Sparse SVM for Sufficient Data Reduction

Shenglong Zhou
shenglong.zhou@soton.ac.uk
School of Mathematics, University of Southampton, UK

Abstract

Kernel-based methods for support vector machines (SVM) have seen a great advantage in various applications. However, they may incur prohibitive computational costs when the involved sample size is on a large scale. 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, the sparsity constrained kernel SVM optimization is taken into consideration and is capable of controlling the number of support vectors. Based on the established optimality conditions associated with the stationary equations, a Newton-type method is cast to tackle the sparsity constrained optimization and turns out to enjoy the one-step convergence property if the starting point is chosen to be close to a local region of a stationary point, leading to a super-fast computational speed. Numerical comparisons with two powerful solvers demonstrate that the proposed method performs exceptionally well, especially for datasets with large numbers of samples, in terms of a much fewer number of support vectors and shorter computational time.

Keywords: data reduction, sparsity constrained kernel SVM, Newton method, one-step convergence property.

1 Introduction

Support Support vector machines (SVM), as one of the most popular classification tools, were first introduced by Vapnik and Cortes [1], with wide applications in machine learning, statistic and pattern recognition. The goal is to find a hyperplane in the input space that best separates the training dataset so that it can well predict the class of some newly input data. The paper focuses on the binary classification problem: Suppose 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. In order to find an optimal hyperplane, there are two possible cases. The training dataset is linearly separable and inseparable 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 designing the loss functions ℓ to deal 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 by taking advantage of the kernel tricks, the dual kernel based SVM optimization is usually taken into account, whose the objective function involves an $m \times m$ order matrix, the kernel matrix. The complexity of computing this kernel matrix is about $O(m^2n)$, which makes it impossible for training on million-size data since the storage of such a scale data requires considerably large memory and

the incurred computational cost is extremely expensive. Hence, to overcome this drawback, data reduction has drawn much attention since it makes use of a small portion of samples.

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

$$\mathbf{w}^* = \sum_{i=1}^{m} \alpha_i^* y_i \mathbf{x}_i, \tag{1.1}$$

where $\boldsymbol{\alpha}^*$ is a solution to the dual kernel-based SVM optimization. The training vectors $\mathbf{x}_i$ corresponding to non-zero 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. Because of this, computations and storage for large scale size data are possible since we only focus on the support vectors. However, solutions to the dual kernel-based SVM optimization are not sparse enough in general. In order to ensure a solution being sufficiently sparse, an impressive body of approaches have been developed. Those methods aiming at reducing the number of support vectors can be categorized into the sparse SVM group. We will explore more in the sequel.

1.1 Selective Literature Review

There is a vast body of work on designing a proper loss function ℓ to cast an efficient soft-margin SVM model, which can be summarized into two categories based on the convexity of ℓ. Convex soft margin loss functions include the famous hinge loss [1], the pinball loss [3, 2], the hybrid Huber loss [4, 5, 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 meanwhile induces the unboundedness, which reduces the robustness of those functions to outliers from the training data. In order to overcome such a drawback, authors in [11],[12] set an upper bound and enforce the loss functions to stop increasing after a certain point. By doing so, the convex loss functions turn to be non-convex. Other non-convex ones consist of the ramp loss [13], the truncated pinball loss [14], the asymmetrical truncated pinball loss [15], the sigmoid loss [16], the normalized sigmoid cost loss [17] and to name a few. Compared with the convex margin loss functions, most non-convex ones are less sensitive to outliers due to the boundedness. However, non-convexity would cause difficulties in computations in practice.

A separate line of research investigates methods to do data reduction, namely reducing the number of the support vectors, which gives rise to the topic of the sparse SVM. One of the earliest attempts was in [18], where it is 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 is adopted as a key component of the method developed in [19], where the training method is to exclude the samples that incur the separation hypersurface highly convoluted so that a few number of the support vectors are enough to describe a less convoluted hypersurface for separating two classes. The method RSVM proposed in [20] randomly picks a subset of the training set, and then searches for a solution (supported only on the picked training set) to a smooth SVM optimization that minimizes the loss on the entire training set. In [21], from (1.1), they substitute $\mathbf{w}$ by the expression $\sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i$ and employ the ℓ_1-norm regularization of $\boldsymbol{\alpha}$ into the soft-margin loss model, which effectively benefits for the variable/sample selection since the ℓ_1-norm regularization is capable of rendering a sparse structure of the solution [22]. Similarly, authors in [23] also takes advantage of the expression from (1.1). They perform a greedy method, where in each step, a new training sample is carefully selected into the set of the training vectors to form a new subproblem. Since the number of the training vectors

is relatively small comparing to the whole size of the training sample, the subproblem is on small scale and thus can be addressed by Newton method quickly. In [24], a subgradient descent algorithm is proposed, in each step, only samples with the correct classification inside the margin but maximizing a gap are 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 those methods perform exceptionally well in terms of reducing the number of support vectors.

1.2 Methodology

Mathematically, the soft-margin SVM takes the form,

$$\min_{\mathbf{w}\in\mathbb{R}^n} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m} \ell\Big[1 - y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle + b)\Big], \tag{1.2}$$

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, whose dual problem is a quadratic kernel based SVM optimization:

$$\begin{aligned} \min_{\boldsymbol{\alpha}\in\mathbb{R}^m} \quad & d(\boldsymbol{\alpha}) := \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i\alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j\rangle - \sum_{i=1}^{m}\alpha_i, \\ \text{s.t.} \quad & \sum_{i=1}^{m} \alpha_i y_i = 0, \quad 0 \le \alpha_i \le C, i \in [m], \end{aligned} \tag{1.3}$$

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

$$\min_{\mathbf{w}\in\mathbb{R}^n} \quad \frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{m} \ell_{cC}\Big[1 - y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle + b)\Big], \tag{1.4}$$

with the soft-margin loss ℓ_{cC} being given by

$$\ell_{cC}(t) := \begin{cases} Ct^2/2, & t \ge 0, \\ ct^2/2, & t < 0, \end{cases} \tag{1.5}$$

where $c > 0$ is chosen to be smaller than C, namely $c < C$. The soft-margin SVM (1.4) implies that it gives penalty C for $t_i := 1 - y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle + b) \ge 0$ and c for $t_i < 0$. 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 (1.2) with $\ell = \ell_{cC}$ enjoys the following form

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^m} \; D(\boldsymbol{\alpha}) := d(\boldsymbol{\alpha}) + \sum_{i=1}^{m} h_{cC}(\alpha_i), \quad \text{s.t.} \; \sum_{i=1}^{m} \alpha_i y_i = 0, \tag{1.6}$$

where h_{cC} is defined by

$$h_{cC}(t) := \begin{cases} t^2/(2C), & t \ge 0, \\ t^2/(2c), & t < 0. \end{cases} \tag{1.7}$$

Motivated by the work in the sparse SVM, in this paper, we aim at solving the following sparsity constrained kernel based SVM optimization problem,

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^m} \; D(\boldsymbol{\alpha}), \quad \text{s.t.} \; \sum_{i=1}^{m} \alpha_i y_i = 0, \; \|\boldsymbol{\alpha}\|_0 \le s, \tag{1.8}$$

where $s \in [m]$ is a given integer satisfying $s \ll m$ and is called the sparsity level, and $\|\boldsymbol{\alpha}\|_0$ is called the zero norm of $\boldsymbol{\alpha}$, counting the number of non-zero elements of $\boldsymbol{\alpha}$. Compared with the classic kernel SVM model (1.3), the problem (1.8) at least has three advantages: A1) The objective function is strongly convex, hence the optimal solutions exist (see Theorem 2.2) and is unique under mild conditions (see Theorem 2.4). While the objective function of (1.3) is convex but not strongly convex when $n \le m$. This means 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 much more tractable. Note that when m is large, those bounded constraints in (1.3) may incur expensive computational costs. A3) Most importantly, the constraint $\|\boldsymbol{\alpha}\|_0 \le s$ manifests that at most s non-zero elements are contained in $\boldsymbol{\alpha}$, i.e., the number of the support vectors is expected to be less than s by (1.1). In this way, the number is able to be controlled and could be small.

1.3 Contributions

The contributions of this paper are summarized as follows.

C1) As we mentioned above, the sparsity constrained model (1.8) has its advantages. As far as we know, this is the first paper that employs the sparsity constraint into the kernel SVM model. Such a constraint allows us to govern the number of support vectors and hence do data reduction sufficiently, which makes the extremely large scale computation possible and significantly reduces the demand for huge volumes of hardware memory.

C2) Based on the established optimality conditions, associated with the stationary equations by Theorem 2.3, a Newton-type method is employed. The method is shown in very low computational complexity and also enjoys one-step convergence property. This convergence result is much better than the locally quadratic convergence property which is usually enjoyed by Newton-type methods. Namely, 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 succeed in designing a mechanism to tune the sparsity level s adaptively.

C3) Numerical experiments have demonstrated that the proposed method performs exceptionally well, especially for datasets with $m \gg n$, namely, the number of samples being far greater than the number of features. When comparing with two powerful existing solvers, it takes much shorter computational time due to a tiny number of support vectors being used.

1.4 Preliminaries

We present some notation to be employed throughout the paper. For the sake of easy reference, we list all relevant ones in the following table including those that have been given in the above part of this section.

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 $\|\boldsymbol{\alpha}\|$.
$\|\boldsymbol{\alpha}\|_0$	The number of non-zero elements of $\boldsymbol{\alpha}$.
$\mathrm{supp}(\boldsymbol{\alpha})$	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, $\{\mathbf{u} \in \mathbb{R}^m : \|\mathbf{u} - \boldsymbol{\alpha}\| < \delta\}$.

