Knowledge-Based Systems 260 (2023) 110074

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

Fast truncated Huber loss SVM for large scale classification

Huajun Wang [a], Yuanhai Shao [b,*]

[a] Department of Mathematics and Statistics, Changsha University of Science and Technology, Changsha, PR China
[b] School of Management, Hainan University, Haikou, PR China

ARTICLE INFO

Article history:
Received 10 August 2022
Received in revised form 22 October 2022
Accepted 23 October 2022
Available online 5 November 2022

Keywords:
Truncated Huber loss
L_{th} proximal operator
L_{th} support vectors
Working set
L_{th}-ADMM

ABSTRACT

Support vector machine (SVM), as a useful tool of classification, has been widely applied in many fields. However, it may incur computationally infeasibility on very large sample datasets. To handle this problem, this paper presents a novel sparse and robust SVM model with truncated Huber loss, which is known as L_{th}-SVM. To this end, we establish a first-order optimality condition of L_{th}-SVM based on the newly introduced P-stationary point, which allows us to define L_{th} support vectors and working set of L_{th}-SVM and proves that all of the L_{th} support vectors are a small portion of the whole training set. Furthermore, we develop a novel efficient alternating direction method of multipliers with working set (dubbed as L_{th}-ADMM) to address L_{th}-SVM, which is proven that sequence produced by L_{th}-ADMM is proved to be a local minimizer of L_{th}-SVM and enjoys a relatively low computational complexity if the size of the working set is very small. We compare L_{th}-ADMM with other nine state-of-the-art solvers on both synthetic and real datasets. The extensive numerical experiments demonstrate that L_{th}-ADMM is capable of delivering higher prediction accuracy, presenting a smaller number of support vectors and running quickly, especially, in large-scale datasets setting.

1. Introduction

Support vector machine (SVM), as one of effective and popular classification tools, was first developed by Cortes and Vapnik [1] and has been extensively applied in many fields, such as disease detection [2–4], pattern recognition [5–7], machine learning [8–10] and so on. In the paper, we focus on a binary classification problem, which is given as below. Given m pairs of training data $\{(\mathbf{x}_i, y_i) : i = 1, 2, \ldots, m\}$, where $\mathbf{x}_i \in \mathbb{R}^n$ denotes the sample vector and $y_i \in \{-1, 1\}$ denotes class labels. The goal is to learn a hyperplane $\langle \mathbf{w}, \mathbf{x}\rangle + b = w_1x_1 + \cdots + w_nx_n + b = 0$ with weight vector $\mathbf{w} \in \mathbb{R}^n$ and bias $b \in \mathbb{R}$ to be estimated through given m pairs of training data. For any newly input vector $\widetilde{\mathbf{x}}$, the class label $\widetilde{y}$ must be predicted such that $\widetilde{y} = -1$ for $\langle \mathbf{w}, \widetilde{\mathbf{x}}\rangle + b < 0$ and $\widetilde{y} = 1$ otherwise. To train optimal hyperplane, there exists two scenarios in the input space: linearly separable and inseparable training datasets.

For the former, we can get the unique optimal hyperplane through addressing the following convex quadratic programming problem. Namely,

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

Here, $[m] := \{1, 2, \ldots, m\}$. Since the model (1) requires correct classifications of all training samples, it is termed as hard-margin SVM.

For the latter, the popular technique is to permit violations the constraints in (1) and then punish these violations in the objective function, which obtains the following unconstrained optimization problem:

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

where $\|\cdot\|$ stands for the Euclidean norm, $\gamma > 0$ denotes a penalty parameter and $\ell(z)$ with $z := 1 - y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle + b)$ represents a loss function. The model (2) allows misclassified training samples, which is named as soft-margin SVM. The first soft-margin SVM is hinge loss SVM (L_h-SVM) [1] and the hinge loss is defined as

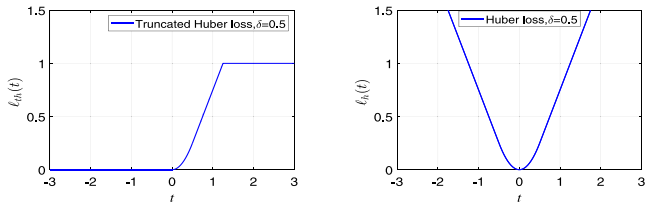
$$\ell_h(z) := \begin{cases} 0, & z < 0, \\ z, & z \geq 0. \end{cases} \tag{3}$$

which is the non-differentiable at $z = 0$. It presents loss at 0 for $z < 0$, which results in the sparsity and provides linear loss for $z \geq 0$, which tends to be sensitive to outliers.

To increase smoothness of L_h-SVM, Zhu et al. in [11] first propose huberized pinball loss SVM and the huberized pinball

* Corresponding author.
E-mail addresses: huajunwang@bjtu.edu.cn (H. Wang), 17118437@bjtu.edu.cn (Y. Shao).

https://doi.org/10.1016/j.knosys.2022.110074
0950-7051/



(a) (b)

Fig. 1. (a) Truncated Huber loss function with $\delta = 0.5$. (b) Huber loss function with $\delta = 0.5$.

loss is defined as

$$\ell_{hp}(z) = \begin{cases} z - \pi/2, & z \geq \pi, \\ z^2/2\pi, & z \in [0, \pi), \\ \beta z^2/2\pi, & z \in (-\pi, 0), \\ \beta(-z - \pi/2), & z < -\pi, \end{cases} \tag{4}$$

where $\pi > 0$ and $\beta \in (0, 1]$. The huberized pinball loss is reduced to the Huber loss (see Fig. 1(b)) if $\beta = 1$. In fact, the huberized pinball loss and Huber loss give an square loss for $z \in (-\pi, \pi)$ and pay a linear loss otherwise, which looses the sparseness and robustness to outliers.

1.1. Truncated Huber loss SVM

To increase sparseness and robustness to outliers of the above two losses, we propose a new sparse and robust loss function, which is shown as

$$\ell_{th}(t) = \begin{cases} 1, & t > 1 + \delta/2, \\ t - \delta/2, & t \in [\delta, 1 + \delta/2], \\ t^2/2\delta, & t \in [0, \delta), \\ 0, & t < 0, \end{cases} \tag{5}$$

where $\delta \in (0, 2)$, which ensures $\delta < 1+\delta/2$. The loss function (5) is known as truncated Huber loss (see Fig. 1(a)). In fact, it gives loss fixed at 1 for $t > 1+\delta/2$, which yields robustness to outliers, while pays loss fixed at 0 for $t < 0$, which receives sparsity. Obviously, truncated Huber loss is sparser or more robust to outliers than hinge loss, huberized pinball loss and Huber loss.

Now, replacing ℓ by ℓ_{th} in (2) enables us to yield a novel sparse and robust SVM model, which is presented as

$$\min_{\mathbf{w}\in\mathbb{R}^n, b\in\mathbb{R}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + \gamma \sum_{i=1}^{m} \ell_{th}(1 - y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b)). \tag{6}$$

The above model are named as truncated Huber loss SVM, which is denoted by L_{th}-SVM. Despite that the nonconvex and nonsmooth of truncated Huber loss function result in difficulty of computation with respect to addressing L_{th}-SVM, however, L_{th}-SVM model enjoys well robustness and sparsity, which can be used for addressing large-scale classification problems. Hence, we carry out this paper along with L_{th}-SVM model.

1.2. Contributions

Contributions of this paper are summarized below.

(C1) We present a new loss function, truncated Huber loss (5), which enables us to construct a new sparse and robust SVM model, see L_{th}-SVM (6). Furthermore, we develop the necessary and sufficient optimality conditions for a local minimizer of L_{th}-SVM (6), see Theorems 3.1 and 3.2.

(C2) Inspired by the P-stationary point, we define the L_{th} support vectors of L_{th}-SVM (8), which proves that the L_{th} support vectors are a small portion of the training set in Theorems 4.1 and 4.2, which makes us to introduce a new working set in each step, see (23).

(C3) To address the L_{th}-SVM (6), we propose a new alternating direction method of multipliers with working set (dubbed as L_{th}-ADMM), which converges to a local minimizer of L_{th}-SVM (6), see Theorem 4.3 and enjoys a relatively low computational complexity, see Remark 4.1.

(C4) The extensive numerical experiments demonstrate that L_{th}-ADMM is capable of delivering higher prediction accuracy, presenting a smaller number of support vectors and running quickly, especially, in large-scale datasets setting.

1.3. Organization

This paper is organized as follows. In Section 2, a brief overview of loss functions will be presented. Section 3 studies the relationships between a P-stationary point and a local minimizer to get optimality conditions of L_{th}-SVM (6). In Section 4, we propose a new alternating direction method of multipliers with working set to handle L_{th}-SVM (6) and establish the properties of convergence and complexity analysis. Numerical experiments demonstrate that our established algorithm shares excellent performance in Section 5. Concluding remarks are presented in the last section.

2. Related work

Most previous work have focused on proposing new loss functions to construct efficient soft-margin SVM models, which is summarized into two categories based on the convexity of loss function.

- The first one consists of the convex loss functions, which include the famous hinge loss function [1,12], generalized hinge loss function [13], pinball loss function [14–18], ε-insensitive pinball loss function [17], least squares loss function [19–21], squared hinge loss function [22–24], Huberized hinge loss function [25], logistic loss function [26]. It is evidently that the convexity of loss function will make the computations of their corresponding SVM models tractable. However, the convex loss function is also a unbounded function, which weakens the robustness of these loss functions to the outliers from the training dataset.

To improve the robustness of these loss functions, upper bounds may be set to stop the increase of loss forcibly beyond a certain threshold. This converts the original convex loss functions into the following non-convex loss functions.

- The other is the non-convex loss functions, which include the truncated hinge loss (ramp loss) function [27–31], the τ-alignment loss [32], the truncated logistic loss function [33], the asymmetrical truncated pinball loss function [34], the truncated pinball loss function [35], generalized exponential loss function [36], generalized logistic loss function [36], sigmoid loss function [37] and so on.

Compared with the convex loss functions, we can obtain that most non-convex loss functions are less sensitive to the outliers because of their boundedness. However, non-convexity of loss functions generally results in difficulties in numerical computations.

3. Optimality conditions of L_{th}-SVM

For convenience, denote

$$G := [y_1\mathbf{x}_1\ y_2\mathbf{x}_2\ \cdots\ y_m\mathbf{x}_m]^\top \in \mathbb{R}^{m\times n}, \tag{7}$$
$$\mathbf{1} := (1, 1, \ldots, 1)^\top \in \mathbb{R}^m,$$
$$\mathbf{y} := (y_1, y_2, \ldots, y_m)^\top \in \mathbb{R}^m,$$
$$\mathbf{h} := \mathbf{1} - G\mathbf{w} - b\mathbf{y} \in \mathbb{R}^m,\ L_{th}(\mathbf{h}) := \sum_{i=1}^{m} \ell_{th}(h_i).$$

Borrowing these notations, the model (6) is equivalently reformulated as

$$\min_{\mathbf{w}\in\mathbb{R}^n, b\in\mathbb{R}, \mathbf{h}\in\mathbb{R}^m} \ \frac{1}{2}\|\mathbf{w}\|^2 + \gamma L_{th}(\mathbf{h}) \quad \text{s.t.} \quad \mathbf{h} + G\mathbf{w} + b\mathbf{y} = \mathbf{1}. \tag{8}$$

We also name model (8) as truncated Huber loss SVM. Prior to the establishment of the optimality condition of L_{th}-SVM (8), we first study subdifferential and proximal operator of truncated Huber loss.

3.1. L_{th} Limiting subdifferential

We begin with presenting the definition of limiting subdifferential of truncated Huber loss, which is dubbed as L_{th} limiting subdifferential.

Definition 3.1 (L_{th} *Limiting Subdifferential [38]*)**.** For truncated Huber loss $L_{th}(\cdot): \mathbb{R}^m \to \mathbb{R}$, the regular and limiting subdifferentials at $\mathbf{h} \in \mathbb{R}^m$ are defined respectively as

$$\widehat{\partial} L_{th}(\mathbf{h}) = \left\{ \mathbf{v} \in \mathbb{R}^m : \liminf_{\widehat{\mathbf{h}}\neq\mathbf{h},\ \widehat{\mathbf{h}}\to\mathbf{h}} \frac{L_{th}(\widehat{\mathbf{h}}) - L_{th}(\mathbf{h}) - \langle \mathbf{v}, \widehat{\mathbf{h}} - \mathbf{h}\rangle}{\|\widehat{\mathbf{h}} - \mathbf{h}\|} \geq 0 \right\},$$
$$\partial L_{th}(\mathbf{h}) = \limsup_{\widehat{\mathbf{h}}\xrightarrow{L_{th}}\mathbf{h}} \widehat{\partial} L_{th}(\widehat{\mathbf{h}}) = \left\{ \mathbf{v} \in \mathbb{R}^m : \exists\, \widehat{\mathbf{h}}_j \xrightarrow{L_{th}} \mathbf{h},\ \mathbf{v}_j \in \widehat{\partial} L_{th}(\widehat{\mathbf{h}}_j) \text{ with } \mathbf{v}_j \to \mathbf{v} \right\},$$

where $\widehat{\mathbf{h}} \xrightarrow{L_{th}} \mathbf{h}$ implies both $\widehat{\mathbf{h}} \to \mathbf{h}$ and $L_{th}(\widehat{\mathbf{h}}) \to L_{th}(\mathbf{h})$.

From the above definition, the explicit expression of L_{th} limiting subdifferential is shown as below.

Lemma 3.1 (*Explicit Expression of L_{th} Limiting Subdifferential*)**.** *For a given $\mathbf{h} \in \mathbb{R}^m$ and $\delta \in (0, 2)$, the L_{th} limiting subdifferential at $\mathbf{h} \in \mathbb{R}^m$ shares the following explicit expression*

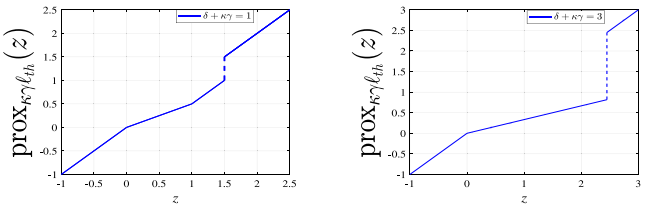
$$\partial L_{th}(\mathbf{h}) := (\partial \ell_{th}(h_1), \ldots, \partial \ell_{th}(h_m))^\top \in \mathbb{R}^m, \tag{9}$$
$$\partial \ell_{th}(h_i) := \begin{cases} 0, & h_i > 1 + \delta/2, \\ \{0, 1\}, & h_i = 1 + \delta/2, \\ 1, & h_i \in (\delta, 1 + \delta/2),\ i \in [m]. \\ t/\delta, & h_i \in [0, \delta], \\ 0, & h_i < 0, \end{cases} \tag{10}$$

3.2. ℓ_{th} Proximal operator

We first introduce the definition of proximal operator of truncated Huber loss function for the one-dimensional case, which is dubbed as ℓ_{th} proximal operator.

Definition 3.2 (ℓ_{th} *Proximal Operator [38]*)**.** For any given $\kappa, \gamma > 0, \delta \in (0, 2)$ and $z \in \mathbb{R}$, the proximal operator of $\ell_{th}(u)$ is defined as

$$\text{prox}_{\kappa\gamma\ell_{th}}(z) = \arg\min_{u\in\mathbb{R}} \ \kappa\gamma\ell_{th}(u) + \frac{1}{2}(u - z)^2. \tag{11}$$



(a) (b)

Fig. 2. (a) The line represents the ℓ_{th} proximal operator with $\delta + \kappa\gamma = 1$. (b) The line denotes the ℓ_{th} proximal operator with $\delta + \kappa\gamma = 3$.

The following lemma states that the ℓ_{th} proximal operator shares an explicit expression by the above definition.

