Two-variable Block Dual Coordinate Descent Methods for Large-scale Linear Support Vector Machines

Chi-Cheng Chiu  
National Taiwan University  
r03922089@ntu.edu.tw

Chih-Jen Lin  
National Taiwan University  
cjlin@csie.ntu.edu.tw

Abstract
Coordinate descent (CD) methods have been a state-of-the-art technique for training large-scale linear SVM. The most used setting is to solve the dual problem of an SVM formulation without the bias term, for which the CD procedure of updating one variable at a time is very simple and easy to implement. In this work, we extend the one-variable setting to use two variables at each CD step. The extension, while looks simple, is not trivial. Some complicated derivations are needed to get a simple CD procedure. Our resulting algorithm is generally competitive with one-variable CD and is superior for difficult problems. We further discuss the two-variable CD for the standard SVM formulation with a bias term. The analysis shows that CD methods are less effective for this SVM formulation, a situation very different from that of kernel SVM. Thus the success of simple one-variable CD in the past decade is not a coincidence. Some design choices such as the SVM formulation considered help to make it computationally efficient. Overall this work sheds many new insights on CD methods for training linear SVM.

Keywords: block coordinate descent, linear support vector machines

1. Introduction

For large and sparse data, linear support vector machines (SVM) have been effective to achieve competitive test accuracy. To train large-scale linear SVM, coordinate descent (CD) methods to solve the dual problem are a state-of-the-art approach. The basic idea of a CD method is to update one variable at a time while fixing others. For a more general setting of updating a subset of variables each time, the method is referred to as block CD.

CD have been a classical optimization approach which can be traced back to, for example, Hildreth (1957) for unconstrained quadratic minimization. However, for linear SVM this type of techniques becomes popular only after Hsieh et al. (2008). They point out that by the special structure of the dual problem of linear SVM, each CD update can be cheaply conducted. Since then, CD has been widely adopted for linear SVM and many subsequent studies including investigation on theoretical properties have been available (e.g., Shalev-Shwartz and Zhang 2013). Another hallmark of CD for linear SVM by Hsieh et al. (2008) is that at each step a simple one-variable sub-problem is minimized and a closed-form solution is available. Thus besides the efficiency, a CD implementation for linear SVM is extremely simple.

While CD for linear SVM has been highly successful, we would like to ask the following questions.

• In many optimization applications, block CD of using more than one variable are used. However, in training linear SVM, block CD is seldom used. One important
reason is that by using the smallest possible number of variables each time (i.e., one), the optimization sub-problems can be easily constructed and solved. In this work, we investigate that if the block size is slightly increased from one to two, can a simple and effective implementation still be possible?

- Hsieh et al. (2008) consider an optimization formulation slightly different from the standard SVM, in which a bias term is included in the decision function. If the bias term is considered, the dual problem contains a linear constraint and each CD step must update at least two variables. In contrast, Hsieh et al. (2008) do not consider a bias term in the decision function, so a simple CD of using one variable is applicable. However, some have criticized the use of the SVM formulation without a bias term. Therefore, an important question is whether the setting in Hsieh et al. (2008) is a must or not. If an effective two-variable block CD can be developed, then probably a bias term can always be considered.

Our results shed some new insights on coordinate descent methods for linear SVM. Main findings can be summarized as follows.

- If the bias term is not considered as in Hsieh et al. (2008), although the dual problem is in a simple quadratic bound-constrained form, developing a two-variable block CD is not straightforward. After some complicated derivations, we are able to have a simple and elegant solution procedure. The resulting two-variable block CD is in general competitive with one-variable CD by Hsieh et al. (2008), and is superior on difficult problems. Details are in Section 3.

- We explain conceptually and experimentally that CD for linear SVM with the bias term is much slower than without. Therefore, the decision in Hsieh et al. (2008) to not have the bias term is very essential for the success of CD for linear SVM. Details are in Section 4.

This finding has some important practical implications. Some methods such as one-class SVM (Schölkopf et al., 2001) or SVDD (Tax and Duin, 2004), the dual problem must have a linear constraint. Thus for linear one-class SVM or SVDD, new CD algorithms must be designed.

A related work to ours is Steinwart et al. (2011), in which two-block CD for kernel SVM is well studied. Ours for linear SVM differs from or improve upon them in the following aspects.

- The two-variable sub-problem at each CD step is the same for kernel and linear SVM. We develop a simpler algorithm than the one used in Steinwart et al. (2011).

- We show that the difference between CD for linear SVM with and without a bias term is much more dramatic than kernel SVM.

Programs used for experiments in this paper can be found at [http://www.csie.ntu.edu.tw/~cjlin/papers/2var_cd](http://www.csie.ntu.edu.tw/~cjlin/papers/2var_cd). Supplementary materials are available at the same page.

1. For example, https://github.com/scikit-learn/scikit-learn/pull/4738
2. Coordinate Descent Method for SVM Dual Problems

In this section we introduce SVM optimization problem and explain how block coordinate descent methods can be effectively applied to train linear SVM.

2.1 Dual SVM Optimization Problem

For a training set of instance-label pairs \((x_i, y_i), i = 1, \ldots, l\), where \(y_i \in \{-1, +1\}\) and \(x_i \in \mathbb{R}^n\), linear SVM solves the following optimization problem.

\[
\min_w \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi(w; x_i, y_i),
\]

where \(\xi(w; x_i, y_i)\) is a loss function and \(C \in (0, \infty)\) is a penalty parameter. The following two loss functions are commonly considered for SVM.

\[
\xi(w; x, y) \equiv \begin{cases} 
\max(0, 1 - yw^T x) & l_1 \text{ loss,} \\
\max(0, 1 - yw^T x)^2 & l_2 \text{ loss.}
\end{cases}
\]

Following the notation in Chiang et al. (2016), if (1) is referred to as the primal problem, then the dual optimization problem is

\[
\min_{\alpha} \frac{1}{2} \alpha^T Q \alpha - e^T \alpha \quad \text{subject to} \quad 0 \leq \alpha_i \leq C_i, \forall i,
\]

where

\[
e = [1, \ldots, 1]^T, \quad C_i = \begin{cases} 
C & l_1 \text{ loss,} \\
\infty & l_2 \text{ loss,}
\end{cases}
\]

and

\[
Q_{ij} = \begin{cases} 
y_i y_j x_i^T x_j + \frac{1}{C_i} & l_2 \text{ loss and } i = j, \\
y_i y_j x_i^T x_j & \text{otherwise.}
\end{cases}
\]

Note that the definition of \(C_i\) can be extended to have a weighted regularization parameter for each instance.

Interestingly, (1) is not the standard SVM formulation, where a bias term \(b\) is included in the model and the loss function becomes

\[
\xi(w; b; x, y) \equiv \begin{cases} 
\max(0, 1 - y(w^T x + b)) & l_1 \text{ loss,} \\
\max(0, 1 - y(w^T x + b)^2 & l_2 \text{ loss.}
\end{cases}
\]

In particular, (5) is traditionally used for kernel SVM, which is a generalization of linear SVM by mapping each \(x_i\) to a higher dimension space:

\[x_i \rightarrow \phi(x_i).\]
The inner product $x_i^T x_j$ in (4) is changed to

$$K(x_i, x_j) = \phi(x_i)^T \phi(x_j),$$

where $K(x_i, x_j)$ is called the kernel function (Boser et al., 1992).

If the bias term $b$ is considered, the dual problem (Cortes and Vapnik, 1995) has an additional linear constraint.

$$\min_{\alpha} \frac{1}{2} \alpha^T Q \alpha - e^T \alpha$$

subject to $0 \leq \alpha_i \leq C_i, i = 1, \ldots, l,$

$$y^T \alpha = 0.$$

The linear constraint may make the dual problem (6) harder to solve than (3). To remove the linear constraint, one can either consider a model without the bias term or have one more constant feature:

$$w \leftarrow \begin{bmatrix} w \\ b \end{bmatrix}, \quad x_i \leftarrow \begin{bmatrix} x_i \\ 1 \end{bmatrix}$$

(7)

The bias term is then regularized, so the optimization problem goes back to be in a form of (1). Some early studies that proposed regularizing the bias term include, for example, Mangasarian and Musicant (1999) and Friess et al. (1998).

While conceptually a dual problem without a linear constraint is easier, for linear SVM no serious study has been made to confirm this conjecture. On the other hand, for kernel SVM, some past works (e.g., Steinwart et al. 2011) have compared CD methods for (6) and (3). Therefore, one of the goals of this work is to fill the gap by studying if the state-of-the-art optimization method performs similarly or not for linear SVM with/without the linear constraint.

### 2.2 Block Coordinate Descent Methods for Linear SVM

To cover both dual optimization problems in (3) and (6), we consider a more general form of the optimization problem.

$$\min_{\alpha} f(\alpha)$$

subject to $\alpha \in F,$

where $F$ is the feasible set and

$$f(\alpha) = \frac{1}{2} \alpha^T Q \alpha + p^T \alpha.$$  (9)

The basic idea of a block coordinate descent method to solve (8) is that at the current $\alpha$, we change elements in a working set $B$ while fix other components. Specifically, if $N \equiv \{1, \ldots, l\} \setminus B$ and $d = [d_B]^T$,

then

$$f\left(\begin{bmatrix} \alpha_B \\ \alpha_N \end{bmatrix} + [d_B]^T \right) = \frac{1}{2} d_B^T Q_B B d_B + \nabla_B f(\alpha)^T d_B + \text{constant},$$
Algorithm 1 A framework of block CD methods

1: Let $\alpha$ be a feasible point
2: while $\alpha$ is not optimal do
3:   Select a working set $B$
4:   Solve the sub-problem (10)
5:   Update $\alpha$ by $\alpha_B \leftarrow \alpha_B + d_B$
6: end while

where $d_B$ is the sub-vector to be used to change $\alpha$. We then minimize the following sub-problem with the variable $d_B$.

$$
\min_{d_B} \frac{1}{2} d_B^T Q_{BB} d_B + \nabla_B f(\alpha)^T d_B
$$

subject to $$
\begin{bmatrix}
\alpha_B \\
\alpha_N
\end{bmatrix} +
\begin{bmatrix}
d_B \\
0
\end{bmatrix} \in F.
$$ (10)

After solving the above sub-problem, we update $\alpha$ by

$$
\alpha_B \leftarrow \alpha_B + d_B.
$$ (11)

A summary of the procedure is in Algorithm 1. Throughout this work, we call the process of finishing an update in (11) a CD step. From (10), important tasks at each CD step are

- constructing the gradient vector $\nabla_B(\alpha)$ in (10),
- selecting the working set $B$, and
- solving the sub-problem.