Let $\mathbb{P}_s$ be defined by

$$\mathbb{P}_s(\boldsymbol{\alpha}) = \mathrm{argmin}_{\mathbf{u} \in \mathbb{R}^m} \Big\{ \|\mathbf{u} - \boldsymbol{\alpha}\| : \|\mathbf{u}\|_0 \leq s \Big\}, \tag{1.9}$$

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 (1.9), we define a useful set by

$$\mathbb{T}_s(\boldsymbol{\alpha}) := \left\{ T \subseteq [m] : \ |T| = s, T \text{ contains indices of} s \text{ largest elements of } |\boldsymbol{\alpha}| \ \right\}. \tag{1.10}$$

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

$$\mathbb{P}_s(\boldsymbol{\alpha}) := \left\{ \begin{bmatrix} \boldsymbol{\alpha}_T \\ 0 \end{bmatrix} : \ T \in \mathbb{T}_s(\boldsymbol{\alpha}) \right\}. \tag{1.11}$$

Since the sth largest element of $|\boldsymbol{\alpha}|$ may not be unique, $\mathbb{T}_s(\boldsymbol{\alpha})$ might have multiple elements, so does $\mathbb{P}_s(\boldsymbol{\alpha})$. For example, $\overline{\boldsymbol{\alpha}} = (1, -1, 0, 0)^\top$, we have $\mathbb{T}_2(\overline{\boldsymbol{\alpha}}) = \{\{1, 2\}\}$, $\mathbb{P}_2(\overline{\boldsymbol{\alpha}}) = \{\overline{\boldsymbol{\alpha}}\}$ and $\mathbb{T}_1(\overline{\boldsymbol{\alpha}}) = \{\{1\}, \{2\}\}$, $\mathbb{P}_1(\overline{\boldsymbol{\alpha}}) = \{(1, 0, 0, 0)^\top, (0, -1, 0, 0)^\top\}$. Now we rewrite $D(\boldsymbol{\alpha})$ in (1.8) as

$$D(\boldsymbol{\alpha}) = \frac{1}{2}\|Q\boldsymbol{\alpha}\|^2 + \frac{1}{2}\langle E(\boldsymbol{\alpha})\boldsymbol{\alpha}, \boldsymbol{\alpha}\rangle - \langle \mathbf{1}, \boldsymbol{\alpha}\rangle, \tag{1.12}$$

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

$$E_{ii}(\boldsymbol{\alpha}) := (E(\boldsymbol{\alpha}))_{ii} = \begin{cases} 1/C, & \alpha_i \geq 0, \\ 1/c, & \alpha_i < 0. \end{cases} \tag{1.13}$$

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

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

One can see that the Hessian matrix is positive definite for any $\boldsymbol{\alpha} \in \mathbb{R}^m$ due to $C > c$ and

$$H(\boldsymbol{\alpha}) \succeq Q^\top Q + I/C = P \succ 0, \tag{1.14}$$

where $A \succeq 0$ ($A \succ 0$) means A is semi-definite (definite) positive. Here $A \succeq B$ stands for $A - B$ being semi-definite positive. Hereafter, for given $\boldsymbol{\alpha} \in \mathbb{R}^m$ and $b \in \mathbb{R}$, let

$$\mathbf{z} \;:=\; (\boldsymbol{\alpha}; b) = \begin{bmatrix} \boldsymbol{\alpha} \\ b \end{bmatrix} \tag{1.15}$$

for notational convenience Based on which, we denote the following functions

$$\begin{aligned} g(\mathbf{z}) \;&:=\; \nabla D(\boldsymbol{\alpha}) + \mathbf{y}b \overset{(1.14)}{=} H(\boldsymbol{\alpha})\boldsymbol{\alpha} - \mathbf{1} + \mathbf{y}b, \\ g_T(\mathbf{z}) \;&:=\; (g(\mathbf{z}))_T, \qquad H_T(\boldsymbol{\alpha}) := (H(\boldsymbol{\alpha}))_{TT}. \end{aligned} \tag{1.16}$$

So $g(\mathbf{z})$ 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 into $\mathbf{z}^*$ 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 (1.8), establishing the optimality condition associated with the η-stationary point. The condition is then equivalently transferred to an stationary equations which allows us to adopt the Newton method in Section 3, where we also prove the proposed method converges to a stationary point within one step. What is more, 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 two powerful solvers are provided. All proofs are presented in the appendix.

2 Optimality

Before we establish the optimality conditions of the sparsity constrained kernel SVM optimization (1.8), we claim that the problem (1.6) is indeed the dual problem of (1.4).

Theorem 2.1 *The problem (1.6) is the dual problem of (1.4) and admits a unique optimal solution, say $\boldsymbol{\alpha}^*$. Furthermore, the optimal solution of the primal problem (1.4) is*

$$\widehat{\mathbf{w}} = Q\boldsymbol{\alpha}^*, \qquad \widehat{b} = \frac{\langle \mathbf{y}_{S_*}, \mathbf{1} - H_{S_*}(\boldsymbol{\alpha}^*)\boldsymbol{\alpha}^*_{S_*} \rangle}{|S_*|}, \tag{2.1}$$

where S_* is the support set of $\boldsymbol{\alpha}^*$, namely,

$$S_* \;:=\; \mathrm{supp}(\boldsymbol{\alpha}^*). \tag{2.2}$$

2.1 η-Stationary Point

In this part, we focus on the sparsity constrained kernel SVM problem (1.8). First of all, we conclude that it admits a global solution/minimizer.

Theorem 2.2 *The global minimizers of* (1.8) *exist.*

To find a solution to the problem, we introduce the concept of an η-stationary point.

Definition 2.1 *We say $\boldsymbol{\alpha}^*$ is an η-stationary point of the problem* (1.8) *with $\eta > 0$ if there is a scalar $b^* \in \mathbb{R}$ such that*

$$\left\{ \begin{array}{rcl} \boldsymbol{\alpha}^* & \in & \mathbb{P}_s\Big[\boldsymbol{\alpha}^* - \eta g(\mathbf{z}^*)\Big], \\ 0 & = & \langle \boldsymbol{\alpha}^*, \mathbf{y} \rangle. \end{array} \right. \tag{2.3}$$

From our notation (1.15) that $\mathbf{z}^* = (\boldsymbol{\alpha}^*; b^*)$. We also say $\mathbf{z}^*$ is an η-stationary point of (1.8) if it satisfies the above conditions. We characterize an η-stationary point of (1.8) as an equation system through the following theorem.

Theorem 2.3 *A point $\mathbf{z}^*$ is an η-stationary point of* (1.8) *with $\eta > 0$ if and only if $\exists\ T_* \in \mathbb{T}_s(\boldsymbol{\alpha}^* - \eta g(\mathbf{z}^*))$ such that*

$$F(\mathbf{z}^*; T_*) \ := \ \left[\begin{array}{c} g_{T_*}(\mathbf{z}^*) \\ \boldsymbol{\alpha}^*_{\overline{T}_*} \\ \langle \boldsymbol{\alpha}^*_{T_*}, \mathbf{y}_{T_*} \rangle \end{array} \right] = 0. \tag{2.4}$$

We call (2.4) the stationary equations. Comparing with those conditions in (2.3), equations (2.4) allow us to employ the Newton method. Moreover, it is worth mentioning that if a point $\mathbf{z}$ is an η-stationary point of (1.8), then for any fixed $T \in \mathbb{T}_s(\boldsymbol{\alpha} - \eta g(\mathbf{z}))$, since $g_T(\mathbf{z}) = H_T(\boldsymbol{\alpha})\boldsymbol{\alpha}_T - \mathbf{1} + \mathbf{y}_T b$ by (1.16), the Jacobian matrix of $F(\mathbf{z}; T)$ can be derived by,

$$\nabla F(\mathbf{z}; T) = \left[\begin{array}{ccc} H_T(\boldsymbol{\alpha}) & 0 & \mathbf{y}_T \\ 0 & I & 0 \\ \mathbf{y}_T^\top & 0 & 0 \end{array} \right], \tag{2.5}$$

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

Remark 2.1 *What is the role of b^*? The second condition in* (2.4) *indicates $S_* \subseteq T_*$. Then the first equation in* (2.4) *and* (1.16) *yields that*

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

which together with (2.1) *means $b^* = \widehat{b}$. Namely, b^* is the bias. Therefore, seeking an η-stationary point can acquire the solution to the dual problem* (1.8) *and the bias b^* to the primal problem* (1.4) *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 the latter is more tractable in numerical computing.