Lemma 3.2 (*Explicit Expression of ℓ_{th} Proximal Operator*)**.** *For any given $\kappa, \gamma > 0, \delta \in (0, 2)$ and $z \in \mathbb{R}$, we have the following explicit expression of ℓ_{th} proximal operator.*

(i) For $\delta + \kappa\gamma \in (0, 2)$, the explicit expression of ℓ_{th} proximal operator at $z \in \mathbb{R}$ is presented as

$$prox_{\kappa\gamma\ell_{th}}(z) := \begin{cases} z, & z > 1 + \frac{\delta+\kappa\gamma}{2}, \\ z \text{ or } z - \kappa\gamma, & z = 1 + \frac{\delta+\kappa\gamma}{2}, \\ z - \kappa\gamma, & z \in (\delta + \kappa\gamma, 1 + \frac{\delta+\kappa\gamma}{2}), \\ \frac{\delta z}{\delta+\kappa\gamma}, & z \in (0, \delta + \kappa\gamma], \\ z, & z \leq 0. \end{cases} \tag{12}$$

(ii) For $\delta + \kappa\gamma \geq 2$, the explicit expression of ℓ_{th} proximal operator at $z \in \mathbb{R}$ is shown as

$$prox_{\kappa\gamma\ell_{th}}(z) := \begin{cases} z, & z > \sqrt{2(\delta + \kappa\gamma)}, \\ z \text{ or } \frac{\delta z}{\delta+\kappa\gamma}, & z = \sqrt{2(\delta + \kappa\gamma)}, \\ \frac{\delta z}{\delta+\kappa\gamma}, & z \in (0, \sqrt{2(\delta + \kappa\gamma)}), \\ z, & z \leq 0. \end{cases} \tag{13}$$

In the following, we use an example to illustrate the ℓ_{th} proximal operator with $\delta + \kappa\gamma \in (0, 2)$ in (12) and $\delta + \kappa\gamma \geq 2$ in (13) (see Fig. 2).

3.3. L_{th} Proximal operator

Based on the separate property of truncated Huber loss, we first introduce the definition of proximal operator of truncated Huber loss for the multi-dimensional case, which is dubbed as L_{th} proximal operator.

Definition 3.3 (L_{th} *Proximal Operator [38]*)**.** For any given $\kappa, \gamma > 0, \delta \in (0, 2)$, the proximal operator of $L_{th}(\mathbf{u})$ at $\mathbf{z} = (z_1, z_2, \ldots, z_m)^\top \in \mathbb{R}^m$ is defined as

$$\text{prox}_{\kappa\gamma L_{th}}(\mathbf{z}) = \arg\min_{\mathbf{u}\in\mathbb{R}^m} \ \kappa\gamma L_{th}(\mathbf{u}) + \frac{1}{2}\|\mathbf{u} - \mathbf{z}\|^2. \tag{14}$$

The following lemma claims that the L_{th} proximal operator shares an explicit expression.

Lemma 3.3 (*Explicit Expression of L_{th} Proximal Operator*)**.** *For any given $\kappa, \gamma > 0, \delta \in (0, 2)$, the explicit expression of L_{th} proximal operator at $\mathbf{z} = (z_1, z_2, \ldots, z_m)^\top \in \mathbb{R}^m$ is presented as*

$$prox_{\kappa\gamma L_{th}}(\mathbf{z}) := \begin{bmatrix} prox_{\kappa\gamma\ell_{th}}(z_1) \\ \vdots \\ prox_{\kappa\gamma\ell_{th}}(z_m) \end{bmatrix} \in \mathbb{R}^m, \tag{15}$$

where $prox_{\kappa\gamma\ell_{th}}(z_i)$ gives in the (12) or (13).

3.4. First-order optimality conditions

This section is devoted to investigate the first-order necessary and sufficient optimality conditions for (8), which will benefit for the algorithmic design. For this purpose, we first introduce the definition of proximal stationary point of (8), which is dubbed as P-stationary point.

Definition 3.4 (*P-stationary Point of (8)*). For a given $\gamma > 0$, $(\mathbf{w}^*; b^*; \mathbf{h}^*)$ is called a P-stationary point of (8) if there is a constant $\kappa > 0$ and a vector $\boldsymbol{\alpha}^* \in \mathbb{R}^m$ such that

$$\begin{cases} \mathbf{w}^* + G^\top \boldsymbol{\alpha}^* = \mathbf{0}, \\ \langle \mathbf{y}, \boldsymbol{\alpha}^* \rangle = 0, \\ \mathbf{h}^* + G\mathbf{w}^* + b^*\mathbf{y} = \mathbf{1}, \\ \mathrm{prox}_{\kappa\gamma L_{th}}(\mathbf{h}^* - \kappa\boldsymbol{\alpha}^*) \ni \mathbf{h}^*. \end{cases} \tag{16}$$

Given $\gamma > 0$, $\delta \in (0, 2)$ and $\mathbf{h}^* \in \mathbb{R}^m$, define $\overline{\delta} := 1 + \delta/2$,

$$\begin{aligned} &\mathbb{S}^* := \{i \in [m] : h_i^* > \overline{\delta}\}, \mathbb{E}^* := \{i \in [m] : h_i^* = \overline{\delta}\}, \\ &\mathbb{T}^* := \{i \in [m] : h_i^* \in (\delta, \overline{\delta})\}, \mathbb{I}^* := \{i \in [m] : h_i^* \in (0, \delta]\}, \\ &\mathbb{O}^* := \{i \in [m] : h_i^* \le 0\}, \mathcal{S}^* := \{i \in [m] : h_i^* > \sqrt{2\delta}\}, \\ &\mathcal{T}^* := \{i \in [m] : h_i^* \in (0, \sqrt{2\delta})\}, \\ &\kappa_1^* := \begin{cases} \min \frac{2(h_i^*-1)-\delta}{\gamma}, & i \in \mathbb{S}^*, \\ +\infty, & \mathbb{S}^* = \emptyset, \end{cases} \\ &\kappa_2^* := \begin{cases} \min \frac{2(1+\delta/2-h_i^*)}{\gamma}, & i \in \mathbb{T}^*, \\ +\infty, & \mathbb{T}^* = \emptyset, \end{cases} \\ &\kappa_3^* := \begin{cases} \min \frac{(h_i^*)^2-2\delta}{2\gamma}, & i \in \mathcal{S}^*, \\ +\infty, & \mathcal{S}^* = \emptyset, \end{cases} \\ &\kappa_4^* := \begin{cases} \min \frac{2\delta^2-\delta(h_i^*)^2}{\gamma(h_i^*)^2}, & i \in \mathcal{T}^* \\ +\infty, & \mathcal{T}^* = \emptyset. \end{cases} \end{aligned} \tag{17}$$

Based on the above definition, the first-order necessary and sufficient condition of (8) is developed in the following two theorems.

Theorem 3.1 (*First-Order Necessary Condition*). *For any given $\gamma > 0$ and $\delta \in (0, 2)$, if $(\mathbf{w}^*; b^*; \mathbf{h}^*)$ is a local minimizer of (8) and $\mathbb{E}^* = \emptyset$, then we obtain that it is a P-stationary point with $0 < \kappa \le \kappa^* := \min\{\kappa_1^*, \kappa_2^*, \kappa_3^*, \kappa_4^*\}$.*

Theorem 3.2 (*First-Order Sufficient Condition*). *For any given $\gamma > 0$ and $\delta \in (0, 2)$, if $(\mathbf{w}^*; b^*; \mathbf{h}^*)$ with $\boldsymbol{\alpha}^*$ is a P-stationary point of (8), then we obtain that the $(\mathbf{w}^*; b^*; \mathbf{h}^*)$ is a local minimizer of (8).*

Theorems 3.1 and 3.2 prove that the newly introduced P-stationary point is the first-order necessary and sufficient conditions, which will be applied to investigate support vectors and fast algorithm of L_{th}-SVM in next part.

4. Fast algorithm

It is widely known that the optimal hyperplane of L_{th}-SVM is decided by its support vectors. When only support vectors are utilized to develop the SVM method, the computational speed is inversely related to the number of support vectors as the utilization of fewer samples in training data during the training of the optimal hyperplane. Thus, decreasing the number of support vectors becomes very key for large-scale SVM problems. Motivated by this, based on the optimality conditions in Section 3, we now define the support vectors of L_{th}-SVM, which is named as L_{th} support vectors.

4.1. L_{th} Support vectors

Theorem 4.1 (L_{th} *Support Vectors for $\delta+\kappa\gamma \in (0, 2)$*). *For $\kappa, \gamma > 0$ and $\delta \in (0, 2)$ satisfying $\delta + \kappa\gamma \in (0, 2)$, if $(\mathbf{w}^*; b^*; \mathbf{h}^*)$ with $\boldsymbol{\alpha}^* \in \mathbb{R}^m$ is a P-stationary point of (8), then we obtain*

$$\mathbf{w}^* = -\sum_{i \in H^*} \alpha_i^* y_i \mathbf{x}_i \text{ and } \alpha_i^* = 0, i \in \overline{H}^*, \tag{18}$$

where $H^* := H_2^* \cup H_3^*$ in (42) and $\overline{H}^* := [m] \backslash H^*$. The $\{\mathbf{x}_i : i \in H^*\}$ are named as the L_{th} support vectors of L_{th}-SVM (8) and satisfy

$$\begin{cases} y_i(\langle \mathbf{w}^*, \mathbf{x}_i \rangle + b^*) \in (1-\delta, 1), & i \in H_2^*, \\ y_i(\langle \mathbf{w}^*, \mathbf{x}_i \rangle + b^*) \in [\frac{\kappa\gamma-\delta}{2}, 1-\delta], & i \in H_3^*. \end{cases} \tag{19}$$

where $\frac{\kappa\gamma-\delta}{2} \in (-\delta, 1-\delta)$. Thus, the L_{th} support vectors are a small portion of the training set.

Theorem 4.2 (L_{th} *Support Vectors for $\delta + \kappa\gamma \ge 2$*). *For $\kappa, \gamma > 0$ and $\delta \in (0, 2)$ satisfying $\delta + \kappa\gamma \ge 2$, if $(\mathbf{w}^*; b^*; \mathbf{h}^*)$ with $\boldsymbol{\alpha}^* \in \mathbb{R}^m$ is a P-stationary point of (8), then we have*

$$\mathbf{w}^* = -\sum_{i \in \mathcal{H}^*} \alpha_i^* y_i \mathbf{x}_i \text{ and } \alpha_i^* = 0, i \in \overline{\mathcal{H}}^*, \tag{20}$$

where $\mathcal{H}^* := \mathcal{H}_2^*$ in (46) and $\overline{\mathcal{H}}^* := [m] \backslash \mathcal{H}^*$. The $\{\mathbf{x}_i \in \mathbb{R}^m : i \in \mathcal{H}^*\}$ are named as the L_{th} support vectors of (8) and meet

$$y_i(\langle \mathbf{w}^*, \mathbf{x}_i \rangle + b^*) \in [1 - \sqrt{2\delta^2/(\delta + \kappa\gamma)}, 1). \tag{21}$$

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

From the above two theorems, we can obtain that the L_{th} support vectors are a small portion of the training set. Motivated by this, based on L_{th} support vectors, we first introduce a novel working set and then use the famous alternating direction method of multipliers (ADMM) with working set to handle L_{th}-SVM.

4.2. L_{th}-ADMM framework

In this subsection, we aim to adopt ADMM with working set to handle the L_{th}-SVM (8), which is called L_{th}-ADMM. The augmented Lagrangian function of the problem (8) can be written as

$$\begin{aligned} \mathcal{L}(\mathbf{w}; b; \mathbf{h}; \boldsymbol{\alpha}) = &\frac{1}{2}\|\mathbf{w}\|^2 + \gamma L_{th}(\mathbf{h}) + \langle \boldsymbol{\alpha}, \mathbf{h} - \mathbf{1} + G\mathbf{w} + b\mathbf{y} \rangle \\ &+ \frac{\nu}{2}\|\mathbf{h} - \mathbf{1} + G\mathbf{w} + b\mathbf{y}\|^2, \end{aligned}$$

where $\boldsymbol{\alpha} \in \mathbb{R}^m$ stands for the Lagrangian multiplier vector and $\nu > 0$ represents a given parameter. Given $(\mathbf{w}^k; b^k; \mathbf{h}^k; \boldsymbol{\alpha}^k)$, L_{th}-ADMM performs the following steps iteratively,

$$\begin{aligned} \mathbf{h}^{k+1} &= \operatorname*{argmin}_{\mathbf{h} \in \mathbb{R}^m} \mathcal{L}(\mathbf{w}^k, b^k, \mathbf{h}, \boldsymbol{\alpha}^k) \\ \mathbf{w}^{k+1} &= \operatorname*{argmin}_{\mathbf{w} \in \mathbb{R}^n} \mathcal{L}(\mathbf{w}, b^k, \mathbf{h}^{k+1}, \boldsymbol{\alpha}^k) + \tfrac{\nu}{2}\|\mathbf{w} - \mathbf{w}^k\|_{B_k}^2 \\ b^{k+1} &= \operatorname*{argmin}_{b \in \mathbb{R}} \mathcal{L}(\mathbf{w}^{k+1}, b, \mathbf{h}^{k+1}, \boldsymbol{\alpha}^k) \\ \boldsymbol{\alpha}^{k+1} &= \boldsymbol{\alpha}^k + \mu\nu(\mathbf{h}^{k+1} - \mathbf{1} + G\mathbf{w}^{k+1} + b^{k+1}\mathbf{y}), \end{aligned} \tag{22}$$

where $\mu \in (0, \frac{1+\sqrt{5}}{2})$ means the dual step-size and $B_k \in \mathbb{R}^{m \times m}$. Here, we define proximal term as

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

It is evidently that if B_k is positive semidefinite matrix, then the framework (22) is known as the standard semi-proximal ADMM [39]. However, in papers [40,41], researchers have also investigated ADMM with the indefinite proximal terms. In other words, the B_k is indefinite matrix. The fundamental principle of choosing B_k is to guarantee the convexity of $\mathbf{w}$-subproblem. Since

$\mathcal{L}(\mathbf{w}; b^k; \mathbf{h}^{k+1}; \boldsymbol{\alpha}^k)$ is strongly convex with regard to $\mathbf{w}$, therefore, B_k can be picked as a negative semidefine matrix.

In order to pick matrix B_k, we introduce a working set based on the L_{th} support vectors, which is denoted as J_k. Define $\mathbf{s}^k := \mathbf{h}^k - \boldsymbol{\alpha}^k/\nu = \mathbf{1} - G\mathbf{w}^k - b^k\mathbf{y} - \boldsymbol{\alpha}^k/\nu$, $\kappa := 1/\nu$,

$$\Gamma_k^1 := \{i \in [m] : s_i^k \in (0, \widetilde{\delta}]\},$$

$$\Gamma_k^2 := \{i \in [m] : s_i^k \in (\widetilde{\delta}, 1 + \frac{\kappa\gamma}{2}) \text{ or } s_i^k = 1 + \frac{\kappa\gamma}{2}, \alpha_i^k \neq 0\},$$

$$I_k := \{i \in [m] : s_i^k \in (0, \sqrt{2\widetilde{\delta}}) \text{ or } s_i^k = \sqrt{2\widetilde{\delta}}, \alpha_i^k \neq 0\}.$$

where $\widetilde{\delta} := \delta + \kappa\gamma$. Based on the above notations, we define working set J_k at the kth step as

$$J_k := \begin{cases} \Gamma_k^1 \cup \Gamma_k^2, & \delta + \kappa\gamma \in (0, 2), \\ I_k, & \delta + \kappa\gamma \geq 2. \end{cases} \tag{23}$$

and $\bar{J}_k := [m] \backslash J_k$. Thus, we pick matrix B_k as

$$B_k = -G_{J_k}^\top G_{\bar{J}_k}. \tag{24}$$

Here, matrix $G_{\bar{J}_k} \in \mathbb{R}^{|\bar{J}_k| \times n}$ represents the sub-matrix containing rows of G indexed on $\bar{J}_k$, where $|\bar{J}_k|$ denotes the cardinality of set $\bar{J}_k$. We will see that if the number of selected working set J_k is very small, the L_{th}-ADMM possesses a considerably low computational complexity (see Remark 4.1) and thus runs super fast (see Section 5.3). More precisely, the explicit expression of the each sub-problem of (22) is derived as below.

