Information Sciences 642 (2023) 119136

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Fast SVM classifier for large-scale classification problems

Huajun Wang [a,*], Genghui Li [b], Zhenkun Wang [c]

[a] Department of Mathematics and Statistics, Changsha University of Science and Technology, Changsha, PR China
[b] School of System Design and Intelligent Manufacturing, Southern University of Science and Technology, Shenzhen, PR China
[c] School of System Design and Intelligent Manufacturing and the Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, PR China

ARTICLE INFO

Keywords:
Support vector machine
Truncated squared hinge loss
Support vectors
Optimality theory
Global convergence
Low computational complexity

ABSTRACT

Support vector machines (SVM), as one of effective and popular classification tools, have been widely applied in various fields. However, they may incur prohibitive computational costs when solving large-scale classification problems. To address this problem, we construct a new fast SVM with a truncated squared hinge loss (dubbed as L_{ts}-SVM). We begin by developing an optimality theory of the nonconvex and nonsmooth L_{ts}-SVM, which makes it convenient for us to investigate the support vectors and working set of L_{ts}-SVM. Based on this, we propose a new and effective global convergence algorithm to address the L_{ts}-SVM. This method is found to enjoy a tremendously low computational complexity, which makes sufficiently decreasing the demand for extremely large-scale computation possible. Numerical comparisons with eight other solvers show that our proposed algorithm achieves excellent performance on large-scale classification problems with regard to shorter computational times, more desirable accuracy levels, fewer support vectors and more robust to outliers.

1. Introduction

Support vector machines (SVM) were first developed by Vapnik and Cortes [1] and had been evolutionary advances in theory and algorithms in the last two decades. The basic idea of SVM is to construct a hyperplane in the feature space that best separates the training set. This paper mainly considers a binary classification problem, which is presented as below. For a given training set $\{(\mathbf{x}_i, y_i) : i \in [m] := \{1, 2, \cdots, m\}\}$, where $\mathbf{x}_i \in \mathbb{R}^n$ are the feature vectors and $y_i \in \{-1, 1\}$ are the labels. The goal of SVM is to construct a hyperplane $\langle \mathbf{w}, \mathbf{x} \rangle + b = 0$ to separate the training set, where parameters $b \in \mathbb{R}$ and $\mathbf{w} \in \mathbb{R}^n$ need to be estimated. To find optimal hyperplane, we need consider two possible cases: linearly separable and inseparable training set. For linearly separable training set, the unique optimal hyperplane can be received by dealing with the following SVM model [1]:

$$\min_{\mathbf{w}\in\mathbb{R}^n, b\in\mathbb{R}} \frac{1}{2}\|\mathbf{w}\|^2, \quad \text{s.t.} \quad y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1, i \in [m], \tag{1}$$

where $\|\cdot\|$ is the Euclidean norm. The model (1) is named as classical hard-margin SVM because it requires correct classifications of all training data.

* Corresponding author.
E-mail addresses: huajunwang@bjtu.edu.cn (H. Wang), genghuili2-c@my.cityu.edu.hk (G. Li), wangzhenkun90@gmail.com (Z. Wang).

https://doi.org/10.1016/j.ins.2023.119136
Received 18 January 2023; Received in revised form 26 March 2023; Accepted 7 May 2023

Available online 18 May 2023

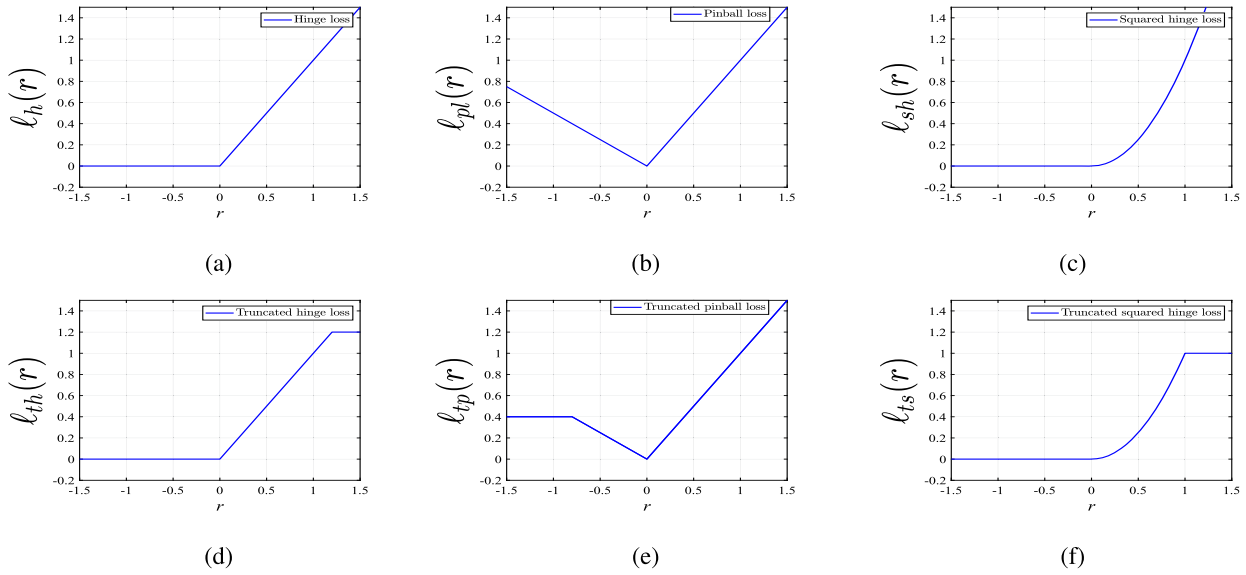

Fig. 1. (a) Hinge loss function. (b) Pinball loss function with $\zeta = 0.5$. (c) Squared hinge loss function. (d) Truncated hinge loss function with $\mu = 1.2$. (e) Truncated pinball loss function with $\zeta = 0.5$ and $\upsilon = 0.4$. (f) Truncated squared hinge loss function.

For linearly inseparable training set, the popular method is to solve the SVM model with loss function,

$$\min_{\mathbf{w}\in\mathbb{R}^n, b\in\mathbb{R}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + \eta \sum_{i=1}^{m} \ell(1 - y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle + b)), \tag{2}$$

where $\eta > 0$ represents the given parameter and the $\ell(r)$ stands for the loss function with $r := 1 - y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle + b)$. The above model is called soft-margin SVM since it allows misclassified training samples. The soft-margin SVM has been widely applied in various fields, such as, disease detection [2–4], machine learning [5–7], image classification [8–10], and so forth.

1.1. Related work

Extensive work has focused on constructing new loss functions $\ell(\cdot)$ to obtain new efficient soft-margin SVM models. Based on the convexity of loss functions $\ell(\cdot)$, they can be summarized into two classes. The first class consists of the convex loss functions. Here, we only review some popular loss functions, which are sufficient for motivating our work in this paper.

a) Hinge loss function. The first soft-margin SVM takes advantage of the hinge loss function [1] (see Fig. 1(a)), which is given by

$$\ell_h(r) := \begin{cases} 0, & r < 0, \\ r, & r \geq 0. \end{cases}$$

The hinge loss function is the non-differentiable at $r = 0$ and gives a loss at 0 for training samples with $r < 0$. Thus, this function leads to the sparsity. However, the hinge loss function pays linear loss for training samples with $r \geq 0$, which losses the robustness to outliers [11]. The hinge loss SVM is a convex optimization model since the hinge loss function is convex function. Other relevant work on the hinge loss SVM includes the sequential minimal optimization (SMO) algorithm [12], augmented Lagrangian method [13], the dual coordinate descent method [14], and so on.

b) Pinball loss function. To improve the effectiveness of hinge loss SVM, Huang et al. in [15] first studied the pinball loss SVM and the pinball loss function (see Fig. 1(b)) is defined as

$$\ell_{pl}(r) := \begin{cases} r, & r \geq 0, \\ \zeta r, & r < 0, \end{cases}$$

where $\zeta \in [0, 1]$. If $\zeta = 0$, then the pinball loss function is reduced to the hinge loss function. In addition, for $\zeta \in (0, 1]$, the pinball loss function gives a linear loss for all training samples, which looses the robustness to outliers and sparsity [16]. There is an impressive body of work proposing algorithms for solving the pinball loss SVM, which includes the traversal algorithm [17], the stochastic subgradient method [18], the SMO algorithm [19].

c) Squared hinge loss function. To improve smoothness of hinge loss SVM, authors in [1] first developed squared hinge loss SVM and the squared hinge loss function (see Fig. 1(c)) is defined as

$$\ell_{sh}(r) := \begin{cases} 0, & r < 0, \\ r^2, & r \geq 0. \end{cases}$$

The squared hinge loss function is a differentiable function and presents loss at 0 for training samples with $r < 0$, which receives the sparsity. However, the squared hinge loss function gives a squared loss for training samples with $r \geq 0$, which losses the robustness to outliers [20]. Other work focuses on proposing new algorithms for solving squared hinge loss SVM, such as, the semismooth Newton method [21], the stochastic gradient algorithm [22], the coordinate descent method [23], and so on.

d) Other convex loss functions include elastic net loss function [24], ε-insensitive pinball loss function [15], huberized pinball loss function [25], and so on.

The above corresponding SVM optimization models are easy to be dealt with because their loss functions are convex functions [26]. Nevertheless, the convexity of these loss functions is the unbounded functions, which weakens robustness to outliers. To overcome such drawback, authors in [27–30] set an upper bound after a certain threshold, which gets the following nonconvex loss functions.

e) Truncated hinge loss (ramp loss) function. To increase the robustness to outliers of hinge loss SVM, Wu et al. in [31] first proposed truncated hinge loss SVM and the truncated hinge loss (see Fig. 1(d)) is defined as

$$\ell_{th}(r) := \begin{cases} 0, & r < 0, \\ t, & r \in (0, \mu), \\ \mu, & r \geq \mu, \end{cases}$$

where $\mu \geq 1$. If $\mu = 1$, then the truncated hinge loss function is reduced to the ramp loss [32,33]. In fact, the truncated hinge loss function gives loss fixed at 0 for training samples with $r < 0$, which gets the sparsity and pays loss fixed at μ for training samples with $r > \mu$, which tends to be robustness to outliers [34]. Other relevant work on the truncated hinge loss SVM includes the convex-concave procedure (CCCP) [35], the branch and bound algorithm [29], the coordinate descent method [36], and so on.

f) Truncated pinball loss function. To increase the sparseness of pinball loss SVM, authors in [37,38] studied truncated pinball loss SVM and the truncated pinball loss function (see Fig. 1(e)) is defined by

$$\ell_{tp}(r) = \begin{cases} r, & r > 0, \\ -\overline{\zeta} r, & r \in [-\upsilon, 0], \\ \overline{\zeta}\upsilon, & r < -\upsilon, \end{cases}$$

where $\overline{\zeta} \in [0, 1], \upsilon > 0$. The truncated pinball loss function gives loss fixed at $\overline{\zeta}\upsilon$ for training samples with $t < -\upsilon$ and pays a linear loss otherwise, which obtains sparsity but tends to be robustness to outliers [39]. The truncated pinball loss SVM is a noncovex optimization model since truncated pinball loss is a noncovex function, which can be addressed by the effective CCCP [38].

g) Other nonconvex loss functions include the truncated logistic loss function [40], the asymmetrical truncated pinball loss function [41], truncated exponential loss function [42], and so on.

Compared with above convex loss functions, we get that most of nonconvex loss functions are robust to feature noises or outliers due to the boundedness of nonconvex loss functions [43]. However, the nonconvexity of these loss functions would lead to difficulties to computations for addressing corresponding SVM optimization models.

1.2. Truncated squared hinge loss SVM

To increase robustness and sparsity of the above losses, we propose a new robust and sparse loss function, which is given by

$$\ell_{ts}(r) := \begin{cases} 1, & r > 1, \\ r^2, & r \in [0, 1], \\ 0, & r < 0. \end{cases} \tag{3}$$

The above new loss function is called truncated squared hinge loss function (see Fig. 1(f)), which has certain desirable properties.

(i) The truncated squared hinge loss function gives loss at 1 for training samples with $r > 1$ and thus gets robustness to outliers, which is more robust than the hinge loss function, the pinball loss function, the squared hinge loss function, the truncated pinball loss function, and so on.
(ii) The truncated squared hinge loss function returns 0 for training samples with $r < 0$ and thus obtains sparsity. Moreover, this function is sparser than the pinball loss function, the elastic net loss function, the huberized pinball loss function, the truncated pinball loss function, and so on.
(iii) The truncated squared hinge loss function is bounded between 0 and 1, which is the same range as that of the 0-1 loss.

However, the truncated squared hinge loss function is a nonconvex and nonsmooth function, which would result in difficulties to computations for solving corresponding truncated squared hinge loss SVM. In (2), we replace $\ell(\cdot)$ by $\ell_{ts}(\cdot)$ and get the following new sparse and robust SVM model

$$\min_{\mathbf{w}\in\mathbb{R}^n, b\in\mathbb{R}} \ \frac{1}{2}\|\mathbf{w}\|^2 + \eta \sum_{i=1}^{m} \ell_{ts}(1 - y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle + b)). \tag{4}$$

Notation in Table 1 enables us to equivalently rewrite model (4) as the following unconstrained optimization model

$$\min_{\mathbf{w}\in\mathbb{R}^n, b\in\mathbb{R}} f(\mathbf{w}; b) := \frac{1}{2}\|\mathbf{w}\|^2 + \eta L_{ts}(\mathbf{e} - H\mathbf{w} - b\mathbf{y}), \tag{5}$$

or the following constrained optimization model with an extra variable $\mathbf{d} \in \mathbb{R}^m$,

$$\begin{aligned} \min_{\mathbf{w}\in\mathbb{R}^n, b\in\mathbb{R}, \mathbf{d}\in\mathbb{R}^m} \ & \frac{1}{2}\|\mathbf{w}\|^2 + \eta L_{ts}(\mathbf{d}) \\ \text{s.t.} \ & \mathbf{d} + H\mathbf{w} + b\mathbf{y} = \mathbf{e}. \end{aligned} \tag{6}$$

The above models (4), (5) and (6) are called truncated squared hinge loss SVM, which is dubbed as L_{ts}-SVM. Fortunately, the L_{ts}-SVM model (6) shares well robustness to outliers and sparsity, which is convenient for us to deal with large-scale SVM problem. Hence, we carry out this paper along with L_{ts}-SVM model (6).

1.3. Our contributions

The aim of this paper is to establish the new optimality theory and develop a new effective algorithm for the L_{ts}-SVM. The key contributions are given as follows.

(i) The new SVM model. We construct a new loss function, truncated squared hinge loss function (3) (see Fig. 1(f)), which is more robust or sparser than many existing loss function (see Subsection 1.2) and allows us to construct a new SVM model, L_{ts}-SVM (6), to deal with the large-scale SVM classification problems. The nonconvexness and nonsmoothness of truncated squared hinge loss lead to difficulty to computation for solving L_{ts}-SVM in general. Nevertheless, a new first-order necessary optimality condition for a local optimal solution of L_{ts}-SVM is established through the newly introduced proximal stationary point, see Theorem 3.1. Moreover, a new first-order sufficient optimality condition as stated in (12) for a local optimal solution of L_{ts}-SVM is further developed without any assumptions, see Theorem 3.2.

(ii) The new support vectors and working set. Since the developed optimality conditions indicating a proximal stationary point are instructive to pursue an optimal solution of L_{ts}-SVM, therefore, based on proximal stationary point, we define the new L_{ts} support vectors as stated in (26) and show that the L_{ts} support vectors are a small portion of the whole training set, see Theorem 4.1. Following the idea of L_{ts} support vectors, we introduce a working set in each step, see (30). This means that indices j of vectors x_j out of this working set will be discarded, which enables tremendously decreasing the demand for large scale computation possible.