Theorem 2.4 *Consider a feasible point $\boldsymbol{\alpha}^*$ to the problem* (1.8)*, namely, $\|\boldsymbol{\alpha}^*\|_0 \le s$ and $\langle \boldsymbol{\alpha}^*, \mathbf{y} \rangle = 0$.*

η-stationary points and local minimizers:

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* $\boldsymbol{\alpha}^*$*.*

η-stationary points and global minimizers:

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* $\eta > C$*.*

3 Newton Method

This section applies the Newton method into solving the equation (2.4). Let $\mathbf{z}^k$ be defined in (1.15) and the current approximation to a solution of (2.4). Choose one index set

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

Then Newton's method for (2.4) takes the following form to get the direction $\mathbf{d}^k \in \mathbb{R}^{m+1}$:

$$\nabla F(\mathbf{z}^k; T_k)\mathbf{d}^k = -F(\mathbf{z}^k; T_k). \tag{3.1}$$

Substituting (2.4) and (2.5) into (3.1) derives

$$\begin{bmatrix} H_{T_k}(\boldsymbol{\alpha}^k) & 0 & \mathbf{y}_{T_k} \\ 0 & I & 0 \\ \mathbf{y}_{T_k}^\top & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{d}^k_{T_k} \\ \mathbf{d}^k_{\overline{T}_k} \\ d^k_{m+1} \end{bmatrix} = - \begin{bmatrix} g_{T_k}(\mathbf{z}^k) \\ \boldsymbol{\alpha}^k_{\overline{T}_k} \\ \langle \boldsymbol{\alpha}^k_{T_k}, \mathbf{y}_{T_k} \rangle \end{bmatrix}. \tag{3.2}$$

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

$$\mathbf{z}^{k+1} = \begin{bmatrix} \boldsymbol{\alpha}^k_{T_k} \\ \boldsymbol{\alpha}^k_{\overline{T}_k} \\ b^k \end{bmatrix} + \begin{bmatrix} \mathbf{d}^k_{T_k} \\ \mathbf{d}^k_{\overline{T}_k} \\ d^k_{m+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\alpha}^k_{T_k} + \mathbf{d}^k_{T_k} \\ 0 \\ b^k + d^k_{m+1} \end{bmatrix}. \tag{3.3}$$

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 (3.2).
  Update $\mathbf{z}^{k+1}$ by (3.3).
  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 (3.2) that $\mathbf{d}^k_{\overline{T}_k} = -\boldsymbol{\alpha}^k_{\overline{T}_k}$ is derived directly. So every new point is always sparse due to

$$\|\boldsymbol{\alpha}^{k+1}\|_0 \overset{(3.3)}{=} \|\boldsymbol{\alpha}^k_{T_k} + \mathbf{d}^k_{T_k}\|_0 \leq |T_k| = s.$$

Moreover, the major computation in (3.2) is from the part on T_k. However, $|T_k| = s$ can be controlled to have a very small scale comparing to $m+1$, which leads to a considerably low computational complexity.

3.1 Complexity Analysis

To derive the Newton direction in (3.2), we need to address the following equations

$$\begin{aligned} d^k_{m+1} &= -\frac{\left\langle \mathbf{y}_{T_k},\ \Theta^{-1} g_{T_k}(\mathbf{z}^k) - \boldsymbol{\alpha}^k_{T_k} \right\rangle}{\langle \mathbf{y}_{T_k},\ \Theta^{-1}\mathbf{y}^k_{T_k}\rangle}, \\ \mathbf{d}^k_{T_k} &= -\Theta^{-1}\left[g_{T_k}(\mathbf{z}^k) + d^k_{m+1}\mathbf{y}_{T_k}\right], \\ \mathbf{d}^k_{\overline{T}_k} &= -\boldsymbol{\alpha}^k_{\overline{T}_k}, \end{aligned} \tag{3.4}$$

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

$$O\left(mn + \min\{n,s\}s^2\right). \tag{3.5}$$

This can be derived by the following analysis:

- Recall the definition (1.14) 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 $|T_k| = s$. And computing its inverse takes complexity at most $O(s^3)$. Therefore, the complexity of deriving Θ^{-1} is $O(\min\{n,s\}s^2)$.

- 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 about $O(mn)$ since one need to calculate $Q^\top(Q_{T_k}\boldsymbol{\alpha}^{k+1}_{T_k})$ and the latter is $O(m + s\ln s)$. Here, we benefit from a MATLAB built-in function `mink` to select the s largest elements.

From (3.5), if s is chosen to be considerably small, (e.g., $s = 0.01m$), then the complexity can be significantly reduced compared to the one of solving the kernel based SVM without applying data reduction strategies. Hence, it makes the extremely large scale computation possible.

3.2 One Step Convergence

For a point $\mathbf{z}^* = (\boldsymbol{\alpha}^*; b^*)$ with $\boldsymbol{\alpha}^*$ feasible to (1.8), define

$$\eta^* := \begin{cases} \|\boldsymbol{\alpha}^*\|_{[s]}\|g(\mathbf{z}^*)\|^{-1}_{[1]}, & \|\boldsymbol{\alpha}^*\|_0 = s, \\ +\infty, & \|\boldsymbol{\alpha}^*\|_0 < s, \end{cases} \tag{3.6}$$

The convergence result is stated by the following theorem.

Theorem 3.1 *Let $\mathbf{z}^*$ be an η-stationary point of the problem (1.8) with $0 < \eta < \eta^*$, where η^* is given by (3.6). Let $\{\mathbf{z}^k\}$ be the sequence generated by Algorithm 1. There always is a $\delta^* > 0$ such that if at one step $\mathbf{z}^k \in N(\mathbf{z}^*, \delta^*)$, then*

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

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

One can discern that Algorithm 1 will terminate at the next step if the current point falls into a local area of an η-stationary point. This means if the starting point by chance is chosen within the local area, then the proposed algorithm will take one step to terminate. Hence, it enjoys a very fast convergence property. Such a convergence results is much better than the locally quadratic convergence property.

3.3 The Sparsity Level Tuning

One major issue we encounter is that the sparsity level s in (1.8) usually is unknown beforehand. And it plays two important roles: (i) The larger s, the better classifications since more samples are taken into consideration. (ii) However, the smaller s, the faster computational speed of Algorithm 1 as the complexity in (3.5) relies on s. Moreover, from (1.1) that the number of support vectors is smaller than $\|\boldsymbol{\alpha}\|_0 \leq s$. Because of this, the smaller s, the smaller the number of support vectors. Therefore, to balance these two aspects, we design the following rule to update the unknown s. Start with a small integer and then increase it until to satisfy the following halting conditions

$$\begin{aligned} \|F(\mathbf{z}^k; T_k)\| &< \epsilon, \\ \left|\mathtt{ACC}(\boldsymbol{\alpha}^k) - \max_{j\in[k-1]} \mathtt{ACC}(\boldsymbol{\alpha}^j)\right| &< 10^{-4}, \end{aligned} \tag{3.7}$$

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

$$\mathtt{ACC}(\boldsymbol{\alpha}) := \left[1 - \frac{1}{m}\|\mathrm{sgn}(X\boldsymbol{\alpha} + b) - \mathbf{y}\|_0\right] \times 100\%, \tag{3.8}$$

with $\mathrm{sgn}(t) = 1$ if $t > 0$ and -1 otherwise. The halting conditions (3.7) mean that $\boldsymbol{\alpha}^k$ is almost an η-stationary point due to the small $\|F(\mathbf{z}^k; T_k)\|$, while the classification accuracy $\mathtt{ACC}(\boldsymbol{\alpha}^k)$ does not increase significantly. Therefore, it is irrational to keep s rising to achieve a better solution since the bigger s would lead to more computational costs. So it is highly suggested to terminate the algorithm if $\boldsymbol{\alpha}^k$ satisfies such halt conditions. Our numerical experiments demonstrate that Algorithm 1 under this scheme works very well. Overall, we derive Algorithm 2: `NSSVM` (Newton method for sparse SVM with adaptively tuning the sparsity level s).

4 Numerical Experiments

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

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

Algorithm 2 `NSSVM`: Newton method for sparse SVM with adaptively tuning s.

Give parameters $C > c, \eta, \epsilon, K > 0, r > 1, s \in [m]$.
Initialize $\mathbf{z}^0$, pick $T_0 \in \mathbb{T}_s(\boldsymbol{\alpha}^0 - \eta g(\mathbf{z}^0))$ and set $k := 0$.
while $\mathbf{z}^k$ violates (3.7) and $k \leq$`MaxIt` **do**
  Update $\mathbf{d}^k$ by solving (3.2).
  Update $\mathbf{z}^{k+1}$ by (3.3).
  Update $s_{k+1} = \begin{cases} rs_k & \text{if } k \text{ is a multiple of 10,} \\ s_k & \text{otherwise.} \end{cases}$
  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 $\mathbf{z}^k$.

4.1 Implementation

The starting point $\mathbf{z}^0$ is initialized as $\boldsymbol{\alpha}^0 = 0$ and $b^0 = \operatorname{sgn}(\langle \mathbf{y}, \mathbf{1} \rangle)$. Parameters are tuned as follows. The maximum number of iteration and the tolerance are set as $K = 1000$, $\epsilon = \max\{\sqrt{m}, \sqrt{n}\}10^{-6}$. Empirically numerical experience has demonstrated that the involved parameters such as C, c or η are suggested to be selected through the cross validation for better results. However, for simplicity, we fix them as $C = 1, c = 0.01, \eta = 1/m$ and $r = 1.15$. The last parameter s_0 is chosen as $s_0 = \lceil \beta \log_{10}(m) \rceil$, where $\lceil a \rceil$ presents the smallest integer that is no less than a, and β is chosen based on the solving problems, e.g., for real datasets

$$\beta = \begin{cases} 1 + 10^{-3}n, & \text{if} \quad m/n < 100, \\ 10^{-2}n, & \text{if} \quad 100 \leq m/n < 60000, \\ 50n, & \text{if} \quad m/n \geq 60000. \end{cases}$$

4.2 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, 31]) *In this example, 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 two classes having equal numbers and then evenly split them into a training and a testing set. Finally, we randomly flip rm labels in the training data, namely, rm samples are treated as outliers.*

Example 4.2 (Real data in higher dimensions) *We select 21 datasets with $m \gg n$ from the libraries: libsvm[†], uci[‡] and kiggle[§]. All datasets are feature-wisely scaled to $[-1, 1]$ and all the classes being not 1 are treated as -1. Their details are presented in Table 1, where two datasets have the testing data. For each of those without the testing data, we split it into two*

[†] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/
[‡] http://archive.ics.uci.edu/ml/datasets.php
[§] https://www.kaggle.com/datasets

parts. The first part containing 90% of samples is treated as the training data and the rest is the testing data.

Table 1: Descriptions of real datasets.

