A Dual Coordinate Descent Method for Large-scale Linear SVM

Cho-Jui Hsieh  
Kai-Wei Chang  
Chih-Jen Lin  
Department of Computer Science, National Taiwan University, Taipei 106, Taiwan

S. Sathiya Keerthi  
Yahoo! Research, Santa Clara, USA

S. Sundararajan  
Yahoo! Labs, Bangalore, India

Abstract

In many applications, data appear with a huge number of instances as well as features. Linear Support Vector Machines (SVM) is one of the most popular tools to deal with such large-scale sparse data. This paper presents a novel dual coordinate descent method for linear SVM with L1- and L2-loss functions. The proposed method is simple and reaches an ϵ-accurate solution in $O(\log(1/\epsilon))$ iterations. Experiments indicate that our method is much faster than state-of-the-art solvers such as Pegasos, TRON, SVMperf, and a recent primal coordinate descent implementation.

1. Introduction

Support vector machines (SVM) (Boser et al., 1992) are useful for data classification. Given a set of instance-label pairs $(x_i, y_i)$, $i = 1, \ldots, l$, $x_i \in \mathbb{R}^m$, $y_i \in \{-1, +1\}$, SVM requires the solution of the following unconstrained 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 $C > 0$ is a penalty parameter. Two common loss functions are:

$$\max(1 - y_i w^T x_i, 0) \text{ and } \max(1 - y_i w^T x_i, 0)^2.$$  

The former is called L1-SVM, while the latter is L2-SVM. In some applications, an SVM problem appears with a bias term $b$. One often deal with this term by appending each instance with an additional dimension:

$$x_i^T \leftarrow [x_i^T, 1] \quad w^T \leftarrow [w^T, b].$$

Problem (1) is often referred to as the primal form of SVM. One may instead solve its dual problem:

$$\min_{\alpha} \quad f(\alpha) = \frac{1}{2} \alpha^T Q \alpha - e^T \alpha$$
subject to $0 \leq \alpha_i \leq U, \forall i,$

where $Q = Q + D, D$ is a diagonal matrix, and $Q_{ij} = y_i y_j x_i^T x_j$. For L1-SVM, $U = C$ and $D_{ii} = 0, \forall i$. For L2-SVM, $U = \infty$ and $D_{ii} = 1/(2C), \forall i$.

An SVM usually maps training vectors into a high-dimensional space via a nonlinear function $\phi(x)$. Due to the high dimensionality of the vector variable $w$, one solves the dual problem (4) by the kernel trick (i.e., using a closed form of $\phi(x_i)^T \phi(x_j)$). We call such a problem as a nonlinear SVM. In some applications, data appear in a rich dimensional feature space, the performances are similar with/without nonlinear mapping. If data are not mapped, we can often train much larger data sets. We indicate such cases as linear SVM; these are often encountered in applications such as document classification. In this paper, we aim at solving very large linear SVM problems.

Recently, many methods have been proposed for linear SVM in large-scale scenarios. For L1-SVM, Zhang (2004), Shalev-Shwartz et al. (2007), Bottou (2007) propose various stochastic gradient descent methods. Collins et al. (2008) apply an exponentiated gradient method. SVMperf (Joachims, 2006) uses a cutting plane technique. Smola et al. (2008) apply bundle methods, and view SVMperf as a special case. For L2-SVM, Keerthi and DeCoste (2005) propose modified Newton methods. A trust region Newton method (TRON) (Lin et al., 2008) is proposed for logistic re-
A Dual Coordinate Descent Method for Large-scale Linear SVM

gression and L2-SVM. These algorithms focus on different aspects of the training speed. Some aim at quickly obtaining a usable model, but some achieve fast final convergence of solving the optimization problem in (1) or (4). Moreover, among these methods, Joachims (2006), Smola et al. (2008) and Collins et al. (2008) solve SVM via the dual (4). Others consider the primal form (1). The decision of using primal or dual is of course related to the algorithm design.

Very recently, Chang et al. (2008) propose using coordinate descent methods for solving primal L2-SVM. Experiments show that their approach more quickly obtains a useful model than some of the above methods. Coordinate descent, a popular optimization technique, updates one variable at a time by minimizing a single-variable sub-problem. If one can efficiently solve this sub-problem, then it can be a competitive optimization method. Due to the non-differentiability of the primal L1-SVM, Chang et al.’s work is restricted to L2-SVM. Moreover, as primal L2-SVM is differentiable but not twice differentiable, certain considerations are needed in solving the single-variable sub-problem.

While the dual form (4) involves bound constraints $0 \leq \alpha_i \leq U$, its objective function is twice differentiable for both L1- and L2-SVM. In this paper, we investigate coordinate descent methods for the dual problem (4). We prove that an $\epsilon$-optimal solution is obtained in $O(\log(1/\epsilon))$ iterations. We propose an implementation using a random order of sub-problems at each iteration, which leads to very fast training. Experiments indicate that our method is more efficient than the primal coordinate descent method. As Chang et al. (2008) solve the primal, they require the easy access of a feature’s corresponding data values. However, in practice one often has an easier access of values per instance. Solving the dual takes this advantage, so our implementation is simpler than Chang et al. (2008).

Early SVM papers (Mangasarian & Musicant, 1999; Friess et al., 1998) have discussed coordinate descent methods for the SVM dual form.\(^1\) However, they do not focus on large data using the linear kernel. Crammer and Singer (2003) proposed an online setting for multi-class SVM without considering large sparse data. Recently, Bordes et al. (2007) applied a coordinate descent method to multi-class SVM, but they focus on nonlinear kernels. In this paper, we point out that dual coordinate descent methods make crucial advantage of the linear kernel and outperform other solvers when the numbers of data and features are both large.

Coordinate descent methods for (4) are related to the popular decomposition methods for training nonlinear SVM. In this paper, we show their key differences and explain why earlier studies on decomposition methods failed to modify their algorithms in an efficient way like ours for large-scale linear SVM. We also discuss the connection to other linear SVM works such as (Crammer & Singer, 2003; Collins et al., 2008; Shalev-Shwartz et al., 2007).

This paper is organized as follows. In Section 2, we describe our proposed algorithm. Implementation issues are investigated in Section 3. Section 4 discusses the connection to other methods. In Section 5, we compare our method with state of the art implementations for large linear SVM. Results show that the new method is more efficient.

\section{A Dual Coordinate Descent Method}

In this section, we describe our coordinate descent method for L1- and L2-SVM. The optimization process starts from an initial point $\alpha^0 \in \mathbb{R}^d$ and generates a sequence of vectors $\{\alpha^k\}_{k=0}^\infty$. We refer to the process from $\alpha^k$ to $\alpha^{k+1}$ as an outer iteration. In each outer iteration we have $l$ inner iterations, so that sequentially $\alpha_1, \alpha_2, \ldots, \alpha_l$ are updated. Each outer iteration thus generates vectors $\alpha^{k,i} \in \mathbb{R}^d$, $i = 1, \ldots, l+1$, such that $\alpha^{k,1} = \alpha^k$, $\alpha^{k,i+1} = \alpha^{k+1}$, and

$$\alpha^{k,i} = [\alpha^{k+1}_1, \ldots, \alpha^{k+1}_{i-1}, \alpha^k_i, \ldots, \alpha^k_l]^T, \quad \forall i = 2, \ldots, l.$$  

