Sparse SVM for Sufficient Data Reduction

Article *in* IEEE Transactions on Pattern Analysis and Machine Intelligence · April 2021

DOI: 10.1109/TPAMI.2021.3075339

Shenglong Zhou
Imperial College London

Sparse SVM for Sufficient Data Reduction

Shenglong Zhou

Abstract—Kernel-based methods for support vector machines (SVM) have shown highly advantageous performance in various applications. However, they may incur prohibitive computational costs for large-scale sample datasets. Therefore, data reduction (reducing the number of support vectors) appears to be necessary, which gives rise to the topic of the sparse SVM. Motivated by this problem, the sparsity constrained kernel SVM optimization has been considered in this paper in order to control the number of support vectors. Based on the established optimality conditions associated with the stationary equations, a Newton-type method is developed to handle the sparsity constrained optimization. This method is found to enjoy the one-step convergence property if the starting point is chosen to be close to a local region of a stationary point, thereby leading to a super-high computational speed. Numerical comparisons with several powerful solvers demonstrate that the proposed method performs exceptionally well, particularly for large-scale datasets in terms of a much lower number of support vectors and shorter computational time.

Index Terms—data reduction, sparsity constrained kernel SVM, Newton method, one-step convergence property.

1 Introduction

Support vector machines (SVM) were first introduced by Cortes and Vapnik [1] and are currently popular classification tools in machine learning, statistic and pattern recognition. The goal is to find a hyperplane in the input space that best separates the training dataset to enable accurate prediction of the class of some newly input data. The paper focuses on the binary classification problem: suppose that we are given a training dataset $\{(\mathbf{x}_i, y_i) : i = 1, 2, \cdots, m\}$, where $\mathbf{x}_i \in \mathbb{R}^n$ is the sample vector and $y_i \in \{-1, 1\}$ is the binary class. The task is to train a hyperplane $\langle \mathbf{w}, \mathbf{x} \rangle + b = w_1x_1 + \cdots + w_nx_n + b = 0$ with variable $\mathbf{w} \in \mathbb{R}^n$ and bias $b \in \mathbb{R}$ to be estimated based on the training dataset. For any newly input vector $\overline{\mathbf{x}}$, one can predict the corresponding class $\overline{y}$ by $\overline{y} = 1$ if $\langle \mathbf{w}, \overline{\mathbf{x}} \rangle + b > 0$ and $\overline{y} = -1$ otherwise. There are two possible scenarios to find an optimal hyperplane: linearly separable and inseparable training datasets in the input space. For the latter, the popular approach is to solve the so-called soft-margin SVM optimization.

There is a vast body of work on the design of the loss functions ℓ for dealing with the soft-margin SVM optimization. One of the most well-known loss functions is the Hinge loss, giving rise to the Hinge soft-margin SVM model. To address such a problem, the dual kernel-based SVM optimization is usually used, for which the objective function involves an $m \times m$-order Gram matrix known as the kernel matrix. The complexity of computing this kernel matrix is approximately $O(m^2n)$, thereby making it impossible to develop methods for training on million-size data since data storage on such a scale requires large memory and the incurred computational cost is prohibitively high. Therefore, to overcome this drawback, data reduction has drawn much attention with the goal of using a small portion of samples to train a classifier.

• *S.L. Zhou is with the Department of Electrical and Electronic Engineering, Imperial College London, London, UK. Email: slzhou2021@163.com.*

Manuscript received 25 Aug 2020; revised 4 Feb 2021; accepted 21 April 2021.

It is well known that by the Representer Theorem, the classifier $\mathbf{w}^*$ can be expressed as

$$\mathbf{w}^* = \sum_{i=1}^{m} \alpha_i^* y_i \mathbf{x}_i, \tag{1}$$

where $\boldsymbol{\alpha}^*$ is a solution to the dual kernel-based SVM optimization. The training vectors $\mathbf{x}_i$ corresponding to nonzero α_i^* are known as the support vectors. If large numbers of coefficients α_i^* are zeros, then the number of the support vectors can be reduced significantly. Therefore, computations and storage for large scale size data are possible since only the support vectors are used. However, solutions to the dual kernel-based SVM optimization are not sparse enough generally. An impressive number of approaches have been developed for guaranteeing that the solution is sufficiently sparse. The methods aiming to reduce the number of support vectors can be categorized as the sparse SVM group. We will explore more in the sequel.

1.1 Selective Literature Review

Extensive work has focused on designing loss functions ℓ to cast efficient soft-margin SVM models that can be summarized into two classes based on the convexity of ℓ. Convex soft margin loss functions include the famous hinge loss [1], the pinball loss [2], [3], the hybrid Huber loss [4]–[6], the square loss [7], [8], the exponential loss [9] and log loss [10]. Convexity makes the computations of their corresponding SVM models tractable, but it also induces the unboundedness, thereby reducing the robustness of these functions to the outliers from the training data. To overcome this drawback, [11],[12] set an upper bound and enforce the loss functions to stop increasing after a certain point. This makes the convex loss functions become nonconvex. Other nonconvex losses include the ramp loss [13], the truncated pinball loss [14], the asymmetrical truncated pinball loss [15], the sigmoid loss [16], and the normalized sigmoid cost loss [17]. Compared with the convex margin loss functions, most nonconvex functions are less sensitive to the outliers due to their boundedness. However, nonconvexity generally gives rise to difficulties in numerical computations.

A research effort following a different direction investigated methods for data reduction, namely, reducing the number of the support vectors, giving rise to the topic of the sparse SVM. One of the earliest attempts can be traced back to [18], where it was suggested to solve the kernel SVM optimization problem to find a solution first and then seek a sparse approximation through support vector regression. This idea was then adopted as a key component of the method developed in [19].The famous reduced SVM (RSVM) in [20] randomly picked a subset of the training set, and then searched for a solution (supported only on the picked training set) to a smooth SVM optimization that minimized the loss on the entire training set. In [21], from (1), they substituted $\mathbf{w}$ by the expression $\sum_{i=1}^m \alpha_i y_i \mathbf{x}_i$ and employed the ℓ_1-norm regularization of $\boldsymbol{\alpha}$ in the soft-margin loss model, leading to effective variable/sample selections because the ℓ_1-norm regularization can render a sparse structure of the solution [22]. Similarly, [23] also took advantage of the expression of (1). They performed a greedy method, where in each step, a new training sample was carefully selected into the set of the training vectors to form a new subproblem. Since the number of the training vectors was relatively small in comparison with the total size of the training samples, the subproblem was on small scale and thus can be addressed by Newton method quickly. In [24], a subgradient descent algorithm was proposed where in each step, only the samples with the correct classification inside the margin but maximizing a gap were selected. Other relevant methods include the so-called reduced set methods [25], [26], the Forgetron algorithm [27], the condensed vector machines training method [28] and those in [29], [30]. Numerical experiments have demonstrated that these methods perform exceptionally well for the reduction in the number of support vectors.

We note that some sparse SVM methods [20], [23] aim to reduce the size of the dual problem, a quadratic kernel-based optimization, to reduce computational cost and the required storage. For the same purpose, two alternatives are available for dealing with data on large scales. The first is to process the data prior to employing a method. For instance, [31], [32] select informative samples and remove the useless samples to reduce the computational cost but while preserving the accuracy. For another example, the 'cascade' SVM [33] and the mixture SVM [34] split the dataset into several disjointed subsets and then carry out parallel optimization to accelerate the computation. The second approach effectively benefits from kernel tricks. Nyström method [35] and the sparse greedy approximation technique [36] can be adopted to construct the Gram matrix to reduce the cost of solving the kernel-based dual problem. Moreover, [37], [38] exploited the low-rank kernel representations to make the interior point method tractable for large scale datasets.

1.2 Methodology

Mathematically, the soft-margin SVM takes the form of

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

where $C > 0$ is a penalty parameter, $\|\cdot\|$ is the Euclidean norm and ℓ is a loss function. One of the most famous loss functions is the Hinge loss $\ell_{\rm H}(t) := \max\{0, t\}$, giving rise to the Hinge loss soft-margin SVM, for which the dual problem is the following quadratic kernel-based SVM optimization,

$$\begin{aligned}\min_{\boldsymbol{\alpha}\in\mathbb{R}^m} \quad & d(\boldsymbol{\alpha}) := \frac{1}{2}\sum_{i=1}^m\sum_{j=1}^m \alpha_i\alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j\rangle - \sum_{i=1}^m \alpha_i, \\ \text{s.t.} \quad & \sum_{i=1}^m \alpha_i y_i = 0, \ \ 0 \le \alpha_i \le C, i \in [m],\end{aligned} \tag{3}$$

where $[m] := \{1, \cdots, m\}$. In this paper, we focus on the following soft-margin SVM,

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

where the soft-margin loss ℓ_{cC} is defined by

$$\ell_{cC}(t) := \begin{cases} Ct^2/2, & t \ge 0, \\ ct^2/2, & t < 0. \end{cases} \tag{5}$$

Here, $c > 0$ is chosen to be smaller than C, namely $c < C$. The soft-margin SVM (4) implies that it gives penalty C for $t_i := 1 - y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle + b) \ge 0$ and c otherwise. Note that ℓ_{cC} is reduced to the squared Hinge loss $(\ell_{\rm H}(t))^2$ when $c = 0$. Theorem 2.1 shows that the dual problem of (4) takes the form of

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^m} \ D(\boldsymbol{\alpha}) := d(\boldsymbol{\alpha}) + \sum_{i=1}^m h_{cC}(\alpha_i), \ \ \text{s.t.} \ \sum_{i=1}^m \alpha_i y_i = 0, \tag{6}$$

where h_{cC} is defined by

$$h_{cC}(t) := \begin{cases} t^2/(2C), & t \ge 0, \\ t^2/(2c), & t < 0. \end{cases} \tag{7}$$

Motivated by the work in the sparse SVM [18]-[30], in this paper, we aim to solve the following sparsity constrained kernel-based SVM optimization problem,

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^m} \ D(\boldsymbol{\alpha}), \ \ \text{s.t.} \ \sum_{i=1}^m \alpha_i y_i = 0, \ \|\boldsymbol{\alpha}\|_0 \le s, \tag{8}$$

where $s \in [m]$ is a given integer satisfying $s \ll m$ and is called the sparsity level, and $\|\boldsymbol{\alpha}\|_0$ is the zero norm of $\boldsymbol{\alpha}$, counting the number of nonzero elements of $\boldsymbol{\alpha}$. Compared with the classic kernel SVM model (3), the problem (8) has at least three advantages: A1) The objective function is strongly convex; hence, the optimal solution exists (see Theorem 2.2) and is unique under mild conditions (see Theorem 2.4). By contrast, the objective function of (3) is convex but not strongly convex when $n \le m$. This means that it may have multiple optimal solutions. A2) Since the bounded constraints $0 \le \alpha_i \le C, i \in [m]$ are absent, the computation is somewhat more tractable. Note that when m is large, these bounded constraints in (3) may incur some computational costs if no additional efforts, such as the use of the perturbed KKT optimality conditions [37], [38], are paid. A3) Most importantly, the constraint $\|\boldsymbol{\alpha}\|_0 \le s$ means that at most s nonzero elements are contained in $\boldsymbol{\alpha}$, i.e., the number of the support vectors is expected to be less than s by (1). Thus, the number can be controlled to be small.

1.3 Contributions

The contributions of this paper are summarized as follows.

C1) As mentioned above, the sparsity constrained model (8) has its advantages. To the best of our knowledge, this is the first paper that employs the sparsity constraint into the kernel SVM model. Such a constraint allows us to control the number of support vectors and hence to perform sufficient data reduction, making extremely large scale computation possible and significantly reducing the demand for huge hardware memory.

C2) Based on the established optimality condition associated with the stationary equations by Theorem 2.3, a Newton-type method is employed. The method shows very low computational complexity and enjoys one-step convergence property. This convergence result is much better than the locally quadratic convergence property that is typical of Newton-type methods. Specifically, the method converges to a stationary point within one step if the chosen starting point is close enough to the stationary point, see Theorem 3.1. Moreover, since the sparsity level s that is used to control the number of support vectors is unknown in practice, the selection of $s \in [m]$ is somewhat tedious. However, we successfully design a mechanism to tune the sparsity level s adaptively.

C3) Numerical experiments have demonstrated that the proposed method performs exceptionally well particularly for datasets with $m \gg n$. Compared with several leading solvers, it takes much shorter computational time due to a tiny number of support vectors used for large scale data.

1.4 Preliminaries

We present some notation to be employed throughout the paper in Table 1.

TABLE 1: List of notations

Notation	Description
$[m]$	The index set $\{1, 2, \cdots, m\}$.
$\|T\|$	The number of elements of an index set $T \subseteq [m]$.
$\overline{T}$	The complementary set of an index set T, namely, $[m] \setminus T$.
$\boldsymbol{\alpha}_T$	The sub-vector of $\boldsymbol{\alpha}$ indexed on T and $\boldsymbol{\alpha}_T \in \mathbb{R}^{\|T\|}$.
$\|\boldsymbol{\alpha}\|$	$:= (\|\alpha\|, \cdots, \|\alpha\|)^\top$.
$\\|\boldsymbol{\alpha}\\|_{[s]}$	The sth largest element of $\|\boldsymbol{\alpha}\|$.
supp($\boldsymbol{\alpha}$)	The support set of $\boldsymbol{\alpha}$, namely, $\{i \in [m] : \alpha_i \neq 0\}$.
$\mathbf{y}$	The labels/classes $(y_1, \ldots, y_m)^\top \in \mathbb{R}^m$.
X	The samples data $[\mathbf{x}_1 \ \cdots \ \mathbf{x}_m]^\top \in \mathbb{R}^{m\times n}$.
Q	$:= [y_1\mathbf{x}_1 \ \cdots \ y_m\mathbf{x}_m] \in \mathbb{R}^{n\times m}$.
Q_T	The sub-matrix containing the columns of Q indexed on T.
$Q_{\Gamma,T}$	The sub-matrix containing the rows of Q_T indexed on Γ.
I	The identity matrix.
P	$:= Q^\top Q + I/C$.
$\mathbf{1}$	$:= (1, \cdots, 1)^\top$ whose dimension varies in the context.
$N(\boldsymbol{\alpha}, \delta)$	The neighbourhood of $\boldsymbol{\alpha}$ with radius $\delta > 0$, namely, $\{\mathbf{u} \in \mathbb{R}^m : \\|\mathbf{u} - \boldsymbol{\alpha}\\| < \delta\}$.

