Dual Coordinate-Descent Methods for Linear One-Class SVM and SVDD

Hung-Yi Chou* Pin-Yen Lin* Chih-Jen Lin*

Abstract

One-class support vector machines (SVM) and support vector data description (SVDD) are two effective outlier detection techniques. They have been successfully applied to many applications under the kernel settings, but for some high dimensional data, linear rather than kernel one-class SVM and SVDD may be more suitable. Past developments on kernel and linear classification have indicated that specially designed optimization algorithms can make the training for linear scenarios much faster. However, we point out that because of some differences from standard linear SVM, existing algorithms may not be efficient for one-class scenarios. We then develop some novel coordinate descent methods for linear one-class SVM and SVDD. Experiments demonstrate their superiority on the convergence speed.

1 Introduction

Outlier detection is widely used in applications of data mining and machine learning. An outlier detection method we are interested in is one-class SVM [15] and its variant SVDD [17]. They were proposed as unsupervised extensions of kernel SVM for two-class classification, and have been used in numerous applications.

To obtain the model of one-class SVM and SVDD, we must solve a quadratic programming problem. The solution procedure has been well studied. In particular, if kernels are used, past works solved the dual problem, where a quadratic function is minimized over some bounded constraints and one linear constraint. Currently, the coordinate descent method (also called the decomposition methods in the SVM community) is the major optimization method to train kernelized one-class SVM or SVDD; see, for example, the implementation in the software LIBSVM [1].

For data sets with a very high number of features, past works on SVM for data classification have shown that the original input space may be rich enough and there is no need to apply the kernel trick (e.g., [9, 8, 11, 6]). In such a situation, although the same optimization algorithm for kernel can be used for linear (i.e., through the linear kernel), existing works demonstrate that specially designed algorithms can dramatically speed up the training process. While linear SVM for classification and regression have been well studied, so far no works have investigated the effectiveness of linear one-class SVM and SVDD, and their efficient training. We illustrate the research status of various scenarios in the following table.

	Two-class SVM	One-class SVM
kernel	studied	studied
linear	studied	not yet

The goal of this work is to partially fill the gap by studying optimization algorithms for linear one-class SVM and SVDD.[1,2]

From the close relationship between standard SVM and one-class SVM, naturally we think that an algorithm suitable for one should work for the other. However, a recent work [2] shows that this thinking may not be correct. They noticed that the success of coordinate descent methods for linear two-class SVM partially relies on a fact that an SVM formulation without the bias term was considered. Specifically, for classification, the standard SVM decision function is

$$\mathrm{sgn}(\boldsymbol{w}^T\boldsymbol{x} + b),$$

where $(\boldsymbol{w}, b)$ is the model with b called a bias term, and $\boldsymbol{x}$ is any feature vector. For data in a high dimensional space, it is known that b can be removed without sacrificing the performance. An important fact is that SVM without a bias term leads to a dual problem not having a linear constraint. The study in [2] pointed out that the coordinate descent (CD) algorithm achieving the state-of-the-art training speed for linear SVM without the bias term has slower convergence for linear SVM with the bias term. The reason is that the linear constraint causes the waste of many CD steps (i.e., variables are not updated) and slows down the overall convergence. Interestingly, for one-class SVM or SVDD, the dual optimization problem

*Computer Science and Information Engineering, National Taiwan Univ. This work was supported in part by MOST of Taiwan grant 107-2221-E-002-167-MY3.

[1] Besides, some users of our LIBLINEAR software [4] have requested a solver for linear one-class SVM and SVDD.

[2] Some works (e.g., [3]) have studied linear one-class SVM, but they do not focus on the optimization algorithms for training as we do here.

inherently contains a linear constraint. Therefore, in [2], they conclude that for linear one-class SVM or SVDD, new algorithms must be developed for efficient training.

In this work, we aim to develop efficient CD algorithms for linear one-class SVM and SVDD. To address the slow-convergence issue caused by the linear constraint, we propose new settings to enlarge the working set of variables selected for update. Thus variables can be changed despite of the appearance of the linear constraint. However, the higher cost in solving the sub-problem at each CD step becomes a concern. Our approach combines effective working-set selections and the approximate solve of the sub-problem to achieve fast convergence.

This work is organized as follows. In Section 2 we review CD algorithms for standard linear SVM and explain the slow-convergence issue when the dual problem has a linear constraint. In Section 3, we propose several CD methods suitable for linear one-class SVM and SVDD. Experiments in Section 4 demonstrate the effectiveness of the proposed methods. Finally, Section 5 concludes this work. One of the proposed methods (Algorithm 5) has been included in the software LIBLINEAR (after version 2.40) for public use. Supplementary materials and programs used for experiments are at `https://www.csie.ntu.edu.tw/~cjlin/papers/linear_oneclass_SVM`.

2 Coordinate Descent Methods for the Dual Problem of Linear SVM with/without the Bias Term

In this section, by reviewing the recent work in [2], we show the need to develop new CD method for linear one-class SVM and linear SVDD.

To begin, we introduce the formulation of linear SVM for classification. For a training set of label-instance pairs $(y_i, \boldsymbol{x}_i), i = 1, \ldots, l$, where $y_i \in \{-1, +1\}$ and $\boldsymbol{x}_i \in R^n$, a standard linear SVM solves the following primal optimization problem:

$$\begin{aligned} \min_{\boldsymbol{w},b} \quad & \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum\nolimits_{i=1}^{l}\xi_i \\ \text{subject to} \quad & y_i\left(\boldsymbol{w}^T\boldsymbol{x}_i + b\right) \geq 1 - \xi_i, \\ & \xi_i \geq 0, i = 1, \ldots, l, \end{aligned}$$

where n is the number of features, $C > 0$ is a penalty parameter, and b is called the bias term.[3] Currently a state-of-the-art approach to train a linear SVM is by the CD method to solve the dual SVM problem [6]. However, in [6] the bias term is omitted. To see if the removal of the bias term contributes to the success of CD methods, [2] studied the difference between CD for linear SVM with and without the bias term. To explain their results we need the following dual SVM problem.