(iii) The new and efficient L_{ts}-ADMM algorithm. When it comes to solving the L_{ts}-SVM, we propose a new and effective alternating direction method of multipliers with working set (L_{ts}-ADMM). In L_{ts}-ADMM, at each round of training, by using the proximal stationary point, the whole training set are randomly divided into two groups: the working set and the non-working set. The working set adopts L_{ts}-ADMM to update their parameters, while parameters in the non-working set remain unchanged, which means that L_{ts}-ADMM updates parameters on reduced datasets instead of the full datasets to solve the L_{ts}-SVM, see Algorithm 1, so the L_{ts}-ADMM enjoys a low computational complexity and hence runs super fast, see Subsection 4.3. Moreover, L_{ts}-ADMM is proved to be global convergence to a proximal stationary point, which must be a local optimal solution of (4), see Theorem 4.2.

(iv) High numerical performance. We compare our algorithm with eight other efficient solvers. Numerical experiments show that our algorithm shares excellent performance with regard to the fewer number of support vectors, shorter computational time and higher prediction accuracy, particularly for large-scale datasets. In addition, the numerical results also demonstrate that our developed algorithm enjoys relatively robustness to the outliers.

1.4. Organization

This paper is organized as follows. In Section 2, we derive explicit expressions of the proximal operator and limiting subdifferential of truncated squared hinge loss function. In Section 3, we establish the optimality theory for local optimal solution of L_{ts}-SVM. In Section 4, we present algorithm L_{ts}-ADMM and then establish its global convergence and computational complexity. In Section 5, we give numerical experiments, including the comparisons with eight other leading solvers. Concluding remarks are made in Section 6.

1.5. Notation

We provide some notation that will be employed throughout the paper in Table 1.

2. Preliminaries

In this part, we first give an explicit expression of the limiting subdifferential of truncated squared hinge loss function and derive an explicit expression of the proximal operator of truncated squared hinge loss function. These explicit expressions will be used to study necessary and sufficient conditions for the local optimal solution of L_{ts}-SVM in the next section and establish a new and efficient algorithm for solving L_{ts}-SVM in Section 4.

Table 1
List of notations.

Notation	Description
$\|\|\cdot\|\|$	The norm induced by the standard inner product (i.e., Euclidean norm).
$\mathbf{y}$	The classes/labels $(y_1, \cdots, y_m)^\top \in \mathbb{R}^m$.
$[m]$	The index set $\{1, 2, \cdots, m\}$.
$\mathbf{X}$	The samples data $[\mathbf{x}_1 \cdots \mathbf{x}_m]^\top \in \mathbb{R}^{m\times n}$.
$:=$	means "define".
$\|\Gamma\|$	The number of elements of an index set $\Gamma \in [m]$.
$\mathbf{d}_\Gamma$	The sub-vector of $\mathbf{d}$ indexed on Γ and $\mathbf{d}_\Gamma \in \mathbb{R}^{\|\Gamma\|}$.
H	$:= [y_1\mathbf{x}_1 \ y_2\mathbf{x}_2 \ \cdots \ y_m\mathbf{x}_m]^\top \in \mathbb{R}^{m\times n}$.
$\overline{\Gamma}$	The complementary set of an index set Γ, namely, $[m]\backslash\Gamma$.
H_Γ	The sub-matrix containing the rows of H indexed on Γ.
$\mathbf{I}$	The identity matrix.
$\mathbf{e}$	$:= (1, \cdots, 1)^\top \in \mathbb{R}^m$ whose dimension varies in the context.
$\mathbf{d}$	$:= \mathbf{e} - H\mathbf{w} - b\mathbf{y} \in \mathbb{R}^m$.
$L_{ts}(\mathbf{d})$	$:= \sum_{i=1}^m \ell_{ts}(d_i)$.
$N(\boldsymbol{\varphi}, \mu)$	The neighborhood of $\boldsymbol{\varphi}$ with radius $\mu > 0$, namely, $\{\mathbf{u} \in \mathbb{R}^m : \|\|\mathbf{u} - \boldsymbol{\varphi}\|\| < \mu\}$.

2.1. Limiting subdifferential of truncated squared hinge loss function

First, we give the definition of the limiting subdifferential of truncated squared hinge loss function, which is dubbed as L_{ts} limiting subdifferential.

Definition 2.1. [44, Definition 8.3] For truncated squared hinge loss $L_{ts}(\cdot) : \mathbb{R}^m \to \mathbb{R}$, its regular and limiting subdifferentials are defined respectively as

$$\widehat{\partial} L_{ts}(\mathbf{d}) = \left\{ \mathbf{v} \in \mathbb{R}^m : \liminf_{\substack{\widehat{\mathbf{d}} \to \mathbf{d} \\ \widehat{\mathbf{d}} \neq \mathbf{d}}} \frac{L_{ts}(\widehat{\mathbf{d}}) - L_{ts}(\mathbf{d}) - \langle \mathbf{v}, \widehat{\mathbf{d}} - \mathbf{d} \rangle}{\|\widehat{\mathbf{d}} - \mathbf{d}\|} \geq 0 \right\},$$

$$\partial L_{ts}(\mathbf{d}) = \limsup_{\widehat{\mathbf{d}} \xrightarrow{L_{ts}} \mathbf{d}} \widehat{\partial} L_{ts}(\widehat{\mathbf{d}}) = \left\{ \mathbf{v} \in \mathbb{R}^m : \exists\, \widehat{\mathbf{d}}_j \xrightarrow{L_{ts}} \mathbf{d},\ \mathbf{v}_j \in \widehat{\partial} L_{ts}(\widehat{\mathbf{d}}_j) \text{ with } \mathbf{v}_j \to \mathbf{v} \right\},$$

where $\widehat{\mathbf{d}} \xrightarrow{L_{ts}} \mathbf{d}$ means both $\widehat{\mathbf{d}} \to \mathbf{d}$ and $L_{ts}(\widehat{\mathbf{d}}) \to L_{ts}(\mathbf{d})$.

Based on the above Definition 2.1, the explicit expression of L_{ts} limiting subdifferential is given in Lemma 2.1.

Lemma 2.1. *Given $\mathbf{d} \in \mathbb{R}^m$, the explicit expression of L_{ts} limiting subdifferential at $\mathbf{d} \in \mathbb{R}^m$ is given by*

$$\partial L_{ts}(\mathbf{d}) := (\partial \ell_{ts}(d_1), \partial \ell_{ts}(d_2), \cdots, \partial \ell_{ts}(d_m))^\top \in \mathbb{R}^m, \tag{7}$$

$$\partial \ell_{ts}(d_i) := \begin{cases} 0, & d_i > 1, \\ \{0, 2d_i\}, & d_i = 1, \\ 2d_i, & d_i \in (0, 1), \\ 0, & d_i \leq 0. \end{cases} \quad i \in [m]. \tag{8}$$

Proof. Because the $L_{ts}(\mathbf{d})$ is separable, therefore, we obtain (7). Next, we prove the explicit expression of ℓ_{ts} limiting subdifferential by two scenarios.
(i) For $d_i \neq 1$, the $\ell_{ts}(d_i)$ is differentiable. Hence, we obtain $\partial \ell_{ts}(d_i) = 0$ for $d_i < 0$ or $d_i > 1$ and $\partial \ell_{ts}(d_i) = 2d_i$ for $d_i \in [0, 1)$.
(ii) For $d_i = 1$, the $\ell_{ts}(d_i)$ is not differentiable. Because $\partial \ell_{ts}(1) = \limsup_{d_i \to 1} \widehat{\partial} \ell_{ts}(d_i)$ and $\widehat{\partial} \ell_{ts}(d_i) = 2d_i$ with $0 < d_i < 1$ and $\widehat{\partial} \ell_{ts}(d_i) = 0$ with $d_i > 1$ from Definition 2.1, thus we get $\partial \ell_{ts}(1) \in \{0, 2d_i\}$.
Based on the above scenarios (i) and (ii), we get (8). □

The explicit expressions (7) and (8) of L_{ts} limiting subdifferential will be used to prove the first-order necessary condition for the local optimal solution of L_{ts}-SVM in Theorem 3.1. Next, we will study the proximal operator of truncated squared hinge loss function.

2.2. Proximal operator of truncated squared hinge loss function

We first give the definition of proximal operator of truncated squared hinge loss function for the one-dimensional case, which is known as ℓ_{ts} proximal operator.

Definition 2.2. [44, Definition 1.22] Given $\kappa, \eta > 0$ and $s \in \mathbb{R}$, the proximal operator of $\ell_{ts}(u)$ is defined as

$$\text{prox}_{\kappa\eta\ell_{ts}}(s) = \arg\min_{u\in\mathbb{R}} \frac{1}{2}(u-s)^2 + \kappa\eta\ell_{ts}(u). \tag{9}$$

Based on the above Definition 2.2, the explicit expression of ℓ_{ts} proximal operator is shown as below.

Lemma 2.2. *Given $\kappa, \eta > 0$ and s, the explicit expression of ℓ_{ts} proximal operator at $s \in \mathbb{R}$ can be shown as*

$$prox_{\kappa\eta\ell_{ts}}(s) := \begin{cases} s, & s > \sqrt{1+2\kappa\eta}, \\ s \text{ or } \frac{s}{1+2\kappa\eta}, & s = \sqrt{1+2\kappa\eta}, \\ \frac{s}{1+2\kappa\eta}, & 0 < s < \sqrt{1+2\kappa\eta}, \\ 0, & s \le 0. \end{cases} \tag{10}$$

Proof. By (3) and (9), we get that $\text{prox}_{\kappa\eta\ell_{ts}}(s)$ is the minimizer of following function

$$\psi(z) := \begin{cases} \psi_1(z) := \kappa\eta + \frac{(z-s)^2}{2}, & z > 1, \\ \psi_2(z) := \kappa\eta + \frac{(1-s)^2}{2}, & z = 1, \\ \psi_3(z) := \kappa\eta z^2 + \frac{(z-s)^2}{2}, & 0 < z < 1, \\ \psi_4(z) := \frac{(z-s)^2}{2}, & z \le 0. \end{cases}$$

Clearly, the minimizers of $\psi_1(z)$, $\psi_2(z)$, $\psi_3(z)$ and $\psi_4(z)$ are received at $z_1^* = s$, $z_2^* = 1$, $z_3^* = \frac{s}{1+2\kappa\eta}$ and $z_4^* = s$ respectively. By comparing the four values of $\psi(z_1^*), \psi(z_2^*), \psi(z_3^*)$ and $\psi(z_4^*)$, we obtain desired conclusion:
(i) As $s > \sqrt{1+2\kappa\eta}$, we get $\min\{\psi(z_2^*), \psi(z_3^*), \psi(z_4^*)\} > \psi(z_1^*)$, which means $z^* = z_1^* = s$.
(ii) As $s = \sqrt{1+2\kappa\eta}$, we obtain $\min\{\psi(z_2^*), \psi(z_4^*)\} > \psi(z_1^*) = \psi(z_3^*)$, which implies $z^* = z_1^* = s$ or $z^* = z_3^* = \frac{s}{1+2\kappa\eta}$.
(iii) As $s \in (0, \sqrt{1+2\kappa\eta})$, we obtain $\min\{\psi(z_1^*), \psi(z_2^*), \psi(z_4^*)\} > \psi(z_3^*)$, which means $z^* = z_3^* = \frac{s}{1+2\kappa\eta}$.
(vi) As $s \le 0$, we get $\min\{\psi(z_1^*), \psi(z_2^*), \psi(z_3^*)\} > \psi(z_4^*)$, which implies $z^* = z_4^* = s$.
Summarizing the above analysis, we obtain (10). □

The explicit expression (10) of ℓ_{ts} proximal operator will be used to prove the first-order necessary and sufficient conditions for the local optimal solution of L_{ts}-SVM in Theorem 3.1 and Theorem 3.2, and establish new efficient algorithm for solving L_{ts}-SVM in Section 4. Next, we give the definition of proximal operator of truncated squared hinge loss function for the multi-dimensional case, which is known as L_{ts} proximal operator.

Definition 2.3. [44, Definition 1.22] Given $\kappa, \eta > 0$, the proximal operator of $L_{ts}(\mathbf{u})$ at $\mathbf{s} = (s_1, \cdots, s_m)^\top \in \mathbb{R}^m$ is defined by

$$\text{prox}_{\kappa\eta L_{ts}}(\mathbf{s}) = \arg\min_{\mathbf{u}\in\mathbb{R}^m} \frac{1}{2}\|\mathbf{u}-\mathbf{s}\|^2 + \kappa\eta L_{ts}(\mathbf{u}).$$

Based on the above Definition 2.3, the explicit expression of L_{ts} proximal operator is given in Lemma 2.3.

Lemma 2.3. *Given $\kappa, \eta > 0$, the explicit expression of L_{ts} proximal operator at $\mathbf{s} = (s_1, s_2, \cdots, s_m)^\top \in \mathbb{R}^m$ can be shown as follows*

$$prox_{\kappa\eta L_{ts}}(\mathbf{s}) := \begin{bmatrix} prox_{\kappa\eta\ell_{ts}}(s_1) \\ \vdots \\ prox_{\kappa\eta\ell_{ts}}(s_m) \end{bmatrix} \in \mathbb{R}^m, \tag{11}$$

where $prox_{\kappa\eta\ell_{ts}}(s_i)$ gives in (10).

Proof. Based on the separate property of $L_{ts}(\mathbf{z})$, we get that $[\text{prox}_{\kappa\eta L_{ts}}(\mathbf{s})]_i = \text{prox}_{\kappa\eta\ell_{ts}}(s_i)$, where

$$\text{prox}_{\kappa\eta\ell_{ts}}(s_i) = \arg\min_{u\in\mathbb{R}} \kappa\eta\ell_{ts}(u) + \frac{1}{2}(u-s_i)^2, i \in [m].$$

Thus, we get (11). □

The explicit expression (11) of L_{ts} proximal operator will be used to study the proximal stationary points as stated in (12) and prove the first-order necessary and sufficient conditions for the local optimal solution of L_{ts}-SVM in Theorem 3.1 and Theorem 3.2. Moreover, the explicit expression (15) will be also used to establish new efficient algorithm for solving L_{ts}-SVM in Section 4.

3. Optimality theory of L_{ts}-SVM

From the perspective of optimization, developing the first-order necessary and sufficient optimality conditions of L_{ts}-SVM is a key step in theoretical analysis, since these conditions provide significant benefits for algorithmic design of L_{ts}-SVM. Therefore, we first use L_{ts} proximal operator in (11) to define the proximal stationary point of L_{ts}-SVM (6). Next, we will prove that the proximal stationary point is the first-order necessary condition for the local optimal solution of L_{ts}-SVM in Theorem 3.1. Finally, we will prove that the proximal stationary point is also the first-order sufficient condition for the local optimal solution of L_{ts}-SVM in Theorem 3.2. For this purpose, we begin with giving the definition of proximal stationary point of L_{ts}-SVM (6).

Definition 3.1. Given $\eta > 0$, the $(\mathbf{d}^*; \mathbf{w}^*; b^*)$ is a proximal stationary point of (6) if there is κ and $\boldsymbol{\beta}^*$ such that

$$\begin{cases} H^\top \boldsymbol{\beta}^* + \mathbf{w}^* &= \mathbf{0}, \\ \langle \mathbf{y}, \boldsymbol{\beta}^* \rangle &= 0, \\ \mathbf{d}^* + H\mathbf{w}^* + b^*\mathbf{y} &= \mathbf{e}, \\ \mathrm{prox}_{\kappa\eta L_{ts}}(\mathbf{d}^* - \kappa\boldsymbol{\beta}^*) &\ni \mathbf{d}^*. \end{cases} \tag{12}$$

Our next result further shows that the proximal stationary point is the first-order necessary condition for the local optimal solution of L_{ts}-SVM. To proceed more, we first introduce some notation, which will be used in Theorem 3.1. For any given $\eta > 0$ and $\mathbf{d}^* \in \mathbb{R}^m$, we define some notation as

$$\mathcal{S}^* := \{i \in [m] : d_i^* > 1\}, \qquad \mathcal{E}^* := \{i \in [m] : d_i^* = 1\},$$
$$\mathcal{T}^* := \{i \in [m] : d_i^* \in (0,1)\}, \ \mathcal{O}^* := \{i \in [m] : d_i^* \le 0\}.$$
$$\kappa_1^* := \begin{cases} \min \frac{(d_i^*)^2 - 1}{2\eta}, & i \in \mathcal{S}^*, \\ +\infty, & \mathcal{S}^* = \emptyset. \end{cases} \quad \kappa_2^* := \begin{cases} \min \frac{1 - (d_i^*)^2}{2\eta (d_i^*)^2}, & i \in \mathcal{T}^*, \\ +\infty, & \mathcal{T}^* = \emptyset. \end{cases}$$

Based on the above notation and proximal stationary point in (12), the following theorem reveals the relationship between a proximal stationary point and a local optimal solution of L_{ts}-SVM.