For updating $\alpha^{k,i}$ to $\alpha^{k,i+1}$, we solve the following one-variable sub-problem:

$$\min_d f(\alpha^{k,i} + d e_i) \quad \text{subject to} \quad 0 \leq \alpha_i + d \leq U,$$  

where $e_i = [0, \ldots, 0, 1, 0, \ldots, 0]^T$. The objective function of (5) is a simple quadratic function of $d$:

$$f(\alpha^{k,i} + d e_i) = \frac{1}{2} P_{ii} d^2 + \nabla_i f(\alpha^{k,i}) d + \text{constant},$$  

where $\nabla_i f$ is the $i$th component of the gradient $\nabla f$. One can easily see that (5) has an optimum at $d = 0$ (i.e., no need to update $\alpha_i$) if and only if

$$\nabla_i^T f(\alpha^{k,i}) = 0,$$  

where $\nabla^T f(\alpha)$ means the projected gradient

$$\nabla_i^T f(\alpha) = \begin{cases} \nabla_i f(\alpha) & \text{if } 0 < \alpha_i < U, \\ \min(0, \nabla_i f(\alpha)) & \text{if } \alpha_i = 0, \\ \max(0, \nabla_i f(\alpha)) & \text{if } \alpha_i = U. \end{cases}$$

\footnote{Note that coordinate descent methods for unconstrained quadratic programming can be traced back to Hildreth (1957).}
Algorithm 1 A dual coordinate descent method for Linear SVM

- Given $\alpha$ and the corresponding $w = \sum_i y_i \alpha_i x_i$.
- While $\alpha$ is not optimal
  For $i = 1, \ldots, l$
    (a) $G = y_i w^T x_i - 1 + D_{ii} \alpha_i$
    (b) $PG = \begin{cases} 
      \min(G, 0) & \text{if } \alpha_i = 0, \\
      \max(G, 0) & \text{if } \alpha_i = U, \\
      G & \text{if } 0 < \alpha_i < U
    \end{cases}$
    (c) If $|PG| \neq 0$,
       $\bar{\alpha}_i \leftarrow \alpha_i$
       $\alpha_i \leftarrow \min(\max(\alpha_i - G/Q_{ii}, 0), U)$
       $w \leftarrow w + (\alpha_i - \bar{\alpha}_i) y_i x_i$

If (7) holds, we move to the index $i + 1$ without updating $\alpha_{i+1}$. Otherwise, we must find the optimal solution of (5). If $Q_{ii} > 0$, easily the solution is:

$$\alpha_{i+1} = \min \left( \left\{ \alpha_i - Q_{ii} f'(\alpha_i) - 1 \right\}, U \right).$$

We thus need to calculate $Q_{ii}$ and $\nabla_i f'(\alpha_i)$. First, $Q_{ii} = x_i^T x_i + D_{ii}$ can be precomputed and stored in the memory. Second, to evaluate $\nabla_i f'(\alpha_i)$, we use

$$\nabla_i f'(\alpha) = (\bar{Q} \alpha + 1 - \sum_{j=1}^l \bar{Q}_{ij} \alpha_j - 1.$$  

$\bar{Q}$ may be too large to be stored, so one calculates $\bar{Q}$'s $i$th row when doing (10). If $\bar{n}$ is the average number of nonzero elements per instance, and $O(\bar{n})$ is needed for each kernel evaluation, then calculating the $i$th row of the kernel matrix takes $O(\bar{n})$. Such operations are expensive. However, for a linear SVM, we can define

$$w = \sum_{j=1}^l y_j \alpha_j x_j,$$

so (10) becomes

$$\nabla_i f'(\alpha) = y_i w^T x_i - 1 + D_{ii} \alpha_i.$$

To evaluate (12), the main cost is $O(\bar{n})$ for calculating $w^T x_i$. This is much smaller than $O(l\bar{n})$. To apply (12), $w$ must be maintained throughout the coordinate descent procedure. Calculating $w$ by (11) takes $O(\bar{n})$ operations, which are too expensive. Fortunately, if $\bar{\alpha}_i$ is the current value and $\alpha_i$ is the value after the updating, we can maintain $w$ by

$$w \leftarrow w + (\alpha_i - \bar{\alpha}_i) y_i x_i.$$

The number of operations is only $O(\bar{n})$. To have the first $w$, one can use $\alpha^0 = 0$ so $w = 0$. In the end, we obtain the optimal $w$ of the primal problem (1) as the primal-dual relationship implies (11).

If $Q_{ii} = 0$, we have $D_{ii} = 0$, $Q_{ii} = x_i^T x_i = 0$, and hence $x_i = 0$. This occurs only in L1-SVM without the bias term by (3). From (12), if $x_i = 0$, then $\nabla_i f'(\alpha_{k,i}) = -1$. As $U = C < \infty$ for L1-SVM, the solution of (5) makes the new $\alpha_{k,i+1} = U$. We can easily include this case in (9) by setting $1/Q_{ii} = \infty$.

Briefly, our algorithm uses (12) to compute $\nabla_i f'(\alpha_{k,i})$, checks the optimality of the sub-problem (5) by (7), updates $\alpha_i$ by (9), and then maintains $w$ by (13). A description is in Algorithm 1. The cost per iteration (i.e., from $\alpha^k$ to $\alpha^{k+1}$) is $O(l\bar{n})$. The main memory requirement is on storing $x_1, \ldots, x_l$. For the convergence, we prove the following theorem using techniques in (Luo & Tseng, 1992):

**Theorem 1** For L1-SVM and L2-SVM, $\{\alpha_{k,i}\}$ generated by Algorithm 1 globally converges to an optimal solution $\alpha^*$. The convergence rate is at least linear: there are $0 < \mu < 1$ and an iteration $k_0$ such that

$$f(\alpha_{k+1}) - f(\alpha^*) \leq \mu(f(\alpha) - f(\alpha^*)), \forall k \geq k_0.$$  

The proof is in Appendix 7.1. The global convergence result is quite remarkable. Usually for a convex but not strictly convex problem (e.g., L1-SVM), one can only obtain that any limit point is optimal. We define an $\epsilon$-accurate solution $\alpha$ if $f(\alpha) \leq f(\alpha^*) + \epsilon$. By (14), our algorithm obtains an $\epsilon$-accurate solution in $O(\log(1/\epsilon))$ iterations.$^2$

3. Implementation Issues

3.1. Random Permutation of Sub-problems

In Algorithm 1, the coordinate descent algorithm solves the one-variable sub-problems in the order of $\alpha_1, \ldots, \alpha_l$. Past results such as (Chang et al., 2008) show that solving sub-problems in an arbitrary order may give faster convergence. This inspires us to randomly permute the sub-problems at each outer iteration. Formally, at the $k$th outer iteration, we permute $\{1, \ldots, l\}$ to $\{\pi(1), \ldots, \pi(l)\}$, and solve sub-problems in the order of $\alpha_{\pi(1)}, \alpha_{\pi(2)}, \ldots, \alpha_{\pi(l)}$. Similar to Algorithm 1, the algorithm generates a sequence $\{\alpha_{k,i}\}$ such that $\alpha_{k,1} = \alpha^k$, $\alpha_{k+1} = \alpha_{k+1}^+$ and

$$\alpha_{k,i} = \begin{cases} 
  \alpha_{i}^k & \text{if } \tau_{k-1}^{\pi(t)} < i, \\
  \alpha_{i}^k & \text{if } \tau_{k-1}^{\pi(t)} \geq i.
\end{cases}$$

$^2$A constant $k_0$ appears in (14). A newer result without needing $k_0$ is in Wang and Lin (2014).
The update from $\alpha^{k,i}$ to $\alpha^{k,i+1}$ is by

$$\alpha^{k,i+1}_i = \alpha^{k,i}_i + \text{arg min}_{0 \leq \alpha^{k,i}_i + d \leq U} f(\alpha^{k,i} + d e_i) \text{ if } \pi^{-1}_k(t) = i.$$ 

We prove that Theorem 1 is still valid. Hence, the new setting obtains an $\epsilon$-accurate solution in $O(\log(1/\epsilon))$ iterations. A simple experiment reveals that this setting of pruning subproblems is much faster than Algorithm 1. The improvement is also bigger than that observed in (Chang et al., 2008) for primal coordinate descent methods.