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & f(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^T Q\boldsymbol{\alpha} - \boldsymbol{e}^T\boldsymbol{\alpha} \\ \text{subject to} \quad & 0 \leq \alpha_i \leq C_i,\ i = 1, \ldots, l, \\ & \boldsymbol{y}^T\boldsymbol{\alpha} = 0 \text{ (if bias is considered)}, \end{aligned} \tag{2.1}$$

where $\boldsymbol{e} = [1, \ldots, 1]^T$, $C_i = C$ and $Q_{ij} = y_i y_j \boldsymbol{x}_i^T\boldsymbol{x}_j$.

We notice that if the bias term is removed, then the dual problem does not have a linear constraint.

2.1 Coordinate Descent Methods for Linear SVM

The basic idea of a CD method is to iteratively minimize the objective function along only a few coordinates at a time. For the current $\boldsymbol{\alpha}$, we change elements in a selected working set B while fixing others in

$$N = \{1, \ldots, l\} \setminus B.$$

That is,

$$\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} + \boldsymbol{d} = \begin{bmatrix} \boldsymbol{\alpha}_B \\ \boldsymbol{\alpha}_N \end{bmatrix} + \begin{bmatrix} \boldsymbol{d}_B \\ \boldsymbol{0} \end{bmatrix},$$

where the sub-vector $\boldsymbol{d}_B$ is used to update $\boldsymbol{\alpha}_B$ and $\boldsymbol{\alpha}_N$ is fixed. The objective function can be rewritten as

$$f(\begin{bmatrix} \boldsymbol{\alpha}_B \\ \boldsymbol{\alpha}_N \end{bmatrix} + \begin{bmatrix} \boldsymbol{d}_B \\ \boldsymbol{0} \end{bmatrix}) = \frac{1}{2}\boldsymbol{d}_B^T Q_{BB}\boldsymbol{d}_B + \nabla_B f(\boldsymbol{\alpha})^T\boldsymbol{d}_B + \text{constant},$$

and $\boldsymbol{d}_B$ is the solution of the following sub-problem

$$\begin{aligned} \min_{\boldsymbol{d}_B} \quad & \frac{1}{2}\boldsymbol{d}_B^T Q_{BB}\boldsymbol{d}_B + \nabla_B f(\boldsymbol{\alpha})^T\boldsymbol{d}_B \\ \text{subject to} \quad & -\alpha_i \leq d_i \leq C_i - \alpha_i, \text{ for } i \in B, \\ & \boldsymbol{y}_B^T\boldsymbol{d}_B = 0 \text{ (if bias is considered)}. \end{aligned} \tag{2.2}$$

In [6], B is chosen to have only one single element so the sub-problem (2.2) has a simple analytical solution. Such a setting is not possible if the bias term is considered. More precisely, if $|B| = 1$ and the constraint $\boldsymbol{y}_B^T\boldsymbol{d}_B = 0$ appears in (2.2), then the only feasible point of the sub-problem is $\boldsymbol{d}_B = \boldsymbol{0}$ and we simply cannot move $\boldsymbol{\alpha}$ at all. Therefore, we must have $|B| \geq 2$ in order to update $\boldsymbol{\alpha}$. It is known [13, 16, 2] that with $|B| = 2$, an analytic solution of the sub-problem is available regardless of whether the linear constraint appears or not. However, if $|B| > 2$, an iterative optimization procedure is needed for solving the sub-problem. We summarize the CD procedure in Algorithm 1.

Past works have thoroughly investigated the following important components of CD methods:
1. the calculation of the gradient $\nabla_B f(\boldsymbol{\alpha})$, and
2. the selection of the working set B.

We discuss them in Sections 2.2 and 2.3, respectively.

[3] If the squared hinge loss is considered, ξ_i in the objective function is replaced by ξ_i^2. All the analysis in this section still applies.

Algorithm 1 A framework of block CD methods

1: Let $\boldsymbol{\alpha}$ be a feasible point
2: **while** $\boldsymbol{\alpha}$ is not optimal **do**
3: Select a working set B
4: Solve the sub-problem (2.2)
5: Update $\boldsymbol{\alpha}$ by $\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} + \boldsymbol{d}$
6: **end while**

2.2 The Calculation of the Gradient $\nabla_B f(\boldsymbol{\alpha})$ For linear SVM, a technique developed in [6] to efficiently calculate $\nabla_B f(\boldsymbol{\alpha})$ is by for $i \in B$,

$$\nabla_i f(\boldsymbol{\alpha}) = \sum_{j=1}^{l} y_i y_j \boldsymbol{x}_i \boldsymbol{x}_j \alpha_j - 1 = y_i \boldsymbol{u}^T \boldsymbol{x}_i - 1, \tag{2.3}$$

where

$$\boldsymbol{u} \equiv \sum_{j=1}^{l} \alpha_j y_j \boldsymbol{x}_j. \tag{2.4}$$

If $\boldsymbol{u}$ is available, then obtaining $\nabla_B f(\boldsymbol{\alpha})$ needs $O(|B|n)$ cost. To maintain $\boldsymbol{u}$, after the sub-problem (2.2) is solved, we can use

$$\boldsymbol{u} \leftarrow \boldsymbol{u} + \sum_{i \in B} d_i y_i \boldsymbol{x}_i, \tag{2.5}$$

which also costs $O(|B|n)$.