(i) Updating $\mathbf{h}^{k+1}$: The $\mathbf{h}$-subproblem of (22) can be simplified into the following form

$$\begin{aligned} &\mathbf{h}^{k+1} \\ &= \operatorname*{argmin}_{\mathbf{h}\in\mathbb{R}^m} \gamma L_{th}(\mathbf{h}) + \langle \boldsymbol{\alpha}^k, \mathbf{h}\rangle + \frac{\nu}{2}\|\mathbf{h} - \mathbf{1} + G\mathbf{w}^k + b^k\mathbf{y}\|^2 \\ &= \operatorname*{argmin}_{\mathbf{h}\in\mathbb{R}^m} \gamma L_{th}(\mathbf{h}) + \frac{\nu}{2}\|\mathbf{h} - (\mathbf{1} - G\mathbf{w}^k - b^k\mathbf{y} - \boldsymbol{\alpha}^k/\nu)\|^2 \\ &= \operatorname*{argmin}_{\mathbf{h}\in\mathbb{R}^m} \gamma L_{th}(\mathbf{h}) + \frac{\nu}{2}\|\mathbf{h} - \mathbf{s}^k\|^2 \\ &= \mathrm{Prox}_{\frac{\gamma}{\nu}L_{th}}(\mathbf{s}^k), \end{aligned}$$

where the last equation is from (15). This together with (12), (13) and the working set (23) leads to

$$\begin{cases} \mathbf{h}_{\Gamma_k^1}^{k+1} = \frac{\delta}{\widetilde{\delta}}\mathbf{s}_{\Gamma_k^1}^k, \mathbf{h}_{\Gamma_k^2}^{k+1} = \widetilde{\mathbf{s}}_{\Gamma_k^2}^k, \mathbf{h}_{\bar{J}_k}^{k+1} = \mathbf{s}_{\bar{J}_k}^k, & \widetilde{\delta} \in (0, 2), \\ \mathbf{h}_{J_k}^{k+1} = \frac{\delta}{\widetilde{\delta}}\mathbf{s}_{J_k}^k, \mathbf{h}_{\bar{J}_k}^{k+1} = \mathbf{s}_{\bar{J}_k}^k, & \widetilde{\delta} \geq 2, \end{cases} \tag{25}$$

where $\widetilde{\delta} := \delta + \kappa\gamma$ and $\widetilde{\mathbf{s}}^k := \mathbf{s}^k - \kappa\gamma\mathbf{1}$. Hence, updating $\mathbf{h}^{k+1}$ becomes very simple.

(ii) Updating $\mathbf{w}^{k+1}$. The $\mathbf{w}$-subproblem of (22) can be simplified into the following form

$$\begin{aligned} \mathbf{w}^{k+1} = \arg\min_{\mathbf{w}\in\mathbb{R}^n} &\frac{1}{2}\|\mathbf{w}\|^2 + \frac{\nu}{2}\|\mathbf{w} - \mathbf{w}^k\|^2_{-G_{J_k}^\top G_{\bar{J}_k}} \\ &+ \langle \boldsymbol{\alpha}^k, G\mathbf{w}\rangle + \frac{\nu}{2}\|\mathbf{h}^{k+1} - \mathbf{1} + G\mathbf{w} + b^k\mathbf{y}\|^2. \end{aligned} \tag{26}$$

Therefore, we can yield the solution to the equation

$$\begin{aligned} \mathbf{0} = &\mathbf{w} - \nu G_{\bar{J}_k}^\top G_{\bar{J}_k}(\mathbf{w} - \mathbf{w}^k) + G^\top\boldsymbol{\alpha}^k \\ &+ \nu G^\top(\mathbf{h}^{k+1} - \mathbf{1} + G\mathbf{w} + b^k\mathbf{y}), \end{aligned} \tag{27}$$

which is equivalent to receive the solution to the equation

$$(I + \nu G_{J_k}^\top G_{J_k})\mathbf{w} = \nu G_{J_k}^\top \boldsymbol{\phi}_{J_k}^k, \tag{28}$$

where $\boldsymbol{\phi}^k := -(\mathbf{h}^{k+1} + b^k\mathbf{y} - \mathbf{1} + \boldsymbol{\alpha}^k/\nu)$. To derive (28) from (27), we employ the following fact that

$$G_{J_k}^\top G_{J_k} = G^\top G - G_{\bar{J}_k}^\top G_{\bar{J}_k}.$$

Hence, the term $G_{\bar{J}_k}$ vanishes in (28). This means the selecting of B_k and the working set J_k allows us to discard the samples $\{\mathbf{x}_j, j \in \bar{J}_k\}$. When the selected $|J_k|$ is very small in the implementation, then this would fasten the computation tremendously of L_{th}-ADMM. In practice, (28) is handled efficiently by the following two cases:

- If $n \leq |J_k|$, we can solve (28) through the form
$$\mathbf{w}^{k+1} = (I + \nu G_{J_k}^\top G_{J_k})^{-1}\nu G_{J_k}^\top \boldsymbol{\phi}_{J_k}^k. \tag{29}$$
- If $n > |J_k|$, we use the Sherman–Morrison–Woodbury formula [42] to calculate the inverse as
$$(I + \nu G_{J_k}^\top G_{J_k})^{-1} = I - \nu G_{J_k}^\top (I + \nu G_{J_k} G_{J_k}^\top)^{-1} G_{J_k}.$$
Then we can update $\mathbf{w}^{k+1}$ as
$$\mathbf{w}^{k+1} = \nu G_{J_k}^\top (I + \nu G_{J_k} G_{J_k}^\top)^{-1} \boldsymbol{\phi}_{J_k}^k. \tag{30}$$

(iii) Updating b^{k+1}. The b-subproblem of (22) can be simplified into the following form

$$b^{k+1} = \arg\min_{b\in\mathbb{R}} \langle \boldsymbol{\alpha}^k, b\mathbf{y}\rangle + \frac{\nu}{2}\|\mathbf{h}^{k+1} - \mathbf{1} + G\mathbf{w}^{k+1} + b\mathbf{y}\|^2.$$

This is a convex differentiable problem, which is addressed efficiently by

$$b^{k+1} = \langle \mathbf{y}, \boldsymbol{\chi}^k\rangle/\|\mathbf{y}\|^2 = \langle \mathbf{y}, \boldsymbol{\chi}^k\rangle/m, \tag{31}$$

where $\boldsymbol{\chi}^k := -G\mathbf{w}^{k+1} + \mathbf{1} - \mathbf{h}^{k+1} - \boldsymbol{\alpha}^k/\nu$.

(iv) Updating $\boldsymbol{\alpha}^{k+1}$. In (22), we update $\boldsymbol{\alpha}^{k+1}$ as below

$$\boldsymbol{\alpha}_{J_k}^{k+1} = \boldsymbol{\alpha}_{J_k}^k + \mu\nu\boldsymbol{\zeta}_{J_k}^{k+1}, \quad \boldsymbol{\alpha}_{\bar{J}_k}^{k+1} = \mathbf{0}, \tag{32}$$

where $\boldsymbol{\zeta}^{k+1} := \mathbf{h}^{k+1} - \mathbf{1} + G\mathbf{w}^{k+1} + b^{k+1}\mathbf{y}$ and setting $\boldsymbol{\alpha}_{\bar{J}_k}^{k+1} = \mathbf{0}$ is from the (18) and (20). In other words, we will remove the part of the Lagrangian multiplier not on the working set.

Based on above analysis, we summarize the framework of our method in Algorithm 1.

Algorithm 1 : L_{th}-ADMM for solving problem (8)

Initialize $(\mathbf{w}^0; b^0; \mathbf{h}^0; \boldsymbol{\alpha}^0)$ and γ, ν, δ, $K > 0$. Set $k = 0$.
while The stop condition does not hold and $k \leq K$ **do**
  Compute J_k as in (23).
  Compute $\mathbf{h}^{k+1}$ by (25).
  Compute $\mathbf{w}^{k+1}$ by (29) if $n \leq |J_k|$ and (30) otherwise.
  Compute b^{k+1} by (31).
  Compute $\boldsymbol{\alpha}^{k+1}$ by (32).
  Set $k = k + 1$.
end while
return the optimal solution $(\mathbf{w}^k, b^k)$ to (8).

4.3. Convergence and complexity analysis

The following theorem states that if the sequence generated by L_{th}-ADMM converges, then we can obtain that it must converge to a local minimizer.

Theorem 4.3. *Let $(\mathbf{w}^*; b^*; \mathbf{h}^*; \boldsymbol{\alpha}^*)$ be the limit point of the sequence $\{(\mathbf{w}^k; b^k; \mathbf{h}^k; \boldsymbol{\alpha}^k)\}$ generated by L_{th}-ADMM. Then we obtain that $(\mathbf{w}^*; b^*; \mathbf{h}^*)$ is a P-stationary point with $\kappa = 1/\nu$. In addition, it is also a local minimizer to (8).*

Remark 4.1. We have some comments in terms of the computational complexity in each iteration of the proposed algorithm L_{th}-ADMM, which is shown as follows.

- To update J_k, the computational complexity is $\mathcal{O}(m)$.
- To compute $\mathbf{h}^{k+1}$, the main term is $G\mathbf{w}^k$, which the computational complexity is about $\mathcal{O}(mn)$.
- To update $\mathbf{w}^{k+1}$, we update (29) if $n \le |J_k|$ and (30) otherwise. For updating $\mathbf{w}^{k+1}$ by (29), the dominant computations are calculating

  $$G_{J_k}^\top G_{J_k} \quad \text{and} \quad (I + \nu G_{J_k}^\top G_{J_k})^{-1}.$$

  For calculating $G_{J_k}^\top G_{J_k}$, the computational complexity is about $\mathcal{O}(n^2|J_k|)$, while for calculating $(I + \nu G_{J_k}^\top G_{J_k})^{-1}$, the computational complexity is about $\mathcal{O}(n^\rho)$, where $\rho \in (2, 3)$. For updating $\mathbf{w}^{k+1}$ by (30), the dominant computations are updating

  $$G_{J_k} G_{J_k}^\top \quad \text{and} \quad (I + \nu G_{J_k} G_{J_k}^\top)^{-1}.$$

  For calculating $G_{J_k} G_{J_k}^\top$, the computational complexity is about $\mathcal{O}(n|J_k|^2)$. For computing $(I + \nu G_{J_k} G_{J_k}^\top)^{-1}$, the computational complexity is about $\mathcal{O}(|J_k|^\rho)$, where $\rho \in (2, 3)$. Based on above analysis, the computational complexity of computing $\mathbf{w}^{k+1}$ in each step is about

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

- To update b^{k+1}, the $G\mathbf{w}^{k+1}$ is the most expensive computation, whose computational complexity is $\mathcal{O}(mn)$.
- To update $\boldsymbol{\alpha}^{k+1}$, the computational complexity is $\mathcal{O}(mn)$.

Based on the above analysis, the whole computational complexity of each step is

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

which represents that the L_{th}-ADMM enjoys a considerably low computational complexity if $\max\{|J_k|, n\} \ll m$.

5. Numerical experiments

This section will report the results of the proposed algorithm L_{th}-ADMM in Algorithm 1 on both synthetic and real data, by utilizing MATLAB (R2018b) on a laptop of 32 GB memory and Core(TM) i7-9880H Inter(R) 2.7 GHz CPU.

Inspired by Theorem 3.2, we will stop algorithm if the point $(\mathbf{w}^k; b^k; \mathbf{h}^k; \boldsymbol{\alpha}^k)$ generated by our proposed algorithm L_{th}-ADMM satisfies (16). Namely,

$$\max\{\pi_1^k, \pi_2^k, \pi_3^k, \pi_4^k\} < \epsilon,$$

where ϵ stands for the tolerance level and

$$\pi_1^k := \frac{\|\mathbf{w}^k + G_{J_k}^\top \boldsymbol{\alpha}_{J_k}^k\|}{1 + \|\mathbf{w}^k\|}, \quad \pi_2^k := \frac{|\langle \mathbf{y}_{J_k}, \boldsymbol{\alpha}_{J_k}^k \rangle|}{1 + |J_k|},$$

$$\pi_3^k := \frac{\|\mathbf{h}^k - \mathbf{1} + G\mathbf{w}^k + b^k \mathbf{y}\|}{\sqrt{m}},$$

$$\pi_4^k := \frac{\|\mathbf{h}^k - \mathrm{prox}_{\frac{\gamma}{\nu} L_{th}}(\mathbf{h}^k - \boldsymbol{\alpha}^k/\nu)\|}{1 + \|\mathbf{h}^k\|}.$$

5.1. Testing examples

We first investigate a synthetic data in $\mathbb{R}^2$, where the features data come from Gaussian normal distribution.

Example 5.1 (*Synthetic Data in $\mathbb{R}^2$ Without Outliers [17]*)**.** We first generate m samples $\mathbf{x}_i$ with positive labels $y_i = +1$, where are come from the Gaussian normal distribution with mean $(0.5, -3)^\top$ and variance Λ, and then generate m samples $\mathbf{x}_j$ with negative labels $y_j = -1$, where are come from the Gaussian normal distribution with mean $(-0.5, 3)^\top$ and variance Π. Here,

$$\Lambda = \Pi = \begin{bmatrix} 0.2 & 0 \\ 0 & 3 \end{bmatrix}.$$

Finally, we evenly divide them into a training and a testing set.

Example 5.2 (*Synthetic Data in $\mathbb{R}^2$ With Outliers*)**.** Firstly, we get $2m$ samples generated in Example 5.1. Then we randomly flip rm labels in each class. In other words, the $2rm$ outliers are generated, where the r is named as the flapping ratio. Finally, we evenly divide them into a training and a testing set.

Example 5.3 (*Real Data Without Outliers*)**.** We select fourteen real datasets from the libraries: libsvm[1] and uci.[2] Their details are provided in Table 3, where the first eight datasets have not the testing data. Moreover, the training size and testing size are denoted as m and m_t respectively and the feature is denoted by n. In our experiments, all the features are scaled to $[-1, 1]$ and all the classes being not 1 are regarded as -1.

Example 5.4 (*Real Data with Outliers*)**.** To observe the influence of the real datasets with outliers on ten solvers, we first pick 6 datasets with small samples or moderate samples in Table 3, which are `col`, `mus`, `aus`, `two`, `adu` and `a6a`. For datasets without the testing samples, we divide them into two parts. The first part contains 10% of samples regarded as the testing samples and the rest are the training samples. Finally, we randomly flip r percentage of training and testing samples, which is known as outliers.

To demonstrate the performance of one solver, we will report five evaluation indicators: the size of working set per iteration (`SWS/ITER`), the total number of iterations (`TNI`), the testing classification accuracy (`ACC`), the CPU time (`CPU`) and the number of support vectors (`NSV`). The `ACC` and `SWS/ITER` are defined as

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

$$\texttt{SWS/ITER} := \frac{\sum_{k=1}^{\overline{k}} |J_k|}{\overline{k}},$$

where $(\overline{\mathbf{w}}, \overline{b})$ is the solution obtained by one classifier and $\{(\overline{\mathbf{x}}_j, \overline{y}_j) : j = 1, \ldots, \overline{m}\}$ is the testing set. $|J_k|$ stands for the size of the working set J_k and $\overline{k}$ stands for the total number of iterations. $\|e\|_0$ implies the zero norm of e. In addition, $\mathrm{sign}(\overline{d}) = 1$ if $\overline{d} > 0$ and $\mathrm{sign}(\overline{d}) = -1$ otherwise. The larger `ACC` (or the smaller `SWS/ITER`, `NSV`, `CPU` or `TNI`) means the better performance.

5.2. Implementation and parameters tuning

We initialize L_{th}-ADMM with $(\mathbf{w}^0; b^0; \mathbf{h}^0; \boldsymbol{\alpha}^0) = \mathbf{0}$ and fix $\mu = 1.618$, $\epsilon = 10^{-3}$ and $K = 10^3$ if no additional information is provided. Recall that the L_{th}-ADMM involves two important parameters γ, ν and the truncated Huber loss has the parameter $\delta \in (0, 2)$. We now adopt Algorithm 1 for turning them as presented below.

5.2.1. Effect of γ

For Example 5.1, we assign fixed values to $\nu = \sqrt{2}$, $\delta = 0.2$ and vary $\gamma \in \{2^{-3}, 2^{-2}, 2^{-1}, 2^0, 2^1, 2^2\}$ and $m \in \{2000, 4000, \ldots, 8000\}$ to examine the effect of γ on Algorithm 1. For each case (γ, m), we run 50 times and present the average results in

[1] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/.