Theorem 3.1 *(First-order necessary condition). Given $\eta > 0$, suppose that $(\mathbf{d}^*; \mathbf{w}^*; b^*)$ is a local optimal solution of (6) and $\mathcal{E}^* = \emptyset$. Then $(\mathbf{d}^*; \mathbf{w}^*; b^*)$ is a proximal stationary point with $0 < \kappa \le \kappa^* := \{\kappa_1^*, \kappa_2^*\}$.*

Proof. Because $(\mathbf{d}^*; \mathbf{w}^*; b^*)$ is a local optimal solution of (6), from [44, Theorem 10.1], we obtain that there exists a $\boldsymbol{\beta}^*$ such that $(\mathbf{d}^*; \mathbf{w}^*; b^*)$ is a KKT point of (6), i.e.,

$$\begin{cases} \mathbf{w}^* + H^\top \boldsymbol{\beta}^* &= \mathbf{0}, \\ \langle \mathbf{y}, \boldsymbol{\beta}^* \rangle &= 0, \\ \mathbf{d}^* + H\mathbf{w}^* + b^*\mathbf{y} &= \mathbf{e}, \\ \boldsymbol{\beta}^* + \eta \partial L_{ts}(\mathbf{d}^*) &\ni \mathbf{0}. \end{cases} \tag{13}$$

Hence, to prove (12), we only need to prove that for any $0 < \kappa \le \kappa^*$, if $(\boldsymbol{\beta}^*; \mathbf{d}^*)$ satisfies $\mathbf{0} \in \boldsymbol{\beta}^* + \eta \partial L_{ts}(\mathbf{d}^*)$, then we obtain $\mathbf{d}^* \in \mathbb{P} := \mathrm{prox}_{\kappa\eta L_{ts}}(\mathbf{d}^* - \kappa\boldsymbol{\beta}^*)$. From (7), (8) and $\mathbf{0} \in \boldsymbol{\beta}^* + \eta \partial L_{ts}(\mathbf{d}^*)$, we have that

$$\beta_i^* \in \begin{cases} 0, & d_i^* > 1, \\ \{0, -2\eta d_i^*\}, & d_i^* = 1, \\ -2\eta d_i^*, & d_i^* \in (0,1), \\ 0, & d_i^* \le 0, \end{cases} \qquad i \in [m]. \tag{14}$$

To prove (12), because $\mathcal{E}^* = \emptyset$, we only consider the following three cases.
(i) For $i \in \mathcal{S}^*$, we get $d_i^* > 1$ and receive $\beta_i^* = 0$ from (14), which implies

$$p_i^* := d_i^* - \kappa\beta_i^* = d_i^* > 1. \tag{15}$$

From $\kappa \le \kappa_1^*$, we have

$$\kappa \le \frac{(d_i^*)^2 - 1}{2\eta} \overset{(15)}{=} \frac{(p_i^*)^2 - 1}{2\eta},$$

which means that

$$p_i^* \ge \sqrt{1 + 2\kappa\eta}.$$

(ii) For $i \in \mathcal{T}^*$, we have $d_i^* \in (0,1)$ and obtain $\beta_i^* = -2\eta d_i^* < 0$ by (14). Hence, we obtain

$$p_i^* = d_i^* - \kappa\beta_i^* = d_i^* + 2\kappa\eta d_i^* = d_i^*(1+2\kappa\eta) > 0. \tag{16}$$

From $\kappa \le \kappa_2^*$, we have

$$\kappa \le \frac{1-(d_i^*)^2}{2\eta(d_i^*)^2},$$

which means

$$2\kappa\eta \le \frac{1}{(d_i^*)^2} - 1.$$

Furthermore, we have

$$(d_i^*)^2(1+2\kappa\eta) \le 1,$$

which implies

$$d_i^*(1+2\kappa\eta) \le \sqrt{1+2\kappa\eta}.$$

This together with (16) results in

$$p_i^* = d_i^*(1+2\kappa\eta) \in (0, \sqrt{1+2\kappa\eta}].$$

(iii) For $i \in \mathcal{O}^*$, from (14), we obtain $d_i^* \le 0$ and $\beta_i^* = 0$, which leads to

$$p_i^* = d_i^* - \kappa\beta_i^* = d_i^* \le 0.$$

From the above (i)-(iii), we get that

$$d_i^* \in \mathrm{prox}_{\kappa\eta\ell_{ts}}(p_i^*) = \begin{cases} p_i^*, & p_i^* > \sqrt{1+2\kappa\eta}, \\ p_i^* \text{ or } \frac{p_i^*}{1+2\kappa\eta}, & p_i^* = \sqrt{1+2\kappa\eta}, \\ \frac{p_i^*}{1+2\kappa\eta}, & 0 < p_i^* < \sqrt{1+2\kappa\eta}, \\ p_i^*, & p_i^* \le 0. \end{cases} \quad \square$$

In Theorem 3.1, we established the first-order necessary condition for L_{ts}-SVM, i.e., a local optimal solution is a proximal stationary point for L_{ts}-SVM. Our next theorem further shows that a proximal stationary point is a local optimal solution for L_{ts}-SVM.

Theorem 3.2 *(First-order sufficient condition). Given $\kappa, \eta > 0$, suppose that $(\mathbf{d}^*; \mathbf{w}^*; b^*)$ with $\boldsymbol{\beta}^*$ is a proximal stationary point of (6). Then the $(\mathbf{d}^*; \mathbf{w}^*; b^*)$ is a local optimal solution of (6).*

Proof. Define $\boldsymbol{\varphi}^* := (\mathbf{d}^*; \mathbf{w}^*; b^*)$ and $\Psi := \{\boldsymbol{\varphi} := (\mathbf{d}; \mathbf{w}; b) : \mathbf{d} + H\mathbf{w} + b\mathbf{y} = \mathbf{e}\}$. Then, for any $\boldsymbol{\varphi} \in \Psi$, we get $\mathbf{d} + H\mathbf{w} + b\mathbf{y} = \mathbf{e}$, which combine with (12) results in

$$-H(\mathbf{w} - \mathbf{w}^*) = \mathbf{d} - \mathbf{d}^* + (b - b^*)\mathbf{y}. \tag{17}$$

By the convexity of $\|\mathbf{w}\|^2$, we have that

$$\begin{aligned} \|\mathbf{w}\|^2 - \|\mathbf{w}^*\|^2 &\ge 2\langle \mathbf{w} - \mathbf{w}^*, \mathbf{w}^* \rangle \\ &\overset{(12)}{=} -2\langle \boldsymbol{\beta}^*, H(\mathbf{w} - \mathbf{w}^*) \rangle \\ &\overset{(17)}{=} 2\langle \boldsymbol{\beta}^*, \mathbf{d} - \mathbf{d}^* \rangle + 2(b - b^*)\langle \boldsymbol{\beta}^*, \mathbf{y} \rangle \\ &\overset{(12)}{=} 2\langle \boldsymbol{\beta}^*, \mathbf{d} - \mathbf{d}^* \rangle. \end{aligned} \tag{18}$$

Next, we prove that $\boldsymbol{\varphi}^*$ is a local optimal solution of problem (6). Namely, there exists a neighborhood $N(\boldsymbol{\varphi}^*, \mu)$ of $\boldsymbol{\varphi}^* \in \Psi$ with $\mu > 0$ such that for any $\boldsymbol{\varphi} \in \Psi \cap N(\boldsymbol{\varphi}^*, \mu)$, we obtain

$$\frac{1}{2}\|\mathbf{w}\|^2 + \eta L_{ts}(\mathbf{d}) \ge \frac{1}{2}\|\mathbf{w}^*\|^2 + \eta L_{ts}(\mathbf{d}^*). \tag{19}$$

For this purpose, we define

$$\begin{aligned} \mu &:= \min\{1 - 1/\sqrt{1+2\kappa\eta}, \sqrt{1+2\kappa\eta} - 1\}, \\ \mu_m &:= \mu/\sqrt{2m}, \end{aligned} \tag{20}$$

and define a local region of $\boldsymbol{\varphi}^* = (\mathbf{d}^*; \mathbf{w}^*; b^*)$ as

$$N(\boldsymbol{\varphi}^*, \mu) := \{\boldsymbol{\varphi} : \|\boldsymbol{\varphi} - \boldsymbol{\varphi}^*\| \le \mu/\sqrt{2}, |d_i - d_i^*| \le \mu_m\}. \tag{21}$$

To show (19), we first define $\mathbf{s}^* := \mathbf{d}^* - \kappa\boldsymbol{\beta}^* \in \mathbb{N}_m$, and

$$\begin{cases} \mathcal{A}_1^* := \{i \in [m] : s_i^* \le 0\}, \\ \mathcal{A}_2^* := \{i \in [m] : s_i^* \in (0, \sqrt{1+2\kappa\eta})\}, \\ \mathcal{A}_3^* := \{i \in [m] : s_i^* = \sqrt{1+2\kappa\eta}, \beta_i^* \ne 0\}, \\ \mathcal{A}_4^* := \{i \in [m] : s_i^* = \sqrt{1+2\kappa\eta}, \beta_i^* = 0\}, \\ \mathcal{A}_5^* := \{i \in [m] : s_i^* > \sqrt{1+2\kappa\eta}\} \\ \Gamma_1^* = \mathcal{A}_1^*, \Gamma_2^* = \mathcal{A}_2^* \cup \mathcal{A}_3^*, \Gamma_3^* = \mathcal{A}_4^* \cup \mathcal{A}_5^*. \end{cases} \tag{22}$$

By (11) and (10), we get that $\mathbf{d}^* \in \mathrm{prox}_{\kappa\eta L_{ts}}(\mathbf{d}^* - \kappa\boldsymbol{\beta}^*)$ is identical to

$$\mathbf{d}^* = \begin{bmatrix} (\mathbf{d}^* - \kappa\boldsymbol{\beta}^*)_{\Gamma_1^*} \\ \frac{(\mathbf{d}^* - \kappa\boldsymbol{\beta}^*)_{\Gamma_2^*}}{1+2\kappa\eta} \\ (\mathbf{d}^* - \kappa\boldsymbol{\beta}^*)_{\Gamma_3^*} \end{bmatrix},$$

which is equivalent to

$$\boldsymbol{\beta}^*_{\Gamma_1^*} = \mathbf{0}_{\Gamma_1^*}, \quad \boldsymbol{\beta}^*_{\Gamma_2^*} = -2\eta\mathbf{d}^*_{\Gamma_2^*}, \quad \boldsymbol{\beta}^*_{\Gamma_3^*} = \mathbf{0}_{\Gamma_3^*}.$$

This combine with (22) leads to

$$\begin{aligned} &\beta_i^* = 0, d_i^* \le 0, i \ \in \Gamma_1^*, \\ &\beta_i^* = -2\eta d_i^*, d_i^* \in [0, 1/\sqrt{1+2\kappa\eta}], i \in \Gamma_2^*, \\ &\beta_i^* = 0, d_i^* \ge \sqrt{1+2\kappa\eta}, i \in \Gamma_3^*. \end{aligned} \tag{23}$$

Denote $\Gamma^* := \Gamma_2^*$ and $\overline{\Gamma}^* := \Gamma_1^* \cup \Gamma_3^*$. To show (19), by (18) and (23), we only need to prove the two results:

$$\eta L_{ts}(\mathbf{d}_{\Gamma^*}) - \eta L_{ts}(\mathbf{d}^*_{\Gamma^*}) + \langle \boldsymbol{\beta}^*_{\Gamma^*}, \mathbf{d}_{\Gamma^*} - \mathbf{d}^*_{\Gamma^*} \rangle \ge 0, \tag{24}$$

$$\eta L_{ts}(\mathbf{d}_{\overline{\Gamma}^*}) - \eta L_{ts}(\mathbf{d}^*_{\overline{\Gamma}^*}) \ge 0. \tag{25}$$

From (3) and (20)-(23), we get the desired conclusion:

(i) For $i \in \Gamma^*$, from $d_i^* \in [0, 1/\sqrt{1+2\kappa\eta}]$ and $|d_i - d_i^*| \le \mu_m$, we obtain $d_i^* - \mu_m \le d_i \le d_i^* + \mu_m < 1$, which means the $\eta L_{ts}(\mathbf{d}_{\Gamma^*})$ is a local convex function in $\Psi \cap N(\boldsymbol{\varphi}^*, \mu)$. Namely, for any $\boldsymbol{\varphi} \in \Psi \cap N(\boldsymbol{\varphi}^*, \mu)$, we get

$$\eta L_{ts}(\mathbf{d}_{\Gamma^*}) - \eta L_{ts}(\mathbf{d}^*_{\Gamma^*}) - \langle \eta\nabla L_{ts}(\mathbf{d}^*_{\Gamma^*}), \mathbf{d}_{\Gamma^*} - \mathbf{d}^*_{\Gamma^*} \rangle \ge 0,$$

which together with $-\eta\nabla L_{ts}(\mathbf{d}^*_{\Gamma^*}) = -2\eta d_i^* = \beta_i^*, i \in F_2^*$ from (23) allows us to obtain (24).

(ii) For $i \in \Gamma_1^*$, by $d_i^* \le 0$ and $|d_i - d_i^*| \le \mu_m$, we get $d_i \le d_i^* + \mu_m < 1$, which means $\ell_{ts}(d_i) \ge \ell_{ts}(d_i^*)$. Hence (25) holds.

(iii) For $i \in \Gamma_3^*$, it follows from $d_i^* \ge \sqrt{1+2\kappa\eta}$ and $|d_i - d_i^*| \le \mu_m$ that we have $d_i \ge d_i^* - \mu_m > 1$, which gets $\ell_{ts}(d_i) = \ell_{ts}(d_i^*)$. Hence, (25) holds.

From the above (i)-(iii), we have (24) and (25). □

In Theorem 3.2, we established the first-order sufficient condition for L_{ts}-SVM, i.e., a proximal stationary point must be a local optimal solution of L_{ts}-SVM without any assumptions, which is convenient for us to characterize support vector of L_{ts}-SVM and adopt the proximal stationary point as a termination rule of our new proposed algorithm in next part.

4. Fast algorithm of L_{ts}-SVM

We note that the decision hyperplane of L_{ts}-SVM is decided by its support vectors. Thus, we only need store the support vectors and can give up other nonsupport vectors. Hence, decreasing the number of support vectors will become very key for handling SVM problems in extremely large sizes. For this purpose, we first define the L_{ts} support vectors and working set strategy of L_{ts}-SVM based on the first-order optimality conditions in Section 3.