This technique is not applicable to kernel SVM, where $\boldsymbol{x}_i$ is mapped to a higher, maybe infinite, dimensional space by a function $\phi(\cdot)$. Then the calculation in (2.3) involves $\boldsymbol{u}^T \phi(\boldsymbol{x})$, a high-dimensional inner product that cannot be directly calculated. Instead, the kernel trick is applied so that by choosing certain $\phi(\boldsymbol{x})$, the following kernel function can be easily calculated

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j).$$

Then the gradient is calculated by

$$\begin{aligned} \nabla_i f(\boldsymbol{\alpha}) &= \sum_{j=1}^{l} y_i y_j \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j) \alpha_j - 1 \\ &= \sum_{j=1}^{l} y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) \alpha_j - 1. \end{aligned} \tag{2.6}$$

If each kernel evaluation costs $O(n)$, then the gradient calculation costs $O(l|B|n)$, which is significantly higher than $O(|B|n)$ by (2.3)-(2.5). We will show in Section 2.3 that this difference causes that suitable working-set selections for linear and kernel SVM are very different.

2.3 Working-set Selection Two major schemes to select the working set for CD methods are

- random or cyclic selection, and
- greedy selection by using the gradient information.

In [6], a greedy selection is not possible because they calculate $\nabla_B f(\boldsymbol{\alpha})$ rather than the full gradient $\nabla f(\boldsymbol{\alpha})$. Thus they consider a random or a cyclic selection of one variable at a time.

In contrast, for kernel SVM, the calculation in (2.6) obtains the kernel sub-matrix $Q_{:,B}$, so we can easily maintain the gradient by

$$\nabla f(\boldsymbol{\alpha} + \boldsymbol{d}) = \nabla f(\boldsymbol{\alpha}) + Q_{:,B} \boldsymbol{d}_B.$$

Because a greedy selection often leads to a better working set, most past works on kernel SVM (e.g., [7, 10, 5]) apply such a setting.

2.4 The Use of the Bias Term We mentioned that depending on whether the bias term is used, the dual SVM problem has or does not have a linear constraint. The behavior of CD methods for both settings has been well studied in the literature. For kernel SVM, traditionally the bias term is used. If without the bias term, [16] studied greedy selections under $|B| = 1$ or 2. Their main results are

- If without the bias term, the CD methods with $|B| = 2$ are faster than those with $|B| = 1$.
- Under $|B| = 2$, dual CD algorithms for SVM without the bias term are slightly faster than those for SVM with the bias term.

Interestingly, the difference between CD algorithms for linear SVM with and without the bias term is more dramatic [2]. We mentioned earlier that because the gradient is not maintained in calculating (2.3)-(2.5), a greedy selection is not applicable. Under random or cyclic selections, [2] explained that CD algorithms converge much faster if the dual problem does not contain a linear constraint (i.e., the bias term is not considered). The reason is that for a working set of only two variables

$$B = \{i, j\},$$

a linear constraint may cause that the sub-problem is easily already optimal. Then α_i, α_j cannot be moved. A simple explanation is that the feasible region becomes more restrictive [2]. From the Karush-Kuhn-Tucker (KKT) condition of (2.1) with a linear constraint, a feasible $\boldsymbol{\alpha}$ is optimal for (2.1) if and only if

$$\max_{t \in I_{\text{up}}(\boldsymbol{\alpha})} -y_t \nabla_t f(\boldsymbol{\alpha}) \le \min_{t \in I_{\text{low}}(\boldsymbol{\alpha})} -y_t \nabla_t f(\boldsymbol{\alpha}), \tag{2.7}$$

where by following the notation in [5], $I_{\text{up}}(\boldsymbol{\alpha})$ and $I_{\text{low}}(\boldsymbol{\alpha})$ are two sets defined as

$$\begin{aligned} I_{\text{up}}(\boldsymbol{\alpha}) &= \{t \mid \alpha_t < C_t, y_t = 1 \text{ or } \alpha_t > 0, y_t = -1\}, \\ I_{\text{low}}(\boldsymbol{\alpha}) &= \{t \mid \alpha_t < C_t, y_t = -1 \text{ or } \alpha_t > 0, y_t = 1\}. \end{aligned} \tag{2.8}$$

Now for a two-variable sub-problem from a working set B, the optimality condition is

$$\max_{t\in I_{\text{up}}(\boldsymbol{\alpha}_B)} -y_t\nabla_t f(\boldsymbol{\alpha}) \le \min_{t\in I_{\text{low}}(\boldsymbol{\alpha}_B)} -y_t\nabla_t f(\boldsymbol{\alpha}). \tag{2.9}$$

In contrast, if the linear constraint is removed from (2.1), the sub-problem's optimality condition is

$$\begin{cases}\nabla_t f(\boldsymbol{\alpha}) \le 0 & \text{if } \alpha_t > 0,\\ \nabla_t f(\boldsymbol{\alpha}) \ge 0 & \text{if } \alpha_t < C_t,\end{cases} \quad \forall t \in B. \tag{2.10}$$

Clearly, indices in B are entangled together in (2.9), but are not in (2.10). With this and other reasons, a randomly selected (α_i, α_j) can easily be already optimal by satisfying (2.9) and cannot be moved; see more details in [2].

To illustrate a two-variable CD method for linear SVM with a linear constraint, we provide details in Algorithm 2. Because the sub-problem may be easily already optimal, to avoid wasting operations, in Algorithm 2 we have several pre-checks. For example, in line 4 by easily checking the boundness of components in $\boldsymbol{\alpha}_B$, we may know that $\boldsymbol{\alpha}_B$ is already optimal for the sub-problem. Then we do not even need to calculate $\nabla_B f(\boldsymbol{\alpha})$ and can move to the next CD step.

The study in [2] concludes that if a two-variable working set is considered, to have an efficient CD procedure, one should use the dual SVM problem without the linear constraint. Unfortunately, for linear one-class SVM or SVDD that we aim to solve, its dual problem inherently has a linear constraint. Therefore, new strategies must be developed in order to avoid the slow convergence of the CD procedure.

2.5 Review of Greedy Working-set Selection for Kernel SVM The discussion in Section 2.4 indicates that if $|B| = 2$ and the dual has a linear constraint, a random or a cyclic working-set selection is not effective. However, a greedy selection is out of question because we have mentioned that maintaining the gradient is extremely expensive. Therefore, we suspect that a setting between greedy and random/cyclic selections might be suitable for our need. To facilitate the development, we review past greedy working-set selections.