[2] http://archive.ics.uci.edu/ml/datasets.php.

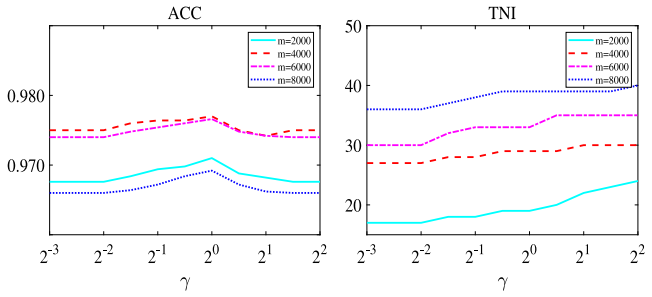


Fig. 3. Effect of γ.

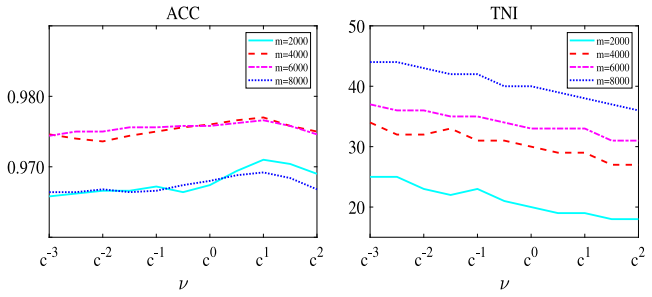


Fig. 4. Effect of ν with $c = \sqrt{2}$.

Fig. 3. We can see that in the range $[2^{-3}, 2^{-2}]$, our algorithm is insensitive to γ. For each m, it is clearly observed that for $\gamma \in [2^{-3}, 2^{-2}]$, the ACC lines are stabilized at a certain level, increased for $\gamma \in [2^{-2}, 2^{0}]$ and then decreased, which indicates that γ of approximately 2^0 achieve the highest classification accuracy for Algorithm 1. Moreover, we also test the real datasets, and omit similar observations. Therefore, we select $\gamma = 1$ in the subsequent numerical experiments if no additional information is provided.

5.2.2. Effect of ν

To observe the effect of ν, we set $\gamma = 1$, $\delta = 0.2$ and vary $\nu \in [c^{-3}, c^{2}]$ with $c = \sqrt{2}$ and $m \in \{2000, 4000, \ldots, 8000\}$. Again, we employ Algorithm 1 for addressing Example 5.1 and the average results are provided in Fig. 4. It is evident that ACC achieve the highest peak at $\nu = \sqrt{2}$. When $\nu > \sqrt{2}$, the bigger ν is, the smaller ACC is and the smaller number of iterations is as well. Moreover, we also test the real datasets, and omit similar observations. To balance the ACC and ITER, if no additional information is provided, we fix $\nu = \sqrt{2}$ in the following numerical experiments.

5.2.3. Effect of δ

We now turn our attention to see the effect of δ. For this purpose, we choose $\gamma = 1$, $\nu = \sqrt{2}$ and vary $\delta \in (0, 2)$ and $m \in \{2000, 4000, \ldots, 8000\}$. We also use Algorithm 1 for dealing with Example 5.1 and the average results are showed in Fig. 5. It is evident that the ACC lines are increased with the ascending of δ but gradually stabilized at a certain level after $\delta > 0.2$. Moreover, the small TNI demonstrate that Algorithm 1 can converge quickly. In addition, we also test the real datasets, and omit similar observations. Hence, in the subsequent numerical experiments, we select $\delta = 0.2$ unless specified otherwise.

5.2.4. Effect of initial points

This part is devoted to examine how the initial points affect the performance of Algorithm 1. For this purpose, we adopt the algorithm for handling Example 5.1 with $m = 5000$ and

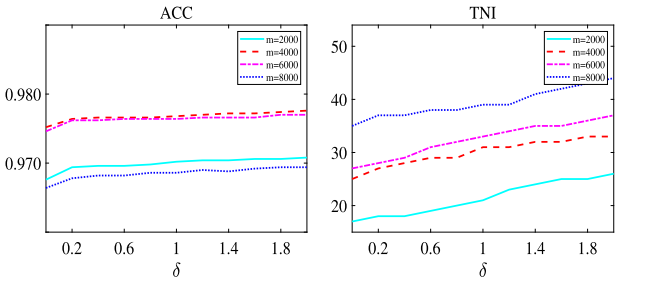


Fig. 5. Effect of δ.

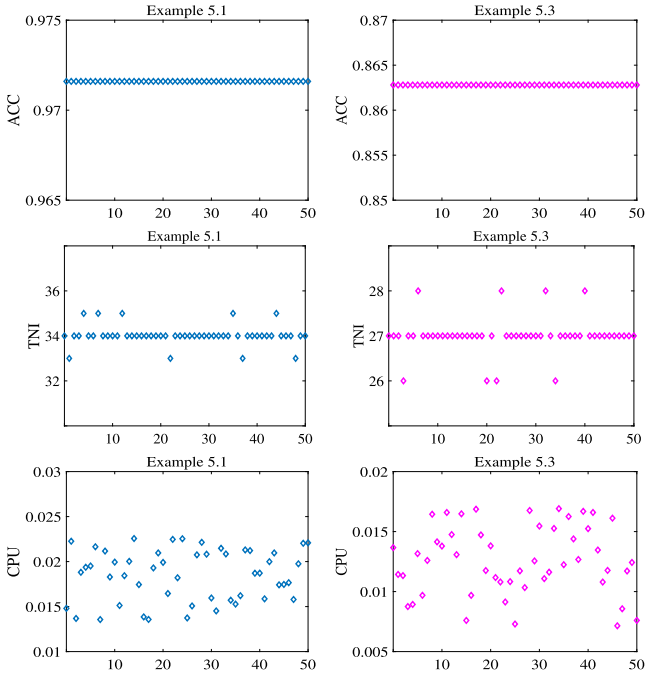


Fig. 6. Robust to the initial points.

Example 5.3 with data aus. We run the our algorithm under 50 different initial points $(\mathbf{w}^0; b^0; \mathbf{h}^0; \boldsymbol{\alpha}^0)$. The first point is $(\mathbf{w}^0; b^0; \mathbf{h}^0; \boldsymbol{\alpha}^0) = \mathbf{0}$ and the other 49 points are randomly produced from the uniform distribution, namely, $(\mathbf{w}^0; b^0; \mathbf{h}^0; \boldsymbol{\alpha}^0) \sim U[0, 1]$. The ACC, TNI and CPU are plotted in Fig. 6, where the x-axis denotes the 50 initial points. It is evidently that all the results stabilize at a certain level, meaning that for these datasets, our algorithm is not sensitive to the choice of the initial points. In addition, we also test the other datasets, and omit similar observations. Therefore, for simplicity, we initialize our algorithm L_{th}-ADMM with $(\mathbf{w}^0; b^0; \mathbf{h}^0; \boldsymbol{\alpha}^0) = \mathbf{0}$ in the sequel.

5.3. Numerical comparisons

To conduct fair comparisons, nine state-of-the-art solvers are selected for comparisons, where eight of them address the SVM problem and one handles the L_2-regularized logistic regression problem.

5.3.1. Benchmark methods

TSVM Truncated Huber loss SVM is addressed by our proposed L_{th}-ADMM.

HSVM Hinge loss SVM can be handled by LibSVM solver [12].

PSVM Hinge loss SVM can be handled by Pegasos algorithm [6], where the maximum number of iterations is fixed as $2m$ and the mini-batch size is fixed as 1.

Table 1
Results of 10 solvers for addressing Example 5.1.

m	ACC (%)									
	TSVM	HSVM	PSVM	ISVM	RSVM	OSVM	VSVM	ASVM	BSVM	RELO
2000	**97.05**	**97.05**	97.01	**97.05**	**97.05**	**97.05**	**97.05**	**97.05**	97.00	97.03
4000	**97.35**	97.25	97.26	97.30	97.33	97.32	97.33	**97.35**	97.30	97.25
6000	**97.33**	97.28	97.16	97.24	**97.33**	**	**97.33**	97.30	**97.33**	97.22
8000	**96.96**	96.91	96.93	96.91	**96.96**	**	96.94	**96.96**	96.89	**96.96**
10 000	**97.23**	97.18	97.16	97.19	97.20	**	97.18	97.18	97.16	97.18
	NSV									
2000	**24**	187	198	2000	96	146	184	192	2000	2000
4000	**37**	301	325	4000	141	289	332	295	4000	4000
6000	**54**	439	453	6000	201	**	444	452	6000	6000
8000	**65**	571	566	8000	223	**	579	563	8000	8000
10 000	**81**	658	669	10 000	240	**	675	648	10 000	10 000
	SWS/ITER									
2000	43	2	**1**	2000	2000	2000	**1**	**1**	2000	2000
4000	65	2	**1**	4000	4000	4000	**1**	**1**	4000	4000
6000	76	2	**1**	6000	6000	6000	**1**	**1**	6000	6000
8000	89	2	**1**	8000	8000	8000	**1**	**1**	8000	8000
10 000	96	2	**1**	10 000	10 000	10 000	**1**	**1**	10 000	10 000
	TNI									
2000	19	259	4000	1216	3(8)	2(13)	4000	4000	14	**8**
4000	29	463	8000	2325	3(16)	3(18)	8000	8000	14	**9**
6000	33	639	12 000	3750	4(15)	**	12 000	12 000	15	**9**
8000	39	772	16 000	5247	4(21)	**	16 000	16 000	16	**9**
10 000	48	961	20 000	6326	5(23)	**	20 000	20 000	16	**10**
	CPU (s)									
2000	**0.007**	0.014	0.028	9.642	3.969	132.5	0.024	0.025	0.221	0.034
4000	**0.009**	0.022	0.089	67.58	16.29	2043	0.087	0.088	0.626	0.112
6000	**0.014**	0.036	0.133	209.9	31.44	>2 h	0.126	0.131	1.200	0.204
8000	**0.021**	0.069	0.194	493.2	65.25	>2 h	0.185	0.188	2.342	0.536
10 000	**0.027**	0.094	0.281	775.3	124.7	>2 h	0.266	0.268	3.951	0.938

ISVM Pinball loss SVM can be solved by the traversal algorithm [15].

RSVM Truncated hinge loss (ramp loss) SVM is tackled by CCCP [30], where the key subproblem of CCCP can be implemented by the MATLAB built-in function quadprog.

OSVM Generalized exponential loss SVM is tackled by the iteratively reweighted algorithm (IRA) [36], where the core subproblem of IRA can be implemented by the CVX.

VSVM Squared hinge loss SVM is implemented by SVRG algorithm [23], where the number of "passes" is fixed as $S = 1$ and the mini-batch size is fixed as 1. Moreover, the default epoch length is fixed as $2m$.

ASVM Squared hinge loss SVM is implemented by Katyusha algorithm [24], where all parameters are selected the same as these for VSVM.

BSVM Least squares loss is tackled by LibLSSVM [21].

RELO L_2-regularized logistic regression can be implemented by Newton algorithm [43].

Overall, all solvers start with the same initial points and the other parameters for each solvers are chosen to be their default values.

5.3.2. Comparisons with synthetic data

(a) Synthetic data without outliers. Firstly, we perform the above ten solvers to address Example 5.1 with sample sizes $m = 2000, 4000, \ldots, 10000$ and $n = 2$, and present the average results of ten solvers in Table 1, where "4(15)" represents the average number of outer (inner subproblem) iterations. In addition, "**" denotes that the average results are not yielded when a solver requires a large memory (denote "––") or uses time longer than two hour (denote ">2 h"). For ACC, obviously, all solvers get desirable ACC and TSVM gets the best classification accuracy. As for NSV, TSVM shares a considerably small number of the support vectors. By contrast, BSVM, ISVM and RELO take all samples as the support vectors. As for SWS/ITER, TSVM, HSVM, PSVM, VSVM and ASVM select a very small portion of samples as the working set. By contrary, others take all samples as the working set. When it comes to TNI, TSVM, RSVM, BSVM and RELO use a small TNI , while others enjoy a large TNI. As for CPU, TSVM uses the shortest CPU because TSVM takes a small TNI and SWS/ITER.

(b) Synthetic data with outliers. To observe the robustness to outliers, for Example 5.2, we investigate $m = 5000$, $n = 2$ and alter $r = 0, 0.05, 0.1, 0.15, 0.2$. The average results of ten solvers are shown in Table 2. In terms of ACC, we can see that with r rising, the smaller ACC are generated by each solver and TSVM gets slightly better ACC, which denotes more robust to outliers than other solvers. When it comes to NSV, TNI and SWS/ITER, we obtain similar observations to that in Table 1. Furthermore, it is clearly observed that for HSVM, PSVM, OSVM, VSVM and ASVM, when the more outliers are added, the more samples become support vectors and bigger values of TNI are generated by ISVM and HSVM. By contrary, with r rising, TSVM takes a small NSV, SWS/ITER and TNI. Overall, TSVM again runs the fastest.

5.3.3. Comparisons with real data

(c) Real data without outliers. We now employ ten solvers to address Example 5.3 and depict the average results of ten solvers in Table 4, where ">5e6" means the number greater than 5 000 000. From Table 4, with regard to ACC and NSV, we can see that our TSVM outperforms the other nine solvers because it yields the best ACC and the smallest NSV. In terms of SWS/ITER, we can observe that TSVM takes advantage of a small samples as the working set, which shows that our constructed working set strategy is very effective in reducing the computational cost.

Table 2
Results of 10 solvers for addressing Example 5.2.

r	ACC (%)									
	TSVM	HSVM	PSVM	ISVM	RSVM	OSVM	VSVM	ASVM	BSVM	RELO
0.00	**97.16**	97.08	97.03	**97.16**	**97.16**	97.12	**97.16**	**97.16**	97.10	97.08
0.05	**92.65**	92.46	92.54	92.60	**92.65**	92.57	92.30	92.35	92.50	92.58
0.10	**87.96**	87.78	87.68	87.90	87.90	87.90	87.46	87.45	87.78	87.70
0.15	**83.06**	82.86	82.98	82.98	**83.06**	83.04	82.88	82.88	82.80	82.93
0.20	**78.29**	78.16	78.21	78.28	78.28	78.20	78.17	78.18	78.12	78.16
	NSV									
0.00	**38**	364	372	5000	184	329	359	357	5000	5000
0.05	**35**	947	942	5000	175	874	953	945	5000	5000
0.10	**32**	1385	1365	5000	170	1015	1373	1389	5000	5000
0.15	**31**	1795	1790	5000	161	1657	1781	1792	5000	5000
0.20	**28**	2160	2177	5000	137	1989	2175	2187	5000	5000
	SWS/ITER									
0.00	58	2	**1**	5000	5000	5000	**1**	**1**	5000	5000
0.05	54	2	**1**	5000	5000	5000	**1**	**1**	5000	5000
0.10	53	2	**1**	5000	5000	5000	**1**	**1**	5000	5000
0.15	51	2	**1**	5000	5000	5000	**1**	**1**	5000	5000
0.20	49	2	**1**	5000	5000	5000	**1**	**1**	5000	5000
	TNI									
0.00	34	584	10 000	3042	3(25)	3(21)	10 000	10 000	15	**9**
0.05	32	3726	10 000	3126	4(18)	3(21)	10 000	10 000	15	**9**
0.10	31	5128	10 000	3268	4(17)	3(21)	10 000	10 000	15	**9**
0.15	28	8423	10 000	3373	5(13)	3(21)	10 000	10 000	15	**9**
0.20	27	10 776	10 000	3443	5(13)	3(21)	10 000	10 000	15	**9**
	CPU (s)									
0.00	**0.016**	0.027	0.117	93.11	22.53	4047	0.108	0.112	0.801	0.149
0.05	**0.016**	0.075	0.119	101.3	20.99	4069	0.114	0.115	0.823	0.131
0.10	**0.015**	0.123	0.118	105.4	19.43	4084	0.111	0.112	0.853	0.147
0.15	**0.015**	0.172	0.118	108.3	18.96	4092	0.110	0.111	0.885	0.152
0.20	**0.014**	0.236	0.119	110.6	18.41	4094	0.115	0.116	0.898	0.165

Table 3
Descriptions of fourteen real datasets.