Data	Datasets	Source		Training	Testing
			n	m	m_t
`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`		uci	9	574914	0
`rlc2`		uci	9	574914	0
`rlc3`		uci	9	574914	0
`rlc4`		uci	9	574914	0
`rlc5`	record linkage	uci	9	574914	0
`rlc6`	comparison patterns	uci	9	574914	0
`rlc7`		uci	9	574914	0
`rlc8`		uci	9	574914	0
`rlc9`		uci	9	574914	0
`rlc10`		uci	9	574914	0
`covt`	covtype.binary	libsvm	54	581012	0
`retb`	real time bidding	kaggle	88	$1e6$	0
`susy`	susy	uci	18	$5e6$	0
`hepm`	hepmass	uci	28	$7e6$	$35e5$
`higg`	higgs	uci	28	$11e6$	0

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 (3.8) where X and $\mathbf{y}$ are the training samples and classes, the testing classification accuracy (`TACC`) by (3.8) where X and $\mathbf{y}$ are the testing samples and classes, and the number of support vectors (`NSV`).

4.3 Effect to s_0

We first to see how the initial sparsity level s_0 would affect `NSSVM`. As mentioned before, we set $s_0 = \lceil \beta \log_{10}(m) \rceil$. Hence, to see the effect to s_0, we alter $\beta \in \{20, 40, \cdots, 100\}$. Average results over 100 trails for `NSSVM` solving Example 4.1 are presented in Figure 1, where one can observe that the larger β is, the higher `ACC` and `TACC` are, which means the bigger values of β lead to the better classifications. However, `ACC` and `TACC` ascend slowly and gradually stabilize at a level along with the rising of β. Moreover, the bigger values of β also result in higher numbers of support vectors (`NSV`). Therefore, there is no necessity to set a relatively big value of β. In our following numerical experiments for Example 4.1, we fix $\beta = 80$ for simplicity.

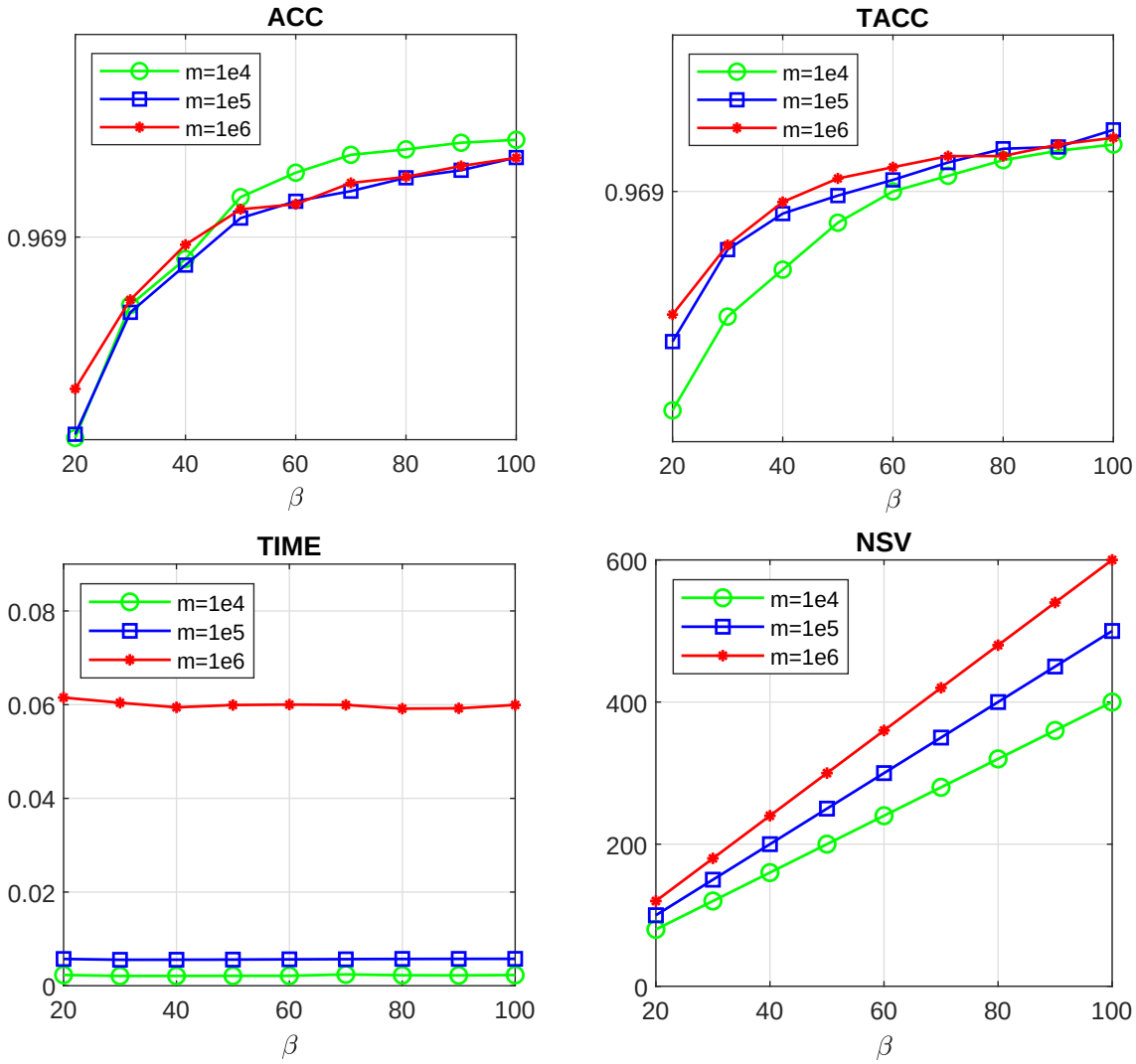

Figure 1: Effect to $s_0 = \lceil \beta \log_{10}(m) \rceil$.

4.4 Numerical Comparisons

(a) Benchmark methods. There are numerous excellent methods have been developed to tackle the SVM [7, 36, 32, 34, 23, 33, 2, 35]. Those methods perform extremely well, especially for datasets in small or mediate size. Some of them calculate the kernel matrix $Q^\top Q$ with size $m \times m$, and thus require a huge volume of hardware memory if m is large (e.g., $m \geq 10^5$). Note that most datasets in Table 1 have at least 10^5 samples. Therefore, we only select a Matlab built-in solver `FITSVM: fitclinear`[¶] and `LIBSVM: liblinear`[‖] [37] for comparisons since they are very fast to deal with those datasets. For the former, we set `Learner` = `'svm'` and `Solver` = `'dual'` in order to obtain the number of the support vectors. For the same reason, the dual problem of the ℓ_2-regularized ℓ_1-loss support vector classification is chosen to

[¶] https://mathworks.com/help/stats/fitclinear.html
[‖] https://www.csie.ntu.edu.tw/~cjlin/liblinear/

compute in LIBSVM. To do so, we set `-s 3` in the function `train` from LIBSVM. It may be much faster if we set `-s 2`, but such a setting suits for computing the primal model and does not render the number of support vectors.

(b) Comparisons for Example 4.1. We first apply three solvers to solve Example 4.1 with small scales $m = 100, 200, 300$ and $r = 0$ and depict there classifiers in Figure 2, where the Bayes classifier is $w_2 = 0.25w_1$, see the black dotted line. Basically, all solvers classify the dataset well, and clearly, NSSVM gets the best classification accuracy.