Theorem 4.1. *Given $\kappa, \eta > 0$, suppose that $(\mathbf{d}^*; \mathbf{w}^*; b^*)$ with $\boldsymbol{\beta}^* \in \mathbb{R}^m$ is a proximal stationary point of (6). Then we obtain*

$$\mathbf{w}^* = -\sum_{i \in \Gamma^*} \beta_i^* y_i \mathbf{x}_i \text{ and } \beta_i^* = 0, i \in \overline{\Gamma}^*, \tag{26}$$

where $\Gamma^* := \Gamma_2^*$ *in (22) and* $\overline{\Gamma}^* := \mathbb{N}_m \backslash \Gamma^*$. *The* $\{\mathbf{x}_i : i \in \Gamma^*\}$ *are dubbed as* L_{ts} *support vectors of (4) and meet*

$$y_i(\langle \mathbf{w}^*, \mathbf{x}_i \rangle + b^*) \in [1 - 1/\sqrt{1+2\kappa\eta}, 1], \ i \in \Gamma^*. \tag{27}$$

Therefore, the L_{ts} *support vectors are a small portion of the whole training set.*

Proof. Because $(\mathbf{d}^*; \mathbf{w}^*; b^*)$ is a proximal stationary point of (6), from Theorem 3.2 and (23), we get

$$\beta_i^* = -2\eta d_i^* \neq 0, i \in \Gamma^*, \quad \beta_i^* = 0, i \in \overline{\Gamma}^*.$$

By the first equation of (12) and $H = [y_1\mathbf{x}_1 \ \cdots \ y_m\mathbf{x}_m]^\top \in \mathbb{R}^{m\times n}$, we get that

$$\mathbf{w}^* = -H_{\Gamma^*}^\top \boldsymbol{\beta}_{\Gamma^*}^* - H_{\overline{\Gamma}^*}^\top \boldsymbol{\beta}_{\overline{\Gamma}^*}^* = -H_{\Gamma^*}^\top \boldsymbol{\beta}_{\Gamma^*}^* = -\sum_{i\in\Gamma^*} \beta_i^* y_i \mathbf{x}_i.$$

From (22) and (23), we obtain $d_i^* \in [0, 1/\sqrt{1+2\kappa\eta}], i \in \Gamma^*$, which combine with the third equation of (12) implies

$$y_i(\langle \mathbf{w}^*, \mathbf{x}_i \rangle + b^*) \in [1 - 1/\sqrt{1+2\kappa\eta}, 1], i \in \Gamma^*,$$

which means that the L_{ts} support vectors are a small portion of the whole training set and completes the proof. □

Theorem 4.1 proves that the L_{ts} support vectors is a small portion of the whole training set and the set Γ^* in (26) gives us a clue to pick L_{ts} support vectors, which is very practical in the following algorithmic design. Motivated by this, based on L_{ts} support vectors, we begin with introducing a working set of L_{ts}-SVM and then adopt the famous alternating direction method of multipliers with working set (L_{ts}-ADMM) for addressing L_{ts}-SVM.

4.1. L_{ts}-ADMM framework

In this part, we will adopt L_{ts}-ADMM for addressing the L_{ts}-SVM (6) and establish its global convergence and low computational complexity. We first provide the framework of ADMM as follows. The augmented Lagrangian function associated with the problem (6) is given by,

$$\mathcal{L}_\tau(\mathbf{w}; b; \mathbf{d}; \boldsymbol{\beta}) = \frac{1}{2}\|\mathbf{w}\|^2 + \eta L_{ts}(\mathbf{d}) + \langle \boldsymbol{\beta}, \mathbf{d} - \mathbf{e} + H\mathbf{w} + b\mathbf{y}\rangle + \frac{\tau}{2}\|\mathbf{d} - \mathbf{e} + H\mathbf{w} + b\mathbf{y}\|^2,$$

where $\tau > 0$ denotes given parameter and $\boldsymbol{\beta} \in \mathbb{R}^m$ stands for Lagrangian multiplier. Given $(\mathbf{w}^k; b^k; \mathbf{d}^k; \boldsymbol{\beta}^k)$, the framework of L_{ts}-ADMM is showed as below:

$$\begin{aligned}
\mathbf{d}^{k+1} &= \operatorname*{argmin}_{\mathbf{d}\in\mathbb{R}^m} \mathcal{L}_\tau(\mathbf{w}^k, b^k, \mathbf{d}, \boldsymbol{\beta}^k) \\
\mathbf{w}^{k+1} &= \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^n} \mathcal{L}_\tau(\mathbf{w}, b^k, \mathbf{d}^{k+1}, \boldsymbol{\beta}^k) + \tfrac{\tau}{2}\|\mathbf{w} - \mathbf{w}^k\|_{G_k}^2 \\
b^{k+1} &= \operatorname*{argmin}_{b\in\mathbb{R}} \mathcal{L}_\tau(\mathbf{w}^{k+1}, b, \mathbf{d}^{k+1}, \boldsymbol{\beta}^k) \\
\boldsymbol{\beta}^{k+1} &= \boldsymbol{\beta}^k + \rho\tau(\mathbf{d}^{k+1} - \mathbf{e} + H\mathbf{w}^{k+1} + b^{k+1}\mathbf{y}),
\end{aligned} \tag{28}$$

where $\rho \in (0, \frac{1+\sqrt{5}}{2})$ and $G_k \in \mathbb{R}^{m\times m}$. We define

$$\|\mathbf{w} - \mathbf{w}^k\|_{G_k}^2 := \langle \mathbf{w} - \mathbf{w}^k, G_k(\mathbf{w} - \mathbf{w}^k)\rangle.$$

To choose G_k, we define a working set of L_{ts}-SVM based on the L_{ts} support vectors of Theorem 4.1, which is denoted as Γ_k. We now define

$$\begin{aligned}
\mathbf{g}^k &:= \mathbf{d}^k - \boldsymbol{\beta}^k/\tau = \mathbf{e} - H\mathbf{w}^k - b^k\mathbf{y} - \boldsymbol{\beta}^k/\tau \\
\Gamma_k^1 &:= \{i \in [m] : g_i^k \in (0, \sqrt{1+2\eta/\tau})\}, \\
\Gamma_k^2 &:= \{i \in [m] : g_i^k = \sqrt{1+2\eta/\tau}, \beta_i^k \neq 0\}.
\end{aligned} \tag{29}$$

The working set Γ_k at the kth step is define as follows

$$\Gamma_k := \Gamma_k^1 \cup \Gamma_k^2, \ \ \overline{\Gamma}_k := [m]\backslash\Gamma_k. \tag{30}$$

Based on which, the G_k is selected as

$$G_k = -H_{\overline{\Gamma}_k}^\top H_{\overline{\Gamma}_k}. \tag{31}$$

Based on this, we update each sub-problem of (28) as below.

(i) Updating $\mathbf{d}^{k+1}$: The $\mathbf{d}$-subproblem of (28) is identical to

$$\begin{aligned}\mathbf{d}^{k+1} &= \underset{\mathbf{d}\in\mathbb{R}^m}{\operatorname{argmin}}\ \eta L_{ts}(\mathbf{d}) + \langle \boldsymbol{\beta}^k, \mathbf{d}\rangle + \frac{\tau}{2}\|\mathbf{d} - \mathbf{e} + H\mathbf{w}^k + b^k\mathbf{y}\|^2 \\ &= \underset{\mathbf{d}\in\mathbb{R}^m}{\operatorname{argmin}}\ \eta L_{ts}(\mathbf{d}) + \frac{\tau}{2}\|\mathbf{d} - (\mathbf{e} - H\mathbf{w}^k - b^k\mathbf{y} - \boldsymbol{\beta}^k/\tau)\|^2 \\ &= \underset{\mathbf{d}\in\mathbb{R}^m}{\operatorname{argmin}}\ \eta L_{ts}(\mathbf{d}) + \frac{\tau}{2}\|\mathbf{d} - \mathbf{g}^k\|^2 \\ &= \operatorname{Prox}_{\frac{\eta}{\tau}L_{ts}}(\mathbf{g}^k),\end{aligned}$$

where the last equation is from (11) with $\kappa = 1/\tau$. This combine with (11) and (30) derives

$$\mathbf{d}^{k+1}_{\Gamma_k} = \frac{\mathbf{g}^k_{\Gamma_k}}{\sqrt{1+2\kappa\eta}},\quad \mathbf{d}^{k+1}_{\overline{\Gamma}_k} = \mathbf{g}^k_{\overline{\Gamma}_k}. \tag{32}$$

(ii) Updating $\mathbf{w}^{k+1}$. The $\mathbf{w}$-subproblem of (28) is identical to

$$\mathbf{w}^{k+1} = \arg\min_{\mathbf{w}\in\mathbb{R}^n} \frac{1}{2}\|\mathbf{w}\|^2 + \frac{\tau}{2}\|\mathbf{w} - \mathbf{w}^k\|^2_{-H^\top_{\overline{\Gamma}_k}H_{\overline{\Gamma}_k}} + \langle \boldsymbol{\beta}^k, H\mathbf{w}\rangle + \frac{\tau}{2}\|\mathbf{d}^{k+1} - \mathbf{e} + H\mathbf{w} + b^k\mathbf{y}\|^2, \tag{33}$$

which is a convex problem. To deal with (33), we can receive the solution to

$$\mathbf{0} = \mathbf{w} - \tau H^\top_{\overline{\Gamma}_k}H_{\overline{\Gamma}_k}(\mathbf{w} - \mathbf{w}^k) + H^\top\boldsymbol{\beta}^k + \tau H^\top(\mathbf{d}^{k+1} - \mathbf{e} + H\mathbf{w} + b^k\mathbf{y}), \tag{34}$$

which is identical to yield the solution to the equation

$$(I + \tau H^\top_{\Gamma_k}H_{\Gamma_k})\mathbf{w} = \tau H^\top_{\Gamma_k}\boldsymbol{\varphi}^k_{\Gamma_k}, \tag{35}$$

where $\boldsymbol{\varphi}^k := -(\mathbf{d}^{k+1} + b^k\mathbf{y} - \mathbf{e} + \boldsymbol{\beta}^k/\tau)$. To derive (35) from (34), we use the fact that

$$H^\top_{\Gamma_k}H_{\Gamma_k} = H^\top H - H^\top_{\overline{\Gamma}_k}H_{\overline{\Gamma}_k}.$$

More precisely, (35) can be addressed efficiently by the following rules:

- If $n \le |\Gamma_k|$, we can handle (35) directly through

$$\mathbf{w}^{k+1} = (I + \tau H^\top_{\Gamma_k}H_{\Gamma_k})^{-1}\tau H^\top_{\Gamma_k}\boldsymbol{\varphi}^k_{\Gamma_k}. \tag{36}$$

- If $n > |\Gamma_k|$, the Sherman-Morrison-Woodbury formula [45] computes the inverse as

$$(I + \tau H^\top_{\Gamma_k}H_{\Gamma_k})^{-1} = I - \tau H^\top_{\Gamma_k}(I + \tau H_{\Gamma_k}H^\top_{\Gamma_k})^{-1}H_{\Gamma_k}.$$

  Thus we update $\mathbf{w}^{k+1}$ by

$$\mathbf{w}^{k+1} = \tau H^\top_{\Gamma_k}(I + \tau H_{\Gamma_k}H^\top_{\Gamma_k})^{-1}\boldsymbol{\varphi}^k_{\Gamma_k}. \tag{37}$$

(iii) Updating b^{k+1}. The b-subproblem of (28) is identical to

$$b^{k+1} = \arg\min_{b\in\mathbb{R}}\ \langle \boldsymbol{\beta}^k, b\mathbf{y}\rangle + \frac{\tau}{2}\|\mathbf{d}^{k+1} - \mathbf{e} + H\mathbf{w}^{k+1} + b\mathbf{y}\|^2,$$

which is addressed efficiently as follows

$$b^{k+1} = \langle \mathbf{y}, \boldsymbol{\zeta}^k\rangle/\|\mathbf{y}\|^2 = \langle \mathbf{y}, \boldsymbol{\zeta}^k\rangle/m, \tag{38}$$

where $\boldsymbol{\zeta}^k := -H\mathbf{w}^{k+1} + \mathbf{e} - \mathbf{d}^{k+1} - \boldsymbol{\beta}^k/\tau$.

(iv) Updating $\boldsymbol{\beta}^{k+1}$. We update $\boldsymbol{\beta}^{k+1}$ in (28) as follows

$$\boldsymbol{\beta}^{k+1}_{\Gamma_k} = \boldsymbol{\beta}^k_{\Gamma_k} + \rho\tau\boldsymbol{\chi}^{k+1}_{\Gamma_k},\quad \boldsymbol{\beta}^{k+1}_{\overline{\Gamma}_k} = \mathbf{0}, \tag{39}$$

where $\boldsymbol{\chi}^{k+1} := \mathbf{d}^{k+1} - \mathbf{e} + H\mathbf{w}^{k+1} + b^{k+1}\mathbf{y}$ and setting $\boldsymbol{\beta}^{k+1}_{\overline{\Gamma}_k} = \mathbf{0}$ follows form the (26).

Overall, updating each subproblem is summarized into Algorithm 1, which is named as L_{ts}-ADMM, an abbreviation for L_{ts}-SVM solved by ADMM with working set.

4.2. Global convergence

The following theorem claims that the L_{ts}-ADMM global converges to a local optimal solution of (6).

Algorithm 1 L_{ts}-ADMM for solving problem (6).

Initialize $(\mathbf{w}^0; b^0; \mathbf{d}^0; \boldsymbol{\beta}^0)$. Set $\eta, \rho, \tau, K > 0$ and $k = 0$.
while The stop condition does not hold and $k \le K$ **do**
  Compute Γ_k as in (30).
  Compute $\mathbf{d}^{k+1}$ by (32).
  Compute $\mathbf{w}^{k+1}$ by (36) if $n \le |\Gamma_k|$ and (37) otherwise.
  Compute b^{k+1} by (38).
  Compute $\boldsymbol{\beta}^{k+1}$ by (39).
  Set $k = k + 1$.
end while
return the final solution $(\mathbf{w}^k, b^k)$ to (6).

Theorem 4.2 *(Global convergence). Let $(\mathbf{w}^*; b^*; \mathbf{d}^*; \boldsymbol{\beta}^*)$ be the limit point of the sequence $\{(\mathbf{w}^k; b^k; \mathbf{d}^k; \boldsymbol{\beta}^k)\}$ generated by L_{ts}-ADMM. Then we yield that $(\mathbf{w}^*; b^*; \mathbf{d}^*)$ is a local optimal solution to (6).*

Proof. Because the $\Gamma_k \subseteq [m]$ has finite many elements, for $k \to \infty$, there exists a subset $T \subseteq \{1, 2, 3, \cdots\}$ such that

$$\Gamma_j \equiv: \Gamma, \quad \forall\, j \in T. \tag{40}$$

For convenience, we now define the sequence $\boldsymbol{\Upsilon}^k := (\mathbf{w}^k, b^k, \mathbf{d}^k, \boldsymbol{\beta}^k)$ and its limit point $\boldsymbol{\Upsilon}^* := (\mathbf{w}^*, b^*, \mathbf{d}^*, \boldsymbol{\beta}^*)$, i.e., $\{\boldsymbol{\Upsilon}^k\} \to \boldsymbol{\Upsilon}^*$. This also means $\{\boldsymbol{\Upsilon}^j\}_{j\in T} \to \boldsymbol{\Upsilon}^*$ and $\{\boldsymbol{\Upsilon}^{j+1}\}_{j\in T} \to \boldsymbol{\Upsilon}^*$. Taking the limit along with T of (39), i.e., $k \in T, k \to \infty$, we yield

$$\begin{cases} \boldsymbol{\beta}^*_{\Gamma} &= \boldsymbol{\beta}^*_{\Gamma} + \rho\tau\boldsymbol{\chi}^*_{\Gamma}, \\ \boldsymbol{\beta}^*_{\overline{\Gamma}} &= \mathbf{0}, \end{cases} \tag{41}$$

which contributes to $\boldsymbol{\chi}^*_{\Gamma} = \mathbf{0}$ because of $\boldsymbol{\chi}^* = \mathbf{d}^* + H\mathbf{w}^* + b^*\mathbf{y} - \mathbf{e}$. Taking the limit along with T of $\mathbf{g}^k$ in (29), we obtain

$$\begin{aligned} \mathbf{g}^* &= \mathbf{e} - H\mathbf{w}^* - b^*\mathbf{y} - \boldsymbol{\beta}^*/\tau \\ &= \mathbf{e} - H\mathbf{w}^* - b^*\mathbf{y} - \mathbf{d}^* + \mathbf{d}^* - \boldsymbol{\beta}^*/\tau \\ &= -\boldsymbol{\chi}^* + \mathbf{d}^* - \boldsymbol{\beta}^*/\tau \end{aligned} \tag{42}$$

