” *Journal of the American Statistical Association*, vol. 102, no. 479, pp. 974-983, 2006.
[38] R. Khemchandani and S. Chandra, “Twin support vector machines for pattern classification,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 5, pp. 905-910, 2007.
[39] Y. Xu, Z. Yang, and X. Pan, “A novel twin support vector machine with pinball loss,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 2, pp. 359-370, 2017.
[40] B. Hong, W. Z. Zhang, W. Liu, J. F. Ye, D. Cai, X. f. He, and J. Wang, “Scaling up sparse support vector machines by simultaneous feature and sample reduction,” *Journal of Machine Learning Research*, vol. 20, no. 121, pp. 1-39, 2019.
[41] C. N. Li, Y. H. Shao, H. J. Wang, Y. T. Zhao, L. W. Huang, N. H. Xiu, and N. Y. Deng, “Single Versus Union: Non-parallel Support Vector Machine Frameworks,” arXiv preprint arXiv:1910.08754, 2019.