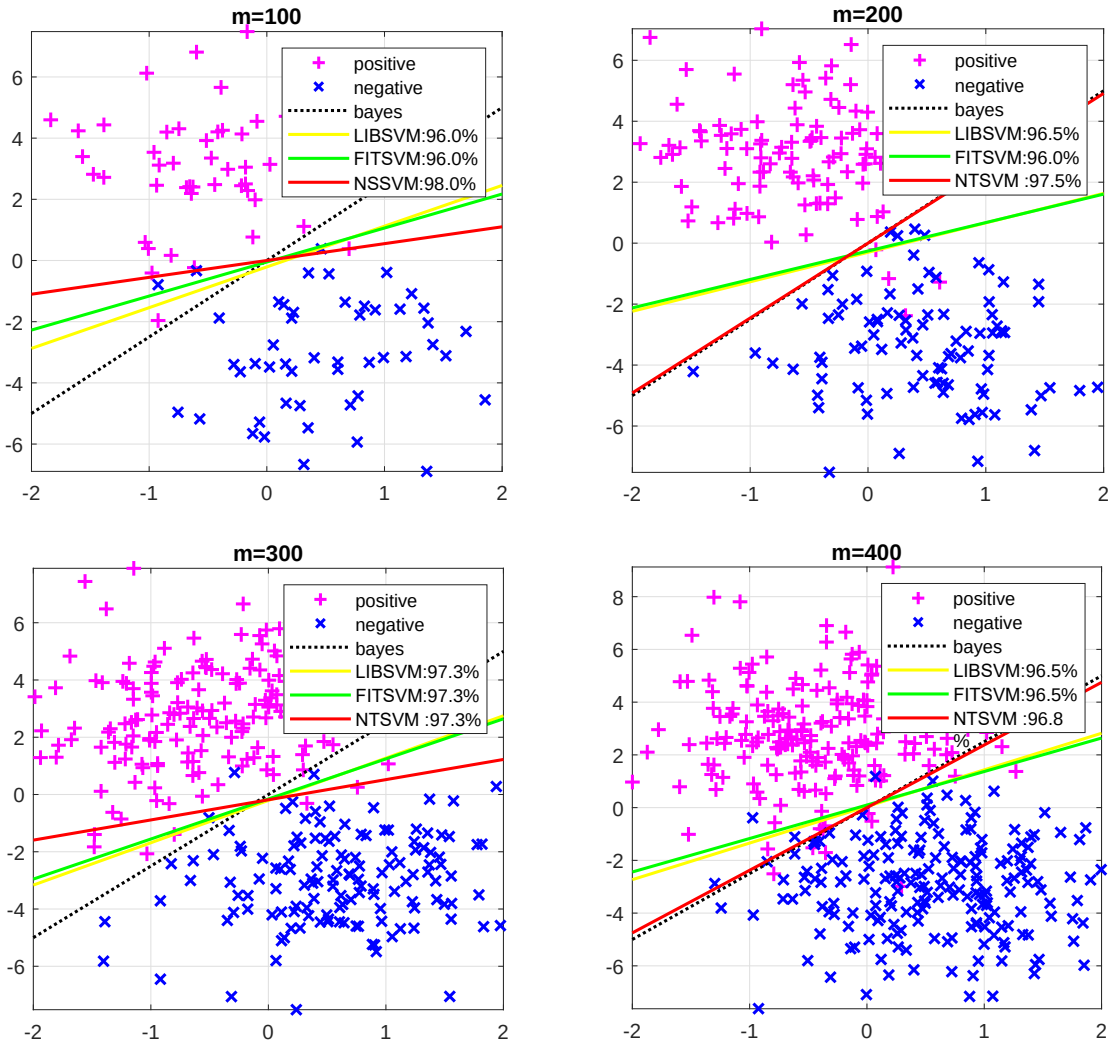

Figure 2: Classifiers by three solvers for Example 4.1.

We then perform three solvers to solve the example with altering the flapping ratio $r \in \{0.02, 0.04, \cdots, 0.2\}$ to see their effects to outliers. Average results over 100 trials are reported in Figure 3, where $m = 10^4$. It can be clearly seen that LIBSVM and NSSVM deliver similar ACC and TACC, and both are higher than those generated by FITSVM. The number of support vectors does not vary with r rising for NSSVM, while is increasing with r rising for the other two methods. Apparently, NSSVM uses the smallest number of support vectors and runs the fastest.

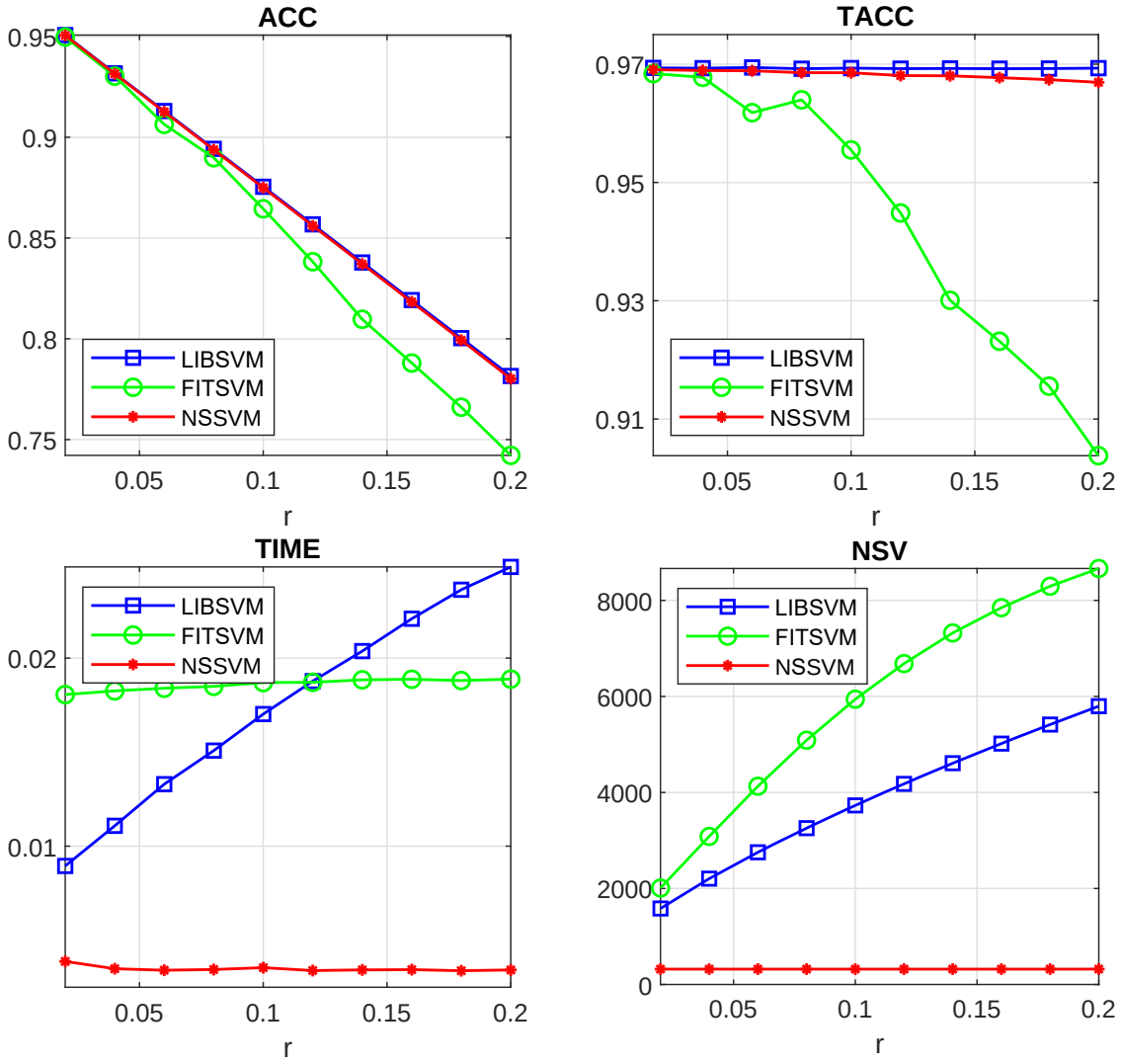

Figure 3: Effect to the outliers for Example 4.1.

Table 2: Results of three solvers for Example 4.2.

m	LIBSVM	FITSVM	NSSVM	LIBSVM	FITSVM	NSSVM
	ACC (%)			TACC (%)		
10^4	87.48	87.35	87.51	96.82	96.63	96.79
10^5	87.54	87.29	87.54	96.95	96.61	96.94
10^6	87.56	87.24	87.53	96.95	96.54	96.91
10^7	−−	86.80	87.51	−−	96.01	96.90
10^8	−−	87.15	87.52	−−	96.43	96.90
	TIME (in seconds)			NSV		
10^4	0.018	0.064	0.026	3773	5950	400
10^5	0.233	0.114	0.010	37054	59563	500
10^6	24.88	2.972	0.114	368473	595891	600
10^7	−−	37.35	0.637	−−	5968891	700
10^8	−−	542.5	11.89	−−	59679125	800

When it comes to the large scales of samples from 10^3 to 10^8, the picture is significantly different. We fix $r = 0.1$ and report the average results over 20 trials in Table 2, where `LIBSVM` runs too long time when $m > 10^6$ and hence its results are omitted. Generally speaking, the training classification accuracy from `NSSVM` and `LIBSVM` are similar, being better than those by `FITSVM`. However, it can be clearly seen that `NSSVM` runs the fastest and uses the much fewer numbers of support vectors. Moreover, the bigger m is, the more evident advantage `NSSVM` has.

(c) Comparisons for Example 4.2. Results of three solvers are reported in Table 3. Again `NSSVM` runs the fastest and generates the smallest numbers of support vectors. Taking the dataset `higg` as an instance, the other two solvers respectively take 2938 and 46.86 seconds to classify the data, by contrast, our proposed method only needs 2.551 seconds, much shorter than the one from `LIBSVM`. Moreover, the number 274229 of support vectors is less than 10% percent of those (2899291 and 8922973) from the other two solvers. For the classification accuracy (training or testing accuracies), there is no big difference among the three solvers.

5 Appendix. Proofs of all theorems

5.1 Proof of Theorem 2.1

By introducing $u_i = 1 - y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b)$, the problem (1.4) is rewritten as

$$\begin{aligned} \min_{\mathbf{w},\mathbf{u}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{m} \ell_{cC}\left(u_i\right), \\ \text{s.t.} \quad & u_i + y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) = 1,\ i \in [m]. \end{aligned} \tag{5.1}$$

To derive the conclusion, we consider three sub-problems:

$$-\frac{1}{2}\|Q\boldsymbol{\alpha}\|^2 = \min_{\mathbf{w}} \ \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{m} \alpha_i y_i \langle \mathbf{w}, \mathbf{x}_i \rangle, \tag{5.2}$$

where the optimal solution is attained at

$$\mathbf{w} = Q\boldsymbol{\alpha}. \tag{5.3}$$

The second sub-problem is

$$-\sum_{i=1}^{m} h_{cC}(\alpha_i) = \min_{\mathbf{u}} \ \sum_{i=1}^{m} \Big\{ \ell_{cC}(u_i) - \alpha_i u_i \Big\}. \tag{5.4}$$

The optimal solution is attained at the optimality condition

$$0 = \ell'_{c,C}(u_i) - \alpha_i \iff \alpha_i = \begin{cases} Cu_i, & u_i \geq 0, \\ cu_i, & u_i < 0, \end{cases} \tag{5.5}$$

which suffices to

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

Table 3: Results of three solvers LIBSVM:LIBSVM, FITSVM:FITSVM and NSSVM for Example 4.2.

