Stochastic Dual Coordinate Ascent for Learning Sign Constrained Linear Predictors

Miya Nakajima, Rikuto Mochida, Yuya Takada, and Tsuyoshi Kato

Graduate School of Science and Technology, Gunma University,
Tenjin, Kiryu, 376-8515, Japan

Abstract. *Sign constraints* are a handy representation of domain-specific prior knowledge that can be incorporated to machine learning. Under the sign constraints, the signs of the weight coefficients for linear predictors cannot be flipped from the ones specified in advance according to the prior knowledge. This paper presents new stochastic dual coordinate ascent (SDCA) algorithms that find the minimizer of the empirical risk under the sign constraints. Generic surrogate loss functions can be plugged into the proposed algorithm with the strong convergence guarantee inherited from the vanilla SDCA. A technical contribution of this work is the finding of an efficient algorithm that performs the SDCA update with a cost linear to the number of input features which coincides with the SDCA update without the sign constraints. Eventually, the computational cost $O(nd)$ is achieved to attain an ϵ-accuracy solution. Pattern recognition experiments were carried out using a classification task for microbiological water quality analysis. The experimental results demonstrate the powerful prediction performance of the sign constraints.

Keywords: sign constraints, convex optimization, stochastic dual coordinate ascent, empirical risk minimization, microbiological water quality analysis.

1 Introduction

Machine learning problems for linear prediction are often formulated as an *empirical risk minimization* (ERM) problem [4]. Let $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n$ be input vectors in $\mathbb{R}^d$, let $\phi_1, \dots, \phi_n : \mathbb{R} \to \mathbb{R}$ be convex loss functions, and let λ be a positive regularization constant. The ERM problem discussed in this paper is described as follows:

$$\begin{aligned} &\min \quad P(\boldsymbol{w}) \quad \text{wrt } \boldsymbol{w} \in \mathbb{R}^d, \\ &\text{where} \quad P(\boldsymbol{w}) := \frac{\lambda}{2}\|\boldsymbol{w}\|^2 + \frac{1}{n}\sum_{i=1}^{n} \phi_i(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle). \end{aligned} \tag{1}$$

Support vector machines (SVM) are recovered if we set the loss functions to the hinge loss $\phi_i(s) = \max(0, 1 - y_i s)$ where $y_i \in \{\pm 1\}$ are the class labels. Setting the loss functions to the log loss $\phi_i(s) = \log(1+\exp(-y_i s))$, logistic regression is obtained. With y_i continuous labels, setting the square error loss function $\phi_i(s) = \frac{1}{2}(y_i - s)^2$ yields the ridge regression.

Recently, Tajima et al [15] constrained the signs of the weights $\boldsymbol{w}$ to the linear SVM algorithm, and demonstrated the effectiveness of the *sign constraints* in the application to a biological sequence classification. The sign constraints are given to some of coefficients in the weight vector $\boldsymbol{w} = [w_1, \dots, w_d]^\top$. For some pre-defined subset of indices $\mathcal{I}_\geq \subseteq [