We discuss some existing developments in Sections 2.3 and 2.4.

2.3 Gradient Calculation for Linear and Kernel SVM

To construct the sub-problem (10), $Q_{BB}$ and $\nabla f(\alpha)$ must be calculated. We show that the situations between kernel and linear are very different. Form (9),

$$
\nabla_B f(\alpha) = Q_{B,:} \alpha + p_B.
$$ (12)

Thus calculating $Q_{B,:}$ is the main computational cost. If kernel is used,

$$
Q_{i,:} \alpha = \sum_{j=1}^{l} Q_{ij} \alpha_j = \sum_{j=1}^{l} y_i y_j K(x_i, x_j) \alpha_j.
$$

Assume each $K(x_i, x_j)$ costs $O(n)$ operations, where $n$ is the number of features. Then (12) requires $O(ln)$ cost. Note that in calculating $Q_{B,:}$, the $Q_{B,B}$ sub-matrix needed in the quadratic form has also been obtained.
Because $Q_B$ has been calculated, if the current $\nabla f(\alpha)$ is available we can easily update it to $\nabla f(\alpha + d)$ by

$$\nabla f(\alpha + d) = \nabla f(\alpha) + Q_{:,B}d_B. \tag{13}$$

In practice, (13) rather than (12) is often implemented. The reason is that with $\alpha = 0$ feasible for (6) and (3), it can be the initial point of the CD procedure. From $\nabla f(0) = -e$, we can use (13) to maintain the gradient $\nabla f(\alpha)$ throughout the optimization process.

For linear SVM, Hsieh et al. (2008) propose a technique to significantly reduce the cost of constructing the sub-problem from $O(ln)$ to $O(|B| n)$. Specifically, they notice that if

$$u \equiv \sum_{j=1}^{l} \alpha_j y_j x_j \tag{14}$$

is available, then

$$Q_{:,\alpha} = \sum_{j=1}^{l} y_i y_j x_i^T x_j \alpha_j = y_i u^T x_i \tag{15}$$

is a simple inner product with $O(n)$ cost. Therefore,

$$\nabla_i f(\alpha) = \begin{cases} y_i u^T x_i - 1 & l1 \text{ loss}, \\ y_i u^T x_i - 1 + \frac{\alpha_i}{2 \epsilon} & l2 \text{ loss}. \end{cases}$$

To maintain $u$ they consider

$$u \leftarrow u + \sum_{j:j \in B} d_j y_j x_j, \tag{16}$$

which costs $O(|B| n)$. Eventually the vector $u$ converges to the primal optimal solution $w$. However, this technique is not applicable to kernel SVM because

$$u = \sum_{j=1}^{l} \alpha_j y_j \phi(x_j)$$

may be an infinite dimensional vector.

We have shown that the cost for constructing (10) is much cheaper than kernel. On the other hand, from (13), $\nabla f(\alpha)$ can be maintained for kernel but not for linear. This situation causes differences in the selection of the working set $B$. Details will be given in Section 2.4

2.4 Working Set Selection for Block CD Methods

Selecting the working set $B$ is a very essential step because a good selection scheme can lead to faster convergence. We will show that the availability of $\nabla f(\alpha)$ or not leads to a significant difference on the working set selection of kernel and linear settings.
We begin with discussing the working set selection for linear SVM without the bias term. The feasible set of (3) is
\[ F = \{ \alpha \mid 0 \leq \alpha_i \leq C, \forall i = 1, \ldots, l \}. \]

Hsieh et al. (2008) consider the simplest setting of using cyclic coordinate descent, so each time
\[ B = \{ i \} \]
is chosen. Clearly the sub-problem (10) can be easily solved by
\[ d = \max(-\alpha_i, \min(C - \alpha_i, -\nabla_i f(\alpha) Q_{ii})). \]

The setting of cyclically updating \( \alpha_1, \ldots, \alpha_l \) is often referred to as the Gauss-Seidel scheme in the literature of CD methods (Tseng and Yun, 2009). However, Hsieh et al. (2008) pointed out that in practice using a random permutation at each cycle leads to faster convergence:
\[ \alpha_{\pi(1)}, \alpha_{\pi(2)}, \ldots, \alpha_{\pi(l)}, \]
where \( \pi(1), \ldots, \pi(l) \) is a permutation of 1, \ldots, l. Alternatively, we may randomly select an index at each CD step to achieve the randomness of the working-set selection. Some have even studied the setting of selecting the index by an adaptive probability distribution (e.g., Glasmachers and Dogan, 2013; Csiba et al., 2015 and references therein). In our extension to select two variables for update, we will show that the randomness is still important, but some new issues may occur.

Subsequently, we refer to the above setting as one-variable CD. A summary of the procedure is in Algorithm 2.

If a bias term is considered, the feasible set of (6) is
\[ F = \{ \alpha \mid y^T \alpha = 0, 0 \leq \alpha_i \leq C_i, \forall i = 1, \ldots, l \}. \]

To have that \( [\alpha_B^+ \alpha_N^-] + [d_B^- 0] \) is feasible, \( d_B \) must satisfies
\[ y_B^T \alpha_B = -y_N^T \alpha_N, \]
\[ 0 \leq \alpha_i + d_i \leq C_i, \forall i \in B. \]

Because of the linear constraint, \( B \) must contain at least two elements. Therefore, regardless of linear or kernel SVM, the above one-variable CD cannot be used.

Next we discuss the working set selection for kernel SVM and show the difference from linear SVM. While a random selection is possible, most existing works (e.g., Joachims, 1998; Platt, 1998; Fan et al., 2005) consider a greedy selection of using the gradient information. Specifically, because the trick (14)-(16) for linear SVM is not applicable, (13) must be conducted and the whole gradient is available. This type of working-set selections is often referred to as the Gauss-Southwell setting in the literature of CD methods (Tseng and Yun, 2009). We omit further details because our focus here is on linear SVM. Interested readers can check Section 4.1 of Hsieh et al. (2008).

For linear SVM, the greedy setting is not considered because calculating the gradient causes the cost of each CD step to become \( l \) times. Although the convergence may be faster, the reduction on the number of CD steps does not compensate the much higher cost per iteration.
Algorithm 2 A one-variable CD by Hsieh et al. (2008) for linear SVM

1: Input: Specify a feasible $\alpha$
2: calculate $u = \sum_j y_j \alpha_j x_j$
3: while $\alpha$ is not optimal do
4: Obtain the permuted indices $\{\pi(1), \pi(2), \ldots, \pi(l)\}$
5: for $j = 1, \ldots, l$ do
6: $i \leftarrow \pi(j)$
7: $G \leftarrow \begin{cases} y_i u^T x_i - 1 & \text{l1 loss} \\ y_i u^T x_i - 1 + \frac{\alpha_i}{2C_i} & \text{l2 loss} \end{cases}$
8: $d = \max(-\alpha_i, \min(C_i - \alpha_i, -G/Q_{ii}))$
9: $\alpha_i \leftarrow \alpha_i + d$
10: $u \leftarrow u + dy_i x_i$
11: end for
12: end while
13: $w \leftarrow u$
14: output: $(w, \alpha)$ as approximate primal and dual solutions.

3. Two-variable Block Coordinate Descent Method

In this section, we extend Algorithm 2 to update two variables at each CD step. We consider the linear SVM problem without the equality constraint; see (3) in Section 2.1.

3.1 The Framework of the Algorithm

To use two rather than one element in each coordinate descent step, we consider a working set

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

and solve the following two-variable sub-problem

$$\min_{d_i, d_j} \frac{1}{2} \begin{bmatrix} d_i & d_j \end{bmatrix} \begin{bmatrix} Q_{ii} & Q_{ij} \\ Q_{ij} & Q_{jj} \end{bmatrix} \begin{bmatrix} d_i \\ d_j \end{bmatrix} + \begin{bmatrix} \nabla_i f(\alpha) \\ \nabla_j f(\alpha) \end{bmatrix} \begin{bmatrix} d_i \\ d_j \end{bmatrix}$$

subject to

$$0 \leq \alpha_i + d_i \leq C_i,$$

$$0 \leq \alpha_j + d_j \leq C_j.$$  \(18\)

For the working-set selection, in Section 2 we mentioned that for the one-variable scenario Hsieh et al. (2008) consider a permuted sequence at the beginning of each cycle

$$\pi(1), \ldots, \pi(l),$$

and cyclically update variables. Now we must choose two variables at a time, so a direct extension is to cyclically consider

$$\pi(1, 2), \ldots, \pi(1, l), \pi(2, 1), \ldots, \pi(l - 1, l),$$  \(19\)
which is a permutation of
\[(1, 2), \ldots, (1, l), (2, 1), \ldots, (l - 1, l).\]
Unfortunately, this setting is not practical because the \(O(l^2)\) storage to store the sequence is prohibitive for large problems. One feasible setting is to permute \(\{1, \ldots, l\}\) first, and for each \(\pi(i)\), another permutation \(\pi'\) of \(\{1, \ldots, l\}\) is generated. The sequence considered is therefore
\[(\pi(1), \pi'(1)), \ldots, (\pi(l), \pi'(1)), (\pi(2), \pi'(1)), \ldots, (\pi(l), \pi'(l)),\] (20)
though some pairs with \(\pi(i) = \pi'(j)\) must be removed. However, a concern is that such a sequence may not be random enough because in every \(l\) pairs, the first index is the same.
Another feasible setting is to randomly select every \(\{i, j\}\) rather than permute indices. The selection scheme turns out to be very important as shown in a thorough experiment in Section 3.4.1. Some interesting findings will be presented, and we conclude that a random selection is more suitable for two-variable CD.