Datasets	Source	Training size m	Testing size m_t	Feature n
Colon-cancer (col)	libsvm	62	0	2000
Australian (aus)	libsvm	690	0	14
Two-norm (two)	uci	7 400	0	20
Mushrooms (mus)	uci	8 124	0	112
Adult (adu)	uci	17 887	0	13
Covtype.binaty (cov)	uci	581 012	0	54
SUSY (sus)	uci	5 000 000	0	18
HIGGS (hig)	uci	11 000 000	0	28
Leukemia (lek)	libsvm	38	34	7129
Splice (spl)	libsvm	1 000	2 175	60
A6a (a6a)	uci	11 220	21 341	123
W6a (w6a)	libsvm	17 188	32 561	300
W8a (w8a)	libsvm	49 749	14 951	300
ijcnn1 (ijc)	libsvm	49 990	91 701	22

With respect to TNI, TSVM adopts a few TNI compared with ISVM, PSVM, VSVM and ASVM. In terms of CPU, PSVM, VSVM and ASVM enjoys the advantage for addressing datasets in small scales, while the TSVM runs super fast for handling big size datasets. Such as, for ijc, TSVM only takes advantage of 0.738 s, while HSVM needs 36.95 s. In addition, in terms of data hig with more than ten million sizes, we can observe that TSVM only takes 19.74 s. Overall, TSVM outperforms the other solvers in terms of the shortest CPU, highest ACC, and smallest NSV for most real datasets.

(d) Real data with outliers. To observe robustness to outliers of each solvers, we adopt these solvers to handle Example 5.4 with altering $r = 0.01, 0.02, \ldots, 0.1$. Apparently, for datasets: mus, adu, two, and a6a, OSVM takes advantage of too long time. Hence, we omit results related to these datasets. With regard to ACC shown in Fig. 7, ACC yielded by all solvers decline with the ascending of r, and TSVM receives the highest ACC. With respect to NSV presented in Fig. 8, it is evidently that BSVM, ISVM and RELO always take advantage of all samples as support vectors, while we can see that lines from TSVM and RSVM either decline or stabilize at a level with r rising, which represents the two solvers are quite robust to outliers. Moreover, TSVM always adopts the fewest NSV. With respect to SWS/ITER in Fig. 9, with the ascending of r, TSVM stabilizes at a level for all real datasets. In terms of TNI in Fig. 10, we can see that with the rising of r, there is no big difference among the these solvers except for HSVM. In terms of CPU, as presented in Fig. 11, except for datasets col and aus with small sizes, TSVM outperforms the other solvers for other datasets.

6. Conclusion

In this paper, we established a new sparse and robust SVM model with truncated Huber loss: L_{th}-SVM and established the first-order necessary and sufficient conditions for the local minimizer of L_{th}-SVM. Furthermore, we designed a new effective algorithm equipped with working set for solving L_{th}-SVM. Theoretically, we proved that the sequence generated by our designed algorithm converges to a local minimizer of L_{th}-SVM and our designed algorithm enjoys a relatively low computational complexity. Numerically, our designed algorithm is capable of running very fast, presenting fewest support vectors and rendering more desirable accuracy to against some other state-of-the-art solvers. Moreover, we strongly feel that the techniques developed in this paper can be used to address L_{th}-SVM with nonlinear kernel,

Table 4
Results of 10 solvers for addressing Example 5.3.

Data	ACC (%)									
	TSVM	HSVM	PSVM	ISVM	RSVM	OSVM	VSVM	ASVM	BSVM	RELO
col	**90.27**	64.52	89.68	77.69	89.68	85.87	89.68	89.68	85.48	86.74
aus	**86.28**	85.51	86.04	85.80	86.02	85.98	86.18	86.23	85.80	86.18
lek	**82.35**	58.82	**82.35**	58.82	76.47	**82.35**	**82.35**	**82.35**	79.41	**82.35**
spl	**88.97**	**88.97**	84.18	85.52	85.47	85.47	85.44	85.33	85.75	85.15
two	**98.37**	98.02	98.10	97.97	98.24	**	**98.37**	98.24	97.97	97.78
mus	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**	**100.0**	**100.0**	**100.0**	**100.0**
adu	**83.94**	83.29	83.29	83.07	83.79	**	83.34	83.90	83.01	82.95
a6a	**84.92**	84.18	84.36	84.69	84.72	**	84.72	84.78	84.55	84.76
w6a	**97.89**	97.21	97.24	97.21	97.86	**	97.61	97.57	97.58	95.13
w8a	**98.66**	98.27	97.43	**	**	**	97.57	97.59	**	**
ijc	**94.36**	92.73	93.49	**	**	**	93.35	93.56	**	**
cov	**72.27**	**	68.93	**	**	**	69.83	69.77	**	**
sus	**67.88**	**	64.28	**	**	**	65.62	65.86	**	**
hig	**65.96**	**	58.12	**	**	**	59.13	59.46	**	**
	NSV									
col	**33**	46	46	54	38	40	45	46	54	54
aus	**29**	203	198	621	89	177	195	202	621	621
lek	**27**	31	33	38	29	31	34	31	38	38
spl	**81**	607	632	1000	87	332	615	612	1000	1000
two	**53**	758	783	6600	108	**	775	788	6600	6600
mus	**152**	550	578	7311	506	**	575	568	7311	7311
adu	**174**	6379	6407	16 098	1247	**	6386	6394	16 098	16 098
a6a	**378**	4346	4562	11 220	1247	**	4575	4582	11 220	11 220
w6a	**466**	1128	1146	17 188	946	**	1152	1138	17 188	17 188
w8a	**794**	2857	2582	**	**	**	2579	2561	**	**
ijc	**315**	8508	8535	**	**	**	8612	8608	**	**
cov	**338**	**	>3e5	**	**	**	>3e5	>3e5	**	**
sus	**812**	**	>2e6	**	**	**	>2e6	>2e6	**	**
hig	**1741**	**	>5e6	**	**	**	>5e6	>5e6	**	**
	SWS/ITER									
col	41	2	**1**	54	54	54	**1**	**1**	54	54
aus	74	2	**1**	621	621	621	**1**	**1**	621	621
lek	32	2	**1**	38	38	38	**1**	**1**	38	38
spl	121	2	**1**	1000	1000	1000	**1**	**1**	1000	1000
two	155	2	**1**	6600	6600	**	**1**	**1**	6600	6600
mus	793	2	**1**	7311	7311	**	**1**	**1**	7311	7311
adu	1188	2	**1**	16 098	16 098	**	**1**	**1**	16 098	16 098
a6a	643	2	**1**	11 220	11 220	**	**1**	**1**	11 220	11 220
w6a	712	2	**1**	17 188	17 188	**	**1**	**1**	17 188	17 188
w8a	1421	2	**1**	**	**	**	**1**	**1**	**	**
ijc	918	2	**1**	**	**	**	**1**	**1**	**	**
cov	1726	**	**1**	**	**	**	**1**	**1**	**	**
sus	3039	**	**1**	**	**	**	**1**	**1**	**	**
hig	3824	**	**1**	**	**	**	**1**	**1**	**	**
	TNI									
col	30	41	108	31	2(2)	2(4)	108	108	**2**	4
aus	26	423	1242	869	2(7)	3(26)	1242	1242	17	**6**
lek	27	89	76	42	2(2)	3(17)	76	76	**2**	25
spl	71	595	2000	1276	2(9)	4(28)	2000	2000	28	**9**
two	62	660	13 200	3417	4(11)	**	13 200	13 200	75	**12**
mus	41	1623	14 622	3685	4(12)	**	14 622	14 622	106	**18**
adu	38	4766	32 196	7720	5(21)	**	32 196	32 196	157	**15**
a6a	188	3032	22 440	6873	5(27)	**	22 440	22 440	289	**16**
w6a	131	1450	34 376	14 417	7(32)	**	34 376	34 376	404	**28**
w8a	**163**	8124	99 498	**	**	**	99 498	99 498	**	**
ijc	**172**	6681	99 980	**	**	**	99 980	99 980	**	**
cov	**155**	**	$1.05e^6$	**	**	**	$1.05e^6$	$1.05e^6$	**	**
sus	**165**	**	$9.0e^6$	**	**	**	$9.0e^6$	$9.0e^6$	**	**
hig	**148**	**	$1.98e^7$	**	**	**	$1.98e^7$	$1.98e^7$	**	**

(*continued on next page*)

multiclass SVM, structural SVM and deep learning models. We will explore more on this topic in future.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

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

Acknowledgments

This work is supported by the National Natural Science Foundation of China (11971052, 11926348-9, 61866010, 11871183).

Table 4 (*continued*).

	CPU (s)									
col	0.031	0.009	0.015	0.010	0.003	1.488	0.012	0.014	**0.001**	0.182
aus	0.011	0.014	**0.004**	0.874	0.650	87.23	**0.004**	**0.004**	0.033	0.021
lek	0.056	0.057	0.029	0.010	0.008	54.36	0.024	0.026	**0.004**	36.10
spl	0.071	0.117	0.036	7.976	0.631	384.2	**0.032**	0.033	0.083	0.151
two	**0.072**	0.265	0.171	516.7	139.2	>2 h	0.164	0.166	2.506	1.591
mus	**0.091**	0.997	0.422	769.5	153.4	>2 h	0.412	0.416	3.419	6.942
adu	**0.812**	3.775	0.775	1633.4	1013.2	>2 h	0.732	0.744	24.58	5.032
a6a	**0.272**	4.405	1.083	1472.5	1037.3	>2 h	1.025	1.031	40.64	6.046
w6a	**0.313**	1.532	1.314	5947.2	2747.4	>2 h	1.186	1.232	170.9	41.21
w8a	**2.916**	64.33	4.863	–	>2 h	>2 h	4.227	4.316	–	–
ijc	**0.738**	36.95	1.526	–	>2 h	>2 h	1.247	1.316	–	–
cov	**4.137**	–	14.37	–	–	–	13.88	13.91	–	–
sus	**13.48**	–	137.6	–	–	–	132.4	133.7	–	–
hig	**19.74**	–	281.3	–	–	–	269.5	270.1	–	–

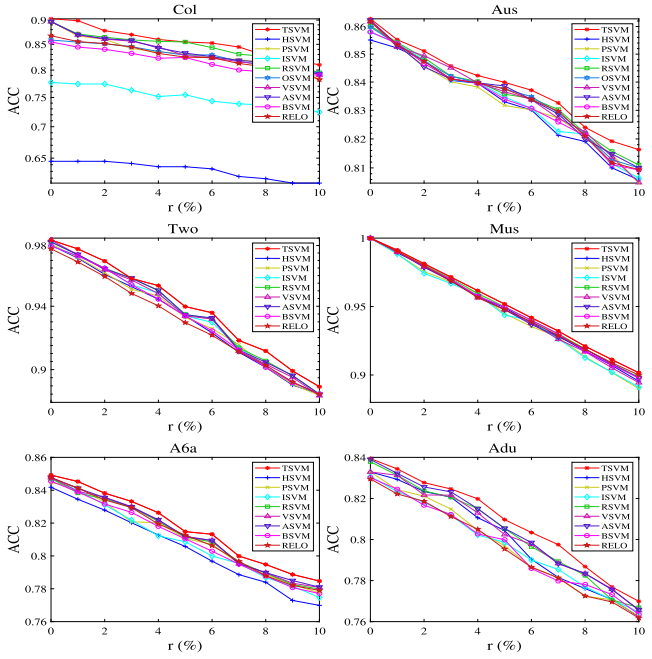


Fig. 7. ACC vs. r of all solvers for addressing six datasets.

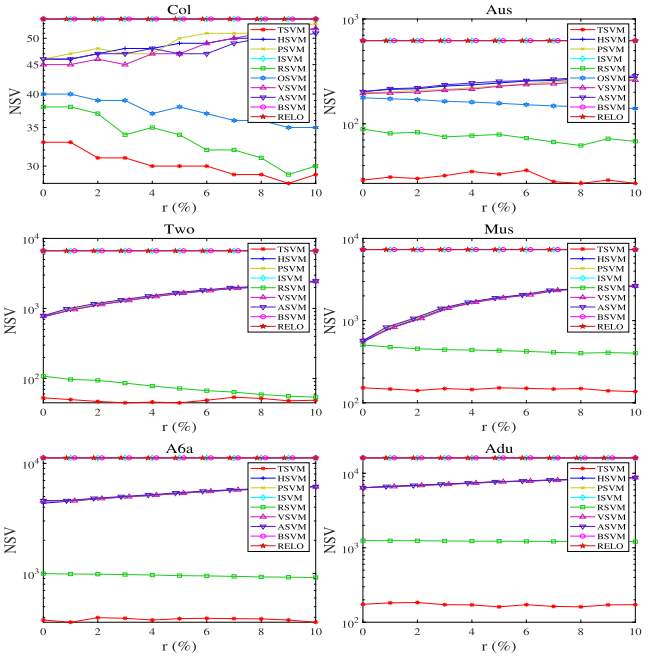


Fig. 8. NSV vs. r of all solvers for addressing six datasets.

Appendix. Proofs of all theorems

A.1. Proof of Lemma 3.1

Proof. From (7), we can yield that the $L_{th}(\mathbf{h})$ is separable, which allows us to obtain (9). Next, we show the explicit expression (10) of ℓ_{th} limiting subdifferential by two cases.

(i) For $h_i \neq 1+\delta/2$, the $\ell_{th}(h_i)$ is differentiable. Therefore, we yield $\partial\ell_{th}(h_i) = 0$ for $h_i < 0$ or $h_i > 1+\delta/2$, $\partial\ell_{th}(h_i) = t/\delta$ for $h_i \in [0, \delta]$ and $\partial\ell_{th}(h_i) = 1$ for $h_i \in (\delta, 1+\delta/2)$.

(ii) For $h_i = 1+\delta/2$, the $\ell_{th}(h_i)$ is not differentiable. Since $\partial\ell_{th}(1+\delta/2) = \limsup_{h_i \to 1+\delta/2} \widehat{\partial} f(h_i)$, from Definition 3.1, we have $\widehat{\partial}\ell_{th}(h_i) = 1$ with $h_i \in (\delta, 1+\delta/2)$ and $\widehat{\partial}\ell_{th}(h_i) = 0$ with $h_i > 1+\delta/2$. Hence, we obtain $\partial\ell_{th}(1+\delta/2) \in \{0, 1\}$.

By the (i) and (ii), we yield (10). This completes the proof. □

A.2. Proof of Lemma 3.2

Proof. From (5) and (11), we yield that $\text{prox}_{\kappa\gamma\ell_{th}}(z)$ is the minimizer of following function

$$\psi(u) := \begin{cases} \psi_1(u) := \kappa\gamma + \frac{(u-z)^2}{2}, & u > 1+\delta/2, \\ \psi_2(u) := \kappa\gamma + \frac{(1+\delta/2-z)^2}{2}, & u = 1+\delta/2, \\ \psi_3(u) := \kappa\gamma(u-\delta/2) + \frac{(u-z)^2}{2}, & u \in [\delta, 1+\delta/2), \\ \psi_4(u) := \kappa\gamma\frac{u^2}{2\delta} + \frac{(u-z)^2}{2}, & u \in (0, \delta), \\ \psi_5(u) := \frac{(u-z)^2}{2}, & u \leq 0. \end{cases}$$

It is evidently that the minimizers of $\psi_1(u)$, $\psi_2(u)$, $\psi_3(u)$, $\psi_4(u)$ and $\psi_5(u)$ are attained at $u_1^* = z$, $u_2^* = 1+\delta/2$, $u_3^* = z-\kappa\gamma$, $u_4^* = \frac{\delta z}{\delta+\kappa\gamma}$ and $u_5^* = z$ respectively.

(i) For $\delta + \kappa\gamma \in (0, 2)$, by comparing the five values of $\psi(u_1^*)$, $\psi(u_2^*)$, $\psi(u_3^*)$, $\psi(u_4^*)$ and $\psi(u_5^*)$, we obtain the following observation:

(a1) As $z > 1 + \frac{\delta+\kappa\gamma}{2}$, we get $\min\{\psi(u_2^*), \psi(u_3^*), \psi(u_4^*), \psi(u_5^*)\} > \psi(u_1^*)$, which implies $u^* = u_1^* = z$.