Taking the limit along with T of (32), we result in

$$\mathbf{d}^*_{\Gamma} = \frac{\mathbf{g}^*_{\Gamma}}{\sqrt{1 + 2\kappa\eta}}, \quad \mathbf{d}^*_{\overline{\Gamma}} = \mathbf{g}^*_{\overline{\Gamma}}, \tag{43}$$

which results in

$$\begin{aligned} \mathbf{d}^*_{\overline{\Gamma}} &= \mathbf{g}^*_{\overline{\Gamma}} \\ &\overset{(42)}{=} -\boldsymbol{\chi}^*_{\overline{\Gamma}} + \mathbf{d}^*_{\overline{\Gamma}} - \boldsymbol{\beta}^*_{\overline{\Gamma}}/\tau \\ &\overset{(41)}{=} -\boldsymbol{\chi}^*_{\overline{\Gamma}} + \mathbf{d}^*_{\overline{\Gamma}}. \end{aligned} \tag{44}$$

This proves $\boldsymbol{\chi}^*_{\overline{\Gamma}} = \mathbf{0}$, which together with $\boldsymbol{\chi}^*_{\Gamma} = \mathbf{0}$ leads to $\boldsymbol{\chi}^* = \mathbf{0}$. Again from (42), we have $\mathbf{g}^* = \mathbf{d}^* - \boldsymbol{\beta}^*/\tau$, which combine with (43) and the (11) contributes to

$$\mathbf{d}^* \in \mathrm{Prox}_{\frac{\eta}{\tau}L_{ts}}(\mathbf{g}^*) = \mathrm{Prox}_{\frac{\eta}{\tau}L_{ts}}(\mathbf{d}^* - \boldsymbol{\beta}^*/\tau). \tag{45}$$

Now taking the limit along with T of (35) derives

$$\begin{aligned} (I + \tau H_{\Gamma}^{\top} H_{\Gamma})\mathbf{w}^* &= \tau H_{\Gamma}^{\top}\boldsymbol{\varphi}^*_{\Gamma} \\ &= -\tau H_{\Gamma}^{\top}(\mathbf{d}^*_{\Gamma} + b^*\mathbf{y}_{\Gamma} - \mathbf{e} + \boldsymbol{\beta}^*_{\Gamma}/\tau) \\ &= -\tau H_{\Gamma}^{\top}(\boldsymbol{\chi}^*_{\Gamma} - H_{\Gamma}\mathbf{w}^* + \boldsymbol{\beta}^*_{\Gamma}/\tau) \\ &= -\tau H_{\Gamma}^{\top}(-H_{\Gamma}\mathbf{w}^* + \boldsymbol{\beta}^*_{\Gamma}/\tau), \end{aligned}$$

where $\boldsymbol{\varphi}^* = -(\mathbf{d}^* + b^*\mathbf{y} - \mathbf{e} + \boldsymbol{\beta}^*/\tau)$ and the last two equations hold because of $\boldsymbol{\chi}^* = \mathbf{d}^* + H\mathbf{w}^* + b^*\mathbf{y} - \mathbf{e} = \mathbf{0}$ and $\boldsymbol{\chi}^*_{\Gamma} = \mathbf{0}$ by (41). The last equation obtains that

$$\mathbf{w}^* = -H_{\Gamma}^{\top}\boldsymbol{\beta}^*_{\Gamma} \overset{(41)}{=} -H^{\top}\boldsymbol{\beta}^*.$$

Finally, taking the limit along with T of (38) derives

$$b^* = \langle \mathbf{y}, \boldsymbol{\zeta}^* \rangle / m = -\langle \mathbf{y}, H\mathbf{w}^* - \mathbf{e} + \mathbf{d}^* + \boldsymbol{\beta}^*/\tau \rangle / m$$

$$
\begin{aligned}
&= -\langle \mathbf{y}, \chi^* - b^*\mathbf{y} + \boldsymbol{\beta}^*/\tau\rangle/m \\
&= -\langle \mathbf{y}, -b^*\mathbf{y} + \boldsymbol{\beta}^*/\tau\rangle/m \\
&= b^* - \langle \mathbf{y}, \boldsymbol{\beta}^*\rangle/(m\tau),
\end{aligned}
$$

which yields $\langle \mathbf{y}, \boldsymbol{\beta}^*\rangle = 0$ and the three equation holds due to $\chi^* = 0$. Summarizing the above analysis, we get

$$
\begin{cases}
\mathbf{w}^* + H^\top \boldsymbol{\beta}^* &= \mathbf{0}, \\
\langle \mathbf{y}, \boldsymbol{\beta}^*\rangle &= \mathbf{0}, \\
\mathbf{d}^* + H\mathbf{w}^* + b^*\mathbf{y} &= \mathbf{e}, \\
\mathrm{prox}_{\frac{\eta}{\tau} L_{ts}}(\mathbf{d}^* - \boldsymbol{\beta}^*/\tau) &\ni \mathbf{d}^*,
\end{cases}
$$

which proves that the $(\mathbf{d}^*; \mathbf{w}^*; b^*)$ is a proximal stationary point with $\kappa = 1/\tau$ of problem (6). From Theorem 3.2, the $(\mathbf{d}^*; \mathbf{w}^*; b^*)$ is a local optimal solution to the problem (6). □

The above theorem proves that if the sequence generated by L_{ts}-ADMM has a limit point, then it must be a proximal stationary point and also a locally optimal solution to (9), which implies that we can adopt the proximal stationary point as a termination rule in terms of guaranteeing the local optimality of a point generated by L_{ts}-ADMM in next section.

4.3. Low computational complexity

For the computational complexity of L_{ts}-ADMM, we have some the following comments:

- For computing Γ_k by (30), the complexity is $\mathcal{O}(m)$.
- For updating $\mathbf{d}^{k+1}$ by (32), its dominant term is $H\mathbf{w}^k$, which uses the complexity about $\mathcal{O}(mn)$.
- For calculating $\mathbf{w}^{k+1}$, we compute (36) if $n \le |\Gamma_k|$ and (37) otherwise. For updating $\mathbf{w}^{k+1}$ by (36), the dominant computations are form

  $$(I + \tau H_{\Gamma_k}^\top H_{\Gamma_k})^{-1} \quad \text{and} \quad H_{\Gamma_k}^\top H_{\Gamma_k}.$$

  The computational complexity of the former is $\mathcal{O}(n^\mu)$, where $\mu \in (2, 3)$ and the computational complexity of the latter is $\mathcal{O}(n^2|\Gamma_k|)$. For updating $\mathbf{w}^{k+1}$ by (37), the main computations are calculating

  $$(I + \tau H_{\Gamma_k} H_{\Gamma_k}^\top)^{-1} \quad \text{and} \quad H_{\Gamma_k} H_{\Gamma_k}^\top.$$

  The computational complexity of the former is $\mathcal{O}(|\Gamma_k|^\mu)$, where $\mu \in (2, 3)$ and the computational complexity of the latter is $\mathcal{O}(n|\Gamma_k|^2)$. Hence, the complexity of updating $\mathbf{w}^{k+1}$ in each step is

  $$\mathcal{O}(\min\{n^2, |\Gamma_k|^2\}\max\{n, |\Gamma_k|\}).$$

- For updating b^{k+1} by (38), the $H\mathbf{w}^{k+1}$ is the most expensive computation, which computational complexity is $\mathcal{O}(mn)$.
- Same as that of updating b^{k+1}, the computational complexity of updating $\boldsymbol{\beta}^{k+1}$ by (39) is $\mathcal{O}(mn)$.

Overall, the whole computational complexity in Algorithm 1 is

$$\mathcal{O}\left(mn + \min\{n^2, |\Gamma_k|^2\}\max\{n, |\Gamma_k|\}\right),$$

which means that the L_{ts}-ADMM possesses a considerably low computational complexity if the number of picked working set $|\Gamma_k|$ or n is very small (i.e., $\max\{|\Gamma_k|, n\} \ll m$).

5. Numerical experiments

This part conducts extensive numerical experiments of L_{ts}-ADMM, against eight leading solvers in Table 3 that are programmed by MATLAB in Table 3. For simplicity, all of these parameters are set to their default values. All the solvers are carried out in MATLAB (R2018b) on a laptop with 32 GB memory and an 2.7 GHz CPU Inter® Core i7-9880H. Inspired by Theorem 3.2 and Theorem 4.2, we take advantage of the proximal stationary point as a stopping criterion in the experiments. In other words, we will terminate L_{ts}-ADMM if the iteration point $(\mathbf{w}^k; b^k; \mathbf{d}^k; \boldsymbol{\beta}^k)$ closely meets (12). Namely,

$$\max\{\xi_1^k, \xi_2^k, \xi_3^k, \xi_4^k\} < \epsilon,$$

where ϵ is the tolerance level and

$$\xi_1^k := \frac{\|\mathbf{w}^k + H_{\Gamma_k}^\top \boldsymbol{\beta}_{\Gamma_k}^k\|}{1 + \|\mathbf{w}^k\|}, \quad \xi_2^k := \frac{|\langle \mathbf{y}_{\Gamma_k}, \boldsymbol{\beta}_{\Gamma_k}^k\rangle|}{1 + |\Gamma_k|},$$

Table 2
The details of real datasets, where the m_1 and m_2 stands for the numbers of total training data and testing data are respectively. The n stands for the feature of training data. The m and m_t stands for the numbers of training data and testing data respectively.

Datasets	Abbreviation	m_1	m_2	m	m_t	n
Lekemia	leke	38	34	38	34	7129
Colon-cancer	colo	62	0	54	8	2000
Australian	aust	690	0	621	69	14
Splice	spli	1000	2175	1000	2175	60
A5a	a5a	6414	26147	6414	26147	123
Two-norm	twon	7400	0	6600	740	20
Mushrooms	mush	8124	0	7311	813	112
W5a	w5a	9888	39861	9888	39861	300
A6a	a6a	11220	21341	11220	21341	123
A7a	a7a	16100	16461	16100	16461	123
W6a	w6a	17188	32561	17188	32561	300
Adult	adul	17887	0	16098	1789	13
A8a	a8a	22696	9865	22696	9865	123
W7a	w7a	24692	25057	24692	25057	300
A9a	a9a	32561	16281	32561	16281	123
W8a	w8a	49749	14951	49749	14951	300
Ijcnn1	ijcn	49990	91701	49990	91701	22
Malware analysis datasets	mada	51959	0	46763	5196	1024
Hospital readmissions binary	hrbn	59557	0	53601	5956	17
Airline passenger satisfaction	apsa	103904	25976	103904	25976	22
Santander customer transaction	sctr	$2e^5$	0	$18e^4$	$2e^4$	200
Skin_nonskin	skin	245056	0	220500	24506	3
Credit card fraud detection	ccfd	284807	0	256326	28481	28
Covtype.binary	covt	581012	0	522910	58102	54
Susy	susy	$5e^6$	0	$45e^5$	$5e^5$	18
Hepmass	hepm	$7e^6$	$35e^6$	$7e^6$	$35e^6$	28
Higgs	higg	$11e^6$	0	$99e^5$	$11e^5$	28

$$\xi_3^k := \frac{\|\mathbf{d}^k - \mathbf{e} + H\mathbf{w}^k + b^k\mathbf{y}\|}{\sqrt{m}}, \quad \xi_4^k := \frac{\|\mathbf{d}^k - \mathrm{prox}_{\frac{\eta}{\tau}L_{ts}}(\mathbf{d}^k - \boldsymbol{\beta}^k/\tau)\|}{1 + \|\mathbf{d}^k\|}.$$

5.1. Testing examples

This part discusses four testing examples, which is given as follows.

Example 5.1 *(Synthetic data in $\mathbb{R}^2$ without outliers [15,17])*. Firstly, for positive labels $y_i = +1$, features $\mathbf{x}_i$ are come from $U(\boldsymbol{\pi}_1, \Omega_1)$ and for negative labels $y_j = -1$, features $\mathbf{x}_j$ are come from $U(\boldsymbol{\pi}_2, \Omega_2)$, where

$$\Omega_1 = \Omega_2 = \begin{bmatrix} 0.2 & 0 \\ 0 & 3 \end{bmatrix}, \boldsymbol{\pi}_1 = \begin{bmatrix} 0.5 \\ -3 \end{bmatrix}, \boldsymbol{\pi}_2 = \begin{bmatrix} -0.5 \\ 3 \end{bmatrix}.$$

We then produce $2m$ samples with two classes having equal numbers. Finally, we evenly split them into a training and a testing samples set.

Example 5.2 *(Synthetic data in $\mathbb{R}^2$ with outliers)*. Based on Example 5.1, we first get $2m$ samples. We then randomly flip θm labels in each class. Namely, the $2\theta m$ outliers are generated, where the θ stands for the flapping ratio. Finally, we evenly divide the $2m$ samples into a training samples set and a testing samples set.

Example 5.3 *(Real data without outliers)*. We select 27 real datasets from the libraries: kaggle,[1] libsvm[2] and uci,[3] which are presented in Table 2. For all datasets, the labels being not 1 are treated as -1 and features are scaled to $[-1, 1]$. For the datasets without the testing data, we perform 10-fold cross validation.

Example 5.4 *(Real data with outliers)*. We pick six real datasets with small or moderate sizes in Table 2, which are colo, aust, adul, twon, mush and a6a. For the first five real datasets, we also perform 10-fold cross validation. Finally, θ percentage of training and testing samples are flipped their labels, which are called outliers.

[1] https://www.kaggle.com/datasets.
[2] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/.
[3] http://archive.ics.uci.edu/ml/datasets.php.

Table 3
Benchmark solvers.

Solvers	References	Models
TSVM	Our paper	SVM with truncated squared hinge loss
HSVM	[13]	SVM with hinge loss
SSVM	[21]	SVM with squared hinge loss
PSVM	[16]	SVM with pinball loss
USVM	[25]	SVM with huberized pinball loss
LORE	[46]	Logistic regression
SLRE	[47]	Sparse logistic regression
RSVM	[37]	SVM with truncated pinball loss
ASVM	[36]	SVM with ramp loss

Table 4
Selection of parameters.

	Cases	τ	η
Example 5.1	$m \le 3 * 10^4$	1	0.5
	$m > 3 * 10^4$	1	1
Example 5.2	$\theta \le 0.1$	0.5	1
	$\theta > 0.1$	1	1
Example 5.3	higg, susy, hepm	2	0.25
	a5a-a9a	0.25	0.5
	other	0.25	1
Example 5.4	a6a	0.25	0.5
	other	0.5	0.5

We present five evaluation indicators to measure the performances of one solver, where the TACC denotes training classification accuracy and the ACC stands for testing classification accuracy. In addition, the TIME stands for training time, the SWS/ITER represents the number of working set of per iteration and NSV stands for the number of support vectors. Especially, the TACC and ACC represent the identification rate. The SWS/ITER is the mean value of the number of working set of per iteration, which stands for the low computational cost of one solver. The TIME is the mean value of training time. It is widely known that the classifier of SVM is dominated by its support vectors, thus the smaller NSV implies better performance of one solver. In addition, the ACC, TACC, NSV and SWS/ITER are defined by

$$\texttt{TACC} := 100\% \times \left[1 - \frac{1}{\overline{m}}\|\text{sign}(\langle \widetilde{\mathbf{w}}, \overline{\mathbf{x}}\rangle + \widetilde{b}) - \overline{y}\|_0\right],$$

$$\texttt{ACC} := 100\% \times \left[1 - \frac{1}{\widetilde{m}}\|\text{sign}(\langle \widetilde{\mathbf{w}}, \widetilde{\mathbf{x}}\rangle + \widetilde{b}) - \widetilde{y}\|_0\right],$$

$$\texttt{SWS/ITER} := \frac{\sum_{k=1}^{\widehat{k}} |\Gamma_k|}{\widehat{k}}, \quad \texttt{NSV} := |\Gamma_{\widehat{k}}|,$$

where the $(\widetilde{\mathbf{w}}, \widetilde{b})$ represents the solution get by one solver, the $\{(\overline{\mathbf{x}}_j, \overline{y}_j) : j = 1, \cdots, \overline{m}\}$ ($\{(\widetilde{\mathbf{x}}_j, \widetilde{y}_j) : j = 1, \cdots, \widetilde{m}\}$) stands for the training (testing) set. We define $\text{sign}(\widehat{a}) = 1$ if $\widehat{a} > 0$ and $\text{sign}(\widehat{a}) = -1$ otherwise. $|\Gamma_k|$ denotes the number of working set Γ_k and $\widehat{k}$ stands for the total number of iterations. $\|c\|_0$ stands for the zero norm of c. The smaller TIME, SWS/ITER and NSV (or the larger ACC and TACC) stands for the better performance.