First we introduce the working-set selection by the maximal violating pair [10]. The basic idea is to find a pair of variables that most violate the optimality condition (2.7). An index pair (i, j) with $i \in I_{\text{up}}(\boldsymbol{\alpha})$ and $j \in I_{\text{low}}(\boldsymbol{\alpha})$ is called a violating pair if

$$-y_i\nabla_i f(\boldsymbol{\alpha}) > -y_j\nabla_j f(\boldsymbol{\alpha}), \tag{2.12}$$

Algorithm 2 A two-variable CD algorithm for solving the dual problem of linear SVM. The bias term is considered, so the dual problem contains a linear constraint.

1: Let $\boldsymbol{\alpha}$ be a feasible point and calculate $Q_{ii}, \forall i$
2: **while** $\boldsymbol{\alpha}$ is not optimal **do**
3: Randomly/cyclicly get a working set $B = \{i, j\}$
4: **if** $I_{\text{up}}(\boldsymbol{\alpha}_B) = \emptyset$ or $I_{\text{low}}(\boldsymbol{\alpha}_B) = \emptyset$ **then**
5: **continue**
6: **end if**
7: Calculate $\nabla_i f(\boldsymbol{\alpha})$ and $\nabla_j f(\boldsymbol{\alpha})$ by (2.3)
8: **if** the optimality condition in (2.9) holds **then**
9: **continue**
10: **end if**
11: Calculate Q_{ij} to form the following sub-problem

$$\begin{aligned}\min \quad & \frac{1}{2}\begin{bmatrix}d_i & d_j\end{bmatrix}\begin{bmatrix}Q_{ii} & Q_{ij}\\ Q_{ij} & Q_{jj}\end{bmatrix}\begin{bmatrix}d_i\\ d_j\end{bmatrix}\\ & + \begin{bmatrix}\nabla_i f(\boldsymbol{\alpha}) & \nabla_j f(\boldsymbol{\alpha})\end{bmatrix}\begin{bmatrix}d_i\\ d_j\end{bmatrix}\\ \text{subject to} \quad & -\alpha_i \le d_i \le C_i - \alpha_i,\\ & -\alpha_j \le d_j \le C_j - \alpha_j,\\ & y_i d_i + y_j d_j = 0.\end{aligned} \tag{2.11}$$

12: Solve the sub-problem.
13: Update (α_i, α_j) and the vector $\boldsymbol{u}$ by (2.5).
14: **end while**

and is further called a maximal violating pair if

$$\begin{aligned} i &\leftarrow \arg\max_t \{-y_t\nabla_t f(\boldsymbol{\alpha}) \mid t \in I_{\text{up}}(\boldsymbol{\alpha})\},\\ j &\leftarrow \arg\min_t \{-y_t\nabla_t f(\boldsymbol{\alpha}) \mid t \in I_{\text{low}}(\boldsymbol{\alpha})\}.\end{aligned} \tag{2.13}$$

A maximal violating pair was used as the working set of a two-variable CD in the popular software LIBSVM [1] for kernel SVM, though they now consider an improved strategy developed in [5].

3 New CD Methods for Linear One-class SVM and Linear SVDD

For linear one-class SVM and SVDD we propose new CD settings that can avoid the waste of many steps when the linear constraint is in the dual problem.

3.1 Dual Forms of Linear One-class SVM and Linear SVDD For a set of points $\boldsymbol{x}_i$, $i = 1, \ldots, l$ without any label information, one-class SVM [15] and SVDD [17] described below were developed to detect the outliers.

Given $0 < \nu < 1$, one-class SVM solves the following

primal problem:

$$\begin{aligned} \min_{\boldsymbol{w},\boldsymbol{\xi},\rho} \quad & \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} - \rho + \frac{1}{\nu l}\sum\nolimits_{i=1}^{l}\xi_i \\ \text{subject to} \quad & \boldsymbol{w}^T\boldsymbol{x}_i \geq \rho - \xi_i, \\ & \xi_i \geq 0, i = 1, \ldots, l. \end{aligned}$$

The dual form is

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\boldsymbol{\alpha}^T Q\boldsymbol{\alpha} \\ \text{subject to} \quad & 0 \leq \alpha_i \leq \frac{1}{\nu l}, i = 1, \ldots, l, \\ & \boldsymbol{e}^T\boldsymbol{\alpha} = 1, \end{aligned} \tag{3.14}$$

where

$$Q_{ij} = \boldsymbol{x}_i^T\boldsymbol{x}_j. \tag{3.15}$$

For SVDD under some $C > 0$ the primal problem is

$$\begin{aligned} \min_{R,\boldsymbol{a},\boldsymbol{\xi}} \quad & R^2 + C\sum\nolimits_{i=1}^{l}\xi_i \\ \text{subject to} \quad & ||\boldsymbol{x}_i - \boldsymbol{a}||^2 \leq R^2 + \xi_i, \\ & \xi_i \geq 0, i = 1, \ldots, l. \end{aligned}$$

The dual form is

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \boldsymbol{\alpha}^T Q\boldsymbol{\alpha} - \sum\nolimits_{i=1}^{l}\alpha_i Q_{ii} \\ \text{subject to} \quad & 0 \leq \alpha_i \leq C, i = 1, \ldots, l, \\ & \boldsymbol{e}^T\boldsymbol{\alpha} = 1, \end{aligned} \tag{3.16}$$

where Q is the same as in (3.15).

To cover (3.14), (3.16), and standard SVM with a linear constraint, from now on we consider the following more general form

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\boldsymbol{\alpha}^T Q\boldsymbol{\alpha} + \boldsymbol{p}^T\boldsymbol{\alpha} \\ \text{subject to} \quad & 0 \leq \alpha_i \leq C_i, i = 1, \ldots, l, \\ & \boldsymbol{y}^T\boldsymbol{\alpha} = \Delta, \end{aligned} \tag{3.17}$$

where $y_i = \pm 1$ and Δ is a constant.