(a2) As $z = 1 + \frac{\delta+\kappa\gamma}{2}$, we have $\min\{\psi(u_2^*), \psi(u_4^*), \psi(u_5^*)\} > \psi(u_1^*) = \psi(u_3^*)$, which means $u^* = u_1^* = z$ or $u^* = u_3^* = z-\kappa\gamma$.

(a3) As $z \in (\delta+\kappa\gamma, 1+\frac{\delta+\kappa\gamma}{2})$, we yield $\min\{\psi(u_1^*), \psi(u_2^*), \psi(u_4^*), \psi(u_5^*)\} > \psi(u_3^*)$, which implies $u^* = u_3^* = z-\kappa\gamma$.

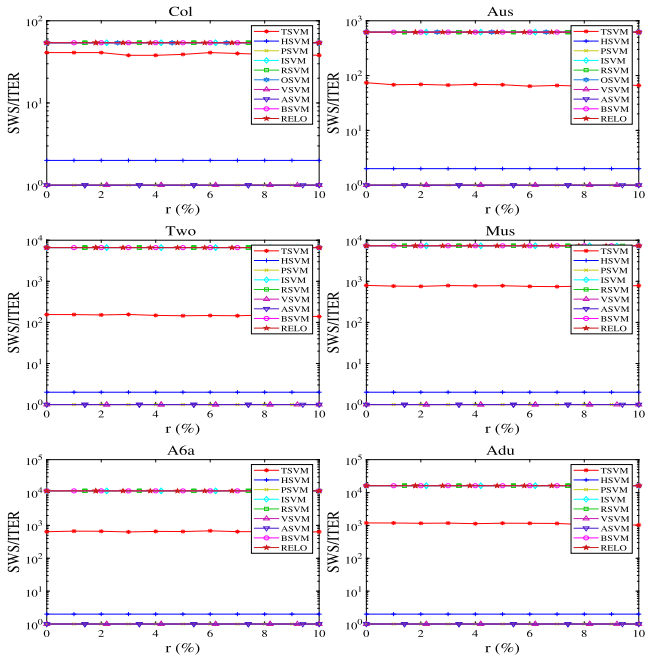


Fig. 9. SWS/ITER vs. r of all solvers for solving six datasets.

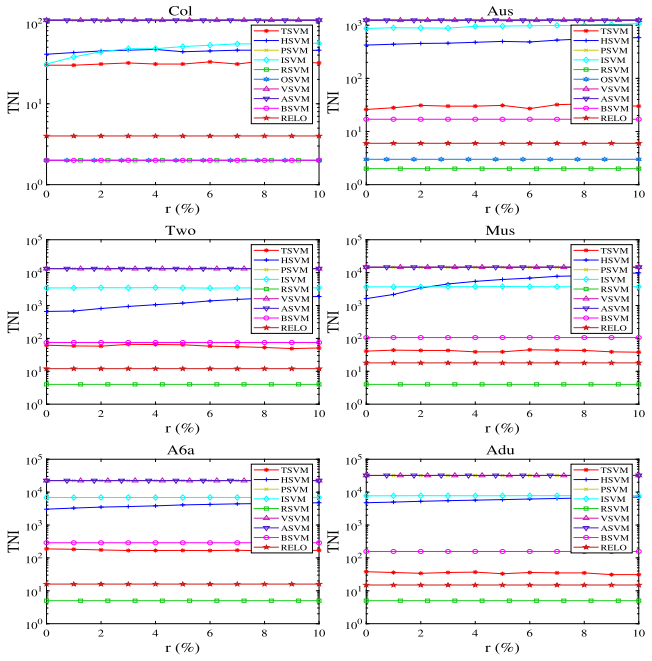


Fig. 10. TNI vs. r of all solvers for solving six datasets.

(a4) As $z \in (0, \delta + \kappa\gamma]$, we obtain $\min\{\psi(u_1^*), \psi(u_2^*), \psi(u_3^*), \psi(u_5^*)\} > \psi(u_4^*)$, which means $u^* = u_4^* = \frac{\delta z}{\delta+\kappa\gamma}$.
(a5) As $z \leq 0$, we receive $\min\{\psi(u_1^*), \psi(u_2^*), \psi(u_3^*), \psi(u_4^*)\} > \psi(u_5^*)$, which implies $u^* = u_5^* = z$.

Summarizing the above analysis, we yield (12).

(ii) For $\delta + \kappa\gamma \geq 2$, by contrasting the five values of $\psi(u_1^*)$, $\psi(u_2^*)$, $\psi(u_3^*)$, $\psi(u_4^*)$ and $\psi(u_5^*)$, we get desired conclusion:
(b1) As $z > \sqrt{2(\delta + \kappa\gamma)}$, we get $\min\{\psi(u_2^*), \psi(u_3^*), \psi(u_4^*), \psi(u_5^*)\} > \psi(u_1^*)$, which implies $u^* = u_1^* = z$.
(b2) As $z = \sqrt{2(\delta + \kappa\gamma)}$, we have $\min\{\psi(u_2^*), \psi(u_3^*), \psi(u_5^*)\} > \psi(u_1^*) = \psi(u_4^*)$, which gets $u^* = u_1^* = z$ or $u^* = u_4^* = \frac{\delta z}{\delta+\kappa\gamma}$.

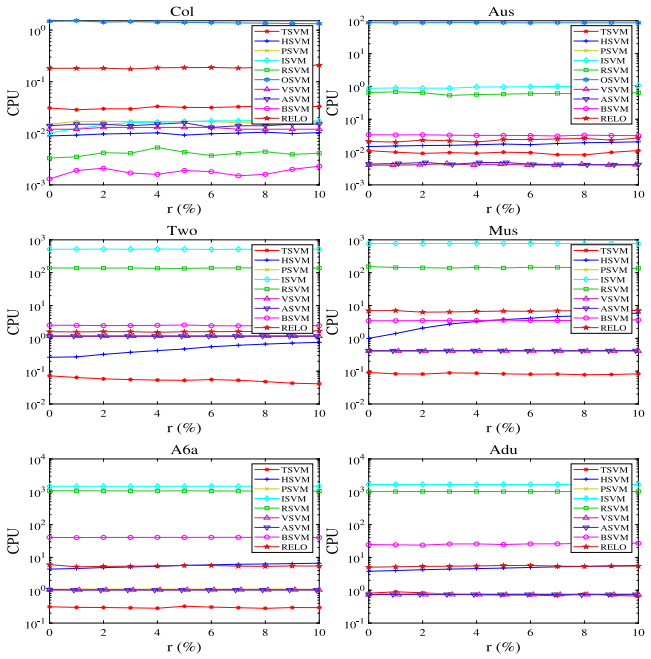


Fig. 11. CPU vs. r of all solvers for addressing six datasets.

(b3) As $z \in [0, \sqrt{2(\delta + \kappa\gamma)})$, we obtain $\min\{\psi(u_1^*), \psi(u_2^*), \psi(u_3^*), \psi(u_5^*)\} > \psi(u_4^*)$, which means $u^* = u_4^* = \frac{\delta z}{\delta+\kappa\gamma}$.
(b4) As $z < 0$, we receive $\min\{\psi(u_1^*), \psi(u_2^*), \psi(u_3^*), \psi(u_4^*)\} > \psi(u_5^*)$, which implies $u^* = u_5^* = z$.

Summarizing the above analysis, we yield (13). This completes the proof. □

A.3. Proof of Lemma 3.3

Proof. From the separate property of $L_{th}(\mathbf{u})$, we can obtain that $[\text{prox}_{\kappa\gamma L_{th}}(\mathbf{z})]_i = \text{prox}_{\kappa\gamma\ell_{th}}(z_i)$, where

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

Thus, we yield (15). This completes the proof. □

A.4. Proof of Theorem 3.1

Proof. Because $(\mathbf{w}^*; b^*; \mathbf{h}^*)$ is a local minimizer of (8), from [38, Theorem 10.1], we can yield that there exists a $\boldsymbol{\alpha}^*$ such that $(\mathbf{w}^*; b^*; \mathbf{h}^*)$ is a KKT point of (8), i.e.,

$$\begin{cases} \mathbf{w}^* + G^\top\boldsymbol{\alpha}^* &= \mathbf{0}, \\ \langle \mathbf{y}, \boldsymbol{\alpha}^* \rangle &= 0, \\ \mathbf{h}^* + G\mathbf{w}^* + b^*\mathbf{y} &= \mathbf{1}, \\ \boldsymbol{\alpha}^* + \gamma\,\partial L_{th}(\mathbf{h}^*) &\ni \mathbf{0}. \end{cases} \tag{33}$$

Hence, to prove (16), we only need to proof that if $(\boldsymbol{\alpha}^*; \mathbf{h}^*)$ satisfies $\mathbf{0} \in \boldsymbol{\alpha}^* + \gamma\,\partial L_{th}(\mathbf{h}^*)$ for any $0 < \kappa \leq \kappa^*$, then we yield $\mathbf{h}^* \in \mathbb{P} := \text{prox}_{\kappa\gamma L_{th}}(\mathbf{h}^* - \kappa\boldsymbol{\alpha}^*)$. From (9), (10) and $\mathbf{0} \in \boldsymbol{\alpha}^* + \gamma\,\partial L_{th}(\mathbf{h}^*)$, we have that

$$\alpha_i^* \in \begin{cases} 0, & h_i^* > 1 + \delta/2, \\ \{-\gamma, 0\}, & h_i^* = 1 + \delta/2, \\ -\gamma, & h_i^* \in (\delta, 1 + \delta/2), \\ -\gamma h_i^*/\delta, & h_i^* \in [0, \delta], \\ 0, & h_i^* < 0. \end{cases} i \in [m]. \tag{34}$$

To show (16), we split the proof into the following two cases: $\delta + \kappa\gamma \in (0, 2)$ and $\delta + \kappa\gamma \geq 2$.

Case (i): For $\gamma > 0, \delta \in (0,2)$ and $\kappa \in (0,\kappa^*]$ satisfying $\delta+\kappa\gamma \in (0,2)$, because $\mathbb{E}^* = \emptyset$, we only need consider the following four scenarios.
(i) For $i \in \mathbb{S}^*$, we have $h_i^* > 1+\delta/2$ and yield $\alpha_i^* = 0$ by (34), which implies that

$$p_i^* := h_i^* - \kappa\alpha_i^* = h_i^* \geq 0. \tag{35}$$

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

$$\kappa \leq \frac{2(h_i^*-1)-\delta}{\gamma} \overset{(35)}{=} \frac{2(p_i^*-1)-\delta}{\gamma},$$

which means that

$$p_i^* \geq 1+\frac{\delta+\kappa\gamma}{2}.$$

(ii) For $i \in \mathbb{T}^*$, we have $h_i^* \in (\delta, 1+\delta/2)$ and get $\alpha_i^* = -\gamma < 0$ from (34). Hence, we obtain

$$p_i^* = h_i^* - \kappa\alpha_i^* = h_i^* + \kappa\gamma > \delta+\kappa\gamma. \tag{36}$$

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

$$\kappa \leq \frac{2(1+\delta/2-h_i^*)}{\gamma},$$

which means

$$p_i^* = h_i^* - \kappa\alpha_i^* \leq 1+\frac{\delta+\kappa\gamma}{2}.$$

This together with (36) results in

$$p_i^* \in (\delta+\kappa\gamma, 1+\frac{\delta+\kappa\gamma}{2}].$$

(iii) For $i \in \mathbb{I}^*$, from (34), we have $h_i^* \in (0,\delta]$ and $\alpha_i^* = -\gamma h_i^*/\delta$ from (34), which implies

$$p_i^* = h_i^* - \kappa\alpha_i^* = h_i^* + \gamma\kappa h_i^*/\delta \in (0, \delta+\kappa\gamma].$$

(iv) For $i \in \mathbb{O}^*$, from (34), we have $h_i^* \leq 0$ and $\alpha_i^* = 0$, which results in

$$p_i^* = h_i^* - \kappa\alpha_i^* = h_i^* \leq 0.$$

From the above (i)-(iv), we obtain that

$$h_i^* \in \mathbb{P}_i := \begin{cases} p_i^*, & p_i^* > 1+\frac{\delta+\kappa\gamma}{2}, \\ p_i^* \text{ or } p_i^*-\kappa\gamma, & p_i^* = 1+\frac{\delta+\kappa\gamma}{2}, \\ p_i^*-\kappa\gamma, & p_i^* \in (\delta+\kappa\gamma, 1+\frac{\delta+\kappa\gamma}{2}), \\ \frac{\delta p_i^*}{\delta+\kappa\gamma}, & p_i^* \in (0, \delta+\kappa\gamma], \\ p_i^*, & p_i^* \leq 0. \end{cases}$$

which completes the proof of Case (i).

Case (ii): For $\gamma > 0, \delta \in (0,2)$ and $\kappa \in (0,\kappa^*]$ satisfying $\delta+\kappa\gamma \geq 2$, we only consider the following three cases.
(i) For $i \in T^* := \mathbb{S}^* \cup \mathbb{E}^* \cup \mathbb{T}^*$, by $\kappa \leq \kappa_3^*$, we have

$$\kappa \leq \kappa_3^* \leq \frac{(h_i^*)^2-2\delta}{2\gamma},$$

which together with $\delta \in (0,2)$ and $\delta+\kappa\gamma \geq 2$, we obtain

$$h_i^* \geq \sqrt{2(\delta+\kappa_3^*\gamma)} \geq \sqrt{2(\delta+\kappa\gamma)} > 1+\delta/2.$$

This combine with (34), we yield $\alpha_i^* = 0$ and obtain

$$p_i^* = h_i^* - \kappa\alpha_i^* = h_i^* \geq \sqrt{2(\delta+\kappa\gamma)}.$$

(ii) For $i \in \mathbb{I}^*$, we have $h_i^* \in (0,\delta]$ and $\alpha_i^* = -\gamma h_i^*/\delta$ from (34). From $\kappa \leq \kappa_4^*$, we obtain

$$\kappa \leq \frac{2\delta^2-\delta(h_i^*)^2}{\gamma(h_i^*)^2},$$

which means

$$p_i^* = h_i^* - \kappa\alpha_i^* \in (0, \sqrt{2(\delta+\kappa\gamma)}].$$

(iii) For $i \in \mathbb{O}^*$, we have $h_i^* \leq 0$ and $\alpha_i^* = 0$, which means that

$$p_i^* = h_i^* - \kappa\alpha_i^* = h_i^* \leq 0.$$

From the above (i)-(iii), we have

$$h_i^* \in \mathbb{P}_i := \begin{cases} p_i^*, & p_i^* > \sqrt{2(\delta+\kappa\gamma)}, \\ p_i^* \text{ or } \frac{\delta p_i^*}{\delta+\kappa\gamma}, & p_i^* = \sqrt{2(\delta+\kappa\gamma)}, \\ \frac{\delta p_i^*}{\delta+\kappa\gamma}, & p_i^* \in (0, \sqrt{2(\delta+\kappa\gamma)}), \\ p_i^*, & p_i^* \leq 0, \end{cases}$$

which completes the proof of Case (ii). □

A.5. Proof of Theorem 3.2

Proof. Define $\Upsilon := \{\boldsymbol{\varphi} := (\mathbf{w}; b; \mathbf{h}) : \mathbf{h} + G\mathbf{w} + b\mathbf{y} = \mathbf{1}\}$ and $\boldsymbol{\varphi}^* := (\mathbf{w}^*; b^*; \mathbf{h}^*)$. Then, for any $\boldsymbol{\varphi} \in \Upsilon$, we can yield $\mathbf{h} + G\mathbf{w} + b\mathbf{y} = \mathbf{1}$, which combine with (16) results in

$$-G(\mathbf{w}-\mathbf{w}^*) = \mathbf{h} - \mathbf{h}^* + (b-b^*)\mathbf{y}. \tag{37}$$

Based on the convexity of $\|\mathbf{w}\|^2$, we receive that

$$\begin{aligned} \|\mathbf{w}\|^2 - \|\mathbf{w}^*\|^2 &\geq 2\langle \mathbf{w}-\mathbf{w}^*, \mathbf{w}^*\rangle \\ &\overset{(16)}{=} -2\langle \boldsymbol{\alpha}^*, G(\mathbf{w}-\mathbf{w}^*)\rangle \\ &\overset{(37)}{=} 2\langle \boldsymbol{\alpha}^*, \mathbf{h}-\mathbf{h}^*\rangle + 2(b-b^*)\langle \boldsymbol{\alpha}^*, \mathbf{y}\rangle \\ &\overset{(16)}{=} 2\langle \boldsymbol{\alpha}^*, \mathbf{h}-\mathbf{h}^*\rangle. \end{aligned} \tag{38}$$

Based on the above result, we now show that $\boldsymbol{\varphi}^*$ is a local minimizer of problem (8). Namely, there is a neighborhood $N(\boldsymbol{\varphi}^*, \eta)$ of $\boldsymbol{\varphi}^* \in \Upsilon$ with $\eta > 0$ such that for any $\boldsymbol{\varphi} \in \Upsilon \cap N(\boldsymbol{\varphi}^*, \eta)$, we get

$$\frac{1}{2}\|\mathbf{w}\|^2 + \gamma L_{th}(\mathbf{h}) \geq \frac{1}{2}\|\mathbf{w}^*\|^2 + \gamma L_{th}(\mathbf{h}^*). \tag{39}$$

For this purpose, we denote

$$\widehat{\eta} := \begin{cases} +\infty, & \mathbf{h}^* \leq 0, \\ \min_i\{h_i^*, h_i^* > 0\}, & \text{otherwise}, \end{cases} \tag{40}$$

$$\widetilde{\eta} := \delta - \sqrt{2\delta^2/(\delta+\kappa\gamma)}, \overline{\eta} := \sqrt{2(\delta+\kappa\gamma)} - (1+\delta/2),$$

$$\eta := \begin{cases} \min\{\frac{\kappa\gamma}{2}, \widehat{\eta}\}, & \delta+\kappa\gamma \in (0,2), \\ \min\{\widehat{\eta}, \widetilde{\eta}, \overline{\eta}\}, & \delta+\kappa\gamma \geq 2, \end{cases}, \quad \eta_m := \eta/\sqrt{2m}.$$

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

$$N(\boldsymbol{\varphi}^*, \eta) := \{\boldsymbol{\varphi} : \|\boldsymbol{\varphi}-\boldsymbol{\varphi}^*\| \leq \eta/\sqrt{2}, |h_i - h_i^*| \leq \eta_m\}. \tag{41}$$

To proof (39), we consider the following two scenarios.