	ACC (%)			TACC (%)			TIME (in seconds)			NSV		
Data	LIBSVM	FITSVM	NSSVM	LIBSVM	FITSVM	NSSVM	LIBSVM	FITSVM	NSSVM	LIBSVM	FITSVM	NSSVM
mrpe	**95.01**	94.78	93.07	**95.25**	94.90	92.57	31.751	2.2828	**1.4842**	42745	29120	**9353**
dhrb	82.70	82.71	**82.96**	83.43	83.43	**83.81**	0.1668	0.0966	**0.0220**	45760	41217	**137**
aips	**87.66**	87.26	85.63	**87.41**	87.08	85.00	0.8731	0.2143	**0.0662**	34746	61160	**243**
sctp	89.95	78.01	**90.55**	90.00	77.80	**90.62**	13.483	1.0879	**0.7188**	76482	115807	**21022**
skin	92.90	92.73	**93.79**	92.66	92.45	**93.56**	0.4214	0.3342	**0.1089**	48372	62609	**2405**
ccfd	99.94	99.94	99.89	99.92	99.92	99.86	2.6611	0.5232	**0.0910**	1583	2884	**425**
rlc1	100.0	100.0	99.98	100.0	100.0	99.97	0.3928	0.6560	**0.0617**	191	342	**47**
rlc2	100.0	100.0	99.98	100.0	100.0	99.98	0.3865	0.5099	**0.0601**	142	295	**47**
rlc3	100.0	100.0	99.99	100.0	100.0	99.99	0.4902	0.5595	**0.0752**	164	337	**47**
rlc4	100.0	100.0	99.99	100.0	100.0	99.99	0.3764	0.6736	**0.0621**	154	333	**47**
rlc5	100.0	100.0	99.98	100.0	100.0	99.97	0.4006	0.5701	**0.0909**	166	331	**47**
rlc6	100.0	100.0	99.99	100.0	100.0	99.99	0.4961	0.5055	**0.0758**	140	317	**47**
rlc7	100.0	100.0	100.0	100.0	100.0	99.99	0.3795	0.5163	**0.0472**	163	305	**47**
rlc8	100.0	100.0	99.99	100.0	100.0	99.99	0.6826	0.6734	**0.0607**	154	332	**47**
rlc9	100.0	100.0	99.96	100.0	100.0	99.95	0.6600	0.5422	**0.0929**	175	345	**47**
rlc10	100.0	100.0	99.99	100.0	100.0	99.98	0.3880	0.5455	**0.0607**	187	340	**47**
covt	**76.29**	75.73	75.14	**76.15**	75.47	75.03	13.228	2.0480	**0.4031**	306358	382638	**1668**
retb	99.81	99.81	99.81	99.80	99.80	99.80	8.9880	4.0662	**0.9898**	130224	421118	**4611**
susy	78.44	78.55	**78.72**	78.39	78.52	**78.69**	596.31	16.685	**2.9198**	2305606	2632113	**107783**
hepm	78.36	83.31	**83.54**	78.33	83.23	**83.52**	1688.9	28.392	**1.6050**	2682838	3698522	**268328**
higg	47.01	63.84	**64.06**	46.98	63.79	**64.00**	2938.7	46.863	**2.5512**	2899291	8922973	**274229**

The third sub-problem is $\min_b \sum_{i=1}^m \alpha_i y_i b = 0$, where the optimal solution is attained at

$$\langle \boldsymbol{\alpha}, \mathbf{y} \rangle = 0. \tag{5.6}$$

These three sub-problems derive the dual problem by

$$\begin{aligned}
& \max_{\boldsymbol{\alpha}} \left\{ \min_{\mathbf{w},\mathbf{u}} \ \frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^m \ell_{cC}(u_i) - \sum_{i=1}^m \alpha_i \Big(u_i + y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - 1 \Big) \right\} \\
= & \max_{\boldsymbol{\alpha}} \left\{ \min_{\mathbf{w}} \ \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^m \alpha_i y_i \langle \mathbf{w}, \mathbf{x}_i \rangle + \sum_{i=1}^m \alpha_i + \right. \\
& \left. \min_{\mathbf{u}} \sum_{i=1}^m \Big(\ell_{cC}(u_i) - \alpha_i u_i \Big) - \min_b \sum_{i=1}^m \alpha_i y_i b \right\} \\
= & \max_{\boldsymbol{\alpha}} \left\{ -\frac{1}{2}\|Q\boldsymbol{\alpha}\|^2 - \sum_{i=1}^m h_{cC}(\alpha_i) + \sum_{i=1}^m \alpha_i : \ \ \langle \boldsymbol{\alpha}, \mathbf{y} \rangle = 0 \right\}. \qquad (5.7)
\end{aligned}$$

For the model (1.4), $\widehat{\mathbf{w}}$ is obtained by (5.3). As for $\widehat{b}$, it follows

$$\begin{aligned}
y_i \widehat{b} &\overset{(5.1)}{=} 1 - u_i - \langle \widehat{\mathbf{w}}, y_i \mathbf{x}_i \rangle \\
&\overset{(5.3)}{=} 1 - u_i - \langle Q\boldsymbol{\alpha}^*, y_i \mathbf{x}_i \rangle \\
&\overset{(5.5),(1.13)}{=} 1 - E_{ii}(\boldsymbol{\alpha}^*)\alpha_i^* - \langle Q\boldsymbol{\alpha}^*, y_i \mathbf{x}_i \rangle, \\
&\overset{(2.2)}{=} 1 - E_{iS_*}(\boldsymbol{\alpha}^*)\alpha_{S_*}^* - \langle Q_{S_*}\boldsymbol{\alpha}_{S_*}^*, Q_i \rangle, \\
&\overset{(1.14)}{=} 1 - H_{iS_*}(\boldsymbol{\alpha}^*)\alpha_{S_*}^*,
\end{aligned}$$

for any $i \in S_* = \mathrm{supp}(\boldsymbol{\alpha}^*)$, where $\boldsymbol{\alpha}^*$ is the optimal solution to (5.7). This leads to

$$\mathbf{y}_{S_*} \widehat{b} = \mathbf{1} - H_{S_* S_*}(\boldsymbol{\alpha}^*)\alpha_{S_*}^* \overset{(1.16)}{=} \mathbf{1} - H_{S_*}(\boldsymbol{\alpha}^*)\boldsymbol{\alpha}_{S_*}^*.$$

Multiplying both sides of the above equation by $\mathbf{y}_{S_*}$ yields b^* in (2.1) due to $\langle \mathbf{y}_{S_*}, \mathbf{y}_{S_*} \rangle = |S_*|$, finishing the proof. □

5.2 Proof of Theorem 2.2

The solution set is non-empty since 0 satisfies the constraints of (1.8). The problem can be written as

$$\min_{|T| \le s, T \subseteq [m]} \Big\{ \min_{\boldsymbol{\alpha} \in \mathbb{R}^m} D(\alpha) : \langle \boldsymbol{\alpha}_T, \mathbf{y}_T \rangle = 0 \Big\}. \tag{5.8}$$

It follows from (1.14) that $D(\cdot)$ is strongly convex. So the inner problem is a strongly convex program which admits a unique solution, say α_T. In addition, the choices of T such that $|T| \le s, T \subseteq [m]$ are finitely many. To derive the global optimal solution, we just pick one T from those choices making $D(\alpha_T)$ the smallest. □

5.3 A Lemma for Theorem 2.4

To prove Theorem 2.4, we need the following Lemma.

Lemma 5.1 *Consider a feasible point $\boldsymbol{\alpha}^*$ to the problem (1.8), namely, $\|\boldsymbol{\alpha}^*\|_0 \leq s$ 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

$$\left\{\begin{array}{rcl} g_{S_*}(\mathbf{z}^*) &=& 0,\\ \langle \boldsymbol{\alpha}^*, \mathbf{y}\rangle &=& 0,\\ \|\boldsymbol{\alpha}^*\|_0 &=& s, \end{array}\right. \quad \text{or} \quad \left\{\begin{array}{rcl} g(\mathbf{z}^*) &=& 0,\\ \langle \boldsymbol{\alpha}^*, \mathbf{y}\rangle &=& 0,\\ \|\boldsymbol{\alpha}^*\|_0 &<& s. \end{array}\right. \tag{5.9}$$

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

proof We only prove that the global minimizer $\boldsymbol{\alpha}^*$ is unique if $\|\boldsymbol{\alpha}^*\|_0 < s$ since the rest parts can be seen in [38, Theorem 3.2]. The condition in (1.14) indicates $D(\boldsymbol{\alpha})$ is a strongly convex function and thus enjoys the property

$$\begin{array}{rcl} D(\boldsymbol{\alpha}) & \overset{(1.14)}{\geq} & D(\boldsymbol{\alpha}') + \langle \nabla D(\boldsymbol{\alpha}'), \boldsymbol{\alpha} - \boldsymbol{\alpha}'\rangle\\ & + & \langle \boldsymbol{\alpha} - \boldsymbol{\alpha}', P(\boldsymbol{\alpha} - \boldsymbol{\alpha}')\rangle/2, \end{array} \tag{5.10}$$

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

$$\begin{array}{rl} & D(\boldsymbol{\alpha}) - D(\boldsymbol{\alpha}^*)\\ \overset{(5.10)}{\geq} & \langle \boldsymbol{\alpha} - \boldsymbol{\alpha}^*, P(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)\rangle/2 + \langle \nabla D(\boldsymbol{\alpha}^*), \boldsymbol{\alpha} - \boldsymbol{\alpha}^*\rangle\\ \overset{(1.16)}{=} & \langle \boldsymbol{\alpha} - \boldsymbol{\alpha}^*, P(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)\rangle/2 + \langle g(\mathbf{z}^*) - \mathbf{y}b^*, \boldsymbol{\alpha} - \boldsymbol{\alpha}^*\rangle\\ \overset{(5.9)}{=} & \langle \boldsymbol{\alpha} - \boldsymbol{\alpha}^*, P(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)\rangle/2 - \langle \mathbf{y}b^*, \boldsymbol{\alpha} - \boldsymbol{\alpha}^*\rangle\\ \overset{(5.9)}{=} & \langle \boldsymbol{\alpha} - \boldsymbol{\alpha}^*, P(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)\rangle/2 \overset{(1.14)}{>} 0, \end{array}$$

where the third equation is 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 $D(\boldsymbol{\alpha}^*) = D(\boldsymbol{\alpha})$, which contradicts with the above inequality. Therefore, $\boldsymbol{\alpha}^*$ is unique. □