3.2 Larger Working Sets and a Two-level CD Framework When $|B| = 2$ is considered, we explained in Section 2.4 that the linear constraint causes that (α_i, α_j) is easily optimal for the sub-problem (2.2) and cannot be updated. A natural idea to address this issue is by considering a larger B. With more variables, $\boldsymbol{\alpha}_B$ is less likely confined by the linear constraint and can be updated.

However, a larger B leads to the more expensive calculation of $\nabla_B f(\boldsymbol{\alpha})$, where the cost is proportional to $|B|$. Further, with $|B| > 2$, the sub-problem no longer has a closed-form solution and the solution procedure can be expensive. If a standard constrained-optimization package is applied, the cost is easily quadratic or even cubic to the size $|B|$. Instead, we can loosely solve the sub-problem. Here we consider applying several two-variable CD steps because each two-variable sub-problem has a closed-form solution. To this end we have a two-level CD procedure in Algorithm 3. In the inner procedure (the **for** loop in Algorithm 3), we run r CD steps, where r is a pre-specified constant.[4]

Algorithm 3 A two-level CD algorithm to solve problem (3.17), where at each inner CD step a two-variable sub-problem is solved

1: Let $\boldsymbol{\alpha}$ be a feasible point and calculate $Q_{ii}, \forall i$
2: **while** $\boldsymbol{\alpha}$ is not optimal **do**
3: Select a working set B
4: **for** $s = 1, \ldots, r$ **do**
5: Select a two-variable set $B_s \subset B$
6: $(i, j) \leftarrow B_s$
7: Calculate Q_{ij} to form the sub-problem (2.11)
8: Solve the sub-problem.
9: Update (α_i, α_j) and the vector $\boldsymbol{u}$ by (2.5).
10: **end for**
11: **end while**

Algorithm 3 is a general framework, and any its realization relies on the specification of

- the outer working set B in line 3, and
- the inner working set B_s with $|B_s| = 2$ in line 5.

For example, if B is a pair of indices from a random or a cyclic selection and we specify $r = 1$ to have $B_1 = B$, then Algorithm 3 is reduced to Algorithm 2, a standard two-variable CD.

In Sections 3.3 and 3.4, we propose two realizations of Algorithm 3 so every inner two-variable CD step has a better chance to update the corresponding variables.

3.3 A Realization of Two-level CD: Outer Random/Cyclic Selection and Inner Greedy Selection Our first proposed setting is a minor extension of the standard CD in Algorithm 2. We still consider a random/cyclic selection of the (outer) working set B, but slightly enlarge the size to be more than two. For example, in later experiments, we consider $|B| = 4$. The remaining issue is to decide two-variable working sets for inner CD steps. That is, from the set B each time we must select two elements. If a random/cyclic selection is considered, from the discussion in Section 2.4 easily variables cannot be updated because of the linear con-

[4] Two-level CD has been used in, for example, [14, 12], though they consider kernel rather than linear SVM.

straint. Then the same issue of many wasted working sets may occur. Therefore, a greedy selection should be considered. Here we consider the maximal violating pair of the sub-problem of $\boldsymbol{\alpha}_B$ and the detailed procedure is in Algorithm 4. Note that $\boldsymbol{\alpha}_B$ is changed along inner iterations, so at each inner CD step, $\nabla_B f(\boldsymbol{\alpha})$ must be re-calculated by (2.3); see line 8 in Algorithm 4.

The following example illustrates our idea. Consider two randomly selected (outer) working sets with $|B| = 2$ and 4, and set $r = 1$ (i.e., we run only one inner CD step). Assume the following situation occurs.

Outer working set	Inner greedily selected pair
$B = \{i, j\}$	$\{i, j\}$
$B = \{i, j, s, t\}$	$\{j, s\}$

The first row corresponds to the standard two-variable CD, for which because of the linear constraint, easily (α_i, α_j) cannot be moved. In contrast, in the second row for the new setting, because B is enlarged to have four elements, a maximal violating pair (α_j, α_s) more likely can be updated. Of course the cost per outer CD step is increased for calculating $\nabla_B f(\boldsymbol{\alpha})$. However, if the selected pair can lead to the update of $\boldsymbol{\alpha}$, the overall convergence may be faster.

Because $|B|$ is small and the greedy selection is considered for the inner CD procedure, an implementation option is to construct the sub-problem (2.2) first and then solve it by a CD procedure for training kernel SVM. Then $\nabla_B f(\boldsymbol{\alpha})$ is cheaply maintained instead of being calculated at each inner CD step. However, in Algorithm 4 we do not consider such a setting because first, we run only a small number of inner CD steps, and second, the implementation of Algorithm 4 is simpler. A detailed discussion about the two implementations is in Section 6 of supplementary materials.

3.4 A Realization of Two-level CD: Outer Greedy Selection and Inner Random/Cyclic Selection In Section 3.3, we consider a working set B with $|B|$ slightly larger than two so $\boldsymbol{\alpha}_B$ has a better chance to be updated even with the presence of the linear constraint. We mentioned that $|B| = 3$ or 4 can be considered. The set B cannot be further enlarged because of the higher cost on calculating $\nabla_B f(\boldsymbol{\alpha})$ in line 8 of Algorithm 4 for the greedy inner working-set selection.