### 3.2 Solving Two-variable Sub-problems

We detailedly discuss how to solve the two-variable sub-problem (18). For the solution procedure, we require that
\[Q_{ii} > 0 \text{ and } Q_{jj} > 0.\] (21)
If (21) does not hold, either \(Q_{ii} = 0\) or \(Q_{jj} = 0\). Assume \(Q_{ii} = 0\). Then
\[Q_{ii} = \|x_i\|^2 = 0\]
implies that \(x_i = 0\). From
\[\nabla_i f(\alpha) = \sum_{j=1}^{l} y_i y_j x_i^T x_j \alpha_j - 1\]
\[= -1 \leq 0, \quad \forall \alpha,\]
by the optimality condition, \(\alpha_i = C_i\) is optimal for the dual problem (3). We can identify these zero instances before running the CD algorithm. Thus throughout this section we assume that (21) always holds.
For easy understanding, we rewrite (18) to a more general two-variable optimization problem:
\[
\min_{d_1, d_2} \frac{1}{2} \begin{bmatrix} d_1 & d_2 \end{bmatrix} \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \end{bmatrix} + \begin{bmatrix} p_1 & p_2 \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \end{bmatrix}
\]
subject to \(L_1 \leq d_1 \leq U_1,\)
\(L_1 \leq d_2 \leq U_2,\)
where \(L_1, U_1, L_2, U_2 \in \mathbb{R}\). Note that (18) is a special case of (22) by
\[U_1 = C_i - \alpha_i, \quad L_1 = -\alpha_i \text{ and } p_1 = \nabla_i f(\alpha).\] (23)
The situation for \(U_2, L_2\) and \(p_2\) is similar. We now split the discussion to two parts according to the Hessian of the objective function is positive definite or only positive semi-definite.
3.2.1 Hessian is Positive Definite

The two-variable sub-problem (22) is the same as the one solved in Steinwart et al. (2011), which studies two-variable CD for kernel SVM. We briefly describe their solution procedure before ours. They begin with considering (22) without constraints. If
\[
\begin{bmatrix}
Q_{11} & Q_{12} \\
Q_{12} & Q_{22}
\end{bmatrix}
\]
is positive definite, then the unconstrained problem has a unique solution \([d^*_1, d^*_2]\), which satisfies
\[
\begin{bmatrix}
Q_{11} & Q_{12} \\
Q_{12} & Q_{22}
\end{bmatrix}
\begin{bmatrix}
d^*_1 \\
d^*_2
\end{bmatrix}
+ \begin{bmatrix}
p_1 \\
p_2
\end{bmatrix} = \begin{bmatrix}
0 \\
0
\end{bmatrix}.
\]

The solution of (24) is
\[
d^*_1 = \frac{-Q_{22}p_1 + Q_{12}p_2}{Q_{11}Q_{22} - Q_{12}^2}, \quad d^*_2 = \frac{-Q_{11}p_2 + Q_{12}p_1}{Q_{11}Q_{22} - Q_{12}^2}.
\]

Denote \(\hat{f}(d_1, d_2)\) as the objective function of (22). If \(d^*_1 > U_1\) and \(d^*_2 \in [L_2, U_2]\), an optimal solution must be on the line of \(d_1 = U_1\). A conceptual proof is in Figure 1a: if a solution \(\hat{d}\) is not on this line, then the line segment connecting \(\hat{d}\) and \(d^*\) leads to a point on \(d_1 = U_1\) with a smaller function value because of the strict convexity of the function \(\hat{f}(d_1, d_2)\). Thus by fixing \(d_1 = U_1\) one can solve a one-variable optimization problem to get the optimal solution \(\bar{d}_2\). Specifically, the optimal solution is
\[
\bar{d}_1 = P[d^*_1],
\]
\[
\bar{d}_2 = \arg \min_{d_2 \in [L_2, U_2]} \frac{1}{2} \begin{bmatrix} d_1 & d_2 \end{bmatrix} \begin{bmatrix} Q_{11} & Q_{12} \\
Q_{12} & Q_{22}
\end{bmatrix} \begin{bmatrix} d_1 \\
Q_{12}
\end{bmatrix} + \begin{bmatrix} p_1 & p_2 \end{bmatrix} \begin{bmatrix} d_1 \\
Q_{12}
\end{bmatrix},
\]
where
\[
P[d_i] = \min(U_i, \max(L_i, d_i)), \quad \forall i = 1, 2,
\]
is a projection operation. However, if
\[
d^*_1 > U_1, \quad d^*_2 > U_2\]
the above argument can only imply that the solution must be on either \(d_1 = U_1\) or \(d_2 = U_2\); see the illustration in Figure 1b. Thus Steinwart et al. (2011) propose solving two one-dimensional problems where one is by fixing \(d_1 = U_1\) and the other is by fixing \(d_2 = U_2\). Then they compare two objective values to decide the solution. However, such a setting may not be ideal because of the following reasons.
Two-variable Block Dual Coordinate Descent Methods for Linear SVMs

\[ \begin{align*}
    d_1 &= U_1 \\
    d_2 &= L_2
\end{align*} \]

(a) If \( d_1^* \geq U_1 \) and \( d_2^* \in [L_2, U_2] \), then the solution is on \( d_1 = U_1 \).

\[ \begin{align*}
    d_1 &= L_1 \\
    d_2 &= U_2
\end{align*} \]

(b) If \( d_1^* \geq U_1 \) and \( d_2^* \geq U_2 \), the solution is on either \( d_1 = U_1 \) or \( d_2 = U_2 \).

Figure 1: Illustrations of different situations of \( d^* \), the solution without constraints.

\[ \begin{align*}
    d_1 &= U_1 \\
    d_2 &= L_2
\end{align*} \]

(a) \( \nabla_1 \hat{f}(P[d_1^*], P[d_2^*]) \leq 0 \) (see the arrow in the figure) and the solution is on \( d_2 = U_2 \).

\[ \begin{align*}
    d_1 &= U_1 \\
    d_2 &= L_2
\end{align*} \]

(b) \( \nabla_2 \hat{f}(P[d_1^*], P[d_2^*]) \leq 0 \) (see the arrow in the figure) and the solution is on \( d_1 = U_1 \).

Figure 2: We can check the optimality condition at the point \( (P[d_1^*], P[d_2^*]) \) to decide which line the optimal solution is at.

1. The implementation is complicated. From Figure 1 eight out-of-boundary cases must be considered.

2. For the situation in Figure 1b the cost is high because we must solve two one-variable sub-problems and compare their objective values.

We notice that for the situation in Figure 1b it is possible to use the gradient information for deciding which boundary line the optimal solution is at. Specifically, in Figure 2 we consider two cases. For the first one in Figure 2a the optimal solution of (22) is on the line

\[ d_2 = U_2, \]
while for the second, it is on
\[ d_1 = U_1. \]

We notice that at the point
\[ (P[d_1^*], P[d_2^*]) = (U_1, U_2), \]
the two cases respectively have
\[ \nabla^2 \hat{f}(P[d_1^*], P[d_2^*]) \leq 0 \quad \text{and} \quad \nabla_1 \hat{f}(P[d_1^*], P[d_2^*]) \leq 0. \]  
\[ (28) \]

Let us look at the case of Figure 2b in detail. With
\[ P[d_1^*] = U_1 \quad \text{and} \quad P[d_2^*] = U_2, \]
the second inequality in (28) means that on the line of
\[ d_2 = U_2, \]
we must increase \( P[d_1^*] = U_1 \) to a larger value (i.e., the negative gradient direction) in order to decrease the function value. However, this is not possible because \( P[d_1^*] \) is already at the upper bound. In other words, the optimality condition of \( d_1 \) has been satisfied. Therefore, the optimal solution must be on the line of
\[ d_1 = U_1. \]

Note that the KKT optimality condition of (22) is for \( i = 1, 2, \)
\[ \nabla_i \hat{f}(d_1, d_2) = (Qd)_i + p_i \begin{cases} 
\geq 0 \quad \text{if} \quad d_i < U_i, \\
\leq 0 \quad \text{if} \quad d_i > L_i.
\end{cases} \]

In our discussion we refer to the above inequality as the optimality condition of \( d_i \). For example, the second inequality in (28) and \( P[d_1^*] = U_1 \) imply that the optimality condition of \( d_1 \) holds.

While the strategy of checking (28) avoids solving the one-variable problems and comparing their objective values, it seems we still need to check all eight cases separately. Fortunately, we can handle Figure 1a and part of Figure 1b together because for the situation in Figure 1a, the following theorem shows that \( (P[d_1^*], P[d_2^*]) \) also satisfies the optimality condition of \( d_1 \).

**Theorem 3.1** If
\[ d_1^* \notin (L_1, U_1), \]
\[ d_2^* \in [L_2, U_2], \]
then \( (P[d_1^*], P[d_2^*]) \) satisfies the optimality condition of \( d_1 \).