5.2. Numerical comparisons

In this part, we will conduct numerical tests on various data to show the performance of L_{ts}-ADMM. Firstly, we will show the performance of L_{ts}-ADMM from four aspects. Then, we compare L_{ts}-ADMM with eight state-of-the-art solvers for synthetic data. Finally, we compare L_{ts}-ADMM with eight state-of-the-art solvers for real data.

5.2.1. Performance test of our algorithm L_{ts}-ADMM

(i) Parameters setting of L_{ts}-ADMM. In our algorithm, we initialize L_{ts}-ADMM with $\mathbf{d}^0 = \boldsymbol{\beta}^0 = \mathbf{0}, \mathbf{w}^0 = \mathbf{e}/100$ and $b^0 = 0$. Parameters are set as $K = 1000$, $\rho = 1.618$ and $\epsilon = 10^{-3}$. For our developed algorithm L_{ts}-ADMM, the two parameters η and τ are trained by the standard 10-fold cross validation, where τ is picked from $\{2^{-7}, 2^{-6}, \cdots, 2^7\}$ and η is selected from $\{2^{-7}, 2^{-6}, \cdots, 2^7\}$. In the following numerical experiments, to avoid using different parameters for different datasets, we pick parameters η and τ as specified in Table 4.

(ii) Performance test of L_{ts}-ADMM. To show the performance of L_{ts}-ADMM, we give the variation diagram of training accuracy (ACC), number of working set (Γ_k), training time (TIME) and tolerance level (ϵ) with the increase of the number of iterations (k)

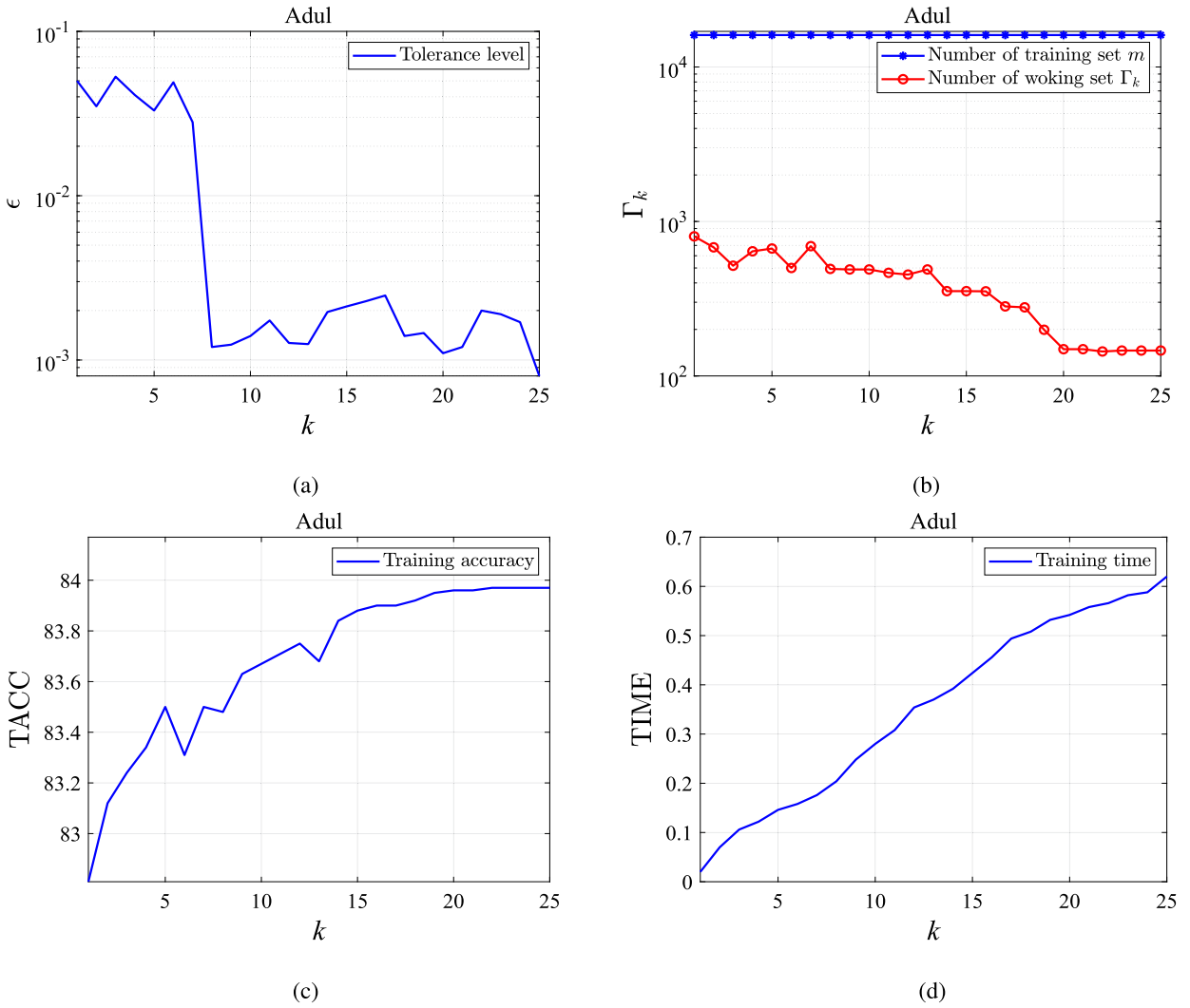

Fig. 2. The influence of the number of iterations (k) for tolerance level (ϵ), number of working set (Γ_k), training accuracy (TACC) and training time (TIME) of our algorithm L_{ts}-ADMM on dataset Adul (the number of training set is $m = 16098$).

on dataset Adul in Fig. 2. Moreover, we also test other synthetic and real datasets, and omit similar observations. We can get the following four results from these observations.

1) (*Global convergence*) From Fig. 2(a), we can see that our algorithm L_{ts}-ADMM global converges to proximal stationary point with 25 steps, which indicates the global converge of L_{ts}-ADMM. Theorem 3.2 states that the proximal stationary point is a locally optimal solution of the L_{ts}-SVM. Hence, L_{ts}-ADMM at least global converges to a locally optimal solution of L_{ts}-SVM.

2) (*Low computational complexity and working set*) From Fig. 2(b), as the increase of the number of iterations, the number of working set (Γ_k) is always small and tends to be stable. In particular, the number of working set ($|\Gamma_k| < 800$) is significantly smaller than the total number of training set $m = 16098$ and the value of ratio ($|\Gamma_k|/m$) is even less than 5% at some iterations, which verifies that L_{ts}-ADMM possesses a considerably low computational complexity since the number of picked working set $|\Gamma_k|$ is very small.

3) (*Training accuracy*). From Fig. 2(c), as the increase of the number of iterations, the TACC lines are increased for $k < 20$ and stabilized at a certain level for $k \geq 20$, which indicates that L_{ts}-ADMM can stable achieve the highest training classification accuracy. This further verifies our claim that L_{ts}-ADMM at least global converges to a locally optimal solution of L_{ts}-SVM.

4) (*Training time*) From Fig. 2(d) and Table 9, we obtain that our algorithm L_{ts}-ADMM gets a very good time performance mainly for three reasons. i) The working set selection strategy makes us using very few number of working set for each iteration, which reduces the computational complexity of our algorithm. ii) For w-subproblem of L_{ts}-ADMM, we use the advanced matrix inversion technique in (36) and (37), which further reduces the computational complexity of our algorithm. iii) We adopt the proximal stationary point as a stopping criterion in the experiments, which ensures that our algorithm converges to the optimal solution by using few iterative steps.

Table 5
Results of 9 solvers for dealing with Example 5.1, where $n = 2$ and m is number of training data.

m	TACC (%)								
	TSVM	HSVM	SSVM	PSVM	USVM	LORE	SLRE	RSVM	RSVM
10000	**97.28**	**97.28**	97.24	**97.28**	97.22	97.25	97.25	97.25	97.25
20000	**97.33**	**97.33**	97.30	97.30	97.26	97.30	97.29	97.27	97.24
30000	**97.37**	**97.37**	97.34	97.33	**97.37**	97.32	97.35	97.32	97.35
40000	**97.27**	97.25	97.24	97.21	96.23	97.22	97.23	97.21	97.20
50000	**97.24**	**97.24**	97.19	97.19	**97.24**	**97.24**	97.23	97.20	97.23
	ACC (%)								
10000	**97.31**	97.28	97.30	97.28	97.26	97.28	97.29	97.27	97.28
20000	**97.35**	97.32	97.31	97.33	97.30	97.32	97.33	97.30	97.32
30000	**97.28**	**97.28**	**97.28**	97.27	97.26	97.25	97.27	97.26	97.27
40000	**97.35**	97.33	97.30	**97.35**	**97.35**	97.31	97.32	**97.35**	**97.35**
50000	**97.33**	97.28	97.28	97.30	97.31	97.30	97.30	97.31	97.32
	NSV								
10000	**99**	667	756	10000	10000	10000	10000	874	583
20000	**133**	1234	1432	20000	20000	20000	20000	1690	1002
30000	**156**	1899	1879	30000	30000	30000	30000	2019	1532
40000	**178**	2632	2678	40000	40000	40000	40000	2893	1876
50000	**191**	3212	3190	50000	50000	50000	50000	3414	2106
	SWS/ITER								
10000	**165**	1345	1379	10000	10000	10000	10000	1678	845
20000	**221**	2432	2671	20000	20000	20000	20000	3028	1651
30000	**332**	4001	4782	30000	30000	30000	30000	4893	2844
40000	**456**	5231	5765	40000	40000	40000	40000	5778	4031
50000	**585**	5443	3190	50000	50000	50000	50000	3414	2890
	TIME (s)								
10000	**0.031**	0.102	0.242	0.094	0.089	0.093	0.084	0.075	0.112
20000	**0.055**	0.288	0.558	0.287	0.249	0.312	0.322	0.388	0.942
30000	**0.083**	0.597	0.914	0.626	0.434	0.654	0.736	0.631	1.103
40000	**0.094**	1.432	1.213	0.985	0.795	0.932	0.969	0.888	1.334
50000	**0.121**	2.311	1.886	1.066	0.997	1.334	1.694	1.067	1.893

5.2.2. Comparisons with synthetic data

(i) Comparisons for Example 5.1. We begin with using nine solvers to address Example 5.1 with different training and testing sample numbers $m \in \{10000, \cdots, 50000\}$ and $n = 2$. As shown in Table 5, we record the average results of nine solvers over 50 times. For TACC and ACC, it is observed that all solvers get desirable TACC and ACC, and TSVM presents the highest TACC and ACC, which means that the TSVM is effective to solve truncated squared hinge loss SVM. When it comes to NSV, we can see that PSVM, USVM, LORE and SLRE use all samples as the support vectors, while others possess a small number of the support vectors. Apparently, TSVM shares a considerably small number of the support vectors. With respect to SWS/ITER, TSVM utilizes a very small portion of samples as the working set. This is not surprising since it uses a relatively small number of support vectors. For TIME, TSVM consumes the shortest TIME, which shows that TSVM is capable of solving L_{ts}-SVM and the computation time of our algorithm is quite small.

(ii) Comparisons for Example 5.2. Now, we would like to see the robustness of all solvers for dealing with Example 5.2. We set $m = 10000, n = 2$ and alter the flapping ratio $\theta \in \{0, 0.05, \cdots, 0.2\}$. We report the average results over 50 times in Table 6. As for TACC and ACC, it is clearly observed that TSVM have more robust to outliers than the others since TSVM receives slightly better TACC and ACC, which demonstrates that TSVM is competitive with other solvers in terms of training and testing classification accuracy. When it comes to NSV and SWS/ITER, we obtain similar observations to that in Table 5. With θ rising, the more samples become support vectors for HSVM, SSVM and RSVM. By contrary, TSVM and ASVM have a small NSV with θ rising. Because of this, TSVM again runs the fastest, which means that TSVM is competitive with other solvers in terms of training time.

5.2.3. Comparisons with real data

(i) Comparisons for Example 5.3. For Example 5.3, we report the average result over 50 times in Table 7, Table 8 and Table 9, where "**" stands for that the results are not received if a solver demands a large memory. For TACC and ACC, it is evident that TSVM outperforms the other solvers with respect to the highest TACC and ACC. Taking col as an instance, USVM, SLRE, RSVM and ASVM get less than 87% correct training classification accuracy. By contrary, our TSVM receives more than 90% correct training classification accuracy. This is probably due to the smallest number of support vectors used. In terms of NSV and SWS/ITER, since TSVM receives the smallest NSV and SWS/ITER, which proves that our proposed working set strategy is very effective for reducing the cost of per iteration. Therefore, TSVM outperforms the eight other solvers. As for TIME, the TSVM runs super fast for dealing with all datasets. Such as, for data higg, TSVM only needs 17.78 seconds, while HSVM uses 561.3 seconds. In summary, TSVM achieves the highest training and testing classification accuracy while running rapidly for large-scale datasets. This is achieved by using a relatively small number of working sets.

Table 6
Results of 9 solvers for dealing with Example 5.2, where we set training data $m = 10000$ and $n = 2$. The θ stands for the noise ratio.

θ	TACC (%)								
	TSVM	HSVM	SSVM	PSVM	USVM	LORE	SLRE	RSVM	ASVM
0.00	**97.28**	**97.28**	97.24	**97.28**	97.22	97.25	97.25	97.25	97.25
0.05	**92.78**	92.75	**92.78**	**92.78**	92.65	92.57	92.46	92.35	97.75
0.10	**88.08**	**88.08**	88.01	88.06	88.10	87.99	87.98	88.05	88.06
0.15	**83.54**	83.53	83.48	**83.54**	83.46	83.44	83.50	**83.54**	83.53
0.20	**78.47**	**78.47**	78.38	78.37	78.38	78.40	78.46	78.43	78.46
	ACC (%)								
0.00	**97.31**	97.28	97.30	97.28	97.26	97.28	97.29	97.27	97.28
0.05	**92.57**	92.55	92.62	92.56	92.63	92.47	92.49	92.45	92.55
0.10	**88.06**	**88.06**	88.03	88.01	88.04	**88.06**	87.99	**88.06**	88.04
0.15	**83.44**	**83.44**	83.38	83.37	83.41	**83.44**	83.40	**83.44**	**83.44**
0.20	**78.34**	78.28	78.28	78.27	78.30	78.29	78.26	78.30	**78.34**
	NSV								
0.00	**99**	667	756	10000	10000	10000	10000	874	583
0.05	**86**	1543	1389	10000	10000	10000	10000	1567	576
0.10	**84**	2772	2126	10000	10000	10000	10000	2290	542
0.15	**77**	3168	2889	10000	10000	10000	10000	3289	483
0.20	**72**	3446	3178	10000	10000	10000	10000	3892	447
	SWS/ITER								
0.00	**165**	1345	1379	10000	10000	10000	10000	1678	845
0.05	**155**	1543	1449	10000	10000	10000	10000	1674	823
0.10	**143**	2564	2357	10000	10000	10000	10000	2347	798
0.15	**132**	2653	2966	10000	10000	10000	10000	3348	776
0.20	**124**	3019	3765	10000	10000	10000	10000	3795	685
	TIME (s)								
0.00	**0.031**	0.102	0.242	0.094	0.089	0.093	0.084	0.075	0.112
0.05	**0.025**	0.123	0.099	0.087	0.099	0.098	0.085	0.094	0.108
0.10	**0.023**	0.132	0.104	0.089	0.093	0.096	0.093	0.098	0.102
0.15	**0.022**	0.138	0.113	0.093	0.096	0.097	0.099	0.088	0.107
0.20	**0.023**	0.144	0.116	0.102	0.110	0.103	0.109	0.097	0.105

(ii) Comparisons for Example 5.4. Finally, we employ nine solvers to address the real datasets in Example 5.4 with changing the flapping ratio $\theta = 0.05, 0.1$ to see their robustness to outliers. We report the average results over 50 independent times in Table 10. With regard to TACC and ACC, TACC and ACC received by nine solvers decline with the ascending of θ, and TSVM gets the highest TACC and ACC. With regard to NSV, PSVM, USVM, LORE and SLRE always select all samples as support vectors. But, TSVMand ASVM either decline or stabilize at a level, which implies that they are quite robust to θ, namely robust to the outliers. In addition, we also can see that TSVM always shares the fewest NSV. In terms of SWS/ITER, TSVM stabilizes at a level for all real datasets with θ rising. With respect to TIME, TSVM outperforms the other solvers for all datasets. These means that our algorithm has better performance and more robust to outliers than other solvers.