Notation in Table 1 enables us to rewrite $D(\boldsymbol{\alpha})$ in (8) as

$$D(\boldsymbol{\alpha}) = \frac{1}{2}\|Q\boldsymbol{\alpha}\|^2 + \frac{1}{2}\langle E(\boldsymbol{\alpha})\boldsymbol{\alpha}, \boldsymbol{\alpha}\rangle - \langle \mathbf{1}, \boldsymbol{\alpha}\rangle, \tag{9}$$

where $E(\boldsymbol{\alpha})$ is a diagonal matrix with

$$E_{ii}(\boldsymbol{\alpha}) := (E(\boldsymbol{\alpha}))_{ii} = \begin{cases} 1/C, & \alpha_i \geq 0, \\ 1/c, & \alpha_i < 0. \end{cases} \tag{10}$$

The gradient $\nabla D(\boldsymbol{\alpha})$ and Hessian $H(\boldsymbol{\alpha})$ of $D(\boldsymbol{\alpha})$ are

$$\nabla D(\boldsymbol{\alpha}) := H(\boldsymbol{\alpha})\boldsymbol{\alpha} - \mathbf{1}, \ \ H(\boldsymbol{\alpha}) := Q^\top Q + E(\boldsymbol{\alpha}). \tag{11}$$

It is observed that the Hessian matrix is positive definite for any $\boldsymbol{\alpha} \in \mathbb{R}^m$ and

$$H(\boldsymbol{\alpha}) \succeq Q^\top Q + I/C = P \succ 0, \tag{12}$$

due to $C > c$, where $A \succeq 0$ ($A \succ 0$) means A is semi-definite (definite) positive. Here, $A \succeq B$ represents that $A - B$ is semi-definite positive. For notational convenience, hereafter, for a given $\boldsymbol{\alpha} \in \mathbb{R}^m$ and $b \in \mathbb{R}$, let

$$\mathbf{z} \ \ := \ \ (\boldsymbol{\alpha}; b) = \begin{bmatrix} \boldsymbol{\alpha} \\ b \end{bmatrix}. \tag{13}$$

Based on this, we denote the following functions

$$\begin{aligned} g(\mathbf{z}) \ \ &:= \nabla D(\boldsymbol{\alpha}) + \mathbf{y}b \overset{(11)}{=} H(\boldsymbol{\alpha})\boldsymbol{\alpha} - \mathbf{1} + \mathbf{y}b, \\ g_T(\mathbf{z}) &:= (g(\mathbf{z}))_T, \quad H_T(\boldsymbol{\alpha}) := (H(\boldsymbol{\alpha}))_{TT}. \end{aligned} \tag{14}$$

Here, $g(\mathbf{z})$ is a vector and $g_T(\mathbf{z})$ is a sub-vector of $g(\mathbf{z})$. $H(\boldsymbol{\alpha})$ is a matrix and $H_T(\boldsymbol{\alpha})$ is the sub-principal matrix of $H(\boldsymbol{\alpha})$ indexed by T. Similar rules are also applied for $\mathbf{z}^*$ and $\mathbf{z}^k$.

1.5 Organization

The rest of the paper is organized as follows. In the next section, we focus on the sparsity constrained model (8), establishing the optimality condition associated with the η-stationary point. The condition is then equivalently transferred to stationary equations that allow us to adopt the Newton method in Section 3 where we also prove that the proposed method converges to a stationary point within one step. Furthermore, a strategy of tuning the sparsity level s is employed to derive `NSSVM`. Numerical experiments are presented in Section 4, where the implementation of `NSSVM` as well as its comparisons with some leading solvers are provided. We draw the conclusion in the last section and present all of the proofs in the appendix.

2 Optimality

Prior to the establishment of the optimality condition of the sparsity constrained kernel SVM optimization (8), hereafter, for a point $\boldsymbol{\alpha}^* \in \mathbb{R}^m$, its support set is always denoted by

$$S_* \ \ := \ \ \text{supp}(\boldsymbol{\alpha}^*). \tag{15}$$

We now claim that (6) is indeed the dual problem of (4).

Theorem 2.1. Problem (6) is the dual problem of (4) and admits a unique optimal solution, say $\boldsymbol{\alpha}^*$. Furthermore, the optimal solution of the primal problem (4) is

$$\widehat{\mathbf{w}} = Q\boldsymbol{\alpha}^*, \ \ \widehat{b} = \frac{\langle \mathbf{y}_{S_*}, \mathbf{1} - H_{S_*}(\boldsymbol{\alpha}^*)\boldsymbol{\alpha}^*_{S_*}\rangle}{|S_*|}. \tag{16}$$

Before we embark on the main theory in this section, we explore more of the sparse constraint. Let $\mathbb{P}_s$ be defined by

$$\mathbb{P}_s(\boldsymbol{\alpha}) = \text{argmin}_{\mathbf{u}}\Big\{\|\mathbf{u} - \boldsymbol{\alpha}\| : \|\mathbf{u}\|_0 \leq s\Big\}, \tag{17}$$

which can be obtained by retaining the s largest elements in magnitude from $\boldsymbol{\alpha}$ and setting the remaining to zero.

$\mathbb{P}_s$ is known as the Hard-thresholding operator. To well characterize the solution of (17), we define a useful set by

$$\mathbb{T}_s(\boldsymbol{\alpha}) := \left\{ T \subseteq [m] : \begin{array}{l} |T| = s, \\ |\alpha_i| \geq |\alpha_j|, \forall i \in T, \forall j \notin T \end{array} \right\}. \quad (18)$$

This definition of $\mathbb{T}_s$ allows us to express $\mathbb{P}_s$ as

$$\mathbb{P}_s(\boldsymbol{\alpha}) := \{(\boldsymbol{\alpha}_T;\ 0) :\ T \in \mathbb{T}_s(\boldsymbol{\alpha})\}. \quad (19)$$

As the sth largest element of $|\boldsymbol{\alpha}|$ may not be unique, $\mathbb{T}_s(\boldsymbol{\alpha})$ may have multiple elements, so does $\mathbb{P}_s(\boldsymbol{\alpha})$. For example, consider $\overline{\boldsymbol{\alpha}} = (1, -1, 0, 0)^\top$. It is easy to check that $\mathbb{T}_1(\overline{\boldsymbol{\alpha}}) = \{\{1\}, \{2\}\}$, $\mathbb{P}_1(\overline{\boldsymbol{\alpha}}) = \{(1, 0, 0, 0)^\top, (0, -1, 0, 0)^\top\}$ and $\mathbb{T}_2(\overline{\boldsymbol{\alpha}}) = \{\{1, 2\}\}$, $\mathbb{P}_2(\overline{\boldsymbol{\alpha}}) = \{\overline{\boldsymbol{\alpha}}\}$.

2.1 η-Stationary Point

In this part, we focus on the sparsity constrained kernel SVM problem (8). First, we conclude that it admits a global solution/minimizer.

Theorem 2.2. The global minimizers of (8) exist.

The Lagrangian function of (8) is $L(\boldsymbol{\alpha}, b) := D(\boldsymbol{\alpha}) + b\langle \boldsymbol{\alpha}, \mathbf{y}\rangle$, where b is the Lagrangian multiplier. To find a solution to (8), we consider its Lagrangian dual problem

$$\max_{b\in\mathbb{R}} \min_{\boldsymbol{\alpha}\in\mathbb{R}^m} \{L(\boldsymbol{\alpha}, b) : \|\boldsymbol{\alpha}\|_0 \leq s\}. \quad (20)$$

We note that $\nabla_{\boldsymbol{\alpha}} L(\boldsymbol{\alpha}, b) = g(\mathbf{z})$ from (14) and $\nabla_b L(\boldsymbol{\alpha}, b) = \langle \boldsymbol{\alpha}, \mathbf{y}\rangle$. Based on these, we introduce the concept of an η-stationary point of the problem (8).

Definition 2.1. We say $\boldsymbol{\alpha}^*$ is an η-stationary point of (8) with $\eta > 0$ if there is $b^* \in \mathbb{R}$ such that

$$\left\{ \begin{array}{rcl} \boldsymbol{\alpha}^* & \in & \mathbb{P}_s\Big[\boldsymbol{\alpha}^* - \eta g(\mathbf{z}^*)\Big], \\ 0 & = & \langle \boldsymbol{\alpha}^*, \mathbf{y}\rangle. \end{array} \right. \quad (21)$$

From the notation (13) that $\mathbf{z}^* = (\boldsymbol{\alpha}^*; b^*)$. We also say $\mathbf{z}^*$ is an η-stationary point of (8) if it satisfies the above conditions. We characterize an η-stationary point of (8) as a system of equations through the following theorem.

Theorem 2.3. A point $\mathbf{z}^*$ is an η-stationary point of (8) with $\eta > 0$ if and only if $\exists\, T_* \in \mathbb{T}_s(\boldsymbol{\alpha}^* - \eta g(\mathbf{z}^*))$ such that

$$F(\mathbf{z}^*; T_*) \ := \ \begin{bmatrix} g_{T_*}(\mathbf{z}^*) \\ \boldsymbol{\alpha}^*_{\overline{T}_*} \\ \langle \boldsymbol{\alpha}^*_{T_*}, \mathbf{y}_{T_*}\rangle \end{bmatrix} = 0. \quad (22)$$

We call (22) the stationary equations that allow for employing the Newton method used to find solutions to system of equations. It is well known that the (generalized) Jacobian of the involved equations is required to update the Newton direction. Therefore, comparing with the conditions in (21), equations in (22) enable easier implementation of the Newton method. More specifically, (21) involves inclusions and its Jacobian is difficult to obtain as $\mathbb{P}_s(\cdot)$ is nondifferentiable. By contrast, for any fixed $T \in \mathbb{T}_s(\boldsymbol{\alpha} - \eta g(\mathbf{z}))$, all functions in $F(\mathbf{z}; T)$ are differentiable and the Jacobian matrix of $F(\mathbf{z}; T)$ can be derived by,

$$\nabla F(\mathbf{z}; T) = \begin{bmatrix} H_T(\boldsymbol{\alpha}) & 0 & \mathbf{y}_T \\ 0 & I & 0 \\ \mathbf{y}_T^\top & 0 & 0 \end{bmatrix}, \quad (23)$$

which is always nonsingular due to $H_T(\boldsymbol{\alpha}) \succ 0$.

Remark 2.1. What is the role of b^*? The second condition in (22) indicates $S_* \subseteq T_*$. Then the first equation in (22) and (14) yields that

$$0 = g_{S_*}(\mathbf{z}^*) = H_{S_*}(\boldsymbol{\alpha}^*)\boldsymbol{\alpha}^*_{S_*} - \mathbf{1} + \mathbf{y}_{S_*} b^*,$$

which together with (16) means $b^* = \widehat{b}$. Namely, b^* is the bias. Therefore, seeking an η-stationary point can acquire the solution to the dual problem (8) and the bias b^* to the primal problem (4) at the same time.

We now reveal the relationships among local minimizers, global minimizers and η-stationary points. These relationships indicate that to pursue a local (or even a global) minimizer, we instead find an η-stationary point because it could be effectively derived from (22) by the numerical method proposed in the next section.

Theorem 2.4. Consider a feasible point $\boldsymbol{\alpha}^*$ to the problem (8), namely, $\|\boldsymbol{\alpha}^*\|_0 \leq s$ and $\langle \boldsymbol{\alpha}^*, \mathbf{y}\rangle = 0$. We have the following relationships.

i) For local minimizers and η-stationary points, it has

a) an η-stationary point $\boldsymbol{\alpha}^*$ is also a local minimizer.

b) a local minimizer $\boldsymbol{\alpha}^*$ is an η-stationary point either for any $\eta > 0$ if $\|\boldsymbol{\alpha}^*\|_0 < s$ or for any $0 < \eta \leq \eta^*$ if $\|\boldsymbol{\alpha}^*\|_0 = s$, where $\eta^* > 0$ is relied on $\boldsymbol{\alpha}^*$.

ii) For global minimizers and η-stationary points, it has

c) if $\|\boldsymbol{\alpha}^*\|_0 < s$, then the local minimizer, the global minimizer and the η-stationary point are identical to each other and unique.

d) if $\|\boldsymbol{\alpha}^*\|_0 = s$, and the point $\boldsymbol{\alpha}^*$ is an η-stationary point with $\eta \geq C$, then it is also a global minimizer. Moreover, it is also unique if $\eta > C$.

Theorem 2.2 states that the global minimizers (also being a local minimizer) of (8) exist. In addition, Theorem 2.4 b) shows that a local minimizer of (8) is an η-stationary point. Therefore, the η-stationary point exists, indicating that there must exist b^* such (21).

3 Newton Method

This section applies the Newton method for solving equation (22). Let $\mathbf{z}^k$ be defined in (13) and the current approximation to a solution of (22). Choose one index set

$$T_k \in \mathbb{T}_s(\boldsymbol{\alpha}^k - \eta g(\mathbf{z}^k)). \quad (24)$$

Then find the Newton direction $\mathbf{d}^k \in \mathbb{R}^{m+1}$ by solving the following linear equations,

$$\nabla F(\mathbf{z}^k; T_k)\mathbf{d}^k = -F(\mathbf{z}^k; T_k). \quad (25)$$

Substituting (22) and (23) into (25), we obtain

$$\begin{bmatrix} H_{T_k}(\boldsymbol{\alpha}^k) & 0 & \mathbf{y}_{T_k} \\ 0 & I & 0 \\ \mathbf{y}_{T_k}^\top & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{d}^k_{T_k} \\ \mathbf{d}^k_{\overline{T}_k} \\ d^k_{m+1} \end{bmatrix} = - \begin{bmatrix} g_{T_k}(\mathbf{z}^k) \\ \boldsymbol{\alpha}^k_{\overline{T}_k} \\ \langle \boldsymbol{\alpha}^k_{T_k}, \mathbf{y}_{T_k}\rangle \end{bmatrix}. \quad (26)$$

After we get the direction, the full Newton step size is taken, namely, $\mathbf{z}^{k+1} = \mathbf{z}^k + \mathbf{d}^k$, to derive

$$\mathbf{z}^{k+1} = \begin{bmatrix} \boldsymbol{\alpha}^k_{T_k} \\ \boldsymbol{\alpha}^k_{\overline{T}_k} \\ b^k \end{bmatrix} + \begin{bmatrix} \mathbf{d}^k_{T_k} \\ \mathbf{d}^k_{\overline{T}_k} \\ d^k_{m+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\alpha}^k_{T_k} + \mathbf{d}^k_{T_k} \\ 0 \\ b^k + d^k_{m+1} \end{bmatrix}. \quad (27)$$

Now we summarize the whole framework in Algorithm 1.