Therefore, we can cover a rather general situation by checking the optimality condition at \( (P[d_1^*], P[d_2^*]) \):
Theorem 3.2 Consider \( P[d_1^*] \) and \( P[d_2^*] \). If \( P[d_1^*] \) is bounded and \((P[d_1^*], P[d_2^*])\) satisfies the optimality condition at \( d_1 \), that is,
\[
Q_{11}P[d_1^*] + Q_{12}P[d_2^*] + p_1 \begin{cases} 
\leq 0 & \text{if } P[d_1^*] = U_1, \\
geq 0 & \text{if } P[d_1^*] = L_1,
\end{cases}
\]
then
\[
\bar{d}_1 = P[d_1^*],
\]
\[
\bar{d}_2 = \arg\min_{d_2 \in [L_2, U_2]} \frac{1}{2} \begin{bmatrix} d_1 & d_2 \end{bmatrix} \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{12} & Q_{22} \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \end{bmatrix} + \begin{bmatrix} p_1 & p_2 \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \end{bmatrix}
\]
\[
= \min(U_2, \max(L_2, -\frac{Q_{12}d_1 + p_2}{Q_{22}}))
\]
(30)
is an optimal solution of (22).

Clearly, the same theorem can hold if the roles of \( d_1^* \) and \( d_2^* \) are swapped. To ensure that every \( d^* \) in the situation of Figure 1b is covered (i.e., Theorem 3.2 on either \( d_1^* \) or \( d_2^* \) is applicable), we need the following theorem.

Theorem 3.3 If
\[
d_1^* \notin (L_1, U_1), \ d_2^* \notin (L_2, U_2),
\]
then \((P[d_1^*], P[d_2^*])\) satisfies either the optimality condition of \( d_1 \) or \( d_2 \).

All proofs are given in appendix.

Based on the above theorems we can derive a simple procedure for solving (22). To begin, if \( d_1^* \notin (L_1, U_1) \) then we know that \( P[d_1^*] \) is bounded. We may apply Theorem 3.2 by checking if \((P[d_1^*], P[d_2^*])\) satisfies the optimality condition of \( d_1 \). If it does, then (30) is an optimal solution.

There are two remaining situations:
\[
d_1^* \in (L_1, U_1)
\]
(31)
or
\[
d_1^* \notin (L_1, U_1) \text{ and } (P[d_1^*], P[d_2^*]) \text{ does not satisfy (29)}. \tag{32}
\]
For both situations, we argue that
\[
\bar{d}_2 = P[d_2^*],
\]
\[
\bar{d}_1 = \min(U_1, \max(L_1, -\frac{Q_{12}d_2 + p_1}{Q_{11}}))
\]
(33)
is an optimal solution. For (31), we can further consider two situations.
\[
d_2^* \in [L_2, U_2], \tag{34}
\]
\[
d_2^* \notin [L_2, U_2]. \tag{35}
\]
If (34) holds, then
\[ P[d_1^*] = d_1^* \text{ and } P[d_2^*] = d_2^* \]
are already an optimal solution. Though we do not need to apply (33), if we do, then \( \bar{d}_1 = d_1^* \) is obtained. On the other hand, if (35) holds, then from Theorem 3.1, \((P[d_1^*], P[d_2^*])\) satisfies the optimality condition of \( d_2 \). With the boundedness of \( P[d_2^*] \), we can apply Theorem 3.2 to have (33).

For the situation of (32), we argue that \( d_2^* \notin [L_2, U_2] \). Otherwise, \( d_2^* \in [L_2, U_2] \) and \( d_1^* \notin (L_1, U_1) \) imply from Theorem 3.1 that \((P[d_1^*], P[d_2^*])\) satisfies the optimality condition of \( d_1 \), a contradiction to the condition in (32). Next, the property \( d_2^* \notin [L_2, U_2] \), (32) and Theorem 3.3 imply that \((P[d_1^*], P[d_2^*])\) must satisfy the optimality condition of \( d_2 \). Then we can apply Theorem 3.2 to have (33).

A summary of the procedure is in Algorithm 3, in which we switch back to \( \alpha_i, \alpha_j \) from \( d_1, d_2 \) for practical implementations. The change is straightforward. For example, (30) becomes (50) in Algorithm 3.

### 3.2.2 Hessian is Positive Semi-definite

For \( l_2 \)-loss SVM, from (4), Hessian is always positive definite. However, for \( l_1 \)-loss SVM, Hessian may be only positive semi-definite and it is possible that
\[
Q_{11}Q_{22} - Q_{12}^2 = 0. \tag{36}
\]
From the assumption in (21),
\[
Q_{11} > 0 \text{ and } Q_{22} > 0,
\]
and therefore (36) implies
\[
Q_{12} \neq 0. \tag{37}
\]

When the Hessian is positive definite, in Section 3.2.1 we calculate \( d_1^* \) and \( d_2^* \), which play an important role in Algorithm 3. With (36), they are not well defined because a division by zero occurs. However, we will show that an extension of (25) to define \( d_1^* \) and \( d_2^* \) is possible. We begin with checking the numerator of \( d_1^* \) and \( d_2^* \) in (25). From
\[
Q_{11}(-Q_{22}p_1 + Q_{12}p_2) = -Q_{12}(-Q_{11}p_2 + Q_{12}p_1),
\]
the two numerators have the following relationship.
\[
-Q_{22}p_1 + Q_{12}p_2 = \frac{-Q_{12}}{Q_{11}}(-Q_{11}p_2 + Q_{12}p_1). \tag{38}
\]
Now assume that
\[-Q_{22}p_1 + Q_{12}p_2 \neq 0. \tag{39}\]
We will discuss later how to handle the situation if this value is zero.
From (36), (38), and (39), we extend (25) to define
\[ d_1^* = \begin{cases} 
\infty & \text{if } -Q_{22}p_1 + Q_{12}p_2 > 0, \\
-\infty & \text{if } -Q_{22}p_1 + Q_{12}p_2 < 0,
\end{cases} \]
\[ d_2^* = \begin{cases} 
\infty & \text{if } -Q_{11}p_2 + Q_{12}p_1 > 0, \\
-\infty & \text{if } -Q_{11}p_2 + Q_{12}p_1 < 0.
\end{cases} \] (40)

These values can be projected to lower or upper bounds if we make the following assumption.

**Assumption 1** We have
\[ -\infty < L_i \leq U_i < \infty, \quad i = 1, 2. \] (41)

For \(l_1\)-loss SVM, whose Hessian may be only positive semi-definite, this assumption holds because in (2) we choose \(C < \infty\).

We show in the following theorem that Theorem 3.2 and Theorem 3.3 can be extended here, so the same Algorithm 3 can be used without modifications. Note that Theorem 3.1 is no longer needed because from Assumption 1 and (40), the condition \(d_2^* \in [L_2, U_2]\) never holds.

**Theorem 3.4** Under Assumption 1, if
\[ Q_{11}Q_{22} - Q_{12}^2 = 0 \] (42)
and \(d_1^*, d_2^*\) are defined as in (40), then Theorems 3.2 and 3.3 hold.

We now show that the same procedure in Algorithm 3 can be used. From (40) and Assumption 1,
\[ d_1^* \notin [L_1, U_1] \text{ and } d_2^* \notin [L_2, U_2]. \] (43)

Then \(P[d_1^*]\) and \(P[d_2^*]\) are bounded. We check if \((P[d_1^*], P[d_2^*])\) satisfies the optimality condition of \(d_1\). If it does, then from Theorem 3.4, we can apply Theorem 3.2 to use (30) for obtaining a solution. Otherwise, from Theorem 3.4 and (43), we apply Theorem 3.3 to have that \((P[d_1^*], P[d_2^*])\) satisfies the optimality condition of \(d_2\). Then we apply Theorem 3.2 to obtain an optimal solution as in (33). Therefore, the solution procedure is exactly the same as the procedure in Algorithm 3 for positive-definite Hessian.

Next we discuss the rare situation where both
\[ Q_{11}Q_{22} - Q_{12}^2 = 0 \] (44)
and
\[ -Q_{22}p_1 + Q_{12}p_2 = 0. \] (45)
The objective function can be written as
\[ \frac{1}{2}Q_{11}d_1^2 + Q_{12}d_1 d_2 + \frac{1}{2}Q_{22}d_2^2 + p_1 d_1 + p_2 d_2 \]
\[ = \frac{1}{2Q_{11}}(Q_{11}d_1 + Q_{12}d_2 + p_1)^2 + \text{constant}, \] (46)
where for the linear term of \( d_2 \), we use (44)-(45) and (37) to have

\[
\frac{Q_{12} p_1 d_2}{Q_{11}} = \frac{Q_{12}^2 p_2 d_2}{Q_{11} Q_{22}} = p_2 d_2. \tag{47}
\]

From (46), the optimization problem becomes to find a point in the feasible region

\[
L_1 \leq d_1 \leq U_1, \quad L_2 \leq d_2 \leq U_2, \tag{48}
\]

that is the closest to the plane

\[
Q_{11} d_1 + Q_{12} d_2 + p_1 = 0. \tag{49}
\]

We then give the details in Appendix B to design Algorithm 5 for finding an optimal solution.

3.3 Shrinking Technique

Because of bound constraints \( 0 \leq \alpha_i \leq C_i \), it is well developed in SVM literature that some bounded components can be tentatively removed in the optimization process. Then we solve smaller problems to reduce the running time, a strategy usually referred to as the shrinking technique (Joachims, 1998). Though several ways are available to implement the shrinking technique, we extend the one proposed by Hsieh et al. (2008) to the two-variable situation. For a bound-constrained convex problem like (3), \( \alpha \) is optimal if and only if the following projected gradient is zero.

\[
\nabla^P_i f(\alpha) = \begin{cases} 
\nabla_i f(\alpha) & \text{if } 0 < \alpha_i < C_i, \\
\min(0, \nabla_i f(\alpha)) & \text{if } \alpha_i = 0, \\
\max(0, \nabla_i f(\alpha)) & \text{if } \alpha_i = C_i.
\end{cases}
\]

For the one-variable CD, let each cycle of updating all the remained variables be an “outer iteration.” Assume at the \((k-1)\)th outer iteration we have the following sequence of iterates.

\[
\alpha^{k-1,1}, \alpha^{k-1,2}, \ldots, \alpha^{k-1,\bar{l}},
\]

where \( \bar{l} \) is the number of remained variables at the beginning of the outer iteration. We further assume that at \( \alpha^{k-1,j} \), the index \( i_j \) is selected for possible update. Hsieh et al. (2008) define the following two values to indicate the violation of the optimality condition.

\[
M^{k-1} = \max_j \nabla^P_{i_j} f(\alpha^{k-1,j}), \quad m^{k-1} = \min_j \nabla^P_{i_j} f(\alpha^{k-1,j}).
\]

Then at each CD step of the next (i.e., the \( k \)th) outer iteration, before updating \( \alpha^{k,j}_{i_j} \) to \( \alpha^{k,j+1}_{i_j} \), the variable \( \alpha_{i_j} \) is shrunken if one of the following two conditions holds:

\[
\begin{align*}
\alpha^{k,j}_{i_j} &= 0 \quad \text{and} \quad \nabla_{i_j} f(\alpha^{k,j}) > M^{k-1}, \\
\alpha^{k,j}_{i_j} &= C_i \quad \text{and} \quad \nabla_{i_j} f(\alpha^{k,j}) < m^{k-1},
\end{align*} \tag{51}
\]
Algorithm 3 A procedure to solve the two-variable sub-problem (22). Note that for practical implementations we switch back to use $\alpha_i, \alpha_j$ rather than $d_1, d_2$.