In this section we propose a viable way of using a large working set B. The procedure is as follows. In the past development of CD methods for kernel SVM, one method to greedily find a relatively larger working set is by selecting the r most violating pairs [7]. We

Algorithm 4 A two-level CD algorithm with an outer random/cyclic selection and an inner greedy selection

1: Let $\boldsymbol{\alpha}$ be a feasible point and calculate $Q_{ii}, \forall i$
2: **while** $\boldsymbol{\alpha}$ is not optimal **do**
3: Randomly/cyclicly select a working set B with $|B|$ slightly larger than 2
4: **for** $s = 1, \ldots, r$ **do**
5: **if** $I_{\text{up}}(\boldsymbol{\alpha}_B) = \emptyset$ or $I_{\text{low}}(\boldsymbol{\alpha}_B) = \emptyset$ **then**
6: **break**
7: **end if**
8: Calculate $\nabla_B f(\boldsymbol{\alpha})$
9: Obtain (i, j) by

$$i \leftarrow \arg\max_t \{-y_t \nabla_t f(\boldsymbol{\alpha}) \mid t \in I_{\text{up}}(\boldsymbol{\alpha}_B)\},$$
$$j \leftarrow \arg\min_t \{-y_t \nabla_t f(\boldsymbol{\alpha}) \mid t \in I_{\text{low}}(\boldsymbol{\alpha}_B)\}.$$

10: **if** (i, j) is not a violating pair **then**
11: **break**
12: **end if**
13: Calculate Q_{ij} to form the sub-problem (2.11)
14: Solve the sub-problem.
15: Update (α_i, α_j) and the vector $\boldsymbol{u}$ by (2.5).
16: **end for**
17: **end while**

sequentially find pairs to form the following working set.

$$B = \{i_1, j_1\} \cup \{i_2, j_2\} \cup \cdots \cup \{i_r, j_r\},$$

where each (i, j) is the maximal violating pair from all the remaining indices. From (2.12), in the end these r pairs satisfy the following relation.

$$\begin{aligned} &\underbrace{-y_{j_1} \nabla_{j_1} f(\boldsymbol{\alpha}) \le \cdots \le -y_{j_r} \nabla_{j_r} f(\boldsymbol{\alpha})}_{j_1, \ldots, j_r \in I_{\text{low}}(\boldsymbol{\alpha})} \\ &< \underbrace{-y_{i_r} \nabla_{i_r} f(\boldsymbol{\alpha}) \le \cdots \le -y_{i_1} \nabla_{i_1} f(\boldsymbol{\alpha})}_{i_1, \ldots, i_r \in I_{\text{up}}(\boldsymbol{\alpha})}. \end{aligned} \tag{3.18}$$

For the inner CD procedure to solve a sub-problem with a large $|B|$, we have mentioned that neither a greedy nor a random/cyclic selection is suitable. Our main idea is that because the outer-level working set B now contains the r most violating pairs, these pairs can be sequentially considered as inner-level working sets. If the vector $\boldsymbol{\alpha}$ is not significantly changed in the inner CD procedure, very likely each pair is still a violating one when it is considered. Thus in most of the r inner CD steps, $\boldsymbol{\alpha}$ can be updated. Details of our procedure are in Algorithm 5. We discuss the convergence of Algorithm 5 in supplementary materials.

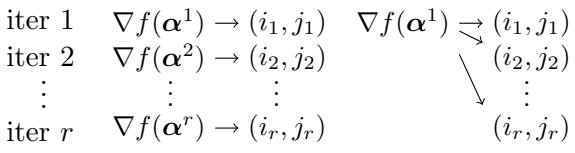
The above interpretation of Algorithm 5 is from the viewpoint of a two-level CD procedure. Instead, we can

Algorithm 5 A two-level CD algorithm with an outer greedy selection and an inner random/cyclic selection

Let $\boldsymbol{\alpha}$ be a feasible point and calculate $Q_{ii}, \forall i$
while $\boldsymbol{\alpha}$ is not optimal **do**
  Calculate the full gradient $\nabla f(\boldsymbol{\alpha})$
  Determine the working set by (3.18)

$$B = \underbrace{\{i_1, j_1\}}_{B_1} \cup \cdots \cup \underbrace{\{i_r, j_r\}}_{B_r}$$

  for $s = 1, \ldots, r$ **do**
    $(i, j) \leftarrow B_s$
    if $I_{\text{up}}(\boldsymbol{\alpha}_{B_s}) = \emptyset$ or $I_{\text{low}}(\boldsymbol{\alpha}_{B_s}) = \emptyset$ **then**
      continue
    end if
    Calculate $\nabla_i f(\boldsymbol{\alpha})$ and $\nabla_j f(\boldsymbol{\alpha})$
    if (i, j) is not a violating pair **then**
      continue
    end if
    Calculate Q_{ij} to form the sub-problem (2.11)
    Solve the sub-problem.
    Update (α_i, α_j) and the vector $\boldsymbol{u}$ by (2.5).
  end for
end while



(a) Two-variable CD by greedy working-set selections (b) Algorithm 5

Figure 1: An illustration showing how Algorithm 5 saves the cost by selecting several working sets per gradient calculation.

consider Algorithm 5 as a particular implementation of two-variable CD. In Figure 1(a), we illustrate a greedy setting of expensively calculating $\nabla f(\boldsymbol{\alpha})$ at each step to select just two elements as the working set. To reduce the cost, in Figure 1(b), which corresponds to Algorithm 5, after having the gradient we select several pairs and use them in the next several CD steps. Thus the gradient is calculated less frequently.

For the cost of getting the r most violating pairs, we use a max heap to maintain the r smallest gradients. The cost is $O(l \log r)$ and the procedure of finding the r largest gradients is similar. If r is chosen to be a larger number (e.g., $l/10$ in our experiment) and if n is not small, then the $O(l \log r)$ cost in finding the working set is usually smaller than the $O(rn)$ cost in constructing and solving the r two-variable sub-problems.

4 Experiments

In this section we compare the proposed methods with Algorithm 2, a standard two-variable CD via a random/cyclic working-set selection.