Algorithm 1 Newton method for sparse SVM.

```
Give parameters $C > c, \eta, \epsilon, K > 0, s \in [m]$.
Initialize $\mathbf{z}^0$, pick $T_0 \in \mathbb{T}_s(\boldsymbol{\alpha}^0 - \eta g(\mathbf{z}^0))$ and set $k := 0$.
while $\|F(\mathbf{z}^k; T_k)\| \geq \epsilon$ and $k \leq K$ do
    Update $\mathbf{d}^k$ by solving (26).
    Update $\mathbf{z}^{k+1}$ by (27).
    Update $T_{k+1} \in \mathbb{T}_s(\boldsymbol{\alpha}^{k+1} - \eta g(\mathbf{z}^{k+1}))$ and $k := k + 1$.
end while
return the solution $\mathbf{z}^k$.
```

It is easy to see from (26) that $\mathbf{d}^k_{\overline{T}_k} = -\boldsymbol{\alpha}^k_{\overline{T}_k}$ is derived directly. Therefore, every new point is always sparse due to

$$\|\boldsymbol{\alpha}^{k+1}\|_0 \overset{(27)}{=} \|\boldsymbol{\alpha}^k_{T_k} + \mathbf{d}^k_{T_k}\|_0 \leq |T_k| = s.$$

Moreover, the major computation in (26) is from the part on T_k. However, $|T_k| = s$ can be controlled to have a very small scale compared to $m + 1$, leading to a considerably low computational complexity.

3.1 Complexity Analysis

To derive the Newton direction in (26), we need to calculate

$$\begin{aligned} d^k_{m+1} &= -\frac{\langle \mathbf{y}_{T_k}, \Theta^{-1} g_{T_k}(\mathbf{z}^k) - \boldsymbol{\alpha}^k_{T_k} \rangle}{\langle \mathbf{y}_{T_k}, \Theta^{-1} \mathbf{y}^k_{T_k} \rangle}, \\ \mathbf{d}^k_{T_k} &= -\Theta^{-1}\left[g_{T_k}(\mathbf{z}^k) + d^k_{m+1}\mathbf{y}_{T_k}\right], \\ \mathbf{d}^k_{\overline{T}_k} &= -\boldsymbol{\alpha}^k_{\overline{T}_k}, \end{aligned} \tag{28}$$

where $\Theta := H_{T_k}(\boldsymbol{\alpha}^k)$. For the computational complexity of Algorithm 1, we observe that calculations of Θ^{-1} and $\mathbb{T}_s$ dominate the whole computation. Their total complexity in each step is approximately

$$O\left(mn + \max\{n, s\}s^2\right). \tag{29}$$

This can be derived by the following analysis:

- Recall the definition (11) of $H(\cdot)$ that
$$\Theta = (E(\boldsymbol{\alpha}^k))_{T_kT_k} + Q^\top_{T_k} Q_{T_k}.$$
The complexity of computing Θ is about $O(ns^2)$ since $|T_k| = s$. Computing its inverse has the complexity of at most $O(s^3)$. Therefore, the complexity of deriving Θ^{-1} is $O(\max\{n, s\}s^2)$.
- To pick T_{k+1} from $\mathbb{T}_s$, we need to compute $g(\mathbf{z}^{k+1})$ and select the k largest elements of $|\boldsymbol{\alpha}^{k+1} - \eta g(\mathbf{z}^{k+1})|$. The complexity of computing the former is $O(mn)$ due to the computation of $Q^\top(Q_{T_k}\boldsymbol{\alpha}^{k+1}_{T_k})$ and the complexity of computing the latter is $O(m + s\ln s)$. Here, we use the MATLAB built-in function `maxk` to select the s largest elements.

We now summarize the complexity of Algorithm 1 and some other methods in Table 2, where in [20] and [39], [40], their proposed method solved a quadratic programming in each iteration k, and N_k was the number of iterations used to solve the quadratic programming. In [37], [38], r was the rank of $Q^\top Q$ or its an approximation. Moreover, $\bar{m}$ was the number of the pre-selected samples from the total samples [20], and m_k was the number of the samples used at the kth iteration [28]. In the setting where data involves large numbers of samples, i.e., m is considerably large in comparison with n, the rank $r \approx n$, which means Algorithm 1 has the lowest computational complexity among the second-order methods. It is well-known that second-order methods can converge quickly within a small number of iterations. Therefore, Algorithm 1 can be executed super-fast, which has been demonstrated by numerical experiments.

TABLE 2: Complexity of different algorithms.

Methods	Reference	Complexity	Descriptions
First-order	[20]	$\mathcal{O}(mn + N_k\bar{m}^2)$	$\bar{m} \in [1, m]$
	[24]	$\mathcal{O}(mn)$	
	[28]	$\mathcal{O}(m_k^3 + mm_k)$	$m_k \in [1, m]$
	[39]	$\mathcal{O}(N_kmn + n\ln(B))$	$B \in (1, n]$
	[40]	$\mathcal{O}(N_km^2)$	
Second-order	[37]	$\mathcal{O}(mr^2)$	$r \in [1, m]$
	[38]	$\mathcal{O}(m^2(n + r))$	$r \in [1, m]$
	[41]	$\mathcal{O}(m^3)$	
	Algorithm 1	$O\left(mn + \max\{n, s\}s^2\right)$	$s \in [1, m]$

3.2 One-Step Convergence

For a point $\mathbf{z}^* = (\boldsymbol{\alpha}^*; b^*)$ with $\boldsymbol{\alpha}^*$ feasible to (8), define

$$\eta^* := \begin{cases} \|\boldsymbol{\alpha}^*\|_{[s]}\|g(\mathbf{z}^*)\|^{-1}_{[1]}, & \|\boldsymbol{\alpha}^*\|_0 = s, \\ +\infty, & \|\boldsymbol{\alpha}^*\|_0 < s, \end{cases} \tag{30}$$

The convergence result is stated by the following theorem.

Theorem 3.1. Let $\mathbf{z}^*$ be an η-stationary point of the problem (8) with $0 < \eta < \eta^*$, where η^* is given by (30). Let $\{\mathbf{z}^k\}$ be the sequence generated by Algorithm 1. There always exists a $\delta^* > 0$ such that if at a certain iteration, $\mathbf{z}^k \in N(\mathbf{z}^*, \delta^*)$, then

$$\mathbf{z}^{k+1} = \mathbf{z}^*, \quad \|F(\mathbf{z}^{k+1}, T_{k+1})\| = 0.$$

Namely, Algorithm 1 terminates at the th$(k + 1)$ step.

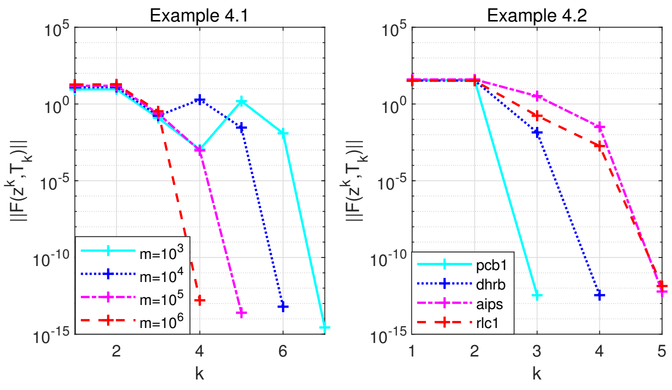

Fig. 1: Decreasing of $\|F(\mathbf{z}^k, T_k)\|$.

We note that Algorithm 1 can terminate at the next step if the current point falls into a local area of an η-stationary point. This means that if by chance the starting point is chosen within the local area, the proposed algorithm will take one step to terminate. Hence, it enjoys a fast convergence property. Of course, this depends on the choices of the initial point. In reality, how close the initial point is to an η-stationary point is unknown. Nevertheless, numerical experiments show that the proposed method is insensitive the choice of the starting point. It can take a few steps to converge. For instance, the results of Algorithm 1 solving Example 4.1 and Example 4.2 with the initial point $\boldsymbol{\alpha}^0 = 0$ and $b^0 = \text{sgn}(\langle \mathbf{y}, \mathbf{1}\rangle)$ were presented in Fig. 1. The line of pcb1 showed that the point $\mathbf{z}^3$ at step 3 may fall into a local area of an η-stationary point since $\|F(\mathbf{z}^4, T_4)\| (\approx 10^{-12})$ almost reached 0 at step 4. Therefore, it terminated at step 4. This showed potential of termination at the next step, even though the initial point was not close to the η-stationary point. Similar observations can be found for all other lines.

Moreover, Theorem 2.4 states that the η-stationary point is a local minimizer of the problem (8). Hence, Algorithm 1 at least converges to a local minimizer. If further $\|\mathbf{z}^*\|_0 < s$ or $\|\mathbf{z}^*\|_0 = s$ and $\eta^* \geq C$, then the η-stationary point is also a global minimizer by Theorem 2.4 c) and d), namely, Algorithm 1 converges to the global minimizer.

3.3 Sparsity Level Tuning

One major issue encountered in practice is that the sparsity level s in (8) is usually unknown beforehand. The level has two important effects: (i) A larger s corresponds to better classifications since more samples are taken into consideration. (ii) However, a smaller s leads to higher computational speed of Algorithm 1 because the complexity in (29) depends on s. Moreover, according to (1) that the number of support vectors is smaller than $\|\boldsymbol{\alpha}\|_0 \leq s$, a smaller s results in a smaller number of support vectors. Therefore, to balance these two requirements, we design the following rule to update the unknown s. We start with a small integer and then increase it until the following halting conditions are satisfied

$$\|F(\mathbf{z}^k; T_k)\| \quad < \quad \epsilon, \tag{31}$$

$$\left|\texttt{ACC}(\boldsymbol{\alpha}^k) - \max_{j\in[k-1]} \texttt{ACC}(\boldsymbol{\alpha}^j)\right| \quad < \quad 10^{-4}, \tag{32}$$

where we admit $\texttt{ACC}(\boldsymbol{\alpha}^{-1}) = 0$ and $\texttt{ACC}(\boldsymbol{\alpha})$ is defined by

$$\texttt{ACC}(\boldsymbol{\alpha}) := \left[1 - \frac{1}{m}\|\text{sgn}(X\boldsymbol{\alpha} + b) - \mathbf{y}\|_0\right] \times 100\%, \tag{33}$$

with $\text{sgn}(t) = 1$ if $t > 0$ and -1 otherwise. The halting condition (31) means that $\boldsymbol{\alpha}^k$ is almost an η-stationary point due to the small $\|F(\mathbf{z}^k; T_k)\|$, while (32) implies that the classification accuracy $\texttt{ACC}(\boldsymbol{\alpha}^k)$ does not increase significantly. Therefore, it is unreasonable to further increase s to achieve a better solution because a larger s will lead to greater computational cost. Our numerical experiments demonstrate that Algorithm 1 under this scheme works very well. Overall, we derive NSSVM in Algorithm 2.

Algorithm 2 NSSVM: Newton method for sparse SVM with adaptively tuning the sparsity level s.

```
Give parameters $C > c, \eta, \epsilon, K > 0, \sigma > 1, s_0 \in [m]$.
Initialize $\mathbf{z}^0$, pick $T_0 \in \mathbb{T}_{s_0}(\boldsymbol{\alpha}^0 - \eta g(\mathbf{z}^0))$ and set $k := 0$.
while $\mathbf{z}^k$ violates (31) and (32) and $k \leq$ MaxIt do
    Update $\mathbf{d}^k$ by solving (26).
    Update $\mathbf{z}^{k+1}$ by (27).
    Update $s_{k+1} = \sigma s_k$ if $k$ is a multiple of 10 or (31) is met, and $s_{k+1} = s_k$ otherwise.
    Update $T_{k+1} \in \mathbb{T}_{s_{k+1}}(\boldsymbol{\alpha}^{k+1} - \eta g(\mathbf{z}^{k+1}))$ and $k := k+1$.
end while
return the solution $\mathbf{z}^k$.
```

4 Numerical Experiments

This part conducts extensive numerical experiments of Algorithm 1 and Algorithm 2 (NSSVM[1]) by using MATLAB (R2019a) on a laptop with 32GB memory and Inter(R) Core(TM) i9-9880H 2.3Ghz CPU.

1. https://github.com/ShenglongZhou/NSSVM

4.1 Testing Examples

We first consider a two-dimensional example with synthetic data, where the features come from Gaussian distributions.

Example 4.1 (Synthetic data in $\mathbb{R}^2$ [6], [42]). Samples $\mathbf{x}_i$ with positive labels $y_i = +1$ are drawn from the normal distribution with mean $(0.5, -3)^\top$ and variance Σ, and samples $\mathbf{x}_j$ with negative labels $y_j = -1$ are drawn from the normal distribution with mean $(-0.5, 3)^\top$ and variance Λ, where Σ and Λ are diagonal matrices with $\Sigma_{11} = \Lambda_{11} = 0.2$, $\Sigma_{22} = \Lambda_{22} = 3$. We generate $2m$ samples with equal numbers of two classes and then evenly split them into a training and a testing set.

Example 4.2 (Real data in higher dimensions). We select 30 datasets with $m \gg n$ from the libraries: libsvm[2], uci[3] and kiggle[4]. Here, $aeb := a \times 10^b$. All datasets are feature-wisely scaled to $[-1, 1]$ and all of the classes that are not 1 are treated as -1. Their details are presented in Table 3, where 7 datasets have the testing data. For each of the dataset without the testing data, we split the dataset into two parts. The first part contains 90% of samples treated as the training data and the rest are the testing data.

To compare the performance of all selected methods, let $\boldsymbol{\alpha}$ be the solution/classifier generated by one method. We report the CPU time (TIME), the training classification accuracy (ACC) by (33) where X and $\mathbf{y}$ are the training samples and classes, the testing classification accuracy (TACC) by (33) where X and $\mathbf{y}$ are the testing samples and classes, and the number of support vectors (NSV).

4.2 Implementation and parameters tuning

The starting point $\mathbf{z}^0$ is initialized as $\boldsymbol{\alpha}^0 = 0$ and $b^0 = \text{sgn}(\langle \mathbf{y}, \mathbf{1}\rangle)$ if no additional information is provided. The maximum number of iterations and the tolerance are set as $K = 1000$, $\epsilon = \max\{\sqrt{m}$ and $\sqrt{n}\}10^{-6}$. Recall that the model (8) involves three important parameters C, c and s and the η-stationary point has the parameter η. We now apply Algorithm 1 for tuning them as described below.

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

TABLE 3: Descriptions of real datasets.