1: Let

$p_i \leftarrow \nabla_i f(\alpha), \ p_j \leftarrow \nabla_j f(\alpha)$.

$\delta \leftarrow Q_{ii}Q_{jj} - Q_{ij}^2$

use $j \leftarrow$ FALSE

2: if $\delta = 0$ and $(-Q_{jj}p_i + Q_{ij}p_j = 0 \ or \ -Q_{ii}p_j + Q_{ij}p_i = 0)$ then

3: $(\tilde{\alpha}_i, \tilde{\alpha}_j) \leftarrow$ run Algorithm 5

4: else

5: calculate

$\bar{\alpha}_i \leftarrow \min(C_i, \max(0, \alpha_i + \frac{-Q_{jj}p_i + Q_{ij}p_j}{\delta}))$

$\bar{\alpha}_j \leftarrow \min(C_j, \max(0, \alpha_j + \frac{-Q_{ii}p_j + Q_{ij}p_i}{\delta}))$

6: if $\bar{\alpha}_i \geq C_i$ then

7: if $Q_{ii}(\bar{\alpha}_i - \alpha_i) + Q_{ij}(\bar{\alpha}_j - \alpha_j) + p_i \leq 0$ then

$\bar{\alpha}_j \leftarrow \min(C_j, \max(0, \alpha_j - \frac{Q_{ij}(\bar{\alpha}_i - \alpha_i) + p_j}{Q_{jj}}))$ (50)

8: else

9: use $j \leftarrow$ TRUE

10: end if

11: else if $\bar{\alpha}_i \leq 0$ then

12: if $Q_{ii}(\bar{\alpha}_i - \alpha_i) + Q_{ij}(\bar{\alpha}_j - \alpha_j) + p_i \geq 0$ then

$\bar{\alpha}_j \leftarrow \min(C_j, \max(0, \alpha_j - \frac{Q_{ij}(\bar{\alpha}_i - \alpha_i) + p_j}{Q_{jj}}))$

13: else

14: use $j \leftarrow$ TRUE

15: end if

16: else

17: use $j \leftarrow$ TRUE

18: end if

19: if use $j =$ TRUE

$\bar{\alpha}_i \leftarrow \min(C_i, \max(0, \alpha_i - \frac{Q_{ij}(\bar{\alpha}_j - \alpha_j) + p_i}{Q_{ii}}))$

20: end if

21: end if
Table 1: Date statistics.

<table>
<thead>
<tr>
<th>data set</th>
<th>#data</th>
<th>#features</th>
<th>data set</th>
<th>#data</th>
<th>#features</th>
</tr>
</thead>
<tbody>
<tr>
<td>ijcnn1</td>
<td>49,990</td>
<td>22</td>
<td>a9a</td>
<td>32,561</td>
<td>123</td>
</tr>
<tr>
<td>news20.binary</td>
<td>19,996</td>
<td>1,355,191</td>
<td>rcv1_train.binary</td>
<td>20,242</td>
<td>47,326</td>
</tr>
<tr>
<td>real-sim</td>
<td>72,309</td>
<td>20,958</td>
<td>yahoojp</td>
<td>176,203</td>
<td>832,026</td>
</tr>
</tbody>
</table>

where

\[ M^{k-1} = \begin{cases} 0 & \text{if } M^{k-1} > 0, \\ \infty & \text{otherwise} \end{cases} \]

\[ m^{k-1} = \begin{cases} 0 & \text{if } m^{k-1} < 0, \\ \infty & \text{otherwise} \end{cases} \]

In (51), \( M^{k-1} \) must be strictly positive, so Hsieh et al. (2008) set it be \( \infty \) if \( M^{k-1} \leq 0 \). The situation for \( m_{k-1} \) is similar. Details of one-variable CD with shrinking can be found in appendix of Hsieh et al. (2008).

To extend the above setting to two-variable block CD, one issue is that we no longer have the concept of outer iterations. The reason is that from Section 3.1 a random working-set selection is practically more viable. Therefore, we can treat a fixed number of CD steps as an outer iteration in order to calculate the above \( M^{k-1} \) and \( m^{k-1} \) values. We choose \( l \) so that the frequency of refreshing \( M^{k-1} \) and \( m^{k-1} \) is similar to that of one-variable CD. Algorithm 4 summarizes the two-variable block CD with the shrinking implementation.

### 3.4 Experiments

For experiments we consider data sets listed in Table 1. Some are dense sets with \( l \gg n \) and some are sparse sets with both large \( l \) and \( n \). Because of space considerations, we present results by using the \( l_2 \) loss, while leave complete results of \( l_1 \) loss in supplementary materials.

For each setting, we show CD steps or training time versus the relative difference to the optimal function value:

\[
\frac{|f(\alpha) - f(\alpha^*)|}{|f(\alpha^*)|},
\]

where \( \alpha^* \) is an approximate optimal solution obtained by running many iterations of the algorithm. The regularization parameter \( C \) is set to be 1 and 8,192.

Our implementation is extended from the software LIBLINEAR (Fan et al. 2008), which provides an implementation of the one-variable CD by Hsieh et al. (2008). Except in Section 3.4.3, we do not incorporate the shrinking technique. All experiments are conducted on a computer with an AMD EPYC 7401 24-Core Processor.

#### 3.4.1 The Importance of Randomness in Working-set Selection

To identify an effective pair-selection scheme for two-variable block CD, we compare the following settings discussed in Section 3.1:

2. All sets except yahoojp are publicly available at https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/
Algorithm 4  Two-variable block CD for solving (3) with a shrinking implementation.

1: Given $\epsilon, \alpha$ and the corresponding $w = \sum_i y_i \alpha_i x_i$.
2: Remove indices with $x_i = 0$
3: Let $\bar{M} \leftarrow \infty, \bar{m} \leftarrow -\infty$ and $A \leftarrow \{1, \ldots, l\}$.
4: while $\alpha$ is not optimal do
5:   Let $M \leftarrow -\infty, m \leftarrow \infty$.
6:   for $t = 1, \ldots, l$ do
7:       select $i, j \in \{1, \ldots, l\}$ with $i \neq j$
8:       $G_i = y_i w^T x_i - 1 + D_{ii} \alpha_i, \ G_j = y_j w^T x_j - 1 + D_{jj} \alpha_j$
9:       $Q_{ij} = y_i y_j w_j^T x_i$
10:      $PG_i \leftarrow 0, \ PG_j \leftarrow 0, \ shrink \leftarrow \text{FALSE}$
11:     for $t = i, j$ do
12:        if $\alpha_t = 0$ then
13:           if $G_t > \bar{M}$ then $A \leftarrow A \setminus \{t\}$ and shrink $\leftarrow \text{TRUE}$
14:           if $G_t < 0$ then $PG_t \leftarrow G_t$
15:        else if $\alpha_t = U_t$ then
16:           if $G_t < \bar{m}$ then $A \leftarrow A \setminus \{t\}$ and shrink $\leftarrow \text{TRUE}$
17:           if $G_t > 0$ then $PG_t \leftarrow G_t$
18:        else
19:           $PG_t \leftarrow G_t$
20:        end if
21:     end for
22:     $M \leftarrow \max(M, PG_t), m \leftarrow \min(m, PG_t)$
23:     if shrink=TRUE then continue
24:     $(\bar{\alpha}_i, \bar{\alpha}_j) \leftarrow \text{Solve (22) by Algorithm 3}$
25:     for $t = i, j$ do
26:       if $\bar{\alpha}_t \neq \alpha_t$ then
27:          $w \leftarrow w + (\bar{\alpha}_t - \alpha_t) y_t x_t$
28:          $\alpha_t \leftarrow \bar{\alpha}_t$
29:       end if
30:     end for
31:     end if
32:     if $M - m < \epsilon$ then
33:       if $A = \{1, \ldots, l\}$ then break
34:     else
35:       $A \leftarrow \{1, \ldots, l\}, \bar{M} \leftarrow \infty, \bar{m} \leftarrow -\infty$. (i.e., no shrinking at the next iteration)
36:     end if
37:     end if
38:     if $M \leq 0$ then $\bar{M} \leftarrow \infty$ else $\bar{M} \leftarrow M$
39:     if $m \geq 0$ then $\bar{m} \leftarrow \infty$ else $\bar{m} \leftarrow m$
40: end while

• perm: a permutation of $O(l^2)$ elements as shown in [19].
Chiu AND Lin.

Figure 3: A comparison of strategies for selecting the two-variable working set. The $x$-axis is the number of CD steps, while the $y$-axis (log-scaled) is the relative difference to the optimal function value. We consider $l2$ loss SVM with $C = 1$ (upper sub-figure), and 8, 192 (lower sub-figure).