### 3.2. Shrinking

Eq. (4) contains constraints $0 \leq \alpha_i \leq U$. If an $\alpha_i$ is 0 or $U$ for many iterations, it may remain the same. To speed up decomposition methods for nonlinear SVM (discussed in Section 4.1), the shrinking technique (Joachims, 1998) reduces the size of the optimization problem without considering some bounded variables. Below we show it is much easier to apply this technique to linear SVM than the nonlinear case.

If $A$ is the subset after removing some elements and $\bar{A} = \{1, \ldots, l\} \setminus A$, then the new problem is

$$\min_{\alpha_A} \frac{1}{2} \alpha^T \bar{Q} \alpha_A + (\bar{Q} \bar{A} \alpha_{\bar{A}} - e_A)^T \alpha_A$$

subject to $0 \leq \alpha_i \leq U, i \in A,$

(15)

where $\bar{Q}_{AA}, \bar{Q}_{A\bar{A}}$ are sub-matrices of $\bar{Q}$, and $\alpha_{\bar{A}}$ is considered as a constant vector. Solving this smaller problem consumes less time and memory. Once (15) is solved, we must check if the vector $\alpha$ is optimal for (4). This check needs the whole gradient $\nabla f(\alpha)$.

$$\nabla_i f(\alpha) = \bar{Q}_{i,AA} \alpha_A + \bar{Q}_{i,A\bar{A}} \alpha_{\bar{A}} - 1,$$

if $i \in A$, and one stores $\bar{Q}_{i,\bar{A}} \alpha_{\bar{A}}$ before solving (15), we already have $\nabla_i f(\alpha)$. However, for $i \notin A$, we must calculate the corresponding rows of $\bar{Q}$. This step, referred to as the reconstruction of gradients in training nonlinear SVM, is very time consuming. It may cost up to $O(l^2 n)$ if each kernel evaluation is $O(n)$.

For linear SVM, solving the smaller problem (15), we still have the vector

$$w = \sum_{i \in A} y_i \alpha_i x_i + \sum_{i \notin A} y_i \alpha_i x_i$$

though only the first part $\sum_{i \in A} y_i \alpha_i x_i$ is updated. Therefore, using (12), $\nabla f(\alpha)$ is easily available. Below we demonstrate a shrinking implementation so that reconstructing the whole $\nabla f(\alpha)$ is never needed.

Our method is related to what LIBSVM (Chang & Lin, 2011) uses. From the optimality condition of bound-constrained problems, $\alpha$ is optimal for (4) if and only if

$$\nabla^P f(\alpha) = 0,$$

where $\nabla^P f(\alpha)$ is the projected gradient defined in (8). We then prove the following result:

**Theorem 2** Let $\alpha^*$ be the convergent point of $\{\alpha^{k,i}\}$.

1. If $\alpha^*_i = 0$ and $\nabla_i f(\alpha^*) > 0$, then $\exists k_i$ such that $\forall k \geq k_i, \forall s, \alpha^{k,s}_i = 0$.
2. If $\alpha^*_i = U$ and $\nabla_i f(\alpha^*) < 0$, then $\exists k_i$ such that $\forall k \geq k_i, \forall s, \alpha^{k,s}_i = U$.
3. $\lim_{k \rightarrow \infty} \min_j \nabla^P_j f(\alpha^{k,j}) = \lim_{k \rightarrow \infty} \min_j \nabla^P_j f(\alpha^{k,j}) = 0$.

The proof is in Appendix 7.3. During the optimization procedure, $\nabla^P f(\alpha^*) \neq 0$, and in general $\max_j \nabla^P_j f(\alpha^*) > 0$ and $\min_j \nabla^P_j f(\alpha^*) < 0$. These two values measure how the current solution violates the optimality condition. In our iterative procedure, what we have are $\nabla_i f(\alpha^{k,i})$, $i = 1, \ldots, l$. Hence, at the $(k - 1)$st iteration, we obtain

$$M^{k-1} = \max_j \nabla^P_j f(\alpha^{k-1,j}), m^{k-1} = \min_j \nabla^P_j f(\alpha^{k-1,j}).$$

Then at each inner step of the $k$th iteration, before updating $\alpha^{k,i}_i$ to $\alpha^{k,i+1}_i$, this element is shrunken if one of the following two conditions holds:

$$\alpha^{k,i}_i = 0 \text{ and } \nabla_i f(\alpha^{k,i}) > \bar{M}^{k-1},$$

$$\alpha^{k,i}_i = U \text{ and } \nabla_i f(\alpha^{k,i}) < \bar{m}^{k-1},$$

(16)

where

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

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

In (16), $\bar{M}^{k-1}$ must be strictly positive, so we set it be $\infty$ if $M^{k-1} \leq 0$. From Theorem 2, elements satisfying the “if condition” of properties 1 and 2 meet (16) after certain iterations, and are then correctly removed for optimization. To have a more aggressive shrinking, one may multiply both $\bar{M}^{k-1}$ and $\bar{m}^{k-1}$ in (16) by a threshold smaller than one.

Property 3 of Theorem 2 indicates that with a tolerance $\epsilon$,

$$M^k - m^k < \epsilon$$

(17)

is satisfied after a finite number of iterations. Hence (17) is a valid stopping condition. We also use it for
A Dual Coordinate Descent Method for Large-scale Linear SVM

Table 1. A comparison between decomposition methods (Decomp.) and dual coordinate descent (DCD). For both methods, we consider that one $\alpha_i$ is updated at a time. We assume Decom. maintains gradients, but DCD does not. The average number of nonzeros per instance is $\bar{n}$.

<table>
<thead>
<tr>
<th>Update $\alpha_i$</th>
<th>Nonlinear SVM</th>
<th>Linear SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maintain $\nabla f(\alpha)$</td>
<td>$O(l)$</td>
<td>$O(l\bar{n})$</td>
</tr>
<tr>
<td>Decom. DCD</td>
<td>$O(l\bar{n})$</td>
<td>NA</td>
</tr>
<tr>
<td>Decom. DCD</td>
<td>$O(l\bar{n})$</td>
<td>NA</td>
</tr>
</tbody>
</table>

smaller problems (15). If at the $k$th iteration, (17) for (15) is reached, we enlarge $A$ to $\{1, \ldots, l\}$, set $M^k = \infty$, $\bar{m}^k = -\infty$ (so no shrinking at the $(k+1)$st iteration), and continue regular iterations. Thus, we do shrinking without reconstructing gradients.

In Appendix 7.4, we provide an algorithm to show the convergence and finite termination of the Algorithm 1 with shrinking.

3.3. An Online Setting

In some applications, the number of instances is huge, so going over all $\alpha_1, \ldots, \alpha_l$ causes an expensive outer iteration. Instead, one can randomly choose an index $i_k$ at a time, and update only $\alpha_{i_k}$ at the $k$th outer iteration. A description is in Algorithm 2. The setting is related to (Crammer & Singer, 2003; Collins et al., 2008). See also the discussion in Section 4.2.