Data	Datasets	Source		Train	Test
			n	m	m_t
pcb1	Polish companies bankruptcy data	uci	64	7027	0
pcb2		uci	64	10173	0
pcb3		uci	64	10503	0
pcb4		uci	64	9792	0
pcb5		uci	64	5910	0
a5a	A5a	libsvm	123	6414	26147
a6a	A6a	libsvm	123	11220	21341
a7a	A7a	libsvm	123	16100	16461
a8a	A8a	libsvm	123	22696	9865
a9a	A9a	libsvm	123	32561	16281
mrpe	malware analysis datasets	kaggle	1024	51959	0
dhrb	hospital readmissions bnary	kaggle	17	59557	0
aips	airline passenger satisfaction	kaggle	22	103904	25976
sctp	santander customer transaction	kaggle	200	200000	0
skin	skin_nonskin	libsvm	3	245056	0
ccfd	credit card fraud dtection	kaggle	28	284807	0
rlc1	record linkage comparison patterns	uci	9	574914	0
···		···	···	···	···
rlc10		uci	9	574914	0
covt	covtype.binary	libsvm	54	581012	0
susy	susy	uci	18	5e6	0
hepm	hepmass	uci	28	7e6	35e5
higg	higgs	uci	28	11e6	0

4.2.1 *Effect of η*

For Example 4.1, we fix $C = 2^{-2}, c = C/2, s = 0.005m$ and vary $\eta \in [10^{-8}, 10^{-2}]$ and $m \in \{2, 4, 6, 8\} \times 10^4$ to examine the effect of η on Algorithm 1. For each case (m, η), we run 100 trials and report the average results in Fig. 2. It is clearly observed that the method was insensitive to η in the range $[10^{-8}, 10^{-6}]$. For each m, the ACC lines were stabilized at a certain level when $\eta \in [10^{-8}, 10^{-6}]$, increased when $\eta \in [10^{-6}, 10^{-4}]$ and then decreased. It appears that η of approximately 10^{-4} gave the best performance for Algorithm 1 in terms of the highest ACC. Moreover, we tested problems with different dimensions and found that $\eta = 1/m$ enabled the algorithm to achieve steady behaviour. Therefore, we set this option for η in the subsequent numerical experiments if no additional information is provided.

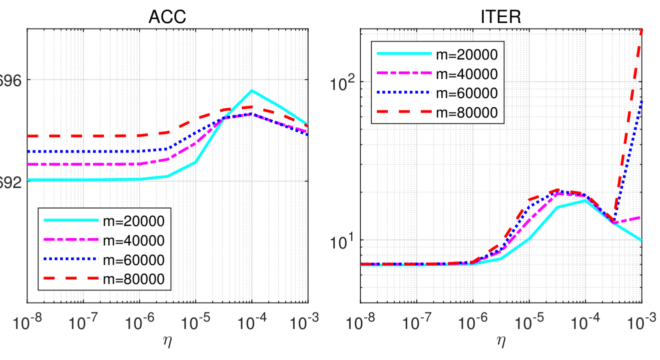

Fig. 2: Effect of η.

4.2.2 *Effect of C and c*

To see the effect of parameters C and c, we choose $C \in \{2^{-3}, 2^{-2}, 2^{-1}, 2^0, 2^1\}$ and $c = aC$, where $a \in [0.001, 1]$. Again, we apply Algorithm 1 for solving Example 4.1 with fixing $m = 10000$ and $s = 0.005m$. The average results were recoded in Fig. 3, where it was observed that ACC reached the highest peak at $a = c/C = 0.01$. When $c/C \geq 0.02$, the bigger C was, the higher ACC was but the larger ITER was as well. To balance the accuracy and the number of iterations, in the following numerical experiments, we set $C = 0.25$ and $c = 0.01C$ unless specified otherwise.

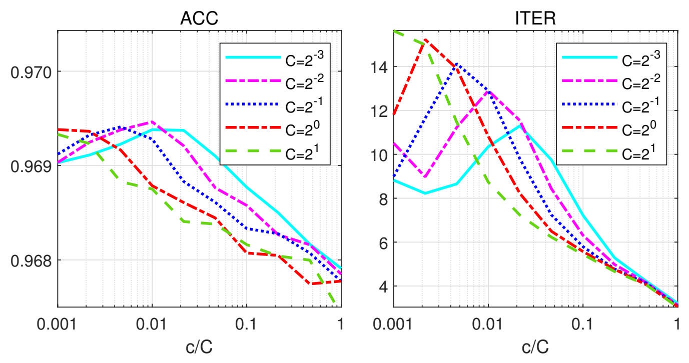

Fig. 3: Effect of C and c.

4.2.3 *Effect of s*

Apparently, the sparsity level s has a strong influence on the solutions to the problem (8) and impacts the computational complexity of Algorithm 1. Below, we use the following rule to tune the sparsity level s,

$$s(\beta) := \lceil \beta n (\log_2(m/n))^2 \rceil,$$

where $\lceil t \rceil$ presents the smallest integer that is no less than t, and β is chosen based on the problem to be solved. We use this rule because it relies on the dimensions of a given problem and allows Algorithm 1 to deliver an overall desirable performance. By setting $s = s(\beta)$, we focus on selecting a proper β. We note that s increases with the rising of β. As demonstrated in Fig. 4, where Example 4.1 was solved again, the average results showed the effect of $\beta \in [0.1, 1]$ (i.e., the effect of s) of the algorithm. It is observed from the figure that ACC increased with increasing s but gradually stabilized at a certain level after $\beta > 0.4$, indicating that there was no need to increase s beyond this point. Furthermore, the small number of iterations showed that Algorithm 1 can converge quickly.

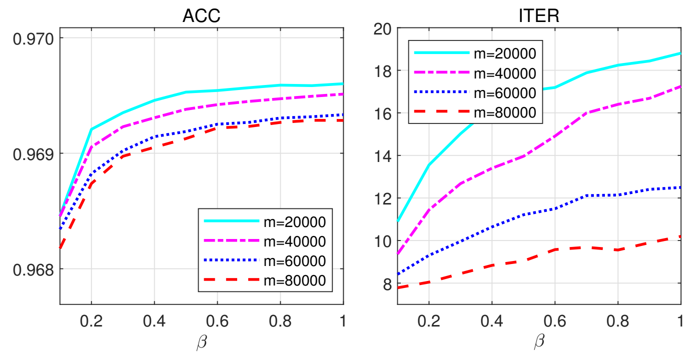

Fig. 4: Effect of s.

4.2.4 *Effect of the initial points*

Theorem 3.1 states that the algorithm can converge at the next step if the initial point is sufficiently close to an η-stationary point that however is unknown beforehand. Therefore, we examine how the initial points affect the performance of Algorithm 1. To carry out this study, we apply the algorithm for solving Example 4.1 with $m = 10^6$ and $s = s(1)$, and Example 4.2 with datasets: `rcl1`, `rcl2`, `rcl3` and $s = s(0.05)$. For each data, we run the algorithm under 50 different initial points $\boldsymbol{\alpha}^0$. The first point is $\boldsymbol{\alpha}^0 = 0$ and the other 49 points have randomly generated entries from the uniform distribution, namely, $\alpha_i^0 \sim U[0, 1]$. The results were plotted in Fig. 5, where the x-axis represents the 50 initial points. For each dataset, all `ACC` stabilized at a certain level, indicating that these initial points had little effect on the accuracy. The results for `ITER` illustrated that the algorithm converged very quickly, stopping within 6 steps for all cases. To summarize, Algorithm 1 was not too sensitive to the choice of the initial point for these datasets. Hence, for simplicity, we initialize our algorithm with $\boldsymbol{\alpha}^0 = 0$ and $b^0 = \mathrm{sgn}(\langle \mathbf{y}, \mathbf{1}\rangle)$ in the sequel.

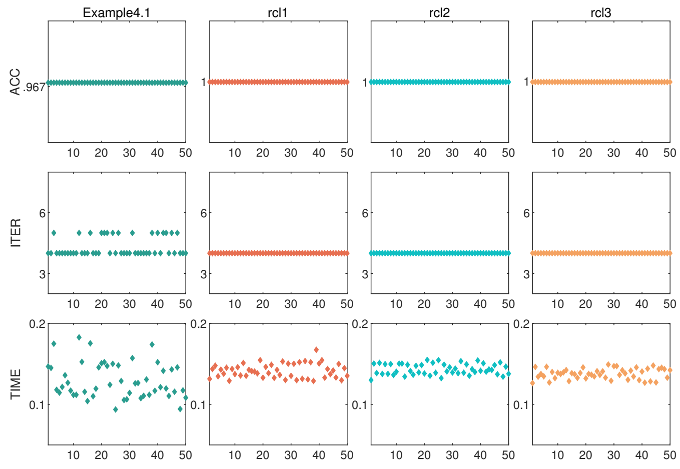

Fig. 5: Robust to the initial points $\boldsymbol{\alpha}^0$.

4.3 Algorithm 1 v.s. Algorithm 2: `NSSVM`

We now turn our attention to `NSSVM` in Algorithm 2. It reduces to Algorithm 1 if we set $\sigma = 1$, namely, $s_k = s_0$ remains invariant. However, to distinguish them, in the following, we always set $\sigma = 1.1$. In this subsection, we focus on how the sparsity level tuning strategy in `NSSVM` affects the final results. For Algorithm 1, we implement it with different fixed $si = s(\beta_i), i = 1, 2, 3$ and name it `Alg1-si`. In particular, we use $\beta_1 = 0.4, \beta_2 = 0.5, \beta_3 = 0.8$. Recall that `NSSVM` also requires an initial sparsity level s_0 that is set as $s_0 = s(0.4) = s1$. Therefore, `NSSVM` and `Alg1-s1` start with the same initial sparsity level.

The average results of two algorithms used to solve Example 4.1 were displayed in Fig. 6. First, despite starting with the same initial sparsity level, `NSSVM` generated higher `ACC` than `Alg1-s1` because it used more support vectors (i.e., higher `NSV`) due to the tuning strategy. Second, `NSSVM` obtained higher `ACC` than that produced by `Alg1-s2` and `Alg1-s3` when $m \geq 70000$, but yielded much lower `NSV`. This indicated that the tuning strategy performed well to find a proper sparsity level to maximize the accuracy. Clearly, `NSSVM` required longer time because the tuning strategy led to more steps to meet the stopping criteria (31). Nevertheless, `NSSVM` still ran quickly, requiring less than 0.03 second and 20 steps to converge.

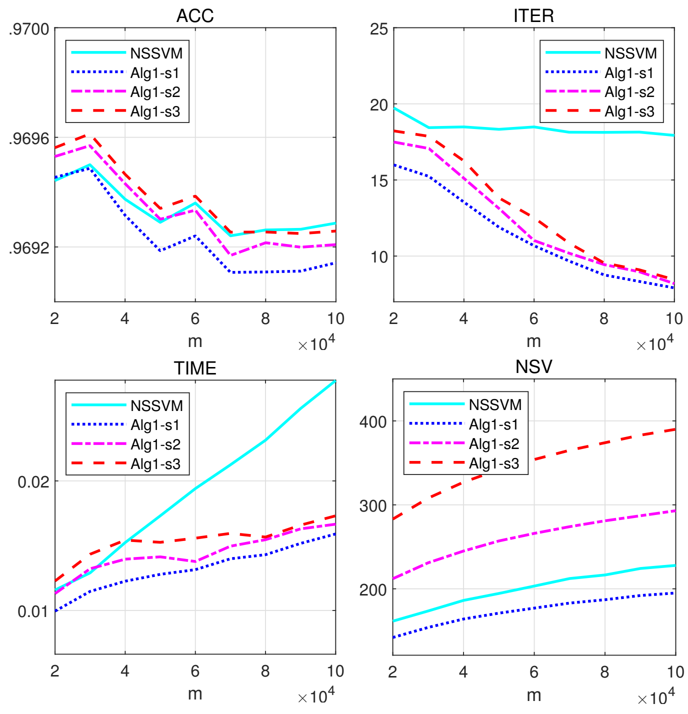

Fig. 6: Algorithm 1 v.s. Algorithm 2

We also compared Algorithm 1 and `NSSVM` for solving the real datasets, and omitted similar observations. In a nutshell, the tuning strategy enables `NSSVM` to deliver higher accuracy with slightly longer computational time. Based on this, we next conduct some numerical comparisons between `NSSVM` and some state-of-the-art solvers.

4.4 Numerical Comparisons

We note that `NSSVM` involves four parameters $(\eta, C, c, s0)$. On the one hand, it is impractical to derive a universal combination of these four parameters for all datasets. On the other hand, tuning them for each dataset will increase the computational cost. Therefore, in the following numerical experiments, we seek to avoid using different parameters for different datasets. Typically, we fix $\eta = 1/m$ and $c = 0.01C$, but select C and s_0 as specified in Table 4.

TABLE 4: Selection of parameters.

	cases	C	s_0
Example 4.1	$m \leq 10^4$	0.25	$s(0.5)$
	$m > 10^4$	0.25	$s(1)$
Example 4.2	`higg, susy, covt, hemp`	$\log_2(m)$	$s(10)$
	`a5a-a9a`	0.25	$s(0.2)$
	other	0.25	$s(0.05)$

4.4.1 *Benchmark methods*

There is an impressive body of work that has developed numerical methods to tackle the SVM, such as those in [43]–[47]. They perform very well, especially for datasets in small or mediate size. Unfortunately, it is unlikely to find a MATLAB implementation for sparse SVM on which we could successfully perform the experiments. For instance, the methods in [45] for RSVM [20] and the one in [24] were programmed by C++ and couldn't run by MATLAB.

TABLE 5: Benchmark methods.

Algs.	Source of Code	Ref.	ℓ
`HSVM`	`libsvm`[5]	[47]	Hinge loss
`SSVM`	`liblssvm`[6]	[43]	Squared Hinge loss
`LSVM`	`liblinear` [7]	[48]	Hinge loss
`FSVM`	`fitclinear` [8]		Hinge loss
`SNSVM`	`Li's lab`[9]	[46]	Squared Hinge loss

Therefore, we select five solvers in Table 5 that do not aim at the sparse SVM. They address the soft-margin SVM models (2) with different loss functions ℓ. Note that `fitclinear` is a Matlab built-in solver with initializing `Solver = 'dual'` in order to obtain the number of the support vectors. For the same reason, we set `-s 3` for `FSVM`. It runs much faster if we set `-s 2`, but such a setting does not offer the number of support vectors. The other three methods also involve some parameters. For simplicity, all of these parameters are set to their default values.

4.4.2 Comparisons for datasets with small sizes

We first applied five solvers for solving Example 4.1 with small scales $m = 200, 400$ and depicted the classifiers in Fig. 7 where the Bayes classifier was $w_2 = 0.25w_1$, see the black dotted line. We did not run `HSVM` because it solved the dual problem and did not provide the solution $(\mathbf{w}, b)$. Overall, all solvers can classify the dataset well.

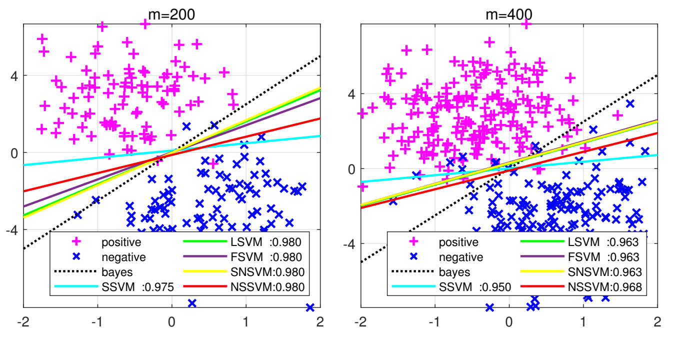

Fig. 7: Classifiers by five solvers for Example 4.1.