- **semi-perm**: the setting in [20] to avoid the $O(l^2)$ storage of perm.

- **random**: a random selection.

In Figure 3, we check the number of CD steps versus the relative function-value decrease. Only small data sets are used for this experiment because of the $O(l^2)$ storage requirement of perm. Results show that semi-perm is generally worse than perm and random. Therefore, randomness in selecting the working set is very essential.

Because perm is not practically feasible, we conclude that among the three options considered here random is the only effective way to implement two-variable block CD for large-scale problems.

### 3.4.2 Comparison with One-variable CD

We compare the following settings.

- **1-CD-perm**: one-variable CD by permuting all indices in the beginning of each outer iteration. This is the method currently implemented in LIBLINEAR though shrinking is disabled.

- **1-CD-random**: one-variable CD by a random selection of indices for update.

- **2-CD-random**: two-variable block CD by a random selection of indices for update. Note that among methods discussed in Section 3.4.1, this setting is the most effective way for two-variable CD when the data set is large.
We present results of using $C = 1$ and 8,192; see Figures 4 and 5, respectively. Results indicate that by using two variables at a time, the number of CD steps is significantly reduced. The reason is apparently that more information is considered. However, the cost per CD step is also higher, so in terms of running time, using two variables is only competitive when $C = 1$. If $C$ is increased to 8,192, the difference on the number of CD steps is bigger, so for some problems, the running time is shorter. It is well known that a large $C$ implies a better fit of the data, so the optimization problem becomes more difficult. In such a situation, the proposed two-variable block CD is very useful.

A side result in Figures 4-5 is the comparison between 1-CD-perm and 1-CD-random. We may expect that such a comparison has been conducted in some existing works, but interestingly we are not aware of any. In Figure 4, 1-CD-perm is consistent better than 1-CD-random and the gap is sometimes significant. The reason might be that because of the random selection, some variables are less frequently updated than others.

### 3.4.3 Effect of Shrinking Techniques

To check the effect of the shrinking technique in two-variable CD, we compare the following settings.

- **1-CD**: this is the same as 1-CD-perm in Section 3.4.2.
- **1-CD-shrinking**: shrinking technique is incorporated into 1-CD.
- **2-CD**: this is the same as 2-CD-random in Section 3.4.2.
- **2-CD-shrinking**: shrinking technique is incorporated into 2-CD.

In Figure 6, we present a timing comparison by using $C = 1$ and 8,192. Results show that shrinking for two-variable CD is generally as effective as for one-variable CD.

### 4. Two-variable Block CD for Linear SVM with a Bias Term

We now move to discuss two-variable block CD for the standard SVM problem (6), in which a bias term is considered. For the working-set selection, following the discussion in Section 3.1, we consider a random selection.

### 4.1 Solving the Two-variable Sub-problem

To solve the dual problem (6) with a linear constraint, let

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

be the working set considered at the current CD step. The two-variable sub-problem is

$$\min_{d_i, d_j} \frac{1}{2} \begin{bmatrix} d_i & d_j \end{bmatrix} \begin{bmatrix} Q_{ii} & Q_{ij} \\ Q_{ij} & Q_{jj} \end{bmatrix} \begin{bmatrix} d_i \\ d_j \end{bmatrix} + [\nabla_i f(\alpha) \nabla_j f(\alpha)] \begin{bmatrix} d_i \\ d_j \end{bmatrix}$$

subject to

$$0 \leq \alpha_i + d_i \leq C_i,$$
$$0 \leq \alpha_j + d_j \leq C_j,$$
$$y_i d_i + y_j d_j = 0.$$  

(53)
Figure 4: A comparison between one-variable and two-variable CD for $l^2$ loss with $C = 1$. For each set, $x$-axis in the upper sub-figure is the number of CD steps, while $x$-axis in the lower sub-figure is the running time (in seconds).

This sub-problem has been routinely solved in CD methods for kernel SVM, where the bias term is often considered. With the linear constraint, the feasible region becomes a line segment as indicated in Figure 7. The situation is very different from the sub-problem without a linear constraint, for which the feasible set is the whole box region. By replacing

$$d_i = -y_iy_jd_j$$
in the objective function, we can easily minimize the following quadratic function of $d_j$

$$
\frac{1}{2}(Q_{ii} + Q_{jj} - 2y_iy_jQ_{ij})d^2_j + (\nabla_j f(\alpha) - y_iy_j \nabla_i f(\alpha))d_j
$$

If the minimum is not on the line segment, we can identify an end point as the optimal solution. Details of the solution procedure can be found in for example, Section 6 of Chang.

Figure 5: A comparison between one-variable and two-variable CD for $l_2$ loss with $C = 8, 192$. For each set, $x$-axis in upper sub-figure is the number of CD steps, while the $x$-axis in the lower sub-figure is the running time (in seconds).
Figure 6: A timing comparison between one-variable and two-variable CD with/without shrinking for $l_2$-loss SVM. The $x$-axis is running time in seconds. Note that the shrinking implementation is stopping-tolerance dependent (see line 32 of Algorithm 4). Thus the curve is generated by several runs of using different tolerances. It may not be strictly decreasing because of timing fluctuation.

Interestingly, though it is easy to derive a solution procedure for solving (53), a comparison shows that Algorithm 3 for solving (22) is shorter in terms of the code.
Two-variable Block Dual Coordinate Descent Methods for Linear SVMs

\[
y_i \alpha_i + y_j \alpha_j = -y_N^T \alpha_N
\]

Figure 7: An illustration of the feasible region of (53). The solid circle on the \( \alpha_i \)-axis indicates the current iterate in our discussion in Section 4.2.

length. One reason is that in Algorithm 3 gradient information (or optimality condition) is used to avoid the exhaustive check of all out-of-boundary cases of \( \alpha_i \) or \( \alpha_j \). Further, for solving (53), we must separately handle the situations of \( y_i = y_j \) and \( y_i = -y_j \).

4.2 Difference Between With and Without the Bias Term

In Section 1, we mentioned that [Hsieh et al. (2008)] did not consider the SVM formulation with a bias term. Therefore, the dual SVM is a bound-constrained problem and one-variable CD is applicable. Following the success in Section 3 for developing a competitive two-variable CD, an interesting question is whether the decision by [Hsieh et al. (2008)] is a must or not. This issue is important because if two-variable CD is equally good for the dual problem with a linear constraint, then we may prefer always using the standard SVM formulation with the bias term. Unfortunately, here we explain that some subtle differences occur when two-variable CD is applied to the dual problem with/without the linear constraint.

If a bias term is considered and assume that \( y_i \neq y_j \), then because of the equality constraint, the feasible region is similar to what Figure 7 shows. Assume the current \( \alpha \) satisfies

\[
\alpha_i \in (0, C_i), \alpha_j = 0; \quad (55)
\]

see the point on the \( x \)-axis of Figure 7. We can change \((\alpha_i, \alpha_j)\) only if it does not satisfy the optimality condition. Now the quadratic function (54) over the constraint

\[
y_i \alpha_i + y_j \alpha_j = -y_N^T \alpha_N \quad (56)
\]

is in one of the two situations illustrated in Figure 8. From Figures 7 and 8, the objective function can be decreased only if the right sub-figure of Figure 8 occurs. Therefore, for the selected \( \{i,j\} \) working set, the chance we can improve the objective function value is only half. In practice, the chance may be even smaller. For example, if at the optimal solution \( \alpha^* \), we have

\[
\alpha_i^* \in (0, 1), \alpha_j = 0, \quad (57)
\]
Figure 8: An illustration of the two situations in minimizing the quadratic function (54) over the constraint (56). The solid circle is assumed to be the current iterate shown on the $x$-axis of Figure 7.

(a) $(\alpha_i, 0)$ is optimal and cannot be further changed.

(b) $(\alpha_i, 0)$, though on the boundary, can be changed to decrease the function value.

Figure 9: A illustration showing that without the linear constraint, in general an iterate $(\alpha_i, 0)$ on the boundary is not optimal for the sub-problem. Thus we can improve the objective function value.

then at the later stage of the optimization process, the iterate $(\alpha_i, 0)$ is likely optimal for the sub-problem (i.e., the right sub-figure of Figure 8 happens).

On the other hand, if the optimization problem does not have the equality constraint, then unless the relationship between the contour and the feasible region is like Figure 9a, we can always change $\alpha_i$ to improve the function value. For example, in Figure 9b, although $(\alpha_i, 0)$ is on the boundary of the feasible region, we can identify another point with a smaller objective function value. In contrast, for the same contour and feasible region, if the linear constraint is imposed, then we are in a situation like the right sub-figure in Figure 8 and the function value cannot be reduced.

We now use mathematical derivations to explain the above analysis. Assume

$$y_i = 1, \ y_j = -1.$$
With the assumption (55), if the bias term is used, the optimality condition is that there exists $b$ such that

$$\nabla_i f(\alpha) + b = 0,$$

$$\nabla_j f(\alpha) - b \geq 0.$$  

This is equivalent to

$$\nabla_j f(\alpha) \geq -\nabla_i f(\alpha)$$

and as we said, the chance that it happens may be only half.

If the bias term is not considered, under the assumption (55), the optimality condition is

$$\nabla_i f(\alpha) = 0 \quad \text{(58)}$$

and

$$\nabla_j f(\alpha) \geq 0. \quad \text{(59)}$$

Clearly, though the chance that (59) holds may be only half, the probability that (58) holds is measure zero. Now we see a crucial difference between with and without the bias term: for the former, in the optimality condition, $(\alpha_i, \alpha_j)$ are tied together. In contrast, for the latter, the optimality conditions of $\alpha_i$ and $\alpha_j$ are independent to each other.

Our discussion so far is by assuming that the current iterates satisfies (55). That is, one element is free, while the other is bounded. In fact, a similar argument can be made if both elements are bounded, though we do not go through details here.