Case (i): For $\delta+\kappa\gamma \in (0,2)$ and $\boldsymbol{\varphi} \in \Upsilon \cap N(\boldsymbol{\varphi}^*, \eta)$. Define $\mathbf{s}^* := \mathbf{h}^* - \kappa\boldsymbol{\alpha}^* \in \mathbb{N}_m$, and

$$\begin{cases} C_1^* := \{i \in [m] : s_i^* \leq 0\}, \\ C_2^* := \{i \in [m] : s_i^* \in (0, \delta+\kappa\gamma)\}, \\ C_3^* := \{i \in [m] : s_i^* \in [\delta+\kappa\gamma, 1+\frac{\delta+\kappa\gamma}{2})\} \\ C_4^* := \{i \in [m] : s_i^* = 1+\frac{\delta+\kappa\gamma}{2}, \alpha_i^* \neq 0\}, \\ C_5^* := \{i \in [m] : s_i^* = 1+\frac{\delta+\kappa\gamma}{2}, \alpha_i^* = 0\}, \\ C_6^* := \{i \in [m] : s_i^* > 1+\frac{\delta+\kappa\gamma}{2}\} \\ H_1^* = C_1^*, H_2^* = C_2^*, H_3^* = C_3^* \cup C_4^*, H_4^* = C_5^* \cup C_6^*. \end{cases} \tag{42}$$

From (12), (15) and (42), we get that $\mathbf{h}^* \in \text{prox}_{\kappa\gamma L_{th}}(\mathbf{h}^* - \kappa\boldsymbol{\alpha}^*)$ is equivalent to

$$\mathbf{h}^* = \begin{bmatrix} (\mathbf{h}^* - \kappa\boldsymbol{\alpha}^*)_{H_1^*} \\ \frac{\delta}{\delta+\kappa\gamma}(\mathbf{h}^* - \kappa\boldsymbol{\alpha}^*)_{H_2^*} \\ (\mathbf{h}^* - \kappa\boldsymbol{\alpha}^* - \kappa\gamma\mathbf{1})_{H_3^*} \\ (\mathbf{h}^* - \kappa\boldsymbol{\alpha}^*)_{H_4^*} \end{bmatrix},$$

which is identical to

$$\boldsymbol{\alpha}^*_{H_1^*} = \mathbf{0}_{H_1^*},\ \boldsymbol{\alpha}^*_{H_2^*} = -\frac{\gamma}{\delta}\mathbf{h}^*_{H_2^*},\ \boldsymbol{\alpha}^*_{H_3^*} = -\gamma\mathbf{1}_{H_3^*},\ \boldsymbol{\alpha}^*_{H_4^*} = \mathbf{0}_{H_4^*}.$$

This combine with (42) results in

$$\begin{aligned} &\alpha_i^* = 0, h_i^* \le 0, i \in H_1^*, \\ &\alpha_i^* = -\frac{\gamma h_i^*}{\delta}, h_i^* \in (0, \delta), i \in H_2^*, \\ &\alpha_i^* = -\gamma, h_i^* \in [\delta, 1 + \frac{\delta - \kappa\gamma}{2}], i \in H_3^*, \\ &\alpha_i^* = 0, h_i^* \ge 1 + \frac{\delta + \kappa\gamma}{2}, i \in H_4^*. \end{aligned} \tag{43}$$

Define $H^* := H_2^* \cup H_3^*$ and $\overline{H}^* := H_1^* \cup H_4^*$. To prove (39), from (38)–(43), we only need to show the following two results:

$$\gamma L_{th}(\mathbf{h}_{H^*}) - \gamma L_{th}(\mathbf{h}^*_{H^*}) + \langle \boldsymbol{\alpha}^*_{H^*}, \mathbf{h}_{H^*} - \mathbf{h}^*_{H^*}\rangle \ge 0, \tag{44}$$

$$\gamma L_{th}(\mathbf{h}_{\overline{H}^*}) - \gamma L_{th}(\mathbf{h}^*_{\overline{H}^*}) \ge 0. \tag{45}$$

It is from (5) and (40)–(43) that we yield the following desired conclusion:

(i) For $i \in H^* = H_2^* \cup H_3^*$, we have $h_i^* \in (0, 1 + \frac{\delta-\kappa\gamma}{2}]$. From $|h_i - h_i^*| \le \eta_m$ and $\eta = \min\{\frac{\kappa\gamma}{2}, \widehat{\eta}\}$ for $\delta + \kappa\gamma \in (0, 2)$ by (40), for any $\boldsymbol{\varphi} \in \Upsilon \cap N(\boldsymbol{\varphi}^*, \eta)$, we obtain $h_i \in (0, 1 + \frac{\delta}{2})$, which means that the $\gamma L_{th}(\mathbf{h}_{H^*})$ is a local convex function in $\Upsilon \cap N(\boldsymbol{\varphi}^*, \eta)$. Namely, for any $\boldsymbol{\varphi} \in \Upsilon \cap N(\boldsymbol{\varphi}^*, \eta)$, we get

$$\gamma L_{th}(\mathbf{h}_{H^*}) - \gamma L_{th}(\mathbf{h}^*_{H^*}) - \langle \gamma\nabla L_{th}(\mathbf{h}^*_{H^*}), \mathbf{h}_{H^*} - \mathbf{h}^*_{H^*}\rangle \ge 0.$$

which together with $\boldsymbol{\alpha}^*_{H_2^*} = -\frac{\gamma}{\delta}\mathbf{h}^*_{H_2^*} = -\gamma\nabla L_{th}(\mathbf{h}^*_{H_2^*})$ and $\boldsymbol{\alpha}^*_{H_3^*} = -\gamma\mathbf{1}_{H_3^*} = -\gamma\nabla L_{th}(\mathbf{h}^*_{H_3^*})$ leads to (44).

(ii) For $i \in H_1^*$, from $h_i^* \le 0$ and $|h_i - h_i^*| \le \eta_m$, we get $z_i \le h_i^* + \eta_m$, which implies $\ell_{th}(h_i) \ge \ell_{th}(h_i^*)$. Hence, (45) holds.

(iii) For $i \in H_4^*$, it follows from $h_i^* \ge 1 + \frac{\delta+\kappa\gamma}{2}$ and $|h_i - h_i^*| \le \eta_m$ that we have $h_i \ge h_i^* - \eta_m > 1 + \frac{\delta}{2}$, which obtains $\ell_{th}(h_i) = \ell_{th}(h_i^*)$. Hence (45) holds.

Summarizing the above (i)–(iii), we get (44) and (45). This completes the proof of Case (i).

Case (ii): For $\delta + \kappa\gamma \ge 2$ and $\boldsymbol{\varphi} \in \Upsilon \cap N(\boldsymbol{\varphi}^*, \eta)$. Denote $\mathbf{s}^* = \mathbf{h}^* - \kappa\boldsymbol{\alpha}^* \in \mathbb{N}_m$, and

$$\begin{cases} D_1^* := \{i \in [m] : s_i^* \le 0\}, \\ D_2^* := \{i \in [m] : s_i^* \in (0, \sqrt{2(\delta+\kappa\gamma)})\} \\ D_3^* := \{i \in [m] : s_i^* = \sqrt{2(\delta+\kappa\gamma)}, \alpha_i^* \ne 0\}, \\ D_4^* := \{i \in [m] : s_i^* = \sqrt{2(\delta+\kappa\gamma)}, \alpha_i^* = 0\}, \\ D_5^* := \{i \in [m] : s_i^* > \sqrt{2(\delta+\kappa\gamma)}\}, \\ \mathcal{H}_1^* := D_1^*, \mathcal{H}_2^* := D_2^* \cup D_3^*, \mathcal{H}_3^* := D_4^* \cup D_5^*. \end{cases} \tag{46}$$

From (13), (15) and (46), we get that $\mathbf{h}^* \in \text{prox}_{\kappa\gamma L_{th}}(\mathbf{h}^* - \kappa\boldsymbol{\alpha}^*)$ is equivalent to

$$\mathbf{h}^* = \begin{bmatrix} (\mathbf{h}^* - \kappa\boldsymbol{\alpha}^*)_{\mathcal{H}_1^*} \\ \frac{\delta}{\delta+\kappa\gamma}(\mathbf{h}^* - \kappa\boldsymbol{\alpha}^*)_{\mathcal{H}_2^*} \\ (\mathbf{h}^* - \kappa\boldsymbol{\alpha}^*)_{\mathcal{H}_3^*} \end{bmatrix},$$

which is identical to

$$\boldsymbol{\alpha}^*_{\mathcal{H}_1^*} = \mathbf{0}_{\mathcal{H}_1^*},\quad \boldsymbol{\alpha}^*_{\mathcal{H}_2^*} = -\frac{\gamma}{\delta}\mathbf{h}^*_{\mathcal{H}_2^*},\quad \boldsymbol{\alpha}^*_{\mathcal{H}_3^*} = \mathbf{0}_{\mathcal{H}_3^*}.$$

This together with (46) leads to

$$\begin{aligned} &\alpha_i^* = 0, h_i^* \le 0,\quad i \in \mathcal{H}_1^*, \\ &\alpha_i^* = -\frac{\gamma}{\delta}h_i^*, h_i^* \in (0, \sqrt{2\delta^2/(\delta+\kappa\gamma)}\,],\quad i \in \mathcal{H}_2^*, \\ &\alpha_i^* = 0, h_i^* \ge \sqrt{2(\delta+\kappa\gamma)}, i \in \mathcal{H}_3^*. \end{aligned} \tag{47}$$

Define $\mathcal{H}^* := \mathcal{H}_2^*$, $\overline{\mathcal{H}}^* := \mathcal{H}_1^* \cup \mathcal{H}_3^*$. To prove (39), we only need to show the following two inequalities:

$$\gamma L_{th}(\mathbf{h}_{\mathcal{H}^*}) - \gamma L_{th}(\mathbf{h}^*_{\mathcal{H}^*}) + \langle \boldsymbol{\alpha}^*_{\mathcal{H}^*}, \mathbf{h}_{\mathcal{H}^*} - \mathbf{h}^*_{\mathcal{H}^*}\rangle \ge 0, \tag{48}$$

$$\gamma L_{th}(\mathbf{h}_{\overline{\mathcal{H}}^*}) - \gamma L_{th}(\mathbf{h}^*_{\overline{\mathcal{H}}^*}) \ge 0. \tag{49}$$

From (5), (40), (41) and (46)–(47), we get desired conclusion:

(i) For $i \in \mathcal{H}^*$, from $h_i^* \in (0, \sqrt{2\delta^2/(\delta+\kappa\gamma)}]$, $|h_i - h_i^*| \le \eta_m$ and (40), we get $h_i \in (0, \delta)$, which means that the $\gamma L_{th}(\mathbf{h}_{\mathcal{H}^*})$ is a local convex function in $\Upsilon \cap N(\boldsymbol{\varphi}^*, \eta)$. Namely, for any $\boldsymbol{\varphi} \in \Upsilon \cap N(\boldsymbol{\varphi}^*, \eta)$, we get

$$\gamma L_{th}(\mathbf{h}_{\mathcal{H}^*}) - \gamma L_{th}(\mathbf{h}^*_{\mathcal{H}^*}) - \langle \gamma\nabla L_{th}(\mathbf{h}^*_{\mathcal{H}^*}), \mathbf{h}_{\mathcal{H}^*} - \mathbf{h}^*_{\mathcal{H}^*}\rangle \ge 0.$$

This together with $\alpha_i^* = -\frac{\gamma}{\delta}h_i^* = \gamma\nabla\ell_{th}(h_i^*), i \in \mathcal{H}_2^*$ from (47) allows us to obtain (49).

(ii) For $i \in \mathcal{H}_1^*$, based on $h_i^* \le 0$ and $|h_i - h_i^*| \le \eta_m$, we receive $h_i \le h_i^* + \eta_m$. This means $\ell_{th}(h_i) \ge \ell_{th}(h_i^*)$. Hence (49) holds.

(iii) For $i \in \mathcal{H}_3^*$, by $h_i^* \ge \sqrt{2(\delta+\kappa\gamma)}$, $|h_i - h_i^*| \le \eta_m$ and $\delta + \kappa\gamma \ge 2$, we get $\ell_{th}(h_i) = \ell_{th}(h_i^*)$. Hence (49) holds.