We then used six solvers to address Example 4.1 with different sample sizes $m \in [2000, 10^4]$. Average results over 100 trials were reported in Fig. 8. It is observed that `NSSVM` obtained the largest `ACC`, but rendered smallest `TACC` for most cases. This was probably due to the smallest number of support vectors used. Evidently, `LSVM` and `SNSVM` ran the fastest, followed by `FSVM` and `NSSVM`.

Next, six methods were used to handle Example 4.2 with small scale datasets. As shown in Table 6, where '−−' denoted that `SSVM` required memory that was beyond the capacity of our laptop; here, again `NSSVM` did not show the advantage of delivering the accuracy. This was not surprising because it used a relatively small number of support vectors. However, it ran the fastest for most datasets.

To summarize, `NSSVM` did not present the advantage of higher accuracy for dealing with small scale datasets. One reason for this is that the solution to the model (8) may not be sufficiently sparse for small scale datasets. In other words, there is no strong need to perform data reduction for these datasets. Hence, in the sequel, we employ four

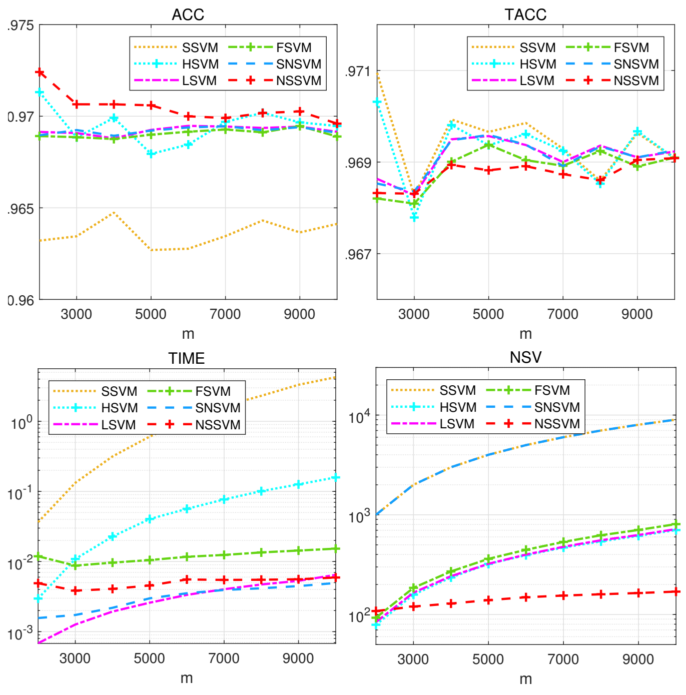

Fig. 8: Average results of six solvers for Example 4.1.

methods to address test problems with datasets on much larger scales. However, we remove `HSVM` and `SSVM` from comparisons in the following since they either are too slow or required memory beyond the capability of our device.

4.4.3 Comparisons for datasets with large sizes

This part focus on instances with much larger scales. First, we employ four algorithms to solve Example 4.1 with m from $[10^5, 10^8]$. The data reported in Table 7 were the results averaged over 20 trials. Since `LSVM` ran for too long when $m > 10^6$, its results were omitted. The training classification accuracy values for `NSSVM`, `SNSVM` and `LSVM` were similar and were better than those obtained by `FSVM`. However, it is clearly observed that `NSSVM` ran the fastest and used much fewer support vectors. Moreover, the superiority of `NSSVM` becomes more evident with larger m.

TABLE 7: Results of four solvers solving Example 4.1 with large-scale datasets.

m	`LSVM`	`FSVM`	`SNSVM`	`NSSVM`	`LSVM`	`FSVM`	`SNSVM`	`NSSVM`
	`ACC` (%)				`TACC` (%)			
10^5	96.95	96.91	96.95	96.95	96.96	96.93	96.96	96.94
10^6	96.94	96.91	96.94	96.94	96.94	96.93	96.94	96.93
10^7	−−	96.92	96.93	96.93	−−	96.93	96.93	96.93
10^8	−−	96.91	96.93	96.93	−−	96.93	96.93	96.93
	`TIME` (in seconds)				`NSV`$/m$			
10^5	0.10	0.10	0.04	0.04	7.83e-2	8.85e-2	1	**5.94e-3**
10^6	3.88	2.43	0.36	**0.33**	7.81e-2	8.87e-2	1	**8.62e-4**
10^7	−−	30.62	3.42	**2.08**	−−	8.88e-2	1	**1.09e-4**
10^8	−−	447.1	34.1	**12.3**	−−	8.89e-2	1	**1.44e-5**

Results obtained by four solvers for classifying real datasets in Example 4.2 on larger scales were reported in Table 8. For the accuracy, there was no significant difference of `ACC` and `TACC` produced by four solvers. For the computational speed, it is clearly observed that `NSSVM` ran the

TABLE 6: Results of six solvers solving Example 4.2 with datasets with small sizes.

	SSVM	HSVM	LSVM	FSVM	SNSVM	NSSVM
			ACC (%)			
pcb1	96.21	96.22	96.22	96.22	96.24	96.24
pcb2	95.95	95.96	95.96	95.96	**95.97**	95.95
pcb3	95.31	95.31	95.31	95.31	95.30	95.31
pcb4	94.71	94.74	94.74	94.74	94.72	94.71
pcb5	93.04	93.12	93.16	93.16	**93.18**	93.04
a5a	24.46	85.06	85.05	81.82	**85.16**	84.10
a6a	−−	**84.89**	84.87	76.11	84.80	83.95
a7a	−−	84.88	**84.96**	81.93	84.89	84.16
a8a	−−	84.69	84.67	83.00	**84.74**	84.01
a9a	−−	84.99	84.99	80.63	84.96	84.04
			TACC (%)			
pcb	**95.73**	95.58	95.58	95.58	95.58	95.58
pcb2	96.85	97.15	97.15	97.15	97.15	97.15
pcb3	95.05	95.05	95.05	95.05	95.05	95.05
pcb4	94.69	94.99	94.99	94.99	94.89	94.99
pcb5	93.06	93.23	93.23	93.23	93.23	93.23
a5a	84.42	84.39	84.39	81.07	**84.50**	83.60
a6a	−−	84.72	84.66	76.12	**84.77**	84.13
a7a	−−	84.84	84.82	82.26	**85.00**	84.00
a8a	−−	85.16	85.13	83.47	**85.42**	84.59
a9a	−−	84.98	**85.00**	80.96	84.94	84.47
			TIME (in seconds)			
pcb1	1.905	0.702	**0.037**	0.241	0.057	0.082
pcb2	4.656	1.577	0.051	0.243	0.064	**0.039**
pcb3	4.987	1.761	0.046	0.064	0.044	**0.019**
pcb4	4.282	1.722	0.048	0.051	0.038	**0.016**
pcb5	1.241	0.805	0.044	0.033	0.021	**0.013**
a5	29.72	5.414	**0.030**	0.110	0.044	0.049
a6a	−−	11.15	**0.051**	0.247	0.107	0.085
a7a	−−	19.15	0.067	**0.072**	0.179	0.097
a8a	−−	33.07	0.105	**0.067**	0.287	0.121
a9a	−−	70.59	0.158	**0.072**	0.361	0.130
			NSV			
pcb1	6325	497	2320	2263	6325	**190**
pcb2	9156	791	4166	3557	9156	**182**
pcb3	9453	981	4692	4375	9453	**184**
pcb4	8813	990	4538	4762	8813	**179**
pcb5	5319	758	2167	2440	5319	**145**
a5a	6414	2294	2332	3891	6414	**882**
a6a	−−	4011	4082	6910	11220	**1148**
a7a	−−	5757	5899	9931	16100	**1339**
a8a	−−	8135	8292	13992	22696	**1534**
a9a	−−	11533	11772	20030	32561	**1754**

fastest for all datasets. With regard to the number of support vectors, NSSVM outperformed the other methods because it obtained the smallest NSV. Taking higg as an instance, the other three solvers respectively produced $0.237m, 0.73m$, and m support vectors and consumed 2938, 46.86 and 61.75 seconds to classify the data. By contrast, our proposed method used $0.00856m$ support vectors and only required 4.78 seconds to achieve the highest classification accuracy.

In summary, NSSVM delivered the desirable classification accuracy and ran rapidly by using a relatively small number of support vectors for large-scale datasets.

5 Conclusion

The sparsity constraint was employed in the kernel-based SVM model (8) allowing for the control of the number of support vectors. Therefore, the memory required for data storage and the computational cost were significantly reduced, enabling large scale computation. We feel that the established theory and the proposed Newton type method deserve to be further explored for using the SVM model with some nonlinear kernel-based models.

TABLE 8: Results of four solvers solving Example 4.2 with datasets with large sizes.

	LSVM	FSVM	SNSVM	NSSVM	LSVM	FSVM	SNSVM	NSSVM
		ACC (%)				TACC (%)		
mrpe	95.01	94.78	**95.08**	95.00	95.25	94.90	95.25	95.25
dhrb	82.70	82.71	**83.25**	82.76	83.43	83.43	**84.11**	83.48
aips	**87.66**	87.26	87.42	86.54	**87.41**	87.08	87.13	85.99
sctp	89.95	78.01	**91.20**	90.50	90.00	77.80	**91.19**	90.51
skin	**92.90**	92.73	92.37	91.04	**92.66**	92.45	92.23	90.63
ccfd	99.94	99.94	99.92	99.94	99.92	99.92	99.88	99.92
rlc1	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
rlc2	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
rlc3	100.0	100.0	100.0	100.0	100.0	100.0	99.99	100.0
rlc4	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
rlc5	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
rlc6	100.0	100.0	100.0	100.0	100.0	100.0	99.99	99.99
rlc7	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
rlc8	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
rlc9	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
rlc10	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
covt	**76.29**	75.73	75.67	75.67	**76.15**	75.47	75.54	75.51
susy	78.44	78.55	78.66	**78.82**	78.39	78.52	78.63	**78.79**
hepm	78.36	83.31	83.63	**83.74**	78.33	83.23	83.61	**83.71**
higg	47.01	63.84	64.13	**64.16**	46.98	63.79	64.07	**64.08**
		TIME (in seconds)				NSV/m		
mrpe	31.75	2.28	5.37	**1.06**	8.23e-1	5.60e-1	1	**3.63e-2**
dhrb	0.17	0.10	0.07	**0.04**	7.68e-1	6.92e-1	1	**2.59e-3**
aips	0.87	0.21	0.18	**0.09**	3.01e-1	5.30e-1	1	**2.33e-3**
sctp	13.48	1.09	10.61	**0.38**	3.82e-1	5.79e-1	1	**5.84e-3**
skin	0.42	0.33	0.11	**0.08**	1.97e-1	2.55e-1	1	**2.20e-4**
ccfd	2.66	0.52	0.40	**0.16**	5.56e-3	1.01e-2	1	**9.41e-4**
rlc1	0.39	0.66	0.25	**0.11**	3.32e-4	5.95e-4	1	**2.17e-4**
rlc2	0.39	0.51	0.26	**0.17**	2.47e-4	5.13e-4	1	**2.40e-4**
rlc3	0.49	0.56	0.23	**0.12**	2.85e-4	5.86e-4	1	**2.17e-4**
rlc4	0.38	0.67	0.25	**0.18**	2.68e-4	5.79e-4	1	**2.40e-4**
rlc5	0.40	0.57	0.26	**0.12**	2.89e-4	5.76e-4	1	**2.17e-4**
rlc6	0.50	0.51	0.27	**0.21**	2.44e-4	5.51e-4	1	**2.64e-4**
rlc7	0.38	0.52	0.25	**0.12**	2.84e-4	5.31e-4	1	**2.17e-4**
rlc8	0.68	0.67	0.25	**0.12**	2.68e-4	5.77e-4	1	**2.17e-4**
rlc9	0.66	0.54	0.25	**0.12**	3.04e-4	6.00e-4	1	**2.17e-4**
rlc10	0.39	0.55	0.25	**0.12**	3.25e-4	5.91e-4	1	**2.17e-4**
covt	13.23	2.05	9.22	**1.11**	4.75e-1	5.93e-1	1	**1.61e-1**
susy	596.3	16.69	16.44	**2.43**	4.15e-1	4.74e-1	1	**1.26e-2**
hepm	1688	28.39	26.97	**3.62**	3.45e-1	4.76e-1	1	**1.27e-2**
higg	2938	46.86	61.75	**4.78**	2.37e-1	7.30e-1	1	**8.56e-3**

Acknowledgements

I sincerely thank the two referees and Professor Naihua Xiu from Beijing Jiaotong University for their valuable suggestions to improve this paper.

References

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

[2] X. Huang, L. Shi and A.K. Suykens, Support vector machine classifier with pinball loss, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(5), 984-997, 2014.

[3] V. Jumutc, X. Huang and A. Suykens, Fixed-size Pegasos for hinge and pinball loss SVM, *International Joint Conference on Neural Networks*, 2013.

[4] S. Rosset and J. Zhu, Piecewise linear regularized solution paths, *The Annals of Statistics*, 35(3), 1012-1030, 2007.

[5] L. Wang, J. Zhu, and H. Zou, Hybrid huberized support vector machines for microarray classification, *Bioinformatics*, 24(3), 412-419, 2008.

[6] Y. Xu, I. Akrotirianakis, and A. Chakraborty, Proximal gradient method for huberized support vector machine, *Pattern Analysis and Applications*, 19(4), 989-1005, 2016.

[7] A. Suykens and J. Vandewalle, Least squares support vector machine classifiers, *Neural Processing Letters*, 9(3), 293-300, 1999.

[8] X. Yang, L. Tan and L. He, A robust least squares support vector machine for regression and classification with noise, *Neurocomputing*, 140, 41-52, 2014.

[9] Y. Freund and R. E. Schapire, A decision theoretic generalization of on-line learning and an application to boosting, *Journal of Computer and System Sciences,* 55(1), 119-139, 1997.

[10] J. Friedman, T. Hastie, and R. Tibshirani, Additive logistic regression: a statistical view of boosting, *Annals of statistics,* 28(2), 337-374, 2000.

[11] L. Mason, P. Bartlett and J. Baxter, Improved generalization through explicit optimization of margins, *Machine Learning,* 38(3), 243-255, 2000.

[12] F. Pérez-Cruz, A. Navia-Vazquez, A. Figueiras-Vidal and A. Artes-Rodriguez. Empirical risk minimization for support vector classifiers, *IEEE Transactions on Neural Networks,* 14(2), 296-303, 2003.

[13] R. Collobert, F. Sinz, J. Weston, and L. Bottou, Large scale transductive SVMs, *Journal of Machine Learning Research,* 7(1), 1687-1712, 2006.