5.4 Proof of Theorem 2.4

a) It follows from [39, Lemma 2.2] that an η-stationary point in (2.3) can be equivalently written as

$$\left\{\begin{array}{rcl} g_{S_*}(\mathbf{z}^*) &=& 0,\\ \eta\|g_{\overline{S}_*}(\mathbf{z}^*)\|_{[1]} &\leq& \|\boldsymbol{\alpha}^*\|_{[s]},\\ \|\boldsymbol{\alpha}^*\|_0 &\leq& s,\\ \langle \boldsymbol{\alpha}^*, \mathbf{y}\rangle &=& 0. \end{array}\right. \tag{5.11}$$

This clearly indicates (5.9) by $\|\boldsymbol{\alpha}^*\|_{[s]} = 0$ if $\|\boldsymbol{\alpha}^*\|_0 < s$. Therefore it is a local minimizer by Lemma 5.1 a).

b) If $\|\boldsymbol{\alpha}^*\|_0 < s$, then Lemma 5.1 a) states that a local minimizer $\boldsymbol{\alpha}^*$ satisfies the second condition in (5.9) which is same as (5.11). 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 (5.9),

$$g_{S_*}(\mathbf{z}^*) \overset{(1.16)}{=} (H(\boldsymbol{\alpha}^*)\boldsymbol{\alpha}^*)_{S_*} - \mathbf{1} + \mathbf{y}_{S_*} b^* = 0,$$

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

$$\begin{aligned}
\|g_{\overline{S}_*}(\mathbf{z}^*)\|_{[1]} &\overset{(1.16)}{=} \|(H(\boldsymbol{\alpha}^*)\boldsymbol{\alpha}^*)_{\overline{S}_*} - \mathbf{1} + \mathbf{y}_{\overline{S}_*} b^*\|_{[1]} \\
&\leq \|(H(\boldsymbol{\alpha}^*)\boldsymbol{\alpha}^*)_{\overline{S}_*} - \mathbf{1}\|_{[1]} + |b^*|\|\mathbf{y}_{\overline{S}_*}\|_{[1]} \\
&= \|(H(\boldsymbol{\alpha}^*)\boldsymbol{\alpha}^*)_{\overline{S}_*} - \mathbf{1}\|_{[1]} + |b^*| \\
&= \|(H(\boldsymbol{\alpha}^*)\boldsymbol{\alpha}^*)_{\overline{S}_*} - \mathbf{1}\|_{[1]} + \|(H(\boldsymbol{\alpha}^*)\boldsymbol{\alpha}^*)_{S_*} - \mathbf{1}\|_{[1]} \\
&\leq \|H(\boldsymbol{\alpha}^*)\boldsymbol{\alpha}^* - \mathbf{1}\|_{[1]} 2 = \|\boldsymbol{\alpha}^*\|_{[s]}/\eta^* \leq \|\boldsymbol{\alpha}^*\|_{[s]}/\eta.
\end{aligned}$$

This verifies the second inequality in (5.11), together with (5.9) claiming the conclusion.

c) An η-stationary point $\boldsymbol{\alpha}^*$ with $\|\boldsymbol{\alpha}^*\|_0 < s$ satisfies the condition (5.11), which is same as the second case $\|\boldsymbol{\alpha}^*\|_0 < s$ in (5.9). Namely, $\boldsymbol{\alpha}^*$ is also a local minimizer, which makes the conclusion immediately from Lemma 5.1 b).

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

$$\boldsymbol{\alpha}^* \overset{(2.3)}{\in} \mathbb{P}_s(\boldsymbol{\alpha}^* - \eta g(\mathbf{z}^*)) \overset{(1.9)}{=} \underset{\|\boldsymbol{\alpha}\|_0 \leq s}{\operatorname{argmin}} \|\boldsymbol{\alpha} - (\boldsymbol{\alpha}^* - \eta g(\mathbf{z}^*))\|,$$

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

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

This suffices to

$$-\|\boldsymbol{\alpha}^* - \boldsymbol{\alpha}\|^2 \leq 2\eta\langle \boldsymbol{\alpha} - \boldsymbol{\alpha}^*, g(\mathbf{z}^*)\rangle. \tag{5.12}$$

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

$$\begin{aligned}
2D(\boldsymbol{\alpha}) - 2D(\boldsymbol{\alpha}^*) &\overset{(5.10)}{\geq} \langle \boldsymbol{\alpha} - \boldsymbol{\alpha}^*, P(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)\rangle + 2\langle \nabla D(\boldsymbol{\alpha}^*), \boldsymbol{\alpha} - \boldsymbol{\alpha}^*\rangle \\
&\overset{(1.16)}{=} \langle \boldsymbol{\alpha} - \boldsymbol{\alpha}^*, P(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)\rangle + 2\langle g(\mathbf{z}^*) - \mathbf{y}b^*, \boldsymbol{\alpha} - \boldsymbol{\alpha}^*\rangle \\
&\overset{(5.11)}{=} \langle \boldsymbol{\alpha} - \boldsymbol{\alpha}^*, P(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)\rangle + 2\langle g(\mathbf{z}^*), \boldsymbol{\alpha} - \boldsymbol{\alpha}^*\rangle \\
&\overset{(5.12)}{\geq} \langle \boldsymbol{\alpha} - \boldsymbol{\alpha}^*, P(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)\rangle - \|\boldsymbol{\alpha}^* - \boldsymbol{\alpha}\|^2/\eta \\
&= \langle \boldsymbol{\alpha} - \boldsymbol{\alpha}^*, (P - I/\eta)(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)\rangle \geq 0,
\end{aligned}$$

where the last inequity follows from

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

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

$$0 = D(\hat{\boldsymbol{\alpha}}) - D(\boldsymbol{\alpha}^*) \geq \langle \hat{\boldsymbol{\alpha}} - \boldsymbol{\alpha}^*, (P - I/\eta)(\hat{\boldsymbol{\alpha}} - \boldsymbol{\alpha}^*)\rangle > 0.$$

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

5.5 Proof of Theorem 2.3

It follows from $\mathbf{z}^*$ being an η-stationary point and (2.3) that $\langle \boldsymbol{\alpha}^*, \mathbf{y}\rangle = 0$ and

$$\boldsymbol{\alpha}^* \in \mathbb{P}_s\left(\boldsymbol{\alpha}^* - \eta g(\mathbf{z}^*)\right) \overset{(1.11)}{=} \left\{ \begin{bmatrix} \boldsymbol{\alpha}^*_T - \eta g_T(\mathbf{z}^*) \\ 0 \end{bmatrix} : T \in \mathbb{T}_s(\boldsymbol{\alpha}^* - \eta g(\mathbf{z}^*)) \right\},$$

which is equivalent to that there is 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 concludes the conclusion immediately. □

5.6 A Lemma for Theorem 3.1

Before proving Theorem 3.1, we first present some properties regarding an η-stationary point of (1.8).

Lemma 5.2 *Let $\mathbf{z}^*$ be an η-stationary point of (1.8) with $0 < \eta < \eta^*$, where η^* is given by (3.6). Then there is always a $\delta^* > 0$ such that for any $\mathbf{z} \in N(\mathbf{z}^*, \delta^*)$ with $\|\boldsymbol{\alpha}\|_0 \leq s$, the following results hold.*

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

$$\mathbb{T}_s(\boldsymbol{\alpha} - \eta g(\mathbf{z}) = \mathbb{T}_s(\boldsymbol{\alpha}^* - \eta g(\mathbf{z}^*)) = \{S_*\}. \tag{5.13}$$

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

$$S_* \subseteq (T_* \cap T). \tag{5.14}$$

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

$$F(\mathbf{z}^*; T) = 0. \tag{5.15}$$

proof a) It follows from Theorem 2.3 and $\mathbf{z}^*$ being an η-stationary point of (1.8) that $F(\mathbf{z}^*; T_*) = 0$ for any $T_* \in \mathbb{T}_s(\boldsymbol{\alpha}^* - \eta g(\mathbf{z}^*))$. We first derive

$$E(\boldsymbol{\alpha}^*)\boldsymbol{\alpha}^* = E(\boldsymbol{\alpha})\boldsymbol{\alpha}^*. \tag{5.16}$$

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

$$\alpha_i^* > 0 \Longrightarrow \alpha_i > 0, \quad \alpha_i^* < 0 \Longrightarrow \alpha_i < 0. \tag{5.17}$$

If (5.17) is not true, then there is an j that violates one of the above relations, namely α_j^* and α_j have different signs. As a consequence, $|\alpha_j^*| < |\alpha_j^* - \alpha_j| \leq \|\boldsymbol{\alpha}^* - \boldsymbol{\alpha}\| < \delta^*$. This is a contradiction for a sufficiently small positive δ^*. Therefore, we must have (5.17). Recall that $E(\boldsymbol{\alpha})$ is a diagonal matrix with diagonal elements given by (1.13). It follows

$$\begin{aligned}
[(E(\boldsymbol{\alpha}^*) - E(\boldsymbol{\alpha}))\boldsymbol{\alpha}^*]_i &= [E_{ii}(\boldsymbol{\alpha}^*) - E_{ii}(\boldsymbol{\alpha})]\alpha_i^* \\
&\overset{(5.17),(1.13)}{=} \begin{cases} (1/C - 1/C)\alpha_i^*, & \alpha_i^* > 0 \\ (1/c - 1/c)\alpha_i^*, & \alpha_i^* < 0 \\ (E_{ii}(\boldsymbol{\alpha}^*) - E_{ii}(\boldsymbol{\alpha}))0, & \alpha_i^* = 0 \end{cases} \\
&= 0.
\end{aligned}$$