6. Conclusion

In this paper, we constructed a new SVM model with the truncated squared hinge loss function: L_{ts}-SVM, which well enjoys robustness and sparsity. To conquer the truncated squared hinge loss SVM, we have developed optimality conditions through the so-called proximal stationary point. These optimality conditions have facilitated to develop a new and efficient method, which shares relatively low computational complexity and highly efficient numerical performance with respect to more desirable accuracy, shorter computational time, fewer support vectors and more robust to outliers, particularly for datasets on large scales.

There are some important issues for the future research. The first issue is that the suggested loss function is not smooth, and not clear whether this really gives any advantage. It may be possible to create a smooth function with similar shape, and there are no examples of it amongst the observed ones, to compare and to show that this is really worse. The another issue is that there is no answer now whether in kernel form the advantage of the developed theoretical results and the proposed algorithm will be hold. We leave these as future research.

CRediT authorship contribution statement

Huajun Wang: Conceptualization, Investigation, Methodology, Software, Writing – original draft. **Genghui Li:** Data curation, Formal analysis, Supervision, Validation, Visualization. **Zhenkun Wang:** Project administration, Writing – review & editing.

Table 7
Results of the testing classification accuracy (ACC) and the training classification accuracy (TACC) of 9 solvers for dealing with Example 5.3.

Datasets	TACC (%)								
	TSVM	HSVM	SSVM	PSVM	USVM	LORE	SLRE	RSVM	ASVM
colo	**90.55**	87.99	87.69	86.64	87.18	86.43	87.12	86.58	86.58
aust	**86.28**	**86.28**	**86.28**	86.01	85.82	85.68	85.31	86.03	86.25
leke	**82.32**	82.31	82.12	82.25	82.17	82.15	82.12	82.17	82.28
spli	**88.97**	**88.97**	**88.97**	88.44	87.97	87.87	88.47	88.23	**88.97**
twon	**98.47**	97.99	97.97	98.17	98.15	98.25	98.11	98.26	98.38
mush	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
adul	**83.97**	83.83	**83.97**	83.45	83.70	83.70	83.49	83.70	**83.97**
a5a	**85.20**	84.98	84.99	84.92	84.92	84.92	84.77	84.98	84.98
a6a	**84.89**	84.68	84.21	**84.89**	84.66	**84.89**	84.21	84.77	84.68
a7a	**84.89**	84.79	84.77	84.57	84.56	84.78	84.67	84.59	84.79
a8a	**84.75**	84.73	84.67	**84.75**	**84.75**	84.70	84.73	84.71	84.71
a9a	**84.92**	**84.92**	84.87	84.83	84.91	84.58	84.78	84.77	84.88
w5a	**98.12**	98.08	98.08	97.68	98.03	98.00	97.92	98.01	98.08
w6a	**97.91**	97.79	97.80	97.18	97.62	97.68	97.51	97.23	97.69
w7a	**98.36**	98.34	**98.36**	97.95	98.17	98.15	98.12	97.95	**98.36**
w8a	**98.72**	**98.72**	98.52	98.44	98.47	98.47	98.67	98.33	98.33
ijcn	**94.54**	94.14	93.97	**94.54**	94.24	94.26	94.02	94.24	94.24
mada	**95.08**	95.01	94.92	94.87	94.88	94.89	94.77	94.69	95.01
hrbn	**87.71**	87.57	87.17	87.34	87.59	87.56	87.49	87.62	87.45
apsa	**87.59**	87.46	87.07	87.34	87.49	87.41	87.29	87.40	87.44
sctr	**91.08**	90.88	90.99	90.72	90.72	91.05	**91.05**	90.78	90.99
skin	**92.88**	92.67	92.21	92.61	92.86	92.67	92.21	92.57	92.62
ccfd	**99.95**	99.92	99.89	99.57	99.58	99.89	99.27	99.59	99.78
covt	**72.56**	72.44	71.55	**	**	**	**	**	**
susy	**78.91**	76.89	75.78	**	**	**	**	**	**
hepm	**83.64**	79.87	**	**	**	**	**	**	**
higg	**65.76**	63.98	**	**	**	**	**	**	**
Datasets	ACC (%)								
	TSVM	HSVM	SSVM	PSVM	USVM	LORE	SLRE	RSVM	ASVM
colo	**89.85**	87.88	86.89	86.45	87.06	86.08	87.09	86.77	86.77
aust	**86.14**	**86.14**	86.10	**86.14**	85.62	85.67	85.31	**86.14**	**86.14**
leke	**82.38**	82.23	82.32	82.25	82.27	82.21	82.14	82.17	82.17
spli	**88.88**	87.99	**88.88**	88.17	88.27	88.37	88.46	**88.88**	**88.88**
twon	**98.37**	**98.37**	98.17	98.15	98.10	98.20	98.21	98.20	98.19
mush	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
adul	**84.96**	84.76	83.98	**84.96**	84.68	84.73	84.49	83.93	83.94
a5a	**85.21**	84.98	85.11	84.98	84.99	84.99	84.97	85.05	85.09
a6a	**84.93**	84.91	84.81	84.78	84.58	84.58	84.57	84.69	84.69
a7a	**84.92**	84.88	84.86	84.77	84.44	84.77	84.86	84.65	84.77
a8a	**84.74**	84.71	**84.74**	84.67	**84.74**	84.58	84.56	**84.74**	84.59
a9a	**84.93**	84.88	84.68	84.86	84.90	84.68	84.88	84.87	84.89
w5a	**98.05**	**98.05**	**98.05**	97.76	**98.05**	98.01	97.97	98.02	98.01
w6a	**97.89**	97.86	97.67	97.58	97.67	97.58	97.66	97.63	97.79
w7a	**98.37**	98.27	97.98	97.97	98.23	98.21	98.14	97.99	98.08
w8a	**98.72**	98.54	98.61	**98.72**	98.54	98.44	98.69	98.46	98.66
ijcn	**94.56**	94.51	94.17	94.18	94.37	94.36	94.22	94.33	94.51
mada	**95.03**	94.98	94.88	94.57	94.74	94.79	94.66	94.68	94.88
hrbn	**87.82**	**87.82**	87.47	87.55	87.71	87.16	87.19	87.60	87.72
apsa	**87.84**	87.81	87.87	87.86	87.76	87.68	87.56	87.48	87.66
sctr	**91.25**	91.22	91.11	90.12	90.12	91.13	91.08	90.18	90.19
skin	**92.97**	92.88	92.28	92.78	92.66	92.77	92.81	92.67	92.72
ccfd	**99.98**	99.87	99.90	99.87	99.88	99.78	99.87	99.89	99.89
covt	**72.34**	72.22	71.15	**	**	**	**	**	**
susy	**78.91**	76.89	75.66	**	**	**	**	**	**
hepm	**83.75**	78.67	**	**	**	**	**	**	**
higg	**65.67**	63.55	**	**	**	**	**	**	**

Declaration of competing interest

We declare that we have no financial and personal relationships with other people or organizations that can inappropriately influence our work, there is no professional or other personal interest of any nature or kind in any product, service and/or company that could be construed as influencing the position presented in, or the review of, the manuscript entitled.

Table 8
Results of the number of working set per iteration (SWS/ITER) and the number of support vectors (NSV) of 9 solvers for dealing with Example 5.3.

Datasets	NSV								
	TSVM	HSVM	SSVM	PSVM	USVM	LORE	SLRE	RSVM	ASVM
colo	**34**	37	43	54	54	54	54	48	37
aust	**26**	332	217	621	621	621	621	301	267
leke	**29**	36	34	37	38	38	38	37	35
spli	**62**	823	692	1000	1000	1000	1000	698	746
twon	**51**	956	1124	6600	6600	6600	6600	1205	821
mush	**112**	1123	1154	7311	7311	7311	7311	1256	929
adul	**146**	8109	8896	16098	16098	16098	16098	9024	6321
a5a	**256**	3321	3492	3414	6414	6414	6414	3021	2787
a6a	**374**	5642	6025	11220	11220	11220	11220	6298	4128
a7a	**467**	6214	6981	16100	16100	16100	16100	6948	5417
a8a	**545**	5674	6931	22696	22696	22696	22696	7355	4288
a9a	**597**	8123	8817	32561	32561	32561	32561	9813	6421
w5a	**412**	2421	3055	9888	9888	9888	9888	3466	2015
w6a	**431**	5765	6038	17188	17188	17188	17188	6002	4336
w7a	**643**	5884	5891	24692	24692	24692	24692	6893	4129
w8a	**758**	6624	7059	49749	49749	49749	49749	7789	5317
ijcn	**331**	11764	11978	49990	49990	49990	49990	11980	9989
mada	**756**	10654	9974	46763	46743	46763	46763	10932	8943
hrbn	**777**	10897	10256	54601	53601	53601	53601	10984	9066
apsa	**884**	18562	17887	103904	103904	103904	103904	16211	15673
sctr	**962**	27778	29873	180000	180000	180000	180000	38887	23193
skin	**998**	27854	30557	220500	220500	220500	220500	28790	23152
ccfd	**876**	30563	32876	256326	256326	256326	256326	31992	28769
covt	**421**	41887	49078	**	**	**	**	**	**
susy	**756**	166989	189032	**	**	**	**	**	**
hepm	**1107**	238865	**	**	**	**	**	**	**
higg	**1578**	567582	**	**	**	**	**	**	**
Datasets	SWS/ITER								
	TSVM	HSVM	SSVM	PSVM	USVM	LORE	SLRE	RSVM	ASVM
colo	**34**	37	43	54	54	54	54	48	36
aust	**65**	212	187	621	621	621	621	256	188
leke	**31**	35	35	38	38	38	38	37	35
spli	**121**	489	397	1000	1000	1000	1000	478	397
twon	**165**	712	794	6600	6600	6600	6600	815	687
mush	**787**	516	734	7311	7311	7311	7311	825	489
adul	**645**	6241	5782	16098	16098	16098	16098	6542	5793
a5a	**598**	1564	1892	6414	6414	6414	6414	2422	1352
a6a	**612**	4354	4562	11220	11220	11220	11220	5642	3998
a7a	**743**	4654	5782	16100	16100	16100	16100	5843	4439
a8a	**854**	4782	5431	22696	22696	22696	22696	6035	4502
a9a	**961**	6453	7853	32561	32561	32561	32561	8826	6033
w5a	**622**	1445	1762	9888	9888	9888	9888	2078	1224
w6a	**976**	4123	4892	17188	17188	17188	17188	4597	3877
w7a	**1189**	5452	5891	24692	24692	24692	24692	6893	5044
w8a	**1354**	4667	6034	49749	49749	49749	49749	6524	4172
ijcn	**898**	8453	8842	49990	49990	49990	49990	7889	7732
mada	**1324**	7352	8804	46763	46743	46763	46763	8712	6893
hrbn	**2014**	8768	9012	54601	53601	53601	53601	9256	8023
apsa	**1234**	12431	11679	103904	103904	103904	103904	12453	10673
sctr	**1886**	21089	24563	180000	180000	180000	180000	34582	19897
skin	**2413**	23452	24098	220500	220500	220500	220500	15489	21946
ccfd	**1989**	21234	25998	256326	256326	256326	256326	26771	20814
covt	**1796**	33215	39078	**	**	**	**	**	**
susy	**3234**	107684	137842	**	**	**	**	**	**
hepm	**3543**	156732	**	**	**	**	**	**	**
higg	**3927**	478654	**	**	**	**	**	**	**

Data availability

No data was used for the research described in the article.

Table 9
Results of the training time (TIME) of 9 solvers for dealing with Example 5.3.

Datasets	TIME								
	TSVM	HSVM	SSVM	PSVM	USVM	LORE	SLRE	RSVM	ASVM
colo	**0.035**	0.077	0.079	0.084	0.089	0.096	0.099	0.063	0.078
aust	**0.022**	0.212	0.441	0.873	0.778	0.714	0.665	0.572	0.209
leke	**0.056**	0.216	0.778	0.567	0.662	0.457	0.365	0.762	0.223
spli	**0.076**	0.534	0.987	1.142	1.225	0.932	0.985	1.231	0.678
twon	**0.069**	0.456	0.889	1.223	1.445	1.089	1.002	1.357	0.554
mush	**0.092**	0.412	0.995	1.087	1.563	1.661	1.438	1.653	0.664
adul	**0.663**	3.278	4.441	4.567	5.932	4.889	4.857	5.034	3.513
a5a	**0.134**	4.023	6.789	6.989	6.887	7.032	7.005	6.414	4.442
a6a	**0.214**	8.038	7.125	10.20	10.78	10.98	10.56	8.678	8.312
a7a	**0.315**	7.996	9.810	9.167	9.991	11.24	11.29	7.678	8.043
a8a	**0.412**	9.213	10.25	13.67	14.38	20.55	20.97	11.56	9.554
a9a	**0.613**	11.12	10.65	26.77	21.88	23.78	24.65	14.87	10.324
w5a	**0.077**	1.003	1.543	2.675	3.089	3.762	3.778	1.324	1.112
w6a	**0.356**	7.123	8.082	14.78	17.69	17.82	17.88	16.89	7.032
w7a	**0.746**	10.11	15.87	26.98	27.89	27.64	28.55	17.89	10.432
w8a	**2.723**	38.32	41.78	42.55	44.85	44.98	45.61	43.21	38.669
ijcn	**0.678**	35.66	27.43	25.78	40.52	41.59	42.39	18.66	35.226
mada	**4.342**	46.33	55.77	55.94	56.78	60.67	65.87	49.31	46.778
hrbn	**3.513**	42.18	48.98	60.66	60.89	62.55	67.43	50.22	41.898
apsa	**6.776**	84.22	90.17	135.7	143.8	141.5	154.3	87.22	90.335
sctr	**8.891**	97.62	112.3	245.2	213.7	222.3	235.8	176.3	112.89
skin	**8.556**	111.4	140.7	500.65	484.2	489.3	487.3	267.3	157.83
ccfd	**9.243**	198.3	254.3	1043	1128	1235	1674	439.2	895.89
covt	**4.342**	265.8	299.9	**	**	**	**	**	**
susy	**12.11**	377.9	388.5	**	**	**	**	**	**
hepm	**14.22**	422.5	**	**	**	**	**	**	**
higg	**17.78**	561.3	**	**	**	**	**	**	**

Acknowledgements

This work is supported by the Changsha Municipal Natural Science Foundation (kq2208214), the Scientific Research Fund of Hunan Provincial Education Department (22C0152), the National Natural Science Foundation of China (62206120, 62106096), Shenzhen Technology Plan (JCYJ20220530113013031), and the National Natural Science Foundation of China (11971052, 11871183).