4. Relations with Other Methods

4.1. Decomposition Methods for Nonlinear SVM

Decomposition methods are one of the most popular approaches for training nonlinear SVM. As the kernel matrix is dense and cannot be stored in the computer memory, decomposition methods solve a sub-problem of few variables at each iteration. Only a small number of corresponding kernel columns are needed, so the memory problem is resolved. If the number of variables is restricted to one, a decomposition method is like the online coordinate descent in Section 3.3, but it differs in the way it selects variables for updating.

It has been shown (Keerthi & DeCoste, 2005) that, for linear SVM decomposition methods are inefficient. On the other hand, here we are pointing out that dual coordinate descent is efficient for linear SVM. Therefore, it is important to discuss the relationship between decomposition methods and our method.

In early decomposition methods that were first proposed (Osuna et al., 1997; Platt, 1998), variables minimized at an iteration are selected by certain heuristics. However, subsequent developments (Joachims, 1998; Chang & Lin, 2011; Keerthi et al., 2001) all use gradient information to conduct the selection. The main reason is that maintaining the whole gradient does not introduce extra cost. Here we explain the detail by assuming that one variable of $\alpha$ is chosen and updated at a time\(^3\). To set-up and solve the sub-problem (6), one uses (10) to calculate $\nabla_i f(\alpha)$. If $O(\bar{n})$ effort is needed for each kernel evaluation, obtaining the $i$th row of the kernel matrix takes $O(l\bar{n})$ effort. If instead one maintains the whole gradient, then $\nabla_i f(\alpha)$ is directly available. After updating $\alpha_i^{k,i}$ to $\alpha_i^{k,i+1}$, we obtain $Q$’s $i$th row (same as the $i$th row due to the symmetry of $Q$), and calculate the new whole gradient:

$$\nabla f(\alpha^{k,i+1}) = \nabla f(\alpha^{k,i}) + \bar{Q}_{:,i}(\alpha_i^{k,i+1} - \alpha_i^{k,i})$$

where $\bar{Q}_{:,i}$ is the $i$th column of $\bar{Q}$. The cost is $O(l\bar{n})$ for $\bar{Q}_{:,i}$ and $O(l)$ for (18). Therefore, maintaining the whole gradient does not cost more. As using the whole gradient implies fewer iterations (i.e., faster convergence due to the ability to choose for updating the variable that violates optimality most), one should take this advantage. However, the situation for linear SVM is very different. With the different way (12) to calculate $\nabla_i f(\alpha)$, the cost to update one $\alpha_i$ is only $O(\bar{n})$. If we still maintain the whole gradient, evaluating (12) $l$ times takes $O(l\bar{n})$ effort. We gather this comparison of different situations in Table 1. Clearly, for nonlinear SVM, one should use decomposition methods by maintaining the whole gradient. However, for linear SVM, if $l$ is large, the cost per iteration without maintaining gradients is much smaller than that with. Hence, the coordinate descent method can be faster than the decomposition method by using many cheap iterations.

An earlier attempt to speed up decomposition methods for linear SVM is (Kao et al., 2004). However, it failed to derive our method here because it does not give up maintaining gradients.

4.2. Existing Linear SVM Methods

We discussed in Section 1 and other places the difference between our method and a primal coordinate descent method (Chang et al., 2008). Below we describe the relations with other linear SVM methods.

We mentioned in Section 3.3 that our Algorithm 2 is related to the online mode in (Collins et al., 2008). They aim at solving multi-class and structured problems. At each iteration an instance is used; then a sub-problem of several variables is solved. They approximately minimize the sub-problem, but for two-class case, one can exactly solve it by (9). For the\(^3\)Solvers like LIBSVM update at least two variables due to a linear constraint in their dual problems. Here (4) has no such a constraint, so selecting one variable is possible.
A Dual Coordinate Descent Method for Large-scale Linear SVM

Table 2. On the right training time for a solver to reduce the primal objective value to within 1% of the optimal value; see (20). Time is in seconds. The method with the shortest running time is boldfaced. Listed on the left are the statistics of data sets: \( l \) is the number of instances and \( n \) is the number of features.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Data statistics</th>
<th>Linear L1-SVM</th>
<th>Linear L2-SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( l ) ( n ) # nonzeros</td>
<td>DCDL1 Pegasos SVM(_{perf})</td>
<td>DCDL2 PCD TRON</td>
</tr>
<tr>
<td>a9a</td>
<td>32,561 123 451,592</td>
<td>0.2 1.1 6.0</td>
<td>0.4 0.1 1.0</td>
</tr>
<tr>
<td>astro-physic</td>
<td>62,369 99,757 4,834,550</td>
<td>0.2 2.8 2.6</td>
<td>0.2 0.5 1.2</td>
</tr>
<tr>
<td>real-sim</td>
<td>72,309 20,958 3,709,083</td>
<td>0.2 2.4 2.4</td>
<td>0.1 2.2 0.9</td>
</tr>
<tr>
<td>news20</td>
<td>19,996 1,355,191 9,097,916</td>
<td>0.5 10.3 20.0</td>
<td>0.2 2.4 5.2</td>
</tr>
<tr>
<td>yahoo-japan</td>
<td>176,203 832,026 23,506,415</td>
<td>1.1 12.7 69.4</td>
<td>1.0 2.9 38.2</td>
</tr>
<tr>
<td>rcv1</td>
<td>677,399 47,236 49,556,258</td>
<td>2.6 21.9 72.0</td>
<td>2.7 5.1 18.6</td>
</tr>
<tr>
<td>yahoo-korea</td>
<td>460,554 3,052,939 156,436,656</td>
<td>8.3 79.7 656.8</td>
<td>7.1 18.4 286.1</td>
</tr>
</tbody>
</table>

batch setting, our approach is different from theirs. The algorithm for multi-class problems in (Crammer & Singer, 2003) is also similar to our online setting. For the two-class case, it solves (1) with the loss function \( \max (-y_i w^T x_i, 0) \), which is different from (2). They do not study data with a large number of features.

Next, we discuss the connection to stochastic gradient descent (Shalev-Shwartz et al., 2007; Bottou, 2007). The most important step of this method is the following update of \( w \):

\[
\begin{align*}
\mathbf{w} &\leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}} (y_i, \mathbf{x}_i),
\end{align*}
\]

(19)

where \( \nabla_{\mathbf{w}} (y_i, \mathbf{x}_i) \) is the sub-gradient of the approximate objective function:

\[
\mathbf{w}^T \mathbf{w}/2 + C \max(1 - y_i \mathbf{w}^T \mathbf{x}_i, 0),
\]

and \( \eta \) is the learning rate (or the step size). While our method is dual-based, we maintain \( \mathbf{w} \) by (13). Both (13) and (19) use one single instance \( \mathbf{x}_i \), but they take different directions \( y_i \mathbf{x}_i \) and \( \nabla_{\mathbf{w}} (y_i, \mathbf{x}_i) \). The selection of the learning rate \( \eta \) may be the subtlest thing in stochastic gradient descent, but for our method this is never a concern. The step size \( (\alpha_i - \bar{\alpha}_i) \) in (13) is governed by solving a sub-problem from the dual.

5. Experiments

In this section, we analyze the performance of our dual coordinate descent algorithm for L1- and L2-SVM. We compare our implementation with state of the art linear SVM solvers. We also investigate how the shrinkage update of \( \mathbf{w} \) may be the subtlest thing in stochastic gradient descent, but for our method this is never a concern. The step size \( (\alpha_i - \bar{\alpha}_i) \) in (13) is governed by solving a sub-problem from the dual.

We set the penalty parameter \( C = 1 \) for comparison. For all data sets, the testing accuracy does not increase.