[14] X. Shen, L. Niu, Z. Qi, and Y. Tian, Support vector machine classifier with truncated pinball loss, *Pattern Recognition,* 68, 2017.

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

[16] F. Pérez-Cruz, A. Navia-Vazquez, P. Alarcón-Dian and A. Artes-Rodriguez, Support vector classifier with hyperbolic tangent penalty function, *Acoustics, Speech, and Signal Processing,* 2000.

[17] L. Mason, J. Baxter, P. Bartlett and M. Frean, Boosting algorithms as gradient descent, *In Advances in neural information processing systems,* 1999.

[18] E. Osuna and G. Federico, Reducing the run-time complexity of support vector machines, *In International Conference on Pattern Recognition*, 1998.

[19] Y. Zhan and D. Shen, Design efficient support vector machine for fast classification, *Pattern Recognition*, 38(1), 157-161, 2005.

[20] Y. Lee and O. Mangasarian, RSVM: Reduced support vector machines, *In Proceedings of the 2001 SIAM International Conference on Data Mining*, 1-17, 2001.

[21] J. Bi, K. Bennett, M. Embrechts, C. Breneman, and M. Song, Dimensionality reduction via sparse support vector machines, *Journal of Machine Learning Research*, 3, 1229-1243, 2003.

[22] R. Tibshirani, Regression shrinkage and selection via the lasso, *Journal of the Royal Statistical Society: Series B (Methodological)*, 58.1, 267-288, 1996.

[23] S. Keerthi, O. Chapelle, and D. DeCoste, Building support vector machines with reduced classifier complexity, *Journal of Machine Learning Research*, 7, 1493-1515, 2006.

[24] A. Cotter, S. Shalev-Shwartz and N. Srebro, Learning optimally sparse support vector machines, *In International Conference on Machine Learning*, 266-274, 2013.

[25] C. Burges, Simplified support vector decision rules, *In ICML*, 96, 71-77,1996 .

[26] C. Burges and B. Schölkopf, Improving the accuracy and speed of support vector machines, *In Advances in neural information processing systems*, 375–381, 1997.

[27] O. Dekel, S. Shalev-Shwartz and Y. Singer, The Forgetron: A kernel-based perceptron on a fixed budget. *In Advances in neural information processing systems*, 259-266, 2006.

[28] D. Nguyen, K. Matsumoto, Y. Takishima and K. Hashimoto, Condensed vector machines: learning fast machine for large data, *IEEE transactions on neural networks*, 21(12), 1903-1914, 2010.

[29] R. Koggalage and S. Halgamuge, Reducing the number of training samples for fast support vector machine classification, *Neural Information Processing-Letters and Reviews*, 2(3), 57-65, 2004.

[30] M. Wu, B. Schölkopf and G. Bakir, Building sparse large margin classifiers, *In Proceedings of the 22nd international conference on Machine learning*, 996-1003, 2005.

[31] S. Wang, Z. Li, C. Liu, X. Zhang and H. Zhang, Training data reduction to speed up SVM training, *Applied intelligence*, 41(2), 405-420, 2014.

[32] N. Panda, E.Y. Chang and G. Wu, Concept boundary detection for speeding up SVMs, *In Proceedings of the 23rd international conference on Machine learning*, pp. 681-688, 2006.

[33] H. Graf, E. Cosatto, L. Bottou, I. Dourdanovic and Vapnik, V., Parallel support vector machines: The cascade SVM. *Advances in neural information processing systems*, 17, pp.521-528, 2004.

[34] R. Collobert, S. Bengio and Y. Bengio, A parallel mixture of SVMs for very large scale problems, Neural computation, 14(5), 1105-1114, 2002.

[35] C. Williams and M. Seeger, Using the Nyström method to speed up kernel machines. I*n Proceedings of the 14th annual conference on neural information processing systems*, No. CONF, 682-688 , 2001.

[36] A. Smola, and B. Schölkopf, Sparse greedy matrix approximation for machine learning, 2000.

[37] S. Fine and K. Scheinberg, Efficient SVM Training Using Low-Rank Kernel Representations, *Journal of Machine Learning Research*, 2, 243–264, 2001.

[38] S. Zhang, R. Shah, and P. Wu, TensorSVM: accelerating kernel machines with tensor engine, *in Proceedings of the 34th ACM International Conference on Supercomputing*, Barcelona, Spain, 1–11, 2020.

[39] M. Tan, L. Wang, and I. W. Tsang, Learning sparse svm for feature selection on very high dimensional datasets, *In ICML*, 2010.

[40] Z. Liu, D. Elashoff and S. Piantadosi, Sparse support vector machines with l_0 approximation for ultra-high dimensional omics data. Artificial intelligence in medicine, 96, 134-141, 2019.

[41] J. Lopez, K. De Brabanter, J. R. Dorronsoro, and J. A. K. Suykens, Sparse LS-SVMs with L_0-norm minimization, *In Proceedings of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 189-194. 2011.

[42] X. Huang, L. Shi and J. Suykens, Solution path for pin-svm classifiers with positive and negative values, *IEEE transactions on neural networks and learning systems*, 28(7), 1584-1593, 2016.

[43] K. Pelckmans, J. Suykens, T. Gestel, J. Brabanter, L. Lukas, B. Hamers, B. Moor and J. Vandewalle, A matlab/c toolbox for least square support vector machines, *ESATSCD-SISTA Technical Report*, 02-145, 2002.

[44] Y. Wu and Y. Liu, Robust truncated hinge loss support vector machines, *Journal of the American Statistical Association*, 102(479), 974-983 ,2007.

[45] K. Lin and C. Lin., A study on reduced support vector machines, *IEEE transactions on Neural Networks*, 14(6), 1449-1459, 2003.

[46] J. Yin and Q. Li, A semismooth Newton method for support vector classification and regression, *Computational Optimization and Applications*, 73(2), 477-508, 2019.

[47] C. Chang and C. Lin, LIBSVM: A library for support vector machines, *ACM transactions on intelligent systems and technology*, 2(3), 1-27, 2011.

[48] R. Fan, K. Chang, C. Hsieh, X. Wang and C. Lin, Liblinear: A library for large linear classification, *Journal of machine learning research*, 9, 1871-1874, 2008.

[49] L. Pan, J. Fan and N. Xiu, Optimality conditions for sparse nonlinear programming, *Science China Mathematics*, 60(5), 759-776, 2017.

[50] A. Beck and Y. Eldar, Sparsity constrained nonlinear optimization: Optimality conditions and algorithms, SIAM Journal on Optimization, 23(3), 1480–1509, 2013.

Shenglong Zhou received the B.Sc. degree in information and computing science in 2011 and the M.Sc. degree in operational research in 2014 from Beijing Jiaotong University, China, and the Ph.D. degree in operational research in 2018 from the University of Southampton, the United Kingdom, where he was a Research Fellow and Teaching Fellow from 2017 to 2021. He is currently a Research Associate at the Department of EEE, Imperial College London. His research interests include the theory and methods of optimization in the fields of sparse, low-rank matrix and bilevel optimization.

Appendix A
Proofs of all Theorems

Proof of Theorem 2.1

proof By introducing $u_i = 1 - y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle + b)$, the problem (4) is rewritten as

$$\begin{aligned}\min_{\mathbf{w},\mathbf{u}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^m \ell_{cC}(u_i), \\ \text{s.t.} \quad & u_i + y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle + b) = 1,\ i \in [m].\end{aligned} \tag{34}$$

The Lagrangian function of the above problem is

$$\begin{aligned}L(\mathbf{w},\mathbf{u},b,\boldsymbol{\alpha}) \ := \ & \frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^m \ell_{cC}(u_i) \\ & - \sum_{i=1}^m \alpha_i\Big(u_i + y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle + b) - 1\Big).\end{aligned}$$

The dual problem of (34) is then given by

$$\max_{\boldsymbol{\alpha}} \min_{\mathbf{w},\mathbf{u},b} L(\mathbf{w},\mathbf{u},b,\boldsymbol{\alpha}). \tag{35}$$

For a fixed $\boldsymbol{\alpha}$, the inner problem $\min_{\mathbf{w},\mathbf{u},b} L(\mathbf{w},\mathbf{u},b,\boldsymbol{\alpha})$ has three variables that are independent of each other. Hence, it can be divided into three sub-problems

$$\begin{aligned}& \min_{\mathbf{w},\mathbf{u},b} \ L(\mathbf{w},\mathbf{u},b,\boldsymbol{\alpha}) \\ = \ & \min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^m \alpha_i y_i \langle \mathbf{w}, \mathbf{x}_i\rangle + \sum_{i=1}^m \alpha_i \\ + \ & \min_{\mathbf{u}} \sum_{i=1}^m \Big(\ell_{cC}(u_i) - \alpha_i u_i\Big) - \min_b \sum_{i=1}^m \alpha_i y_i b.\end{aligned} \tag{36}$$

For the sub-problem with respect to $\mathbf{w}$, the optimal solution $\mathbf{w}$ is attained at $w_j - \sum_{i=1}^m \alpha_i y_i x_{ij} = 0, i \in [n]$, namely,

$$\mathbf{w} = Q\boldsymbol{\alpha}. \tag{37}$$

using notation Q in Table 1. Substituting (37) into the following objective function yields

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^m \alpha_i y_i \langle \mathbf{w}, \mathbf{x}_i\rangle = -\frac{1}{2}\|Q\boldsymbol{\alpha}\|^2. \tag{38}$$

For the sub-problem with respect to $\mathbf{u}$, the optimal solution $\mathbf{u}$ is attained at, for $i \in [m]$,

$$0 = \ell'_{c,C}(u_i) - \alpha_i \iff \alpha_i = \begin{cases} Cu_i, & u_i \geq 0, \\ cu_i, & u_i < 0, \end{cases} \tag{39}$$

where $\ell'_{c,C}(u_i)$ is the derivative of $\ell'_{c,C}$ at u_i that leads to

$$\begin{aligned}\ell_{cC}(u_i) - \alpha_i u_i \ &= \begin{cases} Cu_i^2/2 - \alpha_i u_i, & u_i \geq 0, \\ cu_i^2/2 - \alpha_i u_i, & u_i < 0, \end{cases} \\ &= \begin{cases} -\alpha_i^2/(2C), & \alpha_i \geq 0, \\ -\alpha_i^2/(2c), & \alpha_i < 0, \end{cases} \\ &= -h_{cC}(\alpha_i).\end{aligned}$$

Therefore, we have the sub-problem

$$\min_{\mathbf{u}} \sum_{i=1}^m \Big(\ell_{cC}(u_i) - \alpha_i u_i\Big) = -\sum_{i=1}^m h_{cC}(\alpha_i). \tag{40}$$

For the third sub-problem with respect to b, the optimal solution is attained at

$$\langle \boldsymbol{\alpha}, \mathbf{y}\rangle = \sum_{i=1}^m \alpha_i y_i = 0. \tag{41}$$

This results in

$$\min\nolimits_b \sum_{i=1}^m \alpha_i y_i b = 0.$$

Overall, the sub-problem (36) becomes

$$\min_{\mathbf{w},\mathbf{u},b} L(\mathbf{w},\mathbf{u},b,\boldsymbol{\alpha}) = -\frac{1}{2}\|Q\boldsymbol{\alpha}\|^2 + \sum_{i=1}^m \alpha_i - h_{cC}(\alpha_i) \tag{42}$$

where $\boldsymbol{\alpha}$ satisfies (41). Hence, the dual problem (35) is equivalent to

$$\begin{aligned}& \min_{\boldsymbol{\alpha}} \Big\{ -\min_{\mathbf{w},\mathbf{u},b} L(\mathbf{w},\mathbf{u},b,\boldsymbol{\alpha})\Big\} \\ = \ & \min_{\boldsymbol{\alpha}} \left\{ \frac{1}{2}\|Q\boldsymbol{\alpha}\|^2 - \sum_{i=1}^m \alpha_i + h_{cC}(\alpha_i) : \langle \boldsymbol{\alpha}, \mathbf{y}\rangle = 0 \right\},\end{aligned} \tag{43}$$

which is same as (6). Let $(\widehat{\mathbf{w}}, \widehat{\mathbf{u}}, \widehat{b})$ and $\boldsymbol{\alpha}^*$ be the optimal solutions to the primal problem (34) and dual problem (43), respectively. Then from (37), $\widehat{\mathbf{w}} = Q\boldsymbol{\alpha}^*$. For $\widehat{b}$, it follows

$$\begin{aligned} y_i\widehat{b} \ & \overset{(34)}{=} \ 1 - \widehat{u}_i - \langle \widehat{\mathbf{w}}, y_i \mathbf{x}_i\rangle \\ & \overset{(37)}{=} \ 1 - \widehat{u}_i - \langle Q\boldsymbol{\alpha}^*, y_i \mathbf{x}_i\rangle \\ & \overset{(39),(10)}{=} \ 1 - E_{ii}(\boldsymbol{\alpha}^*)\alpha_i^* - \langle Q\boldsymbol{\alpha}^*, y_i \mathbf{x}_i\rangle, \\ & \overset{(15)}{=} \ 1 - E_{iS_*}(\boldsymbol{\alpha}^*)\alpha_{S_*}^* - \langle Q_{S_*}\boldsymbol{\alpha}_{S_*}^*, Q_i\rangle, \\ & \overset{(11)}{=} \ 1 - H_{iS_*}(\boldsymbol{\alpha}^*)\alpha_{S_*}^*, \end{aligned}$$

for any $i \in S_* = \text{supp}(\boldsymbol{\alpha}^*)$. This leads to

$$\widehat{b} = y_i - y_i H_{iS_*}(\boldsymbol{\alpha}^*)\alpha_{S_*}^*, \forall\, i \in S_*.$$

Summing both sides of the above equation for all $i \in S_*$ yields $\widehat{b}$ in (16), completing the proof. □

Proof of Theorem 2.2

proof The solution set is nonempty since 0 satisfies the constraints of (8). The problem can be written as

$$\min_{|T|\leq s, T\subseteq[m]} \Big\{ \min_{\boldsymbol{\alpha}\in\mathbb{R}^m} D(\alpha) : \langle \boldsymbol{\alpha}_T, \mathbf{y}_T\rangle = 0 \Big\}. \tag{44}$$

It follows from (12) that $D(\cdot)$ is strongly convex. Therefore, the inner problem is a strongly convex program that admits a unique solution, say α_T. In addition, the choices of T such that $|T| \leq s, T \subseteq [m]$ are finitely many. To derive the global optimal solution, we pick one T from the choices making $D(\alpha_T)$ the smallest. □

Proof of Theorem 2.3

proof It follows from $\mathbf{z}^*$ being an η-stationary point and (21) that $\langle \boldsymbol{\alpha}^*, \mathbf{y}\rangle = 0$ and

$$\begin{aligned}\boldsymbol{\alpha}^* \ & \in \ \mathbb{P}_s(\boldsymbol{\alpha}^* - \eta g(\mathbf{z}^*)) \\ & \overset{(19)}{=} \left\{ \begin{bmatrix} \boldsymbol{\alpha}_T^* - \eta g_T(\mathbf{z}^*) \\ 0 \end{bmatrix} : T \in \mathbb{T}_s(\boldsymbol{\alpha}^* - \eta g(\mathbf{z}^*)) \right\},\end{aligned}$$

which is equivalent to stating that there exists a $T_* \in \mathbb{T}_s(\boldsymbol{\alpha}^* - \eta g(\mathbf{z}^*))$ satisfying $\boldsymbol{\alpha}_{\overline{T}_*}^* = 0$ and $0 = g_{T_*}(\mathbf{z}^*)$. This can reach the conclusion immediately. □

A Lemma for Theorem 2.4

To prove Theorem 2.4, we need the following Lemma.