4.1 Data and Settings for Experiments We consider some binary classification sets though label information is not used for training. These sets are the same as those used in [2]. Detailed data statistics can be found in their supplementary materials. For one-class SVM, we consider $\nu = 0.1$. For SVDD, $C = 1/(\nu l)$ is used. Experiments of using other ν and C values are given in supplement materials though results are similar. For the initial $\boldsymbol{\alpha}$, we follow the setting in LIBSVM [1] for kernel one-class SVM and have

$$\boldsymbol{\alpha} = \begin{cases} [\underbrace{\frac{1}{\nu l}, \ldots, \frac{1}{\nu l}}_{\lfloor \nu l \rfloor}, 1 - \frac{\lfloor \nu l \rfloor}{\nu l}, 0, \ldots, 0]^T & \text{one-class SVM,} \\ [\underbrace{C, \ldots, C}_{\lfloor 1/C \rfloor}, 1 - \lfloor \frac{1}{C} \rfloor C, 0, \ldots, 0]^T & \text{SVDD.} \end{cases}$$

To check the convergence speed, we draw figures to show the relationship between

- the cumulative number of $O(n)$ operations, and
- the relative difference to the objective function value at an optimal solution $\boldsymbol{\alpha}^*$.[5]

$$\frac{|f(\boldsymbol{\alpha}) - f(\boldsymbol{\alpha}^*)|}{|f(\boldsymbol{\alpha}^*)|}. \tag{4.19}$$

For one-class SVM, $|f(\boldsymbol{\alpha}^*)|$ is often close to zero so we follow LIBSVM [1] to solve a scaled problem of (3.14).

$$\begin{aligned} \min_{\bar{\boldsymbol{\alpha}}} \quad & \frac{1}{2} \bar{\boldsymbol{\alpha}}^T Q \bar{\boldsymbol{\alpha}} \\ \text{subject to} \quad & 0 \le \bar{\alpha}_i \le 1, i = 1, \ldots, l, \\ & \boldsymbol{e}^T \bar{\boldsymbol{\alpha}} = \nu l. \end{aligned}$$

We consider the number of $O(n)$ operations because it is proportional to the cost of our algorithms. In supplementary materials we show figures to confirm that checking the number of $O(n)$ operations is almost the same as checking the running time.

4.2 Results and Analysis We compare the following approaches.

- cyclic-2cd: This is a standard two-variable CD in Algorithm 2 via a random/cyclic working-set selection. Following the suggestion in [2], at each cycle we permute all instance indices $1, \ldots, l$ to $\pi(1), \ldots, \pi(l)$, and sequentially consider the following working sets:

$$(\pi(1), \pi(2)), (\pi(3), \pi(4)), \ldots, (\pi(l-1), \pi(l)). \tag{4.20}$$

[5] In practice $\boldsymbol{\alpha}^*$ is not available, so an approximation is obtained by running enough iterations of the algorithm.

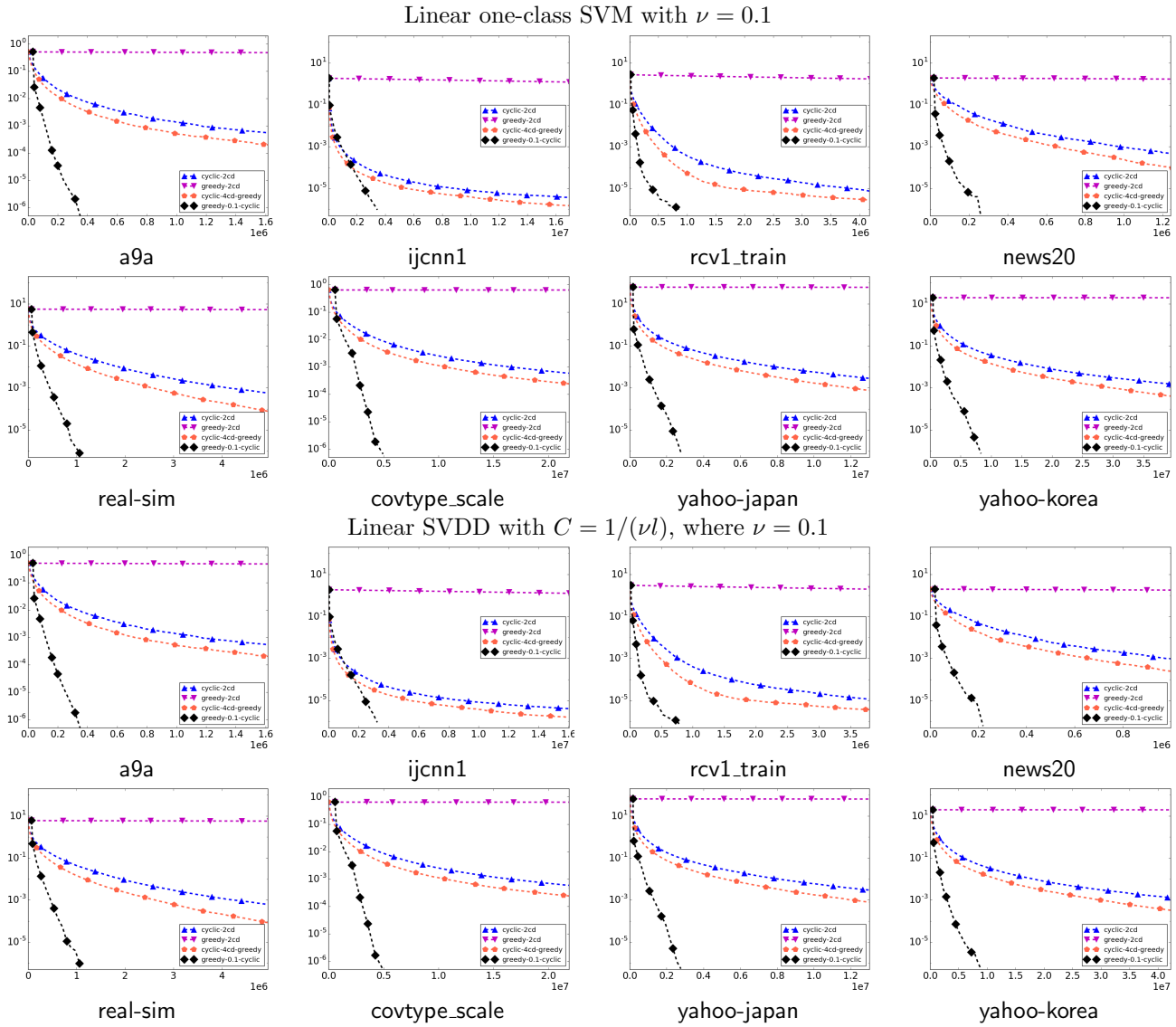


Figure 2: A comparison on the convergence speed of different methods (upper: linear one-class SVM, lower: linear SVDD). The x-axis is the cumulative number of $O(n)$ operations, while the y-axis is the relative difference to the optimal function value defined in (4.19).