References

[1] C. Cortes, V.N. Vapnik, Support vector networks, Mach. Learn. 20 (3) (1995) 273–297.
[2] X.Y. Pang, Y. Zhang, Y.T. Xu, A novel multi-task twin-hypersphere support vector machine for classification, Inf. Sci. 598 (2022) 37–56.
[3] Z. Wang, Y.H. Shao, L. Bai, L.M. Liu, N.Y. Deng, Insensitive stochastic gradient twin support vector machines for large scale problems, Inf. Sci. 462 (2018) 114–131.
[4] H.J. Wang, Y.H. Shao, S.L. Zhou, C. Zhang, N.H. Xiu, Support vector machine classifier via $L_{0/1}$ soft-margin loss, IEEE Trans. Pattern Anal. Mach. Intell. 40 (10) (2022) 7253–7265.
[5] D.B. Niu, C.J. Wang, P.P. Tang, Q.S. Wang, E. Song, An efficient algorithm for a class of large-scale support vector machines exploiting hidden sparsity, IEEE Trans. Signal Process. 99 (2022) 1–16.
[6] H.M. Wang, Y.T. Xu, Scaling up twin support vector regression with safe screening rule, Inf. Sci. 465 (2018) 174–190.
[7] H.J. Wang, Y.H. Shao, Fast truncated Huber loss SVM for large scale classification, Knowl.-Based Syst. 26 (2022) 1–17.
[8] S.L. Zhou, Sparse SVM for sufficient data reduction, IEEE Trans. Pattern Anal. Mach. Intell. 44 (9) (2022) 5560–5571.
[9] W.C. Wu, Y.T. Xu, X.Y. Pang, A hybrid acceleration strategy for nonparallel support vector machine, Inf. Sci. 546 (2021) 543–558.
[10] X.L. Pan, Y.T. Xu, A safe reinforced feature screening strategy for lasso based on feasible solutions, Inf. Sci. 477 (2019) 132–147.
[11] Z. Akram-Ali-Hammouri, M. Fernandez-Delgado, E. Cernadas, S. Barro, Fast support vector classification for large-scale problems, IEEE Trans. Pattern Anal. Mach. Intell. 44 (10) (2022) 6184–6195.
[12] P. Chen, R. Fan, C. Lin, A study on SMO-type decomposition methods for support vector machines, IEEE Trans. Neural. Netw. Learn. 17 (4) (2006) 893–908.
[13] Y.Q. Yan, Q.N. Li, An efficient augmented Lagrangian method for support vector machine, Optim. Methods Softw. 35 (4) (2021) 855–883.
[14] C. Hsieh, K. Chang, C. Lin, S.S. Keerthi, S. Sundararajan, A dual coordinate descent method for large-scale linear SVM, in: Proc. 25th Int. Conf. Mach. Learn., 2008, pp. 408–415.
[15] X.L. Huang, L. Shi, J.A.K. Suykens, Support vector machine classifier with pinball loss, IEEE Trans. Pattern Anal. Mach. Intell. 36 (5) (2014) 984–997.
[16] M. Tanveer, S. Sharma, R. Rastogi, P. Anand, Sparse support vector machine with pinball loss, Trans. Emerg. Telecommun. Technol. 32 (2) (2022) 1–13.
[17] X.L. Huang, L. Shi, J.A.K. Suykens, Solution path for pin-SVM classifiers with positive and negative τ values, IEEE Trans. Neural. Netw. Learn. 28 (7) (2017) 1584–1593.
[18] V. Jumutc, X.L. Huang, J.A.K. Suykens, Fixed-size pegasos for hinge and pinball loss SVM, in: Proc. IEEE Int. Joint Conf. Neural Netw., 2013, pp. 1–7.
[19] X.L. Huang, L. Shi, J.A.K. Suykens, Sequential minimal optimization for SVM with pinball loss, Neurocomputing 149 (2015) 1596–1603.
[20] J. Park, Y. Choi, J. Byun, J. Lee, S. Park, Support vector classifier for multi-class classification, Inf. Sci. 619 (2023) 889–907.
[21] J. Yin, Q.N. Li, A semismooth Newton method for support vector classification and regression, Comput. Optim. Appl. 73 (2) (2019) 477–508.
[22] Z. Allen-Zhu, Katyusha: the first direct acceleration of stochastic gradient methods, J. Mach. Learn. Res. 18 (221) (2018) 1–51.
[23] K. Chang, C. Hsieh, C. Lin, Coordinate descent method for large-scale L2-loss linear support vector machines, J. Mach. Learn. Res. 9 (2008) 1369–1398.
[24] H.M. Wang, Y.T. Xu, A safe double screening strategy for elastic net support vector machine, Inf. Sci. 582 (2022) 382–397.

Table 10
Results of 9 solvers for dealing with Example 5.4, where the θ stands for the noise ratio.

Datasets	TACC (%)								
	θ	TSVM	HSVM	SSVM	PSVM	USVM	LORE	SLRE	RSVM	ASVM
colo	0.05	**85.78**	85.64	85.68	85.47	85.28	85.38	85.68	85.19	**85.78**
	0.1	**80.73**	80.58	80.04	80.28	80.18	80.23	80.02	80.40	80.38
aust	0.05	**84.76**	84.66	84.35	84.55	84.35	84.45	84.47	84.32	84.34
	0.1	**81.27**	81.21	80.18	80.47	80.44	80.33	81.07	80.52	80.54
twon	0.05	**95.18**	94.98	95.10	95.20	95.07	95.04	95.04	94.97	94.88
	0.1	**88.88**	88.78	88.56	88.67	88.64	88.77	88.57	88.89	88.86
mush	0.05	**95.09**	95.08	95.12	94.98	95.01	94.90	95.19	95.04	95.07
	0.1	**90.23**	89.20	89.36	90.02	90.02	89.78	89.79	90.09	90.18
a6a	0.05	**82.44**	82.29	82.24	82.34	81.91	82.07	82.06	82.21	82.13
	0.1	**78.67**	78.45	78.43	78.37	77.97	77.89	78.22	78.65	78.58
adul	0.05	**81.89**	81.82	81.49	81.66	81.13	80.56	81.06	81.27	81.88
	0.1	**77.34**	77.24	77.13	77.18	77.12	76.77	77.16	77.09	77.28
Datasets	ACC (%)									
	θ	TSVM	HSVM	SSVM	PSVM	USVM	LORE	SLRE	RSVM	ASVM
colo	0.05	**85.52**	85.35	85.34	85.27	85.23	85.34	85.24	85.12	85.38
	0.1	**80.91**	80.89	80.24	80.33	80.35	80.47	80.12	80.55	80.90
aust	0.05	**84.89**	84.67	84.65	84.46	84.72	84.33	84.45	84.61	84.77
	0.1	**81.56**	81.51	80.32	80.43	80.35	80.34	81.11	80.12	80.44
twon	0.05	**95.44**	95.29	95.14	95.17	95.17	95.14	95.22	95.08	95.28
	0.1	**88.89**	88.44	88.11	88.21	88.56	88.66	88.39	88.34	88.79
mush	0.05	**95.43**	95.31	95.12	95.08	95.11	94.98	95.21	95.12	95.30
	0.1	**90.45**	90.26	89.96	90.77	90.34	89.98	89.98	89.89	90.28
a6a	0.05	**82.55**	82.32	82.12	82.41	81.41	82.34	82.41	82.32	82.48
	0.1	**78.78**	78.48	78.66	78.67	78.27	78.29	78.24	78.15	78.67
adul	0.05	**81.99**	81.68	81.54	81.55	81.65	80.66	81.73	81.57	81.87
	0.1	**77.55**	77.43	77.16	77.28	77.14	76.97	77.24	77.13	77.46
Datasets	NSV									
	θ	TSVM	HSVM	SSVM	PSVM	USVM	LORE	SLRE	RSVM	ASVM
colo	0.05	**32**	41	45	54	54	54	54	46	37
	0.1	**33**	43	46	54	54	54	54	48	36
aust	0.05	**27**	324	445	621	621	621	621	562	248
	0.1	**24**	377	689	621	621	621	621	531	224
twon	0.05	**135**	1065	1130	6600	6600	6600	6600	1256	816
	0.1	**129**	1342	1578	6600	6600	6600	6600	1356	799
mush	0.05	**756**	1143	1156	7311	7311	7311	7311	1253	902
	0.1	**723**	1335	1564	7311	7311	7311	7311	1678	861
a6a	0.05	**632**	5121	7854	11220	11220	11220	11220	6784	4061
	0.1	**621**	6785	7789	11220	11220	11220	11220	7725	3862
adul	0.05	**1199**	7632	8213	16098	16098	16098	16098	8194	6089
	0.1	**1094**	8012	8867	16098	16098	16098	16098	8613	5542
Datasets	SWS/ITER									
	θ	TSVM	HSVM	SSVM	PSVM	USVM	LORE	SLRE	RSVM	ASVM
colo	0.05	**32**	44	48	54	54	54	54	48	36
	0.1	**35**	47	49	54	54	54	54	49	35
aust	0.05	**31**	434	525	621	621	621	621	547	173
	0.1	**48**	478	543	621	621	621	621	574	168
twon	0.05	**277**	1555	1771	6600	6600	6600	6600	1436	642
	0.1	**195**	1789	1897	6600	6600	6600	6600	1836	595
mush	0.05	**992**	1671	2087	7311	7311	7311	7311	1964	452
	0.1	**902**	1876	2013	7311	7311	7311	7311	2267	415
a6a	0.05	**775**	6512	8045	11220	11220	11220	11220	7284	3583
	0.1	**754**	7342	8214	11220	11220	11220	11220	7987	3184
adul	0.05	**1889**	8786	8672	16098	16098	16098	16098	8576	5082
	0.1	**1702**	8908	8998	16098	16098	16098	16098	8897	4683
Datasets	TIME(s)									
	θ	TSVM	HSVM	SSVM	PSVM	USVM	LORE	SLRE	RSVM	ASVM
colo	0.05	**0.033**	0.081	0.086	0.087	0.091	0.097	0.102	0.073	0.075
	0.1	**0.038**	0.089	0.089	0.088	0.090	0.098	0.101	0.079	0.071
aust	0.05	**0.017**	0.223	0.478	0.877	0.779	0.712	0.669	0.589	0.204
	0.1	**0.019**	0.214	0.499	0.875	0.778	0.717	0.673	0.623	0.201

Table 10 (*continued*)

Datasets	TIME(s)								
	θ	TSVM	HSVM	SSVM	PSVM	USVM	LORE	SLRE	RSVM	ASVM
twon	0.05	**0.072**	0.567	0.912	1.232	1.456	1.091	1.011	1.369	0.539
	0.1	**0.063**	0.555	0.968	1.228	1.444	1.091	1.009	1.390	0.512
mush	0.05	**0.083**	0.467	1.289	1.088	1.576	1.667	1.445	1.768	0.613
	0.1	**0.079**	0.512	1.445	1.089	1.598	1.669	1.467	1.879	0.601
a6a	0.05	**0.263**	7.339	7.722	10.24	10.79	10.93	10.59	8.904	8.132
	0.1	**0.287**	7.321	7.928	10.26	10.82	10.95	10.58	9.123	7.998
adul	0.05	**0.856**	3.925	4.841	4.558	5.943	4.892	4.822	5.567	3.442
	0.1	**0.852**	3.872	5.233	4.577	5.934	4.897	4.828	5.981	3.226

[25] W.X. Zhu, Y.Y. Song, Y.Y. Xiao, Support vector machine classifier with huberized pinball loss, Eng. Appl. Artif. Intell. 91 (2022) 1–16.
[26] Q. Wang, Z. Liu, T. Zhang, H. Alasmary, M. Waqas, Z. Halim, Y. Li, Deep convolutional cross-connected kernel mapping support vector machine based on select dropout, Inf. Sci. 626 (2023) 694–709.
[27] H.J. Wang, Y.H. Shao, N.H. Xiu, Proximal operator and optimality conditions for ramp loss SVM, Optim. Lett. 16 (3) (2022) 999–1014.
[28] X.L. Huang, L. Shi, J.A.K. Suykens, Ramp loss linear programming support vector machine, J. Mach. Learn. Res. 15 (2014) 2185–2211.
[29] J.P. Brooks, Support vector machines with the ramp loss and the hard margin loss, Oper. Res. 59 (2) (2011) 467–479.
[30] B. Liu, R. Huang, Y. Xiao, J. Liu, K. Wang, L. Li, Q. Chen, Adaptive robust Adaboost-based twin support vector machine with universum data, Inf. Sci. 609 (2022) 1334–1352.
[31] Y.C. Wu, Y.F. Liu, Robust truncated hinge loss support vector machines, J. Am. Stat. Assoc. 102 (479) (2007) 974–983.
[32] X.Y. Pang, J. Zhao, Y.T. Xu, A novel ramp loss-based multi-task twin support vector machine with multi-parameter safe acceleration, Neural Netw. 150 (2022) 194–212.
[33] H.R. Wang, Y.T. Xu, Z.J. Zhou, Ramp loss KNN-weighted multi-class twin support vector machine, Soft Comput. 26 (14) (2022) 6591–6618.
[34] Y. Sun, S. Ding, L. Guo, Z. Zhang, Hypergraph regularized semi-supervised support vector machine, Inf. Sci. 591 (2022) 400–421.
[35] X.T. Shen, G.C. Tseng, X.G. Zhang, W.H. Wong, On ψ-learning, J. Am. Stat. Assoc. 98 (463) (2003) 724–734.
[36] X.M. Xi, X.L. Huang, J.A.K. Suykens, S.N. Wang, Coordinate descent algorithm for ramp loss linear programming support vector machines, Neural Process. Lett. 43 (2022) 887–903.
[37] H.R. Wang, Y.T. Xu, Z.J. Zhou, Twin-parametric margin support vector machine with truncated pinball loss, Neural Comput. Appl. 33 (8) (2021) 3781–3798.
[38] X. Shen, L.F. Niu, Z.Q. Qi, Y.J. Tian, Support vector machine classifier with truncated pinball loss, Pattern Recognit. 68 (2017) 199–210.
[39] H.J. Wang, Y.H. Shao, Sparse and robust SVM classifier for large scale classification, Appl. Intell. (2023), https://doi.org/10.1007/s10489-023-04511-w.
[40] S.Y. Park, Y.F. Liu, Robust penalized logistic regression with truncated loss functions, Can. J. Stat. 39 (2) (2011) 300–323.
[41] L.M. Yang, H.W. Dong, Support vector machine with truncated pinball loss and its application in pattern recognition, Chemom. Intell. Lab. Syst. 177 (2018) 89–99.
[42] Y.L. Feng, Y. Yang, X.L. Huang, S. Mehrkanoon, J.A.K. Suykens, Robust support vector machines for classification with nonconvex and smooth losses, Neural Comput. 28 (6) (2016) 1217–1247.
[43] R.M. Pattanayak, H.S. Behera, S. Panigrahi, A novel high order hesitant fuzzy time series forecasting by using mean aggregated membership value with support vector machine, Inf. Sci. 626 (2023) 494–523.
[44] R.T. Rockafellar, R.J. Wets, Variational Analysis, Springer Science and Business Media, 2009.
[45] G. Golub, C.F. Van-Loan, Matrix Computations, Johns Hopkins University Press, 1996.
[46] S.L. Zhou, N.H. Xiu, H.D. Qi, Global and quadratic convergence of newton hard-thresholding pursuit, J. Mach. Learn. Res. 22 (2021) 1–45.
[47] R. Wang, N.H. Xiu, S.L. Zhou, An extended newton-type algorithm for L2-regularized sparse logistic regression and its efficiency for classifying large-scale datasets, J. Comput. Appl. Math. (2022), https://doi.org/10.1016/j.cam.2022.113656 (online).