Lemma A.1. Consider a feasible point $\boldsymbol{\alpha}^*$ to the problem (8), namely, $\|\boldsymbol{\alpha}^*\|_0 \le s$ and $\langle \boldsymbol{\alpha}^*, \mathbf{y}\rangle = 0$.

a) It is a local minimizer if and only if there is $b^* \in \mathbb{R}$ such that

$$\left\{\begin{array}{rcl} g_{S_*}(\mathbf{z}^*) &=& 0,\\ \langle \boldsymbol{\alpha}^*, \mathbf{y}\rangle &=& 0,\\ \|\boldsymbol{\alpha}^*\|_0 &=& s, \end{array}\right. \quad \text{or} \quad \left\{\begin{array}{rcl} g(\mathbf{z}^*) &=& 0,\\ \langle \boldsymbol{\alpha}^*, \mathbf{y}\rangle &=& 0,\\ \|\boldsymbol{\alpha}^*\|_0 &<& s. \end{array}\right. \tag{45}$$

b) If $\|\boldsymbol{\alpha}^*\|_0 < s$, then the local minimizer and the global minimizer are identical to each other and unique.

proof We only prove that the global minimizer $\boldsymbol{\alpha}^*$ is unique if $\|\boldsymbol{\alpha}^*\|_0 < s$. The proofs of the remaining parts can be seen in [49, Theorem 3.2]. The condition in (12) indicates $D(\boldsymbol{\alpha})$ is a strongly convex function and thus enjoys the property

$$\begin{array}{rcl} D(\boldsymbol{\alpha}) &\overset{(12)}{\ge}& D(\boldsymbol{\alpha}') + \langle \nabla D(\boldsymbol{\alpha}'), \boldsymbol{\alpha} - \boldsymbol{\alpha}'\rangle \\ &+& \langle \boldsymbol{\alpha} - \boldsymbol{\alpha}', P(\boldsymbol{\alpha} - \boldsymbol{\alpha}')\rangle/2, \end{array} \tag{46}$$

for any $\boldsymbol{\alpha}, \boldsymbol{\alpha}' \in \mathbb{R}^m$. If there is another global minimizer $\boldsymbol{\alpha} \ne \boldsymbol{\alpha}^*$, then the strong convexity of $D(\cdot)$ gives rise to

$$\begin{array}{rl} & D(\boldsymbol{\alpha}) - D(\boldsymbol{\alpha}^*) \\ \overset{(46)}{\ge} & \langle \boldsymbol{\alpha} - \boldsymbol{\alpha}^*, P(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)\rangle/2 + \langle \nabla D(\boldsymbol{\alpha}^*), \boldsymbol{\alpha} - \boldsymbol{\alpha}^*\rangle \\ \overset{(14)}{=} & \langle \boldsymbol{\alpha} - \boldsymbol{\alpha}^*, P(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)\rangle/2 + \langle g(\mathbf{z}^*) - \mathbf{y}b^*, \boldsymbol{\alpha} - \boldsymbol{\alpha}^*\rangle \\ \overset{(45)}{=} & \langle \boldsymbol{\alpha} - \boldsymbol{\alpha}^*, P(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)\rangle/2 - \langle \mathbf{y}b^*, \boldsymbol{\alpha} - \boldsymbol{\alpha}^*\rangle \\ \overset{(45)}{=} & \langle \boldsymbol{\alpha} - \boldsymbol{\alpha}^*, P(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)\rangle/2 \\ \overset{(12)}{>} & 0, \end{array}$$

where the third equation is valid because global minimizers $\boldsymbol{\alpha}$ and $\boldsymbol{\alpha}^*$ satisfy $\langle \boldsymbol{\alpha}^*, \mathbf{y}\rangle = \langle \boldsymbol{\alpha}, \mathbf{y}\rangle = 0$. It follows from the global optimality that $D(\boldsymbol{\alpha}^*) = D(\boldsymbol{\alpha})$, contradicting the above inequality. Therefore, $\boldsymbol{\alpha}^*$ is unique. □

Proof of Theorem 2.4

proof a) It follows from [50, Lemma 2.2] that an η-stationary point in (21) can be equivalently written as

$$\left\{\begin{array}{rcl} g_{S_*}(\mathbf{z}^*) &=& 0,\\ \eta\|g_{\overline{S}_*}(\mathbf{z}^*)\|_{[1]} &\le& \|\boldsymbol{\alpha}^*\|_{[s]},\\ \|\boldsymbol{\alpha}^*\|_0 &\le& s,\\ \langle \boldsymbol{\alpha}^*, \mathbf{y}\rangle &=& 0. \end{array}\right. \tag{47}$$

This clearly indicates (45) by $\|\boldsymbol{\alpha}^*\|_{[s]} = 0$ if $\|\boldsymbol{\alpha}^*\|_0 < s$. Therefore it is a local minimizer by Lemma A.1 a).

b) If $\|\boldsymbol{\alpha}^*\|_0 < s$, then Lemma A.1 a) states that a local minimizer $\boldsymbol{\alpha}^*$ satisfies the second condition in (45) which is same as (47). Therefore, it is also an η-stationary point for any $\eta > 0$. If $\|\boldsymbol{\alpha}^*\|_0 = s$, then $\|\boldsymbol{\alpha}^*\|_{[s]} > 0$, which implies

$$\eta^* := \|\boldsymbol{\alpha}^*\|_{[s]}/(2\|H(\boldsymbol{\alpha}^*)\boldsymbol{\alpha}^* - \mathbf{1}\|_{[1]}) > 0.$$

A local minimizer satisfies the first condition in (45),

$$g_{S_*}(\mathbf{z}^*) \overset{(14)}{=} (H(\boldsymbol{\alpha}^*)\boldsymbol{\alpha}^*)_{S_*} - \mathbf{1} + \mathbf{y}_{S_*}b^* = 0,$$

which gives rise to

$$|b^*| = \|\mathbf{y}_{S_*}\|_{[1]}|b^*| = \|(H(\boldsymbol{\alpha}^*)\boldsymbol{\alpha}^*)_{S_*} - \mathbf{1}\|_{[1]}$$

because of $|\mathbf{y}| = \mathbf{1}$. In addition, $0 < \eta < \eta^*$ gives rise to

$$\begin{array}{rl} & \|g_{\overline{S}_*}(\mathbf{z}^*)\|_{[1]} \\ \overset{(14)}{=} & \|(H(\boldsymbol{\alpha}^*)\boldsymbol{\alpha}^*)_{\overline{S}_*} - \mathbf{1} + \mathbf{y}_{\overline{S}_*}b^*\|_{[1]} \\ \le & \|(H(\boldsymbol{\alpha}^*)\boldsymbol{\alpha}^*)_{\overline{S}_*} - \mathbf{1}\|_{[1]} + |b^*|\|\mathbf{y}_{\overline{S}_*}\|_{[1]} \\ = & \|(H(\boldsymbol{\alpha}^*)\boldsymbol{\alpha}^*)_{\overline{S}_*} - \mathbf{1}\|_{[1]} + |b^*| \\ = & \|(H(\boldsymbol{\alpha}^*)\boldsymbol{\alpha}^*)_{\overline{S}_*} - \mathbf{1}\|_{[1]} + \|(H(\boldsymbol{\alpha}^*)\boldsymbol{\alpha}^*)_{S_*} - \mathbf{1}\|_{[1]} \\ \le & \|H(\boldsymbol{\alpha}^*)\boldsymbol{\alpha}^* - \mathbf{1}\|_{[1]}2 \\ = & \|\boldsymbol{\alpha}^*\|_{[s]}/\eta^* \\ \le & \|\boldsymbol{\alpha}^*\|_{[s]}/\eta. \end{array}$$

This verifies the second inequality in (47), together with (45) claiming the conclusion.

c) An η-stationary point $\boldsymbol{\alpha}^*$ with $\|\boldsymbol{\alpha}^*\|_0 < s$ satisfies the condition (47), which is same as the second case $\|\boldsymbol{\alpha}^*\|_0 < s$ in (45). Namely, $\boldsymbol{\alpha}^*$ is also a local minimizer, which makes the conclusion immediately from Lemma A.1 b).

d) An η-stationary point $\boldsymbol{\alpha}^*$ with $\|\boldsymbol{\alpha}^*\|_0 = s$ satisfies

$$\boldsymbol{\alpha}^* \overset{(21)}{\in} \mathbb{P}_s(\boldsymbol{\alpha}^* - \eta g(\mathbf{z}^*)) \overset{(17)}{=} \underset{\|\boldsymbol{\alpha}\|_0 \le s}{\operatorname{argmin}} \|\boldsymbol{\alpha} - (\boldsymbol{\alpha}^* - \eta g(\mathbf{z}^*))\|,$$

which means for any feasible point, namely, $\|\boldsymbol{\alpha}\|_0 \le s$ and $\langle \boldsymbol{\alpha}, \mathbf{y}\rangle = 0$, we have

$$\|\boldsymbol{\alpha}^* - (\boldsymbol{\alpha}^* - \eta g(\mathbf{z}^*))\|^2 \le \|\boldsymbol{\alpha} - (\boldsymbol{\alpha}^* - \eta g(\mathbf{z}^*))\|^2.$$

This suffices to

$$-\|\boldsymbol{\alpha}^* - \boldsymbol{\alpha}\|^2 \le 2\eta\langle \boldsymbol{\alpha} - \boldsymbol{\alpha}^*, g(\mathbf{z}^*)\rangle. \tag{48}$$

The strong and quadratic convexity of $D(\cdot)$ gives rise to

$$\begin{array}{rl} & 2D(\boldsymbol{\alpha}) - 2D(\boldsymbol{\alpha}^*) \\ \overset{(46)}{\ge} & \langle \boldsymbol{\alpha} - \boldsymbol{\alpha}^*, P(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)\rangle + 2\langle \nabla D(\boldsymbol{\alpha}^*), \boldsymbol{\alpha} - \boldsymbol{\alpha}^*\rangle \\ \overset{(14)}{=} & \langle \boldsymbol{\alpha} - \boldsymbol{\alpha}^*, P(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)\rangle + 2\langle g(\mathbf{z}^*) - \mathbf{y}b^*, \boldsymbol{\alpha} - \boldsymbol{\alpha}^*\rangle \\ \overset{(47)}{=} & \langle \boldsymbol{\alpha} - \boldsymbol{\alpha}^*, P(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)\rangle + 2\langle g(\mathbf{z}^*), \boldsymbol{\alpha} - \boldsymbol{\alpha}^*\rangle \\ \overset{(48)}{\ge} & \langle \boldsymbol{\alpha} - \boldsymbol{\alpha}^*, P(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)\rangle - \|\boldsymbol{\alpha}^* - \boldsymbol{\alpha}\|^2/\eta \\ = & \langle \boldsymbol{\alpha} - \boldsymbol{\alpha}^*, (P - I/\eta)(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)\rangle \\ \ge & 0, \end{array}$$

where the last inequity follows from

$$P - I/\eta = [1/C - 1/\eta]\, I + Q^\top Q \succeq 0$$

by $\eta \ge C$. Therefore, $\boldsymbol{\alpha}^*$ is a global minimizer. If there is another global minimizer $\hat{\boldsymbol{\alpha}} \ne \boldsymbol{\alpha}^*$, then the strictness $\succ$ in the above condition by $\eta > C$ leads to a contradiction,

$$\begin{array}{rcl} 0 &=& D(\hat{\boldsymbol{\alpha}}) - D(\boldsymbol{\alpha}^*) \\ &\ge& \langle \hat{\boldsymbol{\alpha}} - \boldsymbol{\alpha}^*, (P - I/\eta)(\hat{\boldsymbol{\alpha}} - \boldsymbol{\alpha}^*)\rangle \\ &>& 0. \end{array}$$

Hence, $\boldsymbol{\alpha}^*$ is unique. The proof is completed. □

A Lemma for Theorem 3.1

Before proving Theorem 3.1, we first present some properties regarding an η-stationary point of (8).