- greedy-2cd: This is similar to cyclic-2cd, though for the working-set selection we choose a maximal violating pair via (2.13). This setting corresponds to the way of training kernel one-class SVM or SVDD.
- cyclic-$|B|$cd-greedy: This is Algorithm 4 and $|B|$ indicates the working-set size. As discussed in Section 3.3, $|B|$ should be only slightly larger than two. Here we consider $|B| = 4$. The word "cyclic" in the name of this approach indicates that our working-set selection is extended from (4.20) to have

$$(\pi(1), \ldots, \pi(|B|)), (\pi(|B|+1), \ldots, \pi(2|B|)), \ldots.$$

The word "greedy" indicates that in the inner CD procedure, each time we greedily select a maximum violating pair from B for update. Regarding the number of inner CD steps, we choose to run only one step (i.e., $r = 1$).

- greedy-R-cyclic: This is Algorithm 5. The value $R \in (0, 1]$ indicates that $r = Rl$ violating pairs are selected as the working set B. We consider $R = 0.1$ for our experiments. The word "cyclic" indicates that the inner working sets are sequentially selected via the way described in (4.20).

Results for linear one-class SVM and linear SVDD are respectively presented in Figure 2. We can make the following observations.

- For both one-class SVM and SVDD, greedy-2cd is much worse than the other three methods. While

a greedy selection guarantees that each step can successfully update $\boldsymbol{\alpha}$, this property does not compensate the much higher cost of each CD step.

- For both one-class SVM and SVDD, cyclic-4cd-greedy performs better than cyclic-2cd. Even though the cost per CD step is higher, because of the larger working set B, more successful updates of $\boldsymbol{\alpha}$ lead to faster convergence.
- For problems such as covtype_scale, greedy-0.1-cyclic converges slower at the very begining of the optimization process. The reason is that at the first outer CD step, $\nabla f(\boldsymbol{\alpha})$ is calculated in $O(ln)$ cost; see (2.3). Only after that the first (α_i, α_j) can possibly be updated. The same result can be found for greedy-2cd though it performs poorly not only at the begining. In contrast, cyclic-2cd and cyclic-4cd-greedy have attempted to update many pairs within the same amount of operations.
- Despite the slower start at the very begining, greedy-0.1-cyclic has the fastest overall convergence. The reason should be that when $\boldsymbol{\alpha}$ is close to an optimal solution, Figure 1 indicates that Algorithm 5 is approaching a true greedy setting. Thus a faster final convergence is achieved.

In summary, greedy-0.1-cyclic is our recommended setting to train linear one-class SVM and SVDD.

5 Discussion and Conclusions

In this work, we have successfully developed efficient and effective coordinate descent algorithms for linear one-class SVM and SVDD. The proposed algorithms maintain the simplicity of existing coordinate descent algorithms for linear SVM for classification, but are much more efficient for one-class scenarios.

References

[1] C.-C. Chang and C.-J. Lin, *LIBSVM: A library for support vector machines*, ACM TIST, 2 (2011), pp. 27:1–27:27.

[2] C.-C. Chiu, P.-Y. Lin, and C.-J. Lin, *Two-variable block dual coordinate descent methods for large-scale linear support vector machines*, in SDM, 2020.

[3] S. M. Erfani, S. Rajasegarar, S. Karunasekera, and C. Leckie, *High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning*, Pattern Recognition, 58 (2016), pp. 121–134.

[4] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, *LIBLINEAR: a library for large linear classification*, JMLR, 9 (2008), pp. 1871–1874.

[5] R.-E. Fan, P.-H. Chen, and C.-J. Lin, *Working set selection using second order information for training SVM*, JMLR, 6 (2005), pp. 1889–1918.

[6] C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S. S. Keerthi, and S. Sundararajan, *A dual coordinate descent method for large-scale linear SVM*, in ICML, 2008.

[7] T. Joachims, *Making large-scale SVM learning practical*, in Advances in Kernel Methods - Support Vector Learning, MIT Press, 1998.

[8] T. Joachims, *Training linear SVMs in linear time*, in KDD, 2006.

[9] S. S. Keerthi and D. DeCoste, *A modified finite Newton method for fast solution of large scale linear SVMs*, JMLR, 6 (2005), pp. 341–361.

[10] S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy, *Improvements to SMO algorithm for SVM regression*, IEEE TNN, 11 (2000), pp. 1188–1193.

[11] C.-J. Lin, R. C. Weng, and S. S. Keerthi, *Trust region Newton method for large-scale logistic regression*, JMLR, 9 (2008), pp. 627–650.

[12] F. Pérez-Cruz, A. R. Figueiras-Vidal, and A. Artés-Rodríguez, *Double chunking for solving SVMs for very large datasets*, in Proceedings of Learning 2004, Spain, 2004.

[13] J. C. Platt, *Fast training of support vector machines using sequential minimal optimization*, in Advances in Kernel Methods - Support Vector Learning, Cambridge, MA, 1998, MIT Press.

[14] S. Rüping, *mySVM - another one of those support vector machines*, 2000. Software available at `http://www-ai.cs.uni-dortmund.de/SOFTWARE/MYSVM/`.

[15] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, *Estimating the support of a high-dimensional distribution*, Neural Comput., 13 (2001), pp. 1443–1471.

[16] I. Steinwart, D. Hush, and C. Scovel, *Training SVMs without offset*, JMLR, 12 (2011), pp. 141–202.

[17] D. M. J. Tax and R. P. W. Duin, *Support vector data description*, MLJ, 54 (2004), pp. 45–66.