Summarizing the above (i)–(iii), we obtain (48) and (49). This completes the proof of Case (ii). □

A.6. Proof of Theorem 4.1

Proof. For $\kappa, \gamma > 0$ and $\delta \in (0, 2)$ satisfying $\delta + \kappa\gamma \in (0, 2)$, since $(\mathbf{w}^*; b^*; \mathbf{h}^*)$ is a P-stationary point of (8), by Theorem 3.2 and (43), we obtain

$$\begin{aligned} &\alpha_i^* = -\frac{\gamma h_i^*}{\delta}, i \in H_2^*,\quad \alpha_i^* = -\gamma, i \in H_3^*, \\ &\alpha_i^* = 0, i \in \overline{H}^*. \end{aligned}$$

From (16) and $G = [y_1\mathbf{x}_1\ \cdots\ y_m\mathbf{x}_m]^\top \in \mathbb{R}^{m\times n}$, we obtain

$$\mathbf{w}^* = -G_{H^*}^\top\boldsymbol{\alpha}^*_{H^*} - G_{\overline{H}^*}^\top\boldsymbol{\alpha}^*_{\overline{H}^*} = -G_{H^*}^\top\boldsymbol{\alpha}^*_{H^*} = -\sum_{i\in H^*}\alpha_i^* y_i\mathbf{x}_i.$$

From (42) and (43), we have $h_i^* \in (0, \delta), i \in H_2^*$ and $h_i^* \in [\delta, 1 + \frac{\delta-\kappa\gamma}{2}], i \in H_3^*$, which with (16) and G results in

$$\begin{cases} y_i(\langle \mathbf{w}^*, \mathbf{x}_i\rangle + b^*) \in (1-\delta, 1), & i \in H_2^*, \\ y_i(\langle \mathbf{w}^*, \mathbf{x}_i\rangle + b^*) \in [\frac{\kappa\gamma-\delta}{2}, 1-\delta], & i \in H_3^*. \end{cases}$$

Hence (19) holds. Finally, since $\kappa, \gamma > 0$ and $\delta \in (0, 2)$ satisfying $\delta + \kappa\gamma \in (0, 2)$, we obtain $\frac{\kappa\gamma-\delta}{2} \in (-\delta, 1-\delta)$. This completes the proof. □

A.7. Proof of Theorem 4.2

Proof. For $\kappa, \gamma > 0$ and $\delta \in (0, 2)$ satisfying $\delta + \kappa\gamma \ge 2$, again, since $(\mathbf{w}^*; b^*; \mathbf{h}^*)$ is a P-stationary point of (8), from Theorem 3.2 and (47), we receive

$$\alpha_i^* = -\frac{\gamma}{\delta}h_i^*, i \in \mathcal{H}^*,\quad \alpha_i^* = 0, i \in \overline{\mathcal{H}}^*,$$

which with the first equation of (16) and $G \in \mathbb{R}^{m\times n}$ leads to

$$\mathbf{w}^* = -G_{\mathcal{H}^*}^\top\boldsymbol{\alpha}^*_{\mathcal{H}^*} - G_{\overline{\mathcal{H}}^*}^\top\boldsymbol{\alpha}^*_{\overline{\mathcal{H}}^*} = -G_{\mathcal{H}^*}^\top\boldsymbol{\alpha}^*_{\mathcal{H}^*} = -\sum_{i\in\mathcal{H}^*}\alpha_i^* y_i\mathbf{x}_i.$$

Moreover, from (47), we get $h_i^* \in (0, \sqrt{2\delta^2/(\delta+\kappa\gamma)}\,], i \in \mathcal{H}^*$, which together with G and (16) leads to

$$y_i(\langle \mathbf{w}^*, \mathbf{x}_i\rangle + b^*) \in [1 - \sqrt{2\delta^2/(\delta+\kappa\gamma)}, 1).$$

Hence (21) holds. This completes the proof. □

A.8. Proof of Theorem 4.3

Proof. It is evidently that the set $J_k \subseteq [m]$ has finite many elements, for $k \to \infty$, and thus we can find a subset $\mathcal{I} \subseteq \{1, 2, 3, \ldots\}$ such that

$$J_i \equiv: J, \quad \forall\, i \in \mathcal{I}. \tag{50}$$

Denote the sequence $\boldsymbol{\Phi}^k := (\mathbf{w}^k, b^k, \mathbf{h}^k, \boldsymbol{\alpha}^k)$ and its limit point $\boldsymbol{\Phi}^* := (\mathbf{w}^*, b^*, \mathbf{h}^*, \boldsymbol{\alpha}^*)$, i.e., $\{\boldsymbol{\Phi}^k\} \to \boldsymbol{\Phi}^*$. This also represents $\{\boldsymbol{\Phi}^i\}_{i\in\mathcal{I}} \to \boldsymbol{\Phi}^*$ and $\{\boldsymbol{\Phi}^{i+1}\}_{i\in\mathcal{I}} \to \boldsymbol{\Phi}^*$. Taking the limit along with $\mathcal{I}$ of (32), i.e., $k \in \mathcal{I}, k \to \infty$, we get

$$\begin{cases} \boldsymbol{\alpha}_J^* &= \boldsymbol{\alpha}_J^* + \mu\nu\boldsymbol{\zeta}_J^*, \\ \boldsymbol{\alpha}_{\bar{J}}^* &= \mathbf{0}, \end{cases} \tag{51}$$

which results in $\boldsymbol{\zeta}_J^* = \mathbf{0}$. Taking the limit along with $\mathcal{I}$ of $\mathbf{s}^k$, we obtain

$$\begin{aligned} \mathbf{s}^* &= \mathbf{1} - G\mathbf{w}^* - b^*\mathbf{y} - \boldsymbol{\alpha}^*/\nu \\ &= \mathbf{1} - G\mathbf{w}^* - b^*\mathbf{y} - \mathbf{h}^* + \mathbf{h}^* - \boldsymbol{\alpha}^*/\nu \\ &= -\boldsymbol{\zeta}^* + \mathbf{h}^* - \boldsymbol{\alpha}^*/\nu \end{aligned} \tag{52}$$

With respect to (25), we take the limit along with $\mathcal{I}$ and receive

$$\begin{cases} \mathbf{h}_{\Gamma^1}^* = \frac{\delta}{\widetilde{\delta}}\mathbf{s}_{\Gamma^1}^*, \mathbf{h}_{\Gamma^2}^* = \mathbf{s}_{\Gamma^2}^* - \kappa\gamma, \mathbf{h}_{\bar{J}}^* = \mathbf{s}_{\bar{J}}^*, & \widetilde{\delta} \in (0, 2), \\ \mathbf{h}_J^* = \frac{\delta}{\widetilde{\delta}}\mathbf{s}_J^*, \mathbf{h}_{\bar{J}}^* = \mathbf{s}_{\bar{J}}^*, & \widetilde{\delta} \geq 2, \end{cases} \tag{53}$$

where $\widetilde{\delta} := \delta + \kappa\gamma$, which allows us to obtain

$$\begin{aligned} \mathbf{h}_{\bar{J}}^* &= \mathbf{s}_{\bar{J}}^* \\ &\overset{(52)}{=} -\boldsymbol{\zeta}_{\bar{J}}^* + \mathbf{h}_{\bar{J}}^* - \boldsymbol{\alpha}_{\bar{J}}^*/\nu \\ &\overset{(51)}{=} -\boldsymbol{\zeta}_{\bar{J}}^* + \mathbf{h}_{\bar{J}}^*. \end{aligned}$$

Thus we obtain $\boldsymbol{\zeta}_{\bar{J}}^* = \mathbf{0}$ and $\boldsymbol{\zeta}^* = \mathbf{0}$. Again from (52), we get $\mathbf{s}^* = \mathbf{h}^* - \boldsymbol{\alpha}^*/\nu$, which together with (53) and (15) yields

$$\mathbf{h}^* \in \mathrm{Prox}_{\frac{\gamma}{\widetilde{\nu}}L_{th}}(\mathbf{s}^*) = \mathrm{Prox}_{\frac{\gamma}{\widetilde{\nu}}L_{th}}(\mathbf{h}^* - \boldsymbol{\alpha}^*/\nu). \tag{54}$$

As for (28), we take the limit along with $\mathcal{I}$ and yield

$$\begin{aligned} (I + \nu G_J^\top G_J)\mathbf{w}^* &= \nu G_J^\top \boldsymbol{\phi}_J^* \\ &= -\nu G_J^\top(\mathbf{h}_J^* + b^*\mathbf{y}_J - \mathbf{1} + \boldsymbol{\alpha}_J^*/\nu) \\ &= -\nu G_J^\top(\boldsymbol{\zeta}_J^* - G_J\mathbf{w}^* + \boldsymbol{\alpha}_J^*/\nu) \\ &= -\nu G_J^\top(-G_J\mathbf{w}^* + \boldsymbol{\alpha}_J^*/\nu), \end{aligned}$$

where $\boldsymbol{\phi}^* = -(\mathbf{h}^* + b^*\mathbf{y} - \mathbf{1} + \boldsymbol{\alpha}^*/\nu)$ and the last two equations hold because of $\boldsymbol{\zeta}^* = \mathbf{h}^* + G\mathbf{w}^* + b^*\mathbf{y} - \mathbf{1} = \mathbf{0}$ and $\boldsymbol{\zeta}_J^* = \mathbf{0}$ by (51). Furthermore, the last equation derives

$$\mathbf{w}^* = -G_J^\top\boldsymbol{\alpha}_J^* \overset{(51)}{=} -G^\top\boldsymbol{\alpha}^*.$$

Finally, for (31), we take the limit along with $\mathcal{I}$ and contribute to

$$\begin{aligned} b^* = \langle\mathbf{y}, \boldsymbol{\chi}^*\rangle/m &= -\langle\mathbf{y}, G\mathbf{w}^* - \mathbf{1} + \mathbf{h}^* + \boldsymbol{\alpha}^*/\nu\rangle/m \\ &= -\langle\mathbf{y}, \boldsymbol{\zeta}^* - b^*\mathbf{y} + \boldsymbol{\alpha}^*/\nu\rangle/m \\ &= -\langle\mathbf{y}, -b^*\mathbf{y} + \boldsymbol{\alpha}^*/\nu\rangle/m \\ &= b^* - \langle\mathbf{y}, \boldsymbol{\alpha}^*\rangle/(m\nu), \end{aligned}$$

which results in $\langle\mathbf{y}, \boldsymbol{\alpha}^*\rangle = 0$ and the three equation holds because of $\boldsymbol{\zeta}^* = 0$. Based on above analysis, we get

$$\begin{cases} \mathbf{w}^* + G^\top\boldsymbol{\alpha}^* &= \mathbf{0}, \\ \langle\mathbf{y}, \boldsymbol{\alpha}^*\rangle &= \mathbf{0}, \\ \mathbf{h}^* + G\mathbf{w}^* + b^*\mathbf{y} &= \mathbf{1}, \\ \mathrm{prox}_{\frac{\gamma}{\widetilde{\nu}}L_{th}}(\mathbf{h}^* - \boldsymbol{\alpha}^*/\nu) &\ni \mathbf{h}^*, \end{cases}$$

which proves that the $(\mathbf{w}^*; b^*; \mathbf{h}^*)$ is a P-stationary point with $\kappa = 1/\nu$ and is a local minimizer to the problem (8) from Theorem 3.2. This completes the proof. □

References

[1] C. Cortes, V.N. Vapnik, Support vector networks, Mach. Learn. 20 (3) (1995) 273–297.
[2] Y.Z. Cao, Y.T. Xu, J.L. Du, Multi-variable estimation-based safe screening rule for small sphere and large margin support vector machine, Knowl.-Based Syst. 191 (2020) 105223.
[3] S. Ertekin, L. Bottou, L.C. Giles, Nonconvex online support vector machines, IEEE Trans. Pattern Anal. Mach. Intell. 33 (2) (2011) 368–381.
[4] J. Zhao, Y. Xu, A safe sample screening rule for universum support vector machines, Knowl.-Based Syst. 138 (2017) 46–57.
[5] 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.
[6] S. Shalev-Shwartz, Y. Singer, N. Srebro, A. Cotter, Pegasos: primal estimated sub-gradient solver for SVM, Math. Program. 127 (1) (2011) 3–30.
[7] Y.H. Shao, Z. Wang, W.J. Chen, N.Y. Deng, A regularization for the projection twin support vector machine, Knowl.-Based Syst. 37 (2013) 203–210.
[8] W. Zhang, T. Yoshida, X. Tang, Text classification based on multi-word with support vector machine, Knowl.-Based Syst. 21 (8) (2008) 879–886.
[9] S.L. Zhou, Sparse SVM for sufficient data reduction, IEEE Trans. Pattern Anal. Mach. Intell. 44 (9) (2022) 5560–5571.
[10] L. Bai, Z. Wang, Y.H. Shao, N.Y. Deng, A novel feature selection method for twin support vector machine, Knowl.-Based Syst. 59 (2014) 1–8.
[11] W.X. Zhu, Y.Y. Song, Y.Y. Xiao, Support vector machine classifier with huberized pinball loss, Eng. Appl. Artif. Intell. 91 (2020) 103635.
[12] C.C. Chang, C.J. Lin, LIBSVM: a library for support vector machines, ACM Trans. Intell. Syst. Technol. 2 (3) (2011) 1–27.
[13] P.L. Bartlett, M.H. Wegkamp, Classification with a reject option using a hinge loss, J. Mach. Learn. Res. 9 (2008) 1823–1840.
[14] 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.
[15] 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.
[16] J. Zhao, Y. Xu, H. Fujita, An improved non-parallel universum support vector machine and its safe sample screening rule, Knowl.-Based Syst. 170 (2019) 79–88.
[17] 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.
[18] Z. Yang, Y. Xu, A safe accelerative approach for pinball support vector machine classifier, Knowl.-Based Syst. 147 (2018) 12–24.
[19] J.A.K. Suykens, J. Vandewalle, Least squares support vector machine classifiers, Neural Process. Lett. 9 (3) (1999) 293–300.
[20] G. Huang, Z. Yang, X. Chen, G. Ji, An innovative one-class least squares support vector machine model based on continuous cognition, Knowl.-Based Syst. (2017) 217–228.
[21] K. Pelckmans, J.A.K. Suykens, T.V. Gestel, J.D. Brabanter, L. Lukas, B. Hamers, B.D. Moor, J. Vandewalle, LSSVM Lab: A Matlab/C Toolbox for Least Squares Support Vector Machines, Tutorial, Vol. 142, KULeuven-ESAT, Leuven, Belgium, 2002, pp. 1–2.
[22] O. Chapelle, Training a support vector machine in the primal, Neural Comput. 19 (5) (2007) 1155–1178.
[23] R. Johnson, T. Zhang, Accelerating stochastic gradient descent using predictive variance reduction, in: Proc. Int. Conf. Neural Inf. Process. Syst., 2013, pp. 315–323.
[24] Z. Allen-Zhu, Katyusha: the first direct acceleration of stochastic gradient methods, J. Mach. Learn. Res. 18 (221) (2018) 1–51.
[25] J.T. Li, Y.M. Jia, Huberized multiclass support vector machine for microarray classification, Acta Autom. Sin. 36 (3) (2010) 399–405.
[26] G. Wahba, Support vector machines reproducing kernel Hilbert spaces and randomized GAC, in: Advances in Kernel Methods–Support Vector Learning, 1998, pp. 69–88.
[27] X.T. Shen, G.C. Tseng, X.G. Zhang, W.H. Wong, On ψ-learning, J. Amer. Statist. Assoc. 98 (463) (2003) 724–734.
[28] 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.
[29] X.L. Huang, L. Shi, J.A.K. Suykens, Ramp loss linear programming support vector machine, J. Mach. Learn. Res. 15 (2014) 2185–2211, 2014.
[30] Y.C. Wu, Y.F. Liu, Robust truncated hinge loss support vector machines, J. Amer. Statist. Assoc. 102 (479) (2007) 974–983.
[31] L. Yin, H. Wang, W. Fan, Active learning based support vector data description method for robust novelty detection, Knowl.-Based Syst. 153 (2018) 40–52.
[32] T. Hazan, J. Keshet, D. McAllester, Direct loss minimization for structured prediction, in: Proc. Int. Conf. Neural Inf. Process. Syst., 2010 pp. 1594–1602.
[33] S.Y. Park, Y.F. Liu, Robust penalized logistic regression with truncated loss functions, Can. J. Stat. 39 (2) (2011) 300–323.

[34] L.M. Yang, H.W. Dong, Support vector machine with truncated pinball loss and its application in pattern recognition, Chemometr. Intell. Lab. Syst. 177 (2018) 89–99.
[35] X. Shen, L.F. Niu, Z. Qi, Y.J. Tian, Support vector machine classifier with truncated pinball loss, Pattern Recognit. 68 (2017) 199–210.
[36] 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.
[37] F. Pérez-Cruz, A. Navia-Vázquez, A.R. Figueiras-Vidal, A. Artes-Rodriguez, Empirical risk minimization for support vector classifiers, IEEE Trans. Neural Netw. 14 (2) (2003) 296–303.
[38] R.T. Rockafellar, R.J. Wets, Variational Analysis, Springer Science and Business Media, 2009.
[39] M. Fazel, T.K. Pong, D.F. Sun, P. Tseng, Hankel matrix rank minimization with applications to system identification and realization, SIAM J. Matrix Anal. Appl. 34 (3) (2013) 946–977.
[40] M. Li, D.F. Sun, K.C. Toh, A majorized ADMM with indefinite proximal terms for linearly constrained convex composite optimization, SIAM J. Optim. 26 (2) (2016) 922–950.
[41] X. Chang, S. Liu, P. Zhao, D. Song, A generalization of linearized alternating direction method of multipliers for solving two-block separable convex programming, J. Comput. Appl. Math. 357 (2) (2019) 251–272.
[42] G. Golub, C.F. Van-Loan, Matrix Computations, Johns Hopkins University Press, 1996.
[43] T.P. Minka, A comparison of numerical optimizers for logistic regression, 2003, Available on http://yaroslavvb.com/papers/minka-comparison.pdf.