\(^4\)The equivalent setting for Pegasos is \( \lambda = 1/(CI) \). For SVM\(_{perf}\), its penalty parameter is \( C_{perf} = 0.01CI \).
A Dual Coordinate Descent Method for Large-scale Linear SVM

(a) L1-SVM: astro-physic
(b) L2-SVM: astro-physic
(c) L1-SVM: news20
(d) L2-SVM: news20
(e) L1-SVM: rcv1
(f) L2-SVM: rcv1

Figure 1. Time versus the relative error (20). DCDL1-S, DCDL2-S are DCDL1, DCDL2 with shrinking. The dotted line indicates the relative error 0.01. Time is in seconds.

Figure 2. Time versus the difference of testing accuracy between the current model and the reference model (obtained using strict stopping conditions). Time is in seconds.

Another evaluation is to consider how fast a solver obtains a model with reasonable testing accuracy. Using the optimal solutions from the above experiment, we generate the reference models for L1- and L2-SVM. We evaluate the testing accuracy difference between the current model and the reference model along the training time. Figure 2 shows the results. Overall, DCDL1 and DCDL2 are more efficient than other solvers. Note that we omit DCDL1-S and DCDL2-S in Figure 2, as the performances with/without shrinking are similar.

Among L1-SVM solvers, SVMperf is competitive with Pegasos for small data. But in the case of a huge number of instances, Pegasos outperforms SVMperf. However, Pegasos has slower convergence than DCDL1. As discussed in Section 4.2, the learning rate of stochastic gradient descent may be the cause, but for DCDL1 we exactly solve sub-problems to obtain the step size in updating $w$. Also, Pegasos has a jumpy test set performance while DCDL1 gives a stable behavior.

To compare these solvers, we consider the CPU time of reducing the relative difference between the primal objective value and the optimum to within 0.01:

$$|f^P(w) - f^P(w^*)|/|f^P(w^*)| \leq 0.01,$$  \hspace{2cm} (20)

where $f^P$ is the objective function of (1), and $f^P(w^*)$ is the optimal value. Note that for consistency, we use primal objective values even for dual solvers. The reference solutions of L1- and L2-SVM are respectively obtained by solving DCDL1 and DCDL2 until the duality gaps are less than $10^{-6}$. Table 2 lists the results. Clearly, our dual coordinate descent method for both L1- and L2-SVM is significantly faster than other solvers. To check details, we choose astro-physic, news20, rcv1, and show the relative error along time in Figure 1. In Section 3.2, we pointed out that the shrinking technique is very suitable for DCD. In Figure 1, we also include them (DCDL1-S and DCDL2-S) for comparison. Like in Table 2, our solvers are efficient for both L1- and L2-SVM. With shrinking, its performance is even better.

After $C \geq 4$. All the above methods are implemented in C/C++ with double precision. Some implementations such as (Bottou, 2007) use single precision to reduce training time, but numerical inaccuracy may occur. We do not include the bias term by (3).
DCDL1/DCDL2 in Table 2), DCDL2 is even better. Regarding TRON, as a Newton method, it possesses fast final convergence. However, since it costs more time at each iteration, it hardly generates a reasonable model quickly. From the experiment results, DCDL2 converges as fast as TRON, but also performs well in early iterations.

Due to the space limitation, we give the following observations without details. First, Figure 1 indicates that our coordinate descent method converges faster for L2-SVM than L1-SVM. As L2-SVM has the diagonal matrix $D$ with $D_{ii} = 1/(2C)$, we suspect that its $\bar{Q}$ is better conditioned, and hence leads to faster convergence. Second, all methods have slower convergence when $C$ is large. However, small $C$’s are usually enough as the accuracy is stable after a threshold. In practice, one thus should try from a small $C$. Moreover, if $n < l$ and $C$ is too large, then our DCDL2 is slower than TRON or PCD (see problem a9a in Table 2, where the accuracy does not change after $C \geq 0.25$). If $n < l$, clearly one should solve the primal, whose number of variables is just $n$. Such data are not our focus. Indeed, with a small number of features, one usually maps data to a higher space and train a nonlinear SVM. Third, we have checked the online Algorithm 2. Its performance is similar to DCDL1 and DCDL2 (i.e., batch setting without shrinking). Fourth, we have investigated real document classification which involves many two-class problems. Using the proposed method as the solver is more efficient than using others.

6. Discussion and Conclusions

We can apply the proposed method to solve regularized least square problems, which have the loss function $(1 - y_i w^T x_i)^2$ in (1). The dual is simply (4) without constraints, so the implementation is simpler.

In summary, we present and analyze an efficient dual coordinate decent method for large linear SVM. It is very simple to implement, and possesses sound optimization properties. Experiments show that our method is faster than state of the art implementations.

References


7. Proofs

7.1. Proof of Theorem 1

We consider the analysis in (Luo & Tseng, 1992), which studies coordinate descent methods for problems in the following form:

\[
\min_{\alpha} \quad f(\alpha) \\
\text{subject to} \quad L_i \leq \alpha_i \leq U_i,
\]

where

\[
f(\alpha) \equiv g(E\alpha) + b^T \alpha,
\]

and \(L_i \leq \alpha_i \leq U_i\) are lower/upper bounds. They establish the linear convergence of the coordinate descent method if (21) satisfies the following conditions:

1. \(E\) has no zero column.
2. The set of optimal solutions for (21), denoted by \(A^*\), is nonempty.
3. The function \(g\) is strongly convex and twice differentiable everywhere.

We explain that both L1-SVM and L2-SVM satisfy the above conditions. Let \(E\) be an \((n + l) \times l\) matrix

\[
E = \begin{bmatrix} y_1 x_1, \ldots, y_l x_l \end{bmatrix},
\]

and \(b = -e\). Then \(f(\alpha)\) defined in (4) can be represented as \(g(E\alpha) + b^T \alpha\) with \(g(\beta) = \beta^T \beta/2\). Clearly, \(f(\alpha)\) and \(g(\beta)\) are closed as they are continuous. Moreover, \(g(\beta)\) is strongly convex and twice differentiable everywhere. Regarding the set of optimal solutions, we consider Weierstrass’ Theorem, which states that any continuous function on a compact set attains its minimum. For L1-SVM, the feasible region \(\{\alpha \mid 0 \leq \alpha_i \leq U\}\) is compact. For L2-SVM, we check the level set \(\{\alpha \mid f(\alpha) \leq 0, \alpha_i \geq 0\}\). Since \(Q\) is positive semidefinite,

\[
\frac{1}{4C} \alpha^T \alpha - e^T \alpha \leq -\frac{1}{2} \alpha^T Q \alpha \leq 0.
\]

Hence,

\[
\frac{1}{4C} \sum_{i=1}^l (\alpha_i - 2C)^2 \leq lC \text{ implies } \alpha_i \leq 2C + C\sqrt{l}.
\]

The level set is thus compact. Therefore, for both L1- and L2-SVM, their optimal solution sets are nonempty.

Next, we check if \(E\) has zero column. This situation happens only if \(x_i = 0\) and \(D_{ii} = 0\), the case of L1-SVM without a bias term. Let \(A = \{i \mid x_i \neq 0\}\). We explain that elements not in this set can be eliminated for consideration, so we still have a matrix \(E\) without zero column. As \(Q_{ij} = 0\), if \(i \notin A\) or \(j \notin A\), essentially we are minimizing

\[
\frac{1}{2} \alpha_A^T Q_A \alpha_A - e_A^T \alpha_A - \sum_{i \notin A} \alpha_i.
\]

Clearly, if \(i \notin A\), then \(\alpha_i = U\) is optimal. Moreover, \(\nabla_i f(\alpha) = -1\) (also see the discussion in Section 2), so at the first iteration of Algorithm 1, \(\alpha_i\) is set to \(U\) and remains the same forever. Therefore, after the first iteration, we minimize a function of \(\alpha_A\), which has a nonzero matrix \(E\). Finally, we have all conditions ready. Theorem 2.1 of (Luo & Tseng, 1992) then implies Theorem 1.