In summary, if in the optimization process many $\alpha$ variables are bounded, we frequently have pairs like those in (55). Then with a high probability the CD step is wasted if the bias term is considered. In Section 4.3 we will experimentally confirm the analysis here.

### 4.3 Experiments

We compare the following two settings.

- **2-CD-nobias**: this is the same as 2-CD-random in Section 3.4.2.
- **2-CD-bias**: the two-variable block CD discussed in Section 4.2 for solving (6).

They differ only in the solved dual optimization problem: a linear equality constraint appears in the problem solved by 2-CD-bias, while does not for 2-CD-nobias. Note that their optimal objective values are different, but it is suitable to compare the convergence speed by using their relative difference to the respective optimal value. By considering the $l_2$ loss, we present results in Figure 10. We show only the convergence along the number of CD steps rather than the running time because both settings have similar cost per CD step.

Results indicate that 2-CD-bias is significantly slower than 2-CD-nobias, an observation fully consistent with the analysis in Section 4.2. We conduct a further investigation in Table 2 by showing the percentage of CD steps that are wasted (i.e., in the CD step the
Figure 10: Comparison of applying two-variable block CD to solve the SVM problem with/without a bias term. We consider the $l_2$ loss and set $C = 1$. The $x$-axis is the cumulative number of CD steps. Because the cost per CD step of both approaches is basically the same, we do not present the results of running time. Shrinking is disabled.

Table 2: The percentage of CD steps that are wasted in the first 200 cycles (i.e., outer iterations).

<table>
<thead>
<tr>
<th>Data set</th>
<th>2-CD-bias</th>
<th>2-CD-nobias</th>
<th>Data set</th>
<th>2-CD-bias</th>
<th>2-CD-nobias</th>
</tr>
</thead>
<tbody>
<tr>
<td>a9a</td>
<td>59.00%</td>
<td>14.77%</td>
<td>rcv1.train.binary</td>
<td>87.82%</td>
<td>66.94%</td>
</tr>
<tr>
<td>ijcnn1</td>
<td>79.88%</td>
<td>53.17%</td>
<td>real-sim</td>
<td>95.17%</td>
<td>22.40%</td>
</tr>
<tr>
<td>yahoojp</td>
<td>83.11%</td>
<td>54.67%</td>
<td>news20.binary</td>
<td>67.12%</td>
<td>21.00%</td>
</tr>
</tbody>
</table>

selected $(\alpha_i, \alpha_j)$ is already optimal for the sub-problem and cannot be further changed. For both settings, a fair amount of CD steps are wasted because with $\alpha = 0$ as the initial point, many pairs have $\alpha_i = \alpha_j = 0$. More importantly, the percentage of 2-CD-nobias is much higher, indicating that it faces more difficulties in reducing the function value by the selected pairs.

Steinwart et al. (2011) conducted a similar comparison for kernel SVM, where greedy working-set selections are used in two-variable CD. From Figure 5 in their work, the difference between with and without bias is much smaller than ours in Figure 10. The reason is apparently that greedy selections avoid the situation of many wasted CD steps described in Section 4.2.
Table 3: The percentage of CD steps that are wasted in the first 10 cycles (i.e., outer iterations).

<table>
<thead>
<tr>
<th>Data set</th>
<th>2-CD-bias</th>
<th>2-CD-nobias</th>
<th>Data set</th>
<th>2-CD-bias</th>
<th>2-CD-nobias</th>
</tr>
</thead>
<tbody>
<tr>
<td>a9a</td>
<td>53.97%</td>
<td>10.39%</td>
<td>rcv1_train.binary</td>
<td>80.60%</td>
<td>40.35%</td>
</tr>
<tr>
<td>ijcnn1</td>
<td>76.30%</td>
<td>70.91%</td>
<td>real-sim</td>
<td>90.51%</td>
<td>39.78%</td>
</tr>
<tr>
<td>yahoojp</td>
<td>75.75%</td>
<td>67.30%</td>
<td>news20_binary</td>
<td>58.73%</td>
<td>17.03%</td>
</tr>
</tbody>
</table>

Table 4: A summary of CD methods’ running time for SVM with/without the bias. Results of kernel SVM are from [Steinwart et al. (2011)]. A > B means A is faster than B.

<table>
<thead>
<tr>
<th>SVM problems and CD methods</th>
<th>no bias</th>
<th>with bias</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-CD</td>
<td>&lt;</td>
<td>&gt; or ≈</td>
</tr>
<tr>
<td>2-CD</td>
<td>≈</td>
<td>≫</td>
</tr>
</tbody>
</table>

5. Discussion and Conclusions

In this work we broadly discuss issues in extending one-variable CD to two-variable CD for linear SVM. For the commonly used linear-SVM setting without considering the bias term, we derive a simple and elegant procedure to solve each sub-problem. The resulting two-variable CD framework is generally competitive and is superior for difficult problems. For the SVM formulation with a bias term, we show that because of the linear constraint in the dual optimization problem, CD methods are less effective. We summarize our findings together with those in [Steinwart et al. (2011)] for kernel SVM in Table 4. Overall, this work sheds many new insights on the CD methods for training large-scale linear SVM.

References


Appendix A. Proofs of Theorems

A.1 Proof of Theorem 3.1

Proof Consider

\[ d_1^* \geq U_1. \]

Then from (24),

\[
Q_{11}U_1 + Q_{12}P[d_2^*] + p_1 \\
= Q_{11}d_1^* + Q_{12}d_2^* + p_1 + Q_{11}(U_1 - d_1^*) \\
= Q_{11}(U_1 - d_1^*) \leq 0, 
\]

so the optimality condition is satisfied. The situation for

\[ d_1^* \leq L_1 \]

is similar. \( \blacksquare \)

A.2 Proof of Theorem 3.2

Proof Because \( \bar{d}_2 \) is an optimal solution of (30), it satisfies the following optimality condition

\[
Q_{12}\bar{d}_1 + Q_{22}\bar{d}_2 + p_2 \\
\begin{cases}
\geq 0 & \text{if } \bar{d}_2 < U_2, \\
\leq 0 & \text{if } \bar{d}_2 > L_2.
\end{cases}
\]

This is already the optimality condition corresponding to \( d_2 \), so our remaining task of proving that \( (\bar{d}_1, \bar{d}_2) \) is optimal is to show that

\[
Q_{11}\bar{d}_1 + Q_{12}\bar{d}_2 + p_1 \\
\begin{cases}
\geq 0 & \text{if } \bar{d}_1 < U_1, \\
\leq 0 & \text{if } \bar{d}_1 > L_1.
\end{cases}
\]

We prove

\[ Q_{11}\bar{d}_1 + Q_{12}\bar{d}_2 + p_1 \leq 0 \]
under the situation of
\[d^*_1 \geq U_1 > L_1, \quad \bar{d}_1 = U_1.\] (62)

The proof of the other situation
\[d^*_1 \leq L_1 < U_1, \quad \bar{d}_1 = L_1\]
is the same. Note that if
\[U_1 = L_1,\]
then (60) directly holds because no \(\bar{d}_1 \in (L_1, U_1).\) Now we consider two cases.

Case 1: \(Q_{12} \geq 0\)

From (24),
\[-(Q_{12}\bar{d}_1 + p_2) - d^*_2 = -Q_{12}\bar{d}_1 - p_2 - Q_{22}d^*_2\]
\[= -Q_{12}(\bar{d}_1 - d^*_1) \geq 0,\] (63)

where (63) is from \(Q_{12} \geq 0\) and (62). Thus in finding \(\bar{d}_2\) in (30) we intend to increase \(d^*_2\) to \(\bar{d}_2\).

We further consider two situations. First,

\[-(Q_{12}\bar{d}_1 + p_2) \leq L_2.\] (64)

Then (63) implies
\[d^*_2 \leq -(Q_{12}\bar{d}_1 + p_2) \leq L_2\]
and with (30),
\[\bar{d}_2 = P[d^*_2] = L_2.\] (64)

By (62), (64), and the assumption in (29), we have the optimality condition
\[Q_{11}\bar{d}_1 + Q_{12}\bar{d}_2 + p_1 = Q_{11}P[d^*_1] + Q_{12}P[d^*_2] + p_1 \leq 0.\] (65)

If on the other hand
\[-(Q_{12}\bar{d}_1 + p_2) > L_2,\] (66)
then (30) and (66) imply
\[\bar{d}_2 = \min(U_2, -(Q_{12}\bar{d}_1 + p_2) \leq -(Q_{12}\bar{d}_1 + p_2), \frac{-(Q_{12}\bar{d}_1 + p_2)}{Q_{22}}.\] (67)
Therefore,
\[ \tilde{d}_2 - d^*_2 \leq - \frac{(Q_{12} \tilde{d}_1 + p_2)}{Q_{22}} - d^*_2 \]
\[ = - \frac{Q_{12} (\tilde{d}_1 - d^*_1)}{Q_{22}}, \]  
(68)
where (68) is from (63). Then
\[ Q_{11} \tilde{d}_1 + Q_{12} \tilde{d}_2 + p_1 = Q_{11} (\tilde{d}_1 - d^*_1) + Q_{12} (\tilde{d}_2 - d^*_2) \]
\[ \leq (\tilde{d}_1 - d^*_1) \frac{Q_{11} Q_{22} - Q_{12}^2}{Q_{22}} \]
\[ \leq 0, \]  
(70)
where (69) is from (24), (70) is from (68), and (71) is from the positive semi-definiteness of \([Q_{11} \ Q_{12} \ Q_{12} \ Q_{22}].\) This leads to the optimality condition in (61).

Case 2: \(Q_{12} < 0\)

By a similar derivation to (63),
\[ - (Q_{12} \tilde{d}_1 + p_2) \frac{Q_{22}}{Q_{22}} - d^*_2 = - \frac{Q_{12} (\tilde{d}_1 - d^*_1)}{Q_{22}} \leq 0. \]
(72)
Thus we intend to decrease \(d^*_2\) to \(\tilde{d}_2\).