Lemma A.2. Let $\mathbf{z}^*$ be an η-stationary point of (8) with $0 < \eta < \eta^*$, where η^* is given by (30). Then there always exists a $\delta^* > 0$ such that for any $\mathbf{z} \in N(\mathbf{z}^*, \delta^*)$ with $\|\boldsymbol{\alpha}\|_0 \le s$, the following results hold.

a) If $\|\boldsymbol{\alpha}^*\|_0 = s$, then

$$\mathbb{T}_s(\boldsymbol{\alpha} - \eta g(\mathbf{z}) = \mathbb{T}_s(\boldsymbol{\alpha}^* - \eta g(\mathbf{z}^*)) = \{S_*\}. \tag{49}$$

If $\|\boldsymbol{\alpha}^*\|_0 < s$, then for any $T \in \mathbb{T}_s(\boldsymbol{\alpha} - \eta g(\mathbf{z}))$ and any $T_* \in \mathbb{T}_s(\boldsymbol{\alpha}^* - \eta g(\mathbf{z}^*))$, it holds

$$S_* \subseteq (T_* \cap T). \tag{50}$$

b) For any $T \in \mathbb{T}_s(\boldsymbol{\alpha} - \eta g(\mathbf{z}))$, it holds

$$F(\mathbf{z}^*; T) = 0. \tag{51}$$

proof a) It follows from Theorem 2.3 and $\mathbf{z}^*$ being an η-stationary point of (8) that $F(\mathbf{z}^*; T_*) = 0$ for any $T_* \in \mathbb{T}_s(\boldsymbol{\alpha}^* - \eta g(\mathbf{z}^*))$. We first derive

$$E(\boldsymbol{\alpha}^*)\boldsymbol{\alpha}^* = E(\boldsymbol{\alpha})\boldsymbol{\alpha}^*. \tag{52}$$

If $\boldsymbol{\alpha}^* = 0$, it is true clearly. If $\boldsymbol{\alpha}^* \neq 0$, we have

$$\alpha_i^* > 0 \Longrightarrow \alpha_i > 0, \quad \alpha_i^* < 0 \Longrightarrow \alpha_i < 0. \tag{53}$$

If (53) is not true, then there is a j that violates one of the above relations, namely α_j^* and α_j have different signs. As a consequence,

$$|\alpha_j^*| < |\alpha_j^* - \alpha_j| \le \|\boldsymbol{\alpha}^* - \boldsymbol{\alpha}\| < \delta^*.$$

This is a contradiction for a sufficiently small positive δ^*. Therefore, we must have (53). Recall that $E(\boldsymbol{\alpha})$ is a diagonal matrix with diagonal elements given by (10). It follows

$$\begin{aligned} & [(E(\boldsymbol{\alpha}^*) - E(\boldsymbol{\alpha}))\boldsymbol{\alpha}^*]_i \\ = \; & [E_{ii}(\boldsymbol{\alpha}^*) - E_{ii}(\boldsymbol{\alpha})]\alpha_i^* \\ \overset{(53),(10)}{=} & \begin{cases} (1/C - 1/C)\alpha_i^*, & \alpha_i^* > 0 \\ (1/c - 1/c)\alpha_i^*, & \alpha_i^* < 0 \\ (E_{ii}(\boldsymbol{\alpha}^*) - E_{ii}(\boldsymbol{\alpha}))0, & \alpha_i^* = 0 \end{cases} \\ = \; & 0. \end{aligned}$$

Therefore, (52) is true, allowing us to derive that

$$\begin{aligned} & \|H(\boldsymbol{\alpha}^*)\boldsymbol{\alpha}^* - H(\boldsymbol{\alpha})\boldsymbol{\alpha}\| \\ \overset{(11)}{=} \; & \|(E(\boldsymbol{\alpha}^*) + Q^\top Q)\boldsymbol{\alpha}^* - (E(\boldsymbol{\alpha}) + Q^\top Q)\boldsymbol{\alpha}\| \\ \le \; & \|E(\boldsymbol{\alpha}^*)\boldsymbol{\alpha}^* - E(\boldsymbol{\alpha})\boldsymbol{\alpha}\| + \|Q\|^2\|\boldsymbol{\alpha}^* - \boldsymbol{\alpha}\| \\ \overset{(52)}{=} \; & \|E(\boldsymbol{\alpha})(\boldsymbol{\alpha}^* - \boldsymbol{\alpha})\| + \|Q\|^2\|\boldsymbol{\alpha}^* - \boldsymbol{\alpha}\| \\ \overset{(10)}{\le} \; & (1/c + \|Q\|^2)\|\boldsymbol{\alpha}^* - \boldsymbol{\alpha}\| \\ \le \; & (1/c + \|Q\|^2)\delta^*, \end{aligned} \tag{54}$$

where $\|Q\|$ is the spectral norm of Q. This suffices to

$$\begin{aligned} & \|g(\mathbf{z}^*) - g(\mathbf{z})\| \\ \overset{(14)}{=} \; & \|H(\boldsymbol{\alpha}^*)\boldsymbol{\alpha}^* - H(\boldsymbol{\alpha})\boldsymbol{\alpha} + (b^* - b)\mathbf{y}\| \\ \le \; & \|H(\boldsymbol{\alpha}^*)\boldsymbol{\alpha}^* - H(\boldsymbol{\alpha})\boldsymbol{\alpha}\| + |b^* - b|\|\mathbf{y}\| \\ \overset{(54)}{\le} \; & (1/c + \|Q\|^2 + \sqrt{m})\delta^*. \end{aligned} \tag{55}$$

Case i) $\|\boldsymbol{\alpha}^*\|_0 = s$. Since $|T_*| = s$ and $\boldsymbol{\alpha}^*_{\overline{T}_*} = 0$ from (22), it holds $T_* = \text{supp}(\boldsymbol{\alpha}^*) = S_*$. If $\|g_{\overline{S}_*}(\mathbf{z}^*)\|_{[1]} = 0$, then

$$\begin{aligned} \|\boldsymbol{\alpha}^*_{\overline{S}_*} - \eta g_{\overline{S}_*}(\mathbf{z}^*)\|_{[1]} & \overset{(22)}{=} 0 < \|\boldsymbol{\alpha}^*_{S_*}\|_{[s]} \\ & \overset{(22)}{=} \|\boldsymbol{\alpha}^*_{S_*} - \eta g_{S_*}(\mathbf{z}^*)\|_{[s]} \end{aligned} \tag{56}$$

If $\|g_{\overline{S}_*}(\mathbf{z}^*)\|_{[1]} \neq 0$, then

$$\begin{aligned} \|\boldsymbol{\alpha}^*_{\overline{S}_*} - \eta g_{\overline{S}_*}(\mathbf{z}^*)\|_{[1]} & \overset{(22)}{=} \eta\|g_{\overline{S}_*}(\mathbf{z}^*)\|_{[1]} \\ & < \eta^*\|g_{\overline{S}_*}(\mathbf{z}^*)\|_{[1]} \\ & \overset{(30)}{=} \|\boldsymbol{\alpha}^*_{S_*}\|_{[s]} \\ & \overset{(22)}{=} \|\boldsymbol{\alpha}^*_{S_*} - \eta g_{S_*}(\mathbf{z}^*)\|_{[s]} \end{aligned}$$

Therefore, both scenarios derive (56). This means S_* contains the s largest elements of $|\boldsymbol{\alpha}^*_{S_*} - \eta g_{S_*}(\mathbf{z}^*)|$, together with the definition of $\mathbb{T}_s$ in (18) indicating

$$\mathbb{T}_s(\boldsymbol{\alpha}^* - \eta g(\mathbf{z}^*)) = \{S_*\}.$$

Next, we show $\mathbb{T}_s(\boldsymbol{\alpha} - \eta g(\mathbf{z})) = \{S_*\}$, i.e., to show

$$\|\boldsymbol{\alpha}_{\overline{S}_*} - \eta g_{\overline{S}_*}(\mathbf{z})\|_{[1]} < \|\boldsymbol{\alpha}_{S_*} - \eta g_{S_*}(\mathbf{z})\|_{[s]} \tag{57}$$

For sufficiently small δ^*, $\boldsymbol{\alpha}$ can be close enough to $\boldsymbol{\alpha}^*$ and $g(\mathbf{z})$ can be close enough to $g(\mathbf{z}^*)$ from (55). These and (57) guarantee (57) immediately.

Case ii) $\|\boldsymbol{\alpha}^*\|_0 < s$. The fact $\boldsymbol{\alpha}^*_{\overline{T}_*} = 0$ from (22) indicates $S_* \subseteq T_*$. We next show $S_* \subseteq T$. If $\boldsymbol{\alpha}^* = 0$, then $S_* = \emptyset \subseteq T$ clearly. Therefore, we focus on $\boldsymbol{\alpha}^* \neq 0$. Since $\|\boldsymbol{\alpha}^*\|_0 < s$, $\|\boldsymbol{\alpha}^*\|_{[s]} = 0$, which together with (47) derives

$$g(\mathbf{z}^*) = 0, \tag{58}$$

implying that the indices of s largest elements of $|\boldsymbol{\alpha}^* - \eta g(\mathbf{z}^*)| = |\boldsymbol{\alpha}^*|$ contain S_*. Note that

$$\boldsymbol{\alpha} - \eta g(\mathbf{z}) = (\boldsymbol{\alpha}_{S_*} - \eta g_{S_*}(\mathbf{z}); \boldsymbol{\alpha}_{\overline{S}_*} - \eta g_{\overline{S}_*}(\mathbf{z}))$$

Again, for sufficiently small δ^*, $\boldsymbol{\alpha}$ and $g(\mathbf{z})$ can be close enough to $\boldsymbol{\alpha}^*$ and $g(\mathbf{z}^*)$ from (55). Therefore, $\boldsymbol{\alpha}_{S_*} - \eta g_{S_*}(\mathbf{z})$ and $\boldsymbol{\alpha}_{\overline{S}_*} - \eta g_{\overline{S}_*}(\mathbf{z})$ are close to $\boldsymbol{\alpha}^*_{S_*} - \eta g_{S_*}(\mathbf{z}^*) = \boldsymbol{\alpha}^*_{S_*}$ and $\boldsymbol{\alpha}^*_{\overline{S}_*} - \eta g_{\overline{S}_*}(\mathbf{z}^*) = 0$, respectively. These imply $S_* \subseteq T$.

b) To prove $F(\mathbf{z}^*; T) = 0$, we need to show

$$F(\mathbf{z}^*; T) = \begin{bmatrix} g_T(\mathbf{z}^*) \\ \boldsymbol{\alpha}^*_{\overline{T}} \\ \langle \boldsymbol{\alpha}^*_T, \mathbf{y}_T \rangle \end{bmatrix} = 0. \tag{59}$$

If $\|\boldsymbol{\alpha}^*\|_0 = s$, then $T = S_* = T_*$ by a), deriving the desired result by (22) immediately. If $\|\boldsymbol{\alpha}^*\|_0 < s$, then $g_T(\mathbf{z}^*) = 0$ by (58). Again from b), $S_* \subseteq (T \cap T_*)$ means $\overline{T} \subseteq \overline{S}_*$, indicating that $\boldsymbol{\alpha}^*_{\overline{T}} = 0$ due to $\boldsymbol{\alpha}^*_{\overline{S}_*} = 0$. Finally,

$$\begin{aligned} \langle \boldsymbol{\alpha}^*_T, \mathbf{y}_T \rangle & = \langle \boldsymbol{\alpha}^*_{S_*}, \mathbf{y}_{S_*} \rangle + \langle \boldsymbol{\alpha}^*_{T \setminus S_*}, \mathbf{y}_{T \setminus S_*} \rangle = \langle \boldsymbol{\alpha}^*_{S_*}, \mathbf{y}_{S_*} \rangle \\ & = \langle \boldsymbol{\alpha}^*_{T_*}, \mathbf{y}_{T_*} \rangle - \langle \boldsymbol{\alpha}^*_{T_* \setminus S_*}, \mathbf{y}_{T_* \setminus S_*} \rangle \overset{(22)}{=} 0. \end{aligned}$$

The proof is finished. □

Proof of Theorem 3.1

proof Consider a point $\mathbf{z}_t^k := \mathbf{z}^* + t(\mathbf{z}^k - \mathbf{z}^*)$ with $t \in [0,1]$. Since $\mathbf{z}^k \in N(\mathbf{z}^*, \delta^*)$, it also holds $\mathbf{z}_t^k \in N(\mathbf{z}^*, \delta^*)$ due to

$$\|\mathbf{z}_t^k - \mathbf{z}^*\| = t\|\mathbf{z}^k - \mathbf{z}^*\| \le \|\mathbf{z}^k - \mathbf{z}^*\| < \delta^*.$$

We first prove that

$$E(\boldsymbol{\alpha}^k) = E(\boldsymbol{\alpha}_t^k). \tag{60}$$

In fact, if $\boldsymbol{\alpha}^* = 0$, then $\boldsymbol{\alpha}_t^k = t\boldsymbol{\alpha}^k$, implying that $\boldsymbol{\alpha}_t^k$ and $\boldsymbol{\alpha}^k$ have the same signs. This together with the definition (10) of $E(\cdot)$ shows (60) immediately. If $\boldsymbol{\alpha}^* \neq 0$, then the same reasoning proving (53) also derives that

$$\begin{array}{lll}
\alpha_i^* > 0 & \Longrightarrow & \alpha_i^k > 0, \ \ (\alpha_t^k)_i = (1-t)\alpha_i^* + t\alpha_i^k > 0,\\
\alpha_i^* < 0 & \Longrightarrow & \alpha_i^k < 0, \ \ (\alpha_t^k)_i = (1-t)\alpha_i^* + t\alpha_i^k < 0,\\
\alpha_i^* = 0 & \Longrightarrow & \qquad\qquad (\alpha_t^k)_i = t\alpha_i^k.
\end{array}$$

These also mean $\boldsymbol{\alpha}_t^k$ and $\boldsymbol{\alpha}^k$ have the same signs. Namely, (60) is true and results in

$$\begin{aligned}
H(\boldsymbol{\alpha}^k) &\overset{(11)}{=} E(\boldsymbol{\alpha}^k) + Q^\top Q\\
&\overset{(60)}{=} E(\boldsymbol{\alpha}_t^k) + Q^\top Q = H(\boldsymbol{\alpha}_t^k),
\end{aligned}$$

contributing to $\nabla F(\mathbf{z}_t^k; T_k) = \nabla F(\mathbf{z}^k; T_k)$ due to

$$\begin{bmatrix} H_{T_k}(\boldsymbol{\alpha}_t^k) & 0 & \mathbf{y}_{T_k}\\ 0 & I & 0\\ \mathbf{y}_{T_k}^\top & 0 & 0 \end{bmatrix} = \begin{bmatrix} H_{T_k}(\boldsymbol{\alpha}^k) & 0 & \mathbf{y}_{T_k}\\ 0 & I & 0\\ \mathbf{y}_{T_k}^\top & 0 & 0 \end{bmatrix}.$$

for any $T_k \in \mathbb{T}_s(\boldsymbol{\alpha}^k - \eta g(\mathbf{z}^k))$. It follows from the mean value theorem that there exists a $\mathbf{z}_t^k$ satisfying

$$\begin{aligned}
F(\mathbf{z}^k; T_k) &\overset{(51)}{=} F(\mathbf{z}^k; T_k) - F(\mathbf{z}^*; T_k)\\
&= \nabla F(\mathbf{z}_t^k; T_k)(\mathbf{z}^k - \mathbf{z})\\
&= \nabla F(\mathbf{z}^k; T_k)(\mathbf{z}^k - \mathbf{z}^*).
\end{aligned}$$

This together with $\nabla F(\mathbf{z}^k; T_k)$ being always nonsingular due to (23) enables to show that

$$\begin{aligned}
\mathbf{z}^* &= \mathbf{z}^k - (\nabla F(\mathbf{z}^k; T_k))^{-1} F(\mathbf{z}^k; T_k)\\
&\overset{(25)}{=} \mathbf{z}^k + \mathbf{d}^k \overset{(27)}{=} \mathbf{z}^{k+1}.
\end{aligned}$$

Finally, for any $T_{k+1} \in \mathbb{T}_s(\boldsymbol{\alpha}^{k+1} - \eta g(\mathbf{z}^{k+1})) = \mathbb{T}_s(\boldsymbol{\alpha}^* - \eta g(\mathbf{z}^*))$, it follows from $\mathbf{z}^*$ being an η-stationary point that

$$\|F(\mathbf{z}^{k+1}, T_{k+1})\| = \|F(\mathbf{z}^*, T_{k+1})\| \overset{(22)}{=} 0.$$

The entire proof is completed. $\square$