7.2. Lemma 1

To prove Theorems 2 and 3, we need the following lemma.
Lemma 1 Assume \( \{\alpha^k\} \) is an infinite sequence of solving (4) and \( \alpha^* \) is any optimal solution. If \( \{\alpha^k\} \) satisfies the following conditions:

1. \( \alpha^{k+1} = \alpha^k + \delta e_i \) for some index \( i \), and \( \delta \) is the optimal solution of

\[
\min_{\delta} f(\alpha^k + \delta e_i) \quad \text{subject to} \quad 0 \leq \alpha^k + \delta \leq U,
\]

(23)

2. every limit point of \( \{\alpha^k\} \) is a stationary point,

then we have

1. If \( \alpha^*_i = 0 \) and \( \nabla_i f(\alpha^*) > 0 \), then \( \exists k_i \) such that \( \forall k \geq k_i, \alpha^k_i = 0 \).
2. If \( \alpha^*_i = U \) and \( \nabla_i f(\alpha^*) < 0 \), then \( \exists k_i \) such that \( \forall k \geq k_i, \alpha^k_i = U \).
3. \( \lim_{k \to \infty} \nabla f(\alpha^k) = 0 \).

Proof. From the primal-dual relation of SVM, we have \( \sum_{i=1}^l y_i \alpha_i^* x_i = w^* \), where \( w^* \) is the optimal solution of the primal. Since primal problem (1) is strictly convex, \( w^* \) is unique. In Appendix 7.1, we show that \( \{\alpha^k\} \) for both L1- and L2-SVM is in a compact set. With condition 2,

\[
\lim_{k \to \infty} \sum_{i=1}^l y_i \alpha^k_i x_i = \lim_{k \to \infty} w^k = w^*.
\]

This indicates that for all \( i \),

\[
\lim_{k \to \infty} \nabla_i f(\alpha^k) = \lim_{k \to \infty} y_i (w^k)^T x_i - 1 = y_i (w^*)^T x_i - 1.
\]

(24)

We define \( d^*_i = \nabla_i f(\alpha^*) = y_i (w^*)^T x_i - 1 \). By (24), there exists an index \( k_i \) such that for all \( k \geq k_i \),

\[
\nabla_i f(\alpha^k) > 0 \text{ if } d^*_i > 0,
\]

\[
\nabla_i f(\alpha^k) < 0 \text{ if } d^*_i < 0.
\]

(25)

When we update \( \alpha^k \) to \( \alpha^{k+1} \) by changing the \( i \)th element, from the KKT condition of the sub-problem,

\[
\nabla_i f(\alpha^{k+1}) > 0 \Rightarrow \alpha^k_i = 0,
\]

\[
\nabla_i f(\alpha^{k+1}) < 0 \Rightarrow \alpha^k_i = U.
\]

(26)

Therefore, after \( k \geq k_i \), either \( \alpha_i \) is upper- (lower-) bounded forever or \( 0 < \alpha_i < U \) and the \( i \)th element is never selected for update. The second situation implies that there is a limit point \( \tilde{\alpha} \) with \( 0 < \tilde{\alpha}_i < U \) and \( \nabla_i f(\tilde{\alpha}) \neq 0 \), a contradiction to the assumption that every limit point is optimal. Hence for \( k > k_i \),

\[
\alpha^k_i = 0 \text{ if } d^*_i > 0,
\]

\[
\alpha^k_i = U \text{ if } d^*_i < 0.
\]

(27)

which are the the first and the second results. This result and (25) imply for \( k \geq k_i \),

\[
\nabla_i^P f(\alpha^k) = 0 \text{ if } d^*_i > 0 \text{ or } d^*_i < 0.
\]

(28)

For any index \( i \) with \( d^*_i \), (24) implies that \( \lim_{k \to \infty} \nabla_i^P f(\alpha^k) = 0 \). With (28), \( \lim_{k \to \infty} \nabla_i^P f(\alpha^k) = 0 \).

7.3. Proof of Theorem 2

From Theorem 1 and the update rule (9), Algorithm 1 satisfies the two conditions in Lemma 1. The results in Lemma 1 directly prove Theorem 2.

7.4. Finite Termination of the Algorithm 1 with Shrinking

Algorithm 3 gives details of Algorithm 1 with the shrinking heuristic. In Algorithm 3, the while loop generates a sequence \( \{\alpha^k \ | \ k = 1, 2, \ldots \} \). To obtain \( \alpha^{k+1} \) from \( \alpha^k \), Step 2 of Algorithm 3 sequentially updates each component of \( \alpha^k \) by solving (5). Vectors \( \alpha^{k,i}, i = 1, \ldots, l+1 \) are generated so that \( \alpha^{k,1} = \alpha^k \), \( \alpha^{k,i+1} = \alpha^{k,i+1} \). Notice that Step 2 only updates unshrunk variables in \( A \), so we define \( \alpha^{k,i+1} = \alpha^{k,i} \) if \( i \notin A \). Moreover, \( \bar{M} \) and \( \bar{m} \) in Algorithm 3 correspond to \( M^{k-1} \) and \( m^{k-1} \) in (16), respectively. \( M \) and \( m \) are \( M^k \) and \( m^k \), respectively. The random permutation of sub-problems discussed in Section 3.1 can be easily incorporated to Algorithm 3, though we omit it here for simplification. In this section, we show that Algorithm 3 terminates after finite iterations.

We assume \( C > 0 \) as otherwise \( \nabla^P f(\alpha) \) in (8) is not well defined.

Lemma 2 Consider the sub-problem (23) of updating the \( i \)th element. We have

\[
f(\alpha^{k,i}) - f(\alpha^{k,i+1}) \geq \frac{1}{2} Q_{ii}(\alpha^{k,i}_i - \alpha^{k,i+1}_i)^2.
\]

Proof. From the KKT condition (26) of the sub-problem,

\[
\nabla_i f(\alpha^{k,i+1}) > 0 \Rightarrow \alpha^{k,i}_i - \alpha^{k,i+1}_i \geq 0;
\]

\[
\nabla_i f(\alpha^{k,i+1}) < 0 \Rightarrow \alpha^{k,i}_i - \alpha^{k,i+1}_i \leq 0.
\]

Therefore,

\[
f(\alpha^{k,i}) - f(\alpha^{k,i+1}) = \nabla_i f(\alpha^{k,i+1})(\alpha^{k,i}_i - \alpha^{k,i+1}_i) + \frac{1}{2} Q_{ii}(\alpha^{k,i}_i - \alpha^{k,i+1}_i)^2,
\]

\[
\geq \frac{1}{2} Q_{ii}(\alpha^{k,i}_i - \alpha^{k,i+1}_i)^2.
\]
Algorithm 3 Dual coordinate descent with shrinking

- Given $\epsilon, \alpha$ and the corresponding $w = \sum_i y_i \alpha_i x_i$.
- Let $M \leftarrow -\infty, \bar{m} \leftarrow -\infty$.
- Let $A \leftarrow \{1, \ldots, l\}$.
- While
  1. Let $M \leftarrow -\infty, m \leftarrow \infty$.
  2. For all $i \in A$
     (a) $G = y_i w^T x_i - 1 + D_i \alpha_i$
     (b) $PG \leftarrow 0$
        If $\alpha_i = 0$,
        $A \leftarrow A \setminus \{i\}$ if $G > M$
        $PG \leftarrow G$ if $G < 0$
        Else If $\alpha_i = U$,
        $A \leftarrow A \setminus \{i\}$ if $G < \bar{m}$
        $PG \leftarrow G$ if $G > 0$
        Else
        $PG \leftarrow G$
     (c) $M \leftarrow \max(M, PG), m \leftarrow \min(m, PG)$
     (d) If $|PG| \neq 0$,
        $\bar{\alpha}_i \leftarrow \alpha_i$
        $\alpha_i \leftarrow \min(\max(\alpha_i - G/\bar{Q}_{ii}, 0), U)$
        $w \leftarrow w + (\alpha_i - \bar{\alpha}_i)y_i x_i$
  3. If $M - m < \epsilon$
     (a) If $A = \{1, \ldots, l\}$,
        BREAK.
     Else,
        $A \leftarrow \{1, \ldots, l\}, \bar{M} \leftarrow -\infty, \bar{m} \leftarrow -\infty$.
        (i.e., no shrinking at the next iteration)
  4. If $M \leq 0$, then $\bar{M} \leftarrow \infty$. Else $\bar{M} \leftarrow M$.
  5. If $m \geq 0$, then $\bar{m} \leftarrow -\infty$. Else $\bar{m} \leftarrow m$.

Theorem 3 Algorithm 3 terminates in finite iterations.

Proof. Similar to the proof of Theorem 1, for any index $i$ with $\bar{Q}_{ii} = 0$, after the first iteration, $\nabla_i f(\alpha) = -1$, and $\alpha_i$ remains at $U$. Thus, we only consider the index set $\{i \mid \bar{Q}_{ii} > 0\}$.

If Algorithm 3 does not terminate, it will generate an infinite sequence $\{\alpha^k \mid k = 1, 2, \ldots\}$. Section 3.2 indicates that if the problem of using variables in $A$ satisfies the stopping condition (15) at iteration $k$, then $A$ becomes the whole set $\{1, \ldots, l\}$ (step 3a of Algorithm 3), and all $\alpha$'s elements are checked at the next iteration. We collect those iterations as a subsequence $\{\alpha^k\}_R$.

We claim that $\{\alpha^k\}_R$ is an infinite sequence. If it is finite, consider iterations after the last element of $R$. Since $l$ is finite, there exists a subset $A \subset \{1, \ldots, l\}$ and a number $p$ such that all iterations after the $p$th iteration are for solving the problem (15) of $\alpha_A$. However, Property 3 of Theorem 2 indicates that our procedure will satisfy (17) in finite iterations. Thus, $\{\alpha^k \mid k = 1, 2, \ldots\}$ is a finite sequence, a contradiction to the assumption that Algorithm 3 does not terminate.

Now consider a subsequence $\bar{R} \subset R$ such that $\{\alpha^k \mid k \in \bar{R}\}$ converges to $\alpha^\ast$. We then claim that

$$\lim_{k \in \bar{R}, k \to \infty} \alpha^{k,i} = \alpha^\ast, \forall i = 1, \ldots, l + 1.$$ (29)

Assume $\sigma = \min_i \bar{Q}_{ii}$. Since we only consider the index set $\{i \mid \bar{Q}_{ii} > 0\}$, we have $\sigma > 0$. Then from Lemma 2 we have

$$f(\alpha^{k,1}) - f(\alpha^{k,2}) \geq \frac{\sigma}{2}(\alpha^{k,1} - \alpha^{k,2})^2.$$ (30)

Furthermore, the sequence $\{f(\alpha^{k,i})\}$ is decreasing, and condition 2 in the proof of Theorem 1 shows that...
We have the optimum of the sub-problem \( \alpha^{k,t} \in \{ t \} \) subject to 0 \( \leq \alpha^*_t + d \leq U \).

By similar derivations, we obtain (29).

If \( \alpha^* \) is not a stationary point, there exists an index \( t \in \{ 1, \ldots, l \} \) such that \( \nabla_{\alpha}^t f(\alpha^*) \neq 0 \). Assume \( \hat{d} \) is the optimum of the sub-problem

\[
\min_d f(\alpha^* + \hat{d} e_t) \quad \text{subject to} \quad 0 \leq \alpha^*_t + d \leq U.
\]

We have

\[
f(\alpha^* + \hat{d} e_t) < f(\alpha^*) \quad (31)
\]

from the assumption that \( \nabla_{\alpha}^t f(\alpha^*) \neq 0 \). Besides, as \( \alpha^{k,t+1} \) solves the sub-problem (6),

\[
f(\min(\max(\alpha^{k,t} + \hat{d} e_t, 0), U e)) \geq f(\alpha^{k,t+1}),
\]

where \( \min(\cdot), \max(\cdot) \) are component-wisely operated, and \( e \) is the vector of ones. Taking the limit of both sides, we get \( f(\alpha^* + \hat{d} e_t) \geq f(\alpha^*) \), a contradiction to (31). Thus, any limit point of \( \{ \alpha^k \}_R \) is an optimal solution.

Since any limit point of \( \{ \alpha^k \}_R \) is optimal, and the update rule ensures the objective value is decreasing through \( k = 1, 2, \ldots \), we have that for all \( i \) any limit point of \( \{ \alpha^{k,i} \} \) is an optimal solution. Then from Lemma 1, \( \nabla_P f(\alpha^{k,i}) \) globally converges to 0, so the stopping condition \( M^k - m^k < \epsilon \) is satisfied in finite iterations.

### 7.5. Convergence of Algorithm 2

Nesterov (2012) studied the convergence of random coordinate descent methods, where Algorithm 2 is a special case. In this section, we rewrite his proof to specifically prove the convergence of Algorithm 2. Assume \( \{ \alpha^k \}_{k=1}^\infty \) is the sequence generated by Algorithm 2 and at the \( k \)th iteration, \( i_k \) is selected so that \( d^k_{i_k} \) is updated. We prove the following convergence result:

**Theorem 4** Let

\[
\phi_k \equiv E_{i_1, \ldots, i_{k-1}} f(\alpha^k).
\]

For any \( k \geq 1 \),

\[
\phi_k - f(\alpha^*) \leq \frac{1}{l+k-1} \left( \frac{1}{2} r^2 + f(\alpha^1) - f(\alpha^*) \right).
\]

In addition, if \( f(\alpha) \) is strongly convex in \( \| \cdot \|_1 \) with a constant \( \sigma \), then

\[
\phi_k - f(\alpha^*) \leq (1 - \frac{2\sigma}{l(1+\sigma)})^{k-1} \left( \frac{1}{2} r^2 + f(\alpha^1) - f(\alpha^*) \right). \quad (32)
\]

Here \( \| \cdot \|_1 \) is defined by

\[
\| d \|_1^2 = \sum_{i=1}^l \tilde{Q}_{ii} d^2_t,
\]

\( \alpha^* \) is an optimal solution, and

\[
r_k^2 \equiv \| \alpha^k - \alpha^* \|_1^2 = \sum_{i=1}^l \tilde{Q}_{ii} (\alpha^k_i - \alpha^*_i)^2. \quad (33)
\]
Proof. By the definition of $r_k^2$ in (33), we have

$$r_{k+1}^2 = r_k^2 + 2\tilde{Q}_{ikk}(\alpha^k_{ik} - \alpha^k_{ik})(\alpha^k_{ik} - \alpha^*_{ik})$$

$$+ \hat{Q}_{ikkk}(\alpha^k_{ik} - \alpha^k_{ik})^2$$

$$= r_k^2 + 2\tilde{Q}_{ikk}(\alpha^k_{ik} - \alpha^k_{ik})(\alpha^k_{ik} - \alpha^*_{ik})$$

$$- \hat{Q}_{ikkk}(\alpha^k_{ik} - \alpha^k_{ik})^2$$

$$\leq r_k^2 + 2\nabla_if(\alpha^k)(\alpha^k_{ik} - \alpha^k_{ik+1})$$

$$- \hat{Q}_{ikkk}(\alpha^k_{ik} - \alpha^k_{ik})^2$$

$$= r_k^2 + 2\nabla_if(\alpha^k)(\alpha^k_{ik} - \alpha^k_{ik})$$

$$+ 2\left(f(\alpha^k) - f(\alpha^k_{ik+1})\right).$$

From (34) to (35), we use the optimality condition of the sub-problem (5):

$$\nabla_if(\alpha^k_{ik+1})(\alpha^k_{ik} - \alpha^k_{ik+1}) \geq 0$$

and

$$\nabla_if(\alpha^k_{ik+1}) = \nabla_if(\alpha^k) + \tilde{Q}_{ikk}(\alpha^k_{ik+1} - \alpha^k_{ik}).$$

Taking expectation on $i_k$, (36) becomes

$$E_i\left[\frac{1}{2}r_{k+1}^2 + f(\alpha^{k+1}) - f(\alpha^*)\right]$$

$$\leq \frac{1}{2}r_k^2 + f(\alpha^k) - f(\alpha^*) + \frac{1}{l}\nabla f(\alpha^k)^T(\alpha^* - \alpha^k).$$

Using (37), for any $k \geq 1$,

$$E_{i_1,\ldots,i_k}\left[\frac{1}{2}r_{k+1}^2 + f(\alpha^{k+1}) - f(\alpha^*)\right]$$

$$\leq \frac{1}{2}r_k^2 + f(\alpha^k) - f(\alpha^*) + \frac{1}{l}\sum_{j=1}^{k}E_{i_1,\ldots,i_{j-1}}[\nabla f(\alpha^j)^T(\alpha^* - \alpha^j)].$$

Since $r_{k+1}^2/2 \geq 0$, (38) becomes

$$\frac{1}{2}r_k^2 + f(\alpha^k) - f(\alpha^*)$$

$$\geq E_{i_1,\ldots,i_k}[f(\alpha^{k+1})] - f(\alpha^*)$$

$$+ \frac{1}{l}\sum_{j=1}^{k}E_{i_1,\ldots,i_{j-1}}[\nabla f(\alpha^j)^T(\alpha^j - \alpha^*)]$$

$$\geq \phi_{k+1} - f(\alpha^*) + \frac{1}{l}\sum_{j=1}^{k}(\phi_j - f(\alpha^*))$$

$$\geq (1 + \frac{k}{l})(\phi_{k+1} - f(\alpha^*)).$$

The derivation from (39) to (40) is by

$$f(\alpha^*) \geq f(\alpha^k) + \nabla f(\alpha^k)^T(\alpha^* - \alpha^k)$$

and (40) to (41) is by the fact that $f(\alpha^k)$ is decreasing.

Then, (41) leads to the first result of the theorem. For the second result, we need the following property that there exists $\sigma > 0$ such that

$$\nabla f(\alpha)^T(\alpha - \alpha^*)$$

$$\geq f(\alpha) - f(\alpha^*) + \frac{\sigma}{2}\|\alpha - \alpha^*\|_1^2$$

$$\geq \sigma\|\alpha - \alpha^*\|_1^2.$$}

We obtain (42) and (43) respectively from that $f(\alpha)$ is strongly convex in $\|\cdot\|_1$ and the optimality at $\alpha^*$:

$$f(\alpha^*) \geq f(\alpha^* + \nabla f(\alpha^*^T(\alpha^* - \alpha^*) + \frac{\sigma}{2}\|\alpha - \alpha^*\|_1^2$$

$$\geq f(\alpha^*) + \frac{\sigma}{2}\|\alpha - \alpha^*\|_1^2.$$}

Using (42)-(43), (37) becomes

$$E_{i_1,\ldots,i_k}\left[\frac{1}{2}r_{k+1}^2 + f(\alpha^{k+1}) - f(\alpha^*)\right]$$

$$\leq \frac{1}{2}r_k^2 + f(\alpha^k) - f(\alpha^*) + \frac{\beta}{l}\nabla f(\alpha^k)^T(\alpha^* - \alpha^k)$$

$$+ \frac{1-\beta}{l}\nabla f(\alpha^k)^T(\alpha^* - \alpha^k)$$

$$\leq \frac{1}{2}r_k^2 + f(\alpha^k) - f(\alpha^*) - \frac{\beta}{l}\left(f(\alpha^k) - f(\alpha^*)\right)$$

$$+ \frac{\sigma}{2}\|\alpha^k - \alpha^*\|_1^2 - \frac{1-\beta}{l}\sigma\|\alpha^k - \alpha^*\|_1^2$$

$$= \frac{1}{2}r_k^2 + f(\alpha^k) - f(\alpha^*) - \frac{\beta}{l}\left(f(\alpha^k) - f(\alpha^*)\right)$$

$$+ \frac{\sigma r_k^2}{2} - \frac{1-\beta}{l}\sigma r_k^2.$$
where $\beta$ is any real number in $[0, 1]$. Let $\beta = \frac{2\sigma}{1+\sigma}$ and we have

$$
(44) = (1 - \frac{2\sigma}{l(1+\sigma)})\frac{1}{2r_k^2} + f(\alpha^k) - f(\alpha^*) .
$$

Taking the expectation of (45) on $i_1, \ldots, i_k$, we obtain the second result (32).

Next, we show that the objective function of L2-SVM is strongly convex in $\|\cdot\|_1$. Because $Q$ is positive semi-definite and $\bar{Q} = Q + D$ in (4) for L2-SVM, we have

$$
(\nabla f(\alpha^1) - \nabla f(\alpha^2))^T(\alpha^1 - \alpha^2) \\
= (\alpha^1 - \alpha^2)^T\bar{Q}(\alpha^1 - \alpha^2) \\
= (\alpha^1 - \alpha^2)^T(Q + D)(\alpha^1 - \alpha^2) \\
\geq \sum_{i=1}^l \frac{1}{2C}(\alpha_i^1 - \alpha_i^2)^2 .
$$

Define

$$
\sigma = \frac{1}{2C \max_i(Q_{ii})},
$$

then

$$
(46) \geq \sigma \sum_{i=1}^l \bar{Q}_{ii}(\alpha_i^1 - \alpha_i^2)^2 = \sigma \|\alpha^1 - \alpha^2\|_1^2 .
$$

Thus, $f(\alpha)$ is strongly convex in $\|\cdot\|_1$, so the second result holds for L2-SVM. For L1-SVM, if $Q$ is strictly convex, we can use a similar argument with $\frac{1}{2C}$ replaced by the smallest positive eigen-value of $Q$ to show that $f(\alpha)$ is strongly convex in $\|\cdot\|_1$. In this case the second result holds for L1-SVM.

8. Detailed Results of a9a, real-sim, yahoo-japan and yahoo-korea Data Sets

In Figures 1 and 2, we presented the results of astrophic, news20 and rcv1. Figures 3 and 4 show the comparison results of other data sets.