We also consider two cases. First,
\[ - (Q_{12} \tilde{d}_1 + p_2) \frac{Q_{22}}{Q_{22}} \geq U_2. \]

Then (72) and (30) imply
\[ \tilde{d}_2 = P[d^*_2] = U_2. \]
By (62) and the assumption in (29), we have the optimality condition in (65). If on the other hand,
\[ - (Q_{12} \tilde{d}_1 + p_2) \frac{Q_{22}}{Q_{22}} < U_2, \]
(73)
then (30) and (73) imply
\[ \tilde{d}_2 = \max(L_2, - \frac{(Q_{12} \tilde{d}_1 + p_2)}{Q_{22}}) \geq - \frac{(Q_{12} \tilde{d}_1 + p_2)}{Q_{22}}. \]
(74)
Therefore,
\[ \tilde{d}_2 - d^*_2 \geq - \frac{(Q_{12} \tilde{d}_1 + p_2)}{Q_{22}} - d^*_2 \]
\[ = - \frac{Q_{12} (\tilde{d}_1 - d^*_1)}{Q_{22}}, \]  
(75)
where (75) is from (72). With \(Q_{12} < 0\), we obtain the same inequalities in (69)-(71) and the optimality condition in (61).
A.3 Proof of Theorem 3.3

Proof We consider

\[ L_i < U_i, \ i = 1, 2, \]  

(76)

because if \( L_i = U_i \), the optimality condition directly holds.

We assume \( P[d_1^*] = U_1 \). The situations of \( P[d_1^*] = L_1 \) is similar because of the symmetry.

Assume the result is wrong. Then \( (P[d_1^*], P[d_2^*]) \) satisfies neither the optimality condition of \( d_1 \) nor that of \( d_2 \). We further consider two cases \( P[d_2^*] = L_2 \) and \( U_2 \).

Case 1: \( P[d_2^*] = L_2 \)

Define \( \Delta_1 \) and \( \Delta_2 \) as

\[
\Delta_1 = P[d_1^*] - d_1^* = U_1 - d_1^* \leq 0, \\
\Delta_2 = P[d_2^*] - d_2^* = L_2 - d_2^* \geq 0.
\]  

(77)

Because optimality conditions are violated, with (76),

\[
Q_{11} P[d_1^*] + Q_{12} P[d_2^*] + p_1 > 0, \\
Q_{12} P[d_1^*] + Q_{22} P[d_2^*] + p_2 < 0.
\]  

(78)

With (24) and (77), (78) becomes

\[
Q_{11} \Delta_1 + Q_{12} \Delta_2 > 0, \\
Q_{12} \Delta_1 + Q_{22} \Delta_2 < 0.
\]  

(79)

We then have

\[ \Delta_1 \neq 0 \text{ or } \Delta_2 \neq 0. \]  

(80)

Otherwise, (79) cannot hold. From (79) and (80),

\[
\begin{bmatrix} \Delta_1 & \Delta_2 \end{bmatrix} \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{12} & Q_{22} \end{bmatrix} \begin{bmatrix} \Delta_1 \\ \Delta_2 \end{bmatrix} < 0.
\]

However,

\[
\begin{bmatrix} Q_{11} & Q_{12} \\ Q_{12} & Q_{22} \end{bmatrix}
\]

is positive semi-definite, so there is a contradiction.

Case 2: \( P[d_2^*] = U_2 \)

We have \( \Delta_1 \leq 0, \Delta_2 \leq 0 \). The violation of the result implies

\[
Q_{11} \Delta_1 + Q_{12} \Delta_2 > 0, \\
Q_{12} \Delta_1 + Q_{22} \Delta_2 > 0.
\]

With (80),

\[
\begin{bmatrix} \Delta_1 & \Delta_2 \end{bmatrix} \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{12} & Q_{22} \end{bmatrix} \begin{bmatrix} \Delta_1 \\ \Delta_2 \end{bmatrix} < 0,
\]

a contradiction to the positive semi-definiteness of \( \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{12} & Q_{22} \end{bmatrix} \).
A.4 Proof of Theorem 3.4

Proof We begin with checking Theorem 3.2. The same proof can almost be used. We also prove only the situation
\[ d^*_1 \geq U_1, \bar{d}_1 = U_1. \]

From the definition in (40), this in fact means
\[ d^*_1 = \infty, \ P[d^*_1] = \bar{d}_1 = U_1. \] (81)

Further, (81) and (40) imply
\[ -Q_{22}p_1 + Q_{12}p_2 > 0. \] (82)

We now consider \( Q_{12} > 0 \), while the proof for \( Q_{12} < 0 \) is similar. Note that we have \( Q_{12} \neq 0 \) from (37).

The same as in Theorem 3.2, we further consider two situations. First,
\[ -(Q_{12}d_1 + p_2) \leq L_2. \]

From (82), \( Q_{12} > 0 \), (38) and (40), we have
\[ d^*_2 = -\infty \leq \frac{-(Q_{12}d_1 + p_2)}{Q_{22}} \leq L_2. \] (83)

Then (64) and (65) follow, so we have the needed optimality condition.

If on the other hand,
\[ -(Q_{12}d_1 + p_2) > L_2, \]

then (74) holds. We now check the optimality condition of \( d_1 \):
\[ Q_{11}d_1 + Q_{12}d_2 + p_1 \]
\[ \leq Q_{11}d_1 + Q_{12} \frac{-(Q_{12}d_1 + p_2)}{Q_{22}} + p_1 \]
\[ = \frac{-Q_{12}p_1 + Q_{22}p_1}{Q_{22}} \]
\[ \leq 0, \] (86)

where (84) is from \( Q_{12} \geq 0 \) and (74), (85) is from (42), and (86) is from (82).

Next, to extend Theorem 3.3 we follow the same setting to consider
\[ d^*_1 = +\infty, \ P[d^*_1] = U_1 \] (87)

and check the two cases \( P[d^*_2] = L_2 \) or \( U_2 \).

Case 1: \( P[d^*_2] = L_2 \)

For this case
\[ d^*_2 = -\infty \text{ and } P[d^*_2] = L_2. \]
From (38) and (40),
\[ Q_{12} > 0. \] (88)
If the result in Theorem 3.3 is wrong, both optimality conditions are violated and
\[
Q_{11}U_1 + Q_{12}L_2 + p_1 > 0,
Q_{12}U_1 + Q_{22}L_2 + p_2 < 0.
\]
With (36) and (88),
\[
Q_{22}(Q_{11}U_1 + Q_{12}L_2 + p_1) > 0 \quad (89)
\]
leads to
\[ -Q_{22}p_1 + Q_{12}p_2 < 0. \] (90)
From (40), we obtain a contradiction to \( d_1^* = \infty \) in (87).
Case 2: \( P[d_2^*] = U_2 \)
For this case
\[ d_2^* = \infty \text{ and } P[d_2^*] = U_2. \]
From (38) and (40),
\[ Q_{12} < 0. \] (91)
The same with the last case, we assume the optimality conditions are violated and therefore
\[
Q_{11}U_1 + Q_{12}L_2 + p_1 > 0,
Q_{12}U_1 + Q_{22}L_2 + p_2 > 0.
\]
With (91), we can have (89) and (90). Then (90) contradicts the assumption.

Appendix B. Solution Procedure when (44) and (45) Both Happen

It is easy to identify an optimal solution by checking the geometric relationship between
the feasible region and the straight line in (49). If \( Q_{12} < 0 \), then the line has a positive
slope. Thus we have the following four possible situations.
If

\[ Q_{11}U_1 + Q_{12}L_2 + p_1 \leq 0, \]  

(92)

then the whole feasible region is on the left side of the line and \((U_1, L_2)\) is the closest point.

If \((92)\) does not hold and

\[ Q_{11}L_1 + Q_{12}L_2 + p_1 \leq 0, \]

then from the figure, a line segment is the interaction between the line and the region. Any point on this line segment is an optimal solution. We can simply consider the interaction point on the horizontal line \(d_2 = L_2\):

\[ \dd{d}_1 = \frac{-Q_{12}L_2 - p_1}{Q_{11}}, \quad \dd{d}_2 = L_2. \]  

(93)

Other situations are similar. A summary of the procedure is in Algorithm 5. For practical implementations we switch from \(d_1, d_2\) back to \(\alpha_i, \alpha_j\) by using (23). Note that the plane \((49)\) becomes

\[ Q_{ii}\dd{\alpha}_i + Q_{ij}\dd{\alpha}_j = \delta \equiv Q_{ii}\alpha_i + Q_{ij}\alpha_j - p_i, \]  

(94)

where \((\dd{\alpha}_i, \dd{\alpha}_j)\) and \((\alpha_i, \alpha_j)\) are the variable and the current iterate, respectively.
Algorithm 5 Solve the sub-problem (22) when (44) and (45) both occur.

1: input: \( \alpha_i, \alpha_j, Q_{ii}, Q_{ij}, C_i, C_j, p_i \)
2: calculate \( \delta \leftarrow Q_{ii} \alpha_i + Q_{ij} \alpha_j - p_i \)
3: if \( Q_{ij} < 0 \) then
4: if \( Q_{ii} C_i \leq \delta \) then return \((C_i, 0)\)
5: else if \( Q_{ij} C_j \geq \delta \) then return \((0, C_j)\)
6: else if \( 0 \leq \delta \) then
7: return \((\frac{\delta}{Q_{ii}}, 0)\)

8: else
9: if \( Q_{ij} > 0 \) then
10: if \( 0 \geq \delta \) then return \((0, 0)\)
11: else if \( Q_{ii} C_i + Q_{ij} C_j \leq \delta \) then return \((C_i, C_j)\)
12: else if \( Q_{ij} C_j \leq \delta \) then
13: return \((\frac{\delta - Q_{ij} C_j}{Q_{ii}}, C_j)\)

14: else
15: return \((0, \frac{\delta}{Q_{ij}})\)

end if