So, (5.16) is true, which allow us to derive that

$$\begin{aligned}
\|H(\boldsymbol{\alpha}^*)\boldsymbol{\alpha}^* - H(\boldsymbol{\alpha})\boldsymbol{\alpha}\| &\overset{(1.14)}{=} \|(E(\boldsymbol{\alpha}^*) + Q^\top Q)\boldsymbol{\alpha}^* - (E(\boldsymbol{\alpha}) + Q^\top Q)\boldsymbol{\alpha}\| \\
&\leq \|E(\boldsymbol{\alpha}^*)\boldsymbol{\alpha}^* - E(\boldsymbol{\alpha})\boldsymbol{\alpha}\| + \|Q\|^2\|\boldsymbol{\alpha}^* - \boldsymbol{\alpha}\| \\
&\overset{(5.16)}{=} \|E(\boldsymbol{\alpha})(\boldsymbol{\alpha}^* - \boldsymbol{\alpha})\| + \|Q\|^2\|\boldsymbol{\alpha}^* - \boldsymbol{\alpha}\| \\
&\overset{(1.13)}{\leq} (1/c + \|Q\|^2)\|\boldsymbol{\alpha}^* - \boldsymbol{\alpha}\| \leq (1/c + \|Q\|^2)\delta^*, \qquad (5.18)
\end{aligned}$$

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

$$\begin{aligned}
\|g(\mathbf{z}^*) - g(\mathbf{z})\| &\overset{(1.16)}{=} \|H(\boldsymbol{\alpha}^*)\boldsymbol{\alpha}^* - H(\boldsymbol{\alpha})\boldsymbol{\alpha} + (b^* - b)\mathbf{y}\| \\
&\leq \|H(\boldsymbol{\alpha}^*)\boldsymbol{\alpha}^* - H(\boldsymbol{\alpha})\boldsymbol{\alpha}\| + |b^* - b|\|\mathbf{y}\| \\
&\overset{(5.18)}{\leq} (1/c + \|Q\|^2 + \sqrt{m})\delta^*. \qquad (5.19)
\end{aligned}$$

Case i) $\|\boldsymbol{\alpha}^*\|_0 = s$. Since $|T_*| = s$ and $\boldsymbol{\alpha}^*_{\overline{T}_*} = 0$ from (2.4), it holds $T_* = \mathrm{supp}(\boldsymbol{\alpha}^*) = S_*$. If $\|g_{\overline{S}_*}(\mathbf{z}^*)\|_{[1]} = 0$, then

$$\|\boldsymbol{\alpha}^*_{\overline{S}_*} - \eta g_{\overline{S}_*}(\mathbf{z}^*)\|_{[1]} \overset{(2.4)}{=} 0 < \|\boldsymbol{\alpha}^*_{S_*}\|_{[s]} \overset{(2.4)}{=} \|\boldsymbol{\alpha}^*_{S_*} - \eta g_{S_*}(\mathbf{z}^*)\|_{[s]} \qquad (5.20)$$

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

$$\begin{aligned}
\|\boldsymbol{\alpha}^*_{\overline{S}_*} - \eta g_{\overline{S}_*}(\mathbf{z}^*)\|_{[1]} &\overset{(2.4)}{=} \eta\|g_{\overline{S}_*}(\mathbf{z}^*)\|_{[1]} < \eta^*\|g_{\overline{S}_*}(\mathbf{z}^*)\|_{[1]} \\
&\overset{(3.6)}{=} \|\boldsymbol{\alpha}^*_{S_*}\|_{[s]} \overset{(2.4)}{=} \|\boldsymbol{\alpha}^*_{S_*} - \eta g_{S_*}(\mathbf{z}^*)\|_{[s]}
\end{aligned}$$

Therefore, both scenarios derive (5.20). 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 (1.10) indicating

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

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

$$\|\boldsymbol{\alpha}_{\overline{S}_*} - \eta g_{\overline{S}_*}(\mathbf{z})\|_{[1]} < \|\boldsymbol{\alpha}_{S_*} - \eta g_{S_*}(\mathbf{z})\|_{[s]} \qquad (5.21)$$

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 (5.19). These and (5.21) guarantee (5.21) immediately.

Case ii) $\|\boldsymbol{\alpha}^*\|_0 < s$. The fact $\boldsymbol{\alpha}^*_{\overline{T}_*} = 0$ from (2.4) 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 (5.11) derives

$$g(\mathbf{z}^*) = 0, \tag{5.22}$$

which means 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 (5.19). 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

$$F(\mathbf{z}^*; T) = \begin{bmatrix} g_T(\mathbf{z}^*) \\ \boldsymbol{\alpha}^*_{\overline{T}} \\ \langle \boldsymbol{\alpha}^*_T, \mathbf{y}_T \rangle \end{bmatrix} = 0. \tag{5.23}$$

If $\|\boldsymbol{\alpha}^*\|_0 = s$, then $T = S_* = T_*$ by a), which shows the result by (2.4) immediately. If $\|\boldsymbol{\alpha}^*\|_0 < s$, then $g_T(\mathbf{z}^*) = 0$ by (5.22). Again from b), $S_* \subseteq (T \cap T_*)$ means $\overline{T} \subseteq \overline{S}_*$, which indicates $\boldsymbol{\alpha}^*_{\overline{T}} = 0$ due to $\boldsymbol{\alpha}^*_{\overline{S}_*} = 0$. Finally,

$$\begin{aligned} \langle \boldsymbol{\alpha}^*_T, \mathbf{y}_T \rangle &= \langle \boldsymbol{\alpha}^*_{S_*}, \mathbf{y}_{S_*} \rangle + \langle \boldsymbol{\alpha}^*_{T \setminus S_*}, \mathbf{y}_{T \setminus S_*} \rangle = \langle \boldsymbol{\alpha}^*_{S_*}, \mathbf{y}_{S_*} \rangle \\ &= \langle \boldsymbol{\alpha}^*_{T_*}, \mathbf{y}_{T_*} \rangle - \langle \boldsymbol{\alpha}^*_{T_* \setminus S_*}, \mathbf{y}_{T_* \setminus S_*} \rangle \overset{(2.4)}{=} 0. \end{aligned}$$

The whole proof is finished. □

5.7 Proof of Theorem 3.1

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

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

We first prove that

$$E(\boldsymbol{\alpha}^k) = E(\boldsymbol{\alpha}^k_t). \tag{5.24}$$

In fact, if $\boldsymbol{\alpha}^* = 0$, then $\boldsymbol{\alpha}^k_t = t\boldsymbol{\alpha}^k$, which means $\boldsymbol{\alpha}^k_t$ and $\boldsymbol{\alpha}^k$ have the same signs. This together with the definition (1.13) of $E(\cdot)$ shows (5.24) immediately. If $\boldsymbol{\alpha}^* \neq 0$, then same reasoning proving (5.17) also derives that

$$\begin{aligned} &\alpha^*_i > 0 \implies \alpha^k_i > 0, \quad (\alpha^k_t)_i = (1-t)\alpha^*_i + t\alpha^k_i > 0, \\ &\alpha^*_i < 0 \implies \alpha^k_i < 0, \quad (\alpha^k_t)_i = (1-t)\alpha^*_i + t\alpha^k_i < 0, \\ &\alpha^*_i = 0 \implies \quad\quad\quad (\alpha^k_t)_i = t\alpha^k_i. \end{aligned}$$

These also mean $\boldsymbol{\alpha}^k_t$ and $\boldsymbol{\alpha}^k$ have the same signs. Namely, (5.24) is true and brings out

$$H(\boldsymbol{\alpha}^k) \overset{(1.14)}{=} E(\boldsymbol{\alpha}^k) + Q^\top Q \overset{(5.24)}{=} E(\boldsymbol{\alpha}^k_t) + Q^\top Q = H(\boldsymbol{\alpha}^k_t),$$

which contributes to

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

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

$$\begin{aligned} F(\mathbf{z}^k; T_k) &\overset{(5.15)}{=} F(\mathbf{z}^k; T_k) - F(\mathbf{z}^*; T_k) \\ &= \nabla F(\mathbf{z}_t^k; T_k)(\mathbf{z}^k - \mathbf{z}^*) \\ &= \nabla F(\mathbf{z}^k; T_k)(\mathbf{z}^k - \mathbf{z}^*), \end{aligned}$$

which together with $\nabla F(\mathbf{z}^k; T_k)$ being always non-singular because of (2.5) suffices to

$$\begin{aligned} \mathbf{z}^* &= \mathbf{z}^k - (\nabla F(\mathbf{z}^k; T_k))^{-1} F(\mathbf{z}^k; T_k) \\ &\overset{(3.1)}{=} \mathbf{z}^k + \mathbf{d}^k \overset{(3.3)}{=} \mathbf{z}^{k+1}. \end{aligned}$$

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

$$\|F(\mathbf{z}^{k+1}, T_{k+1})\| = \|F(\mathbf{z}^*, T_{k+1})\| \overset{(2.4)}{=} 0.$$

The whole proof is completed. □

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

[26] 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.

[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] 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.

[31] 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.

[32] 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.

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

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

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

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

[37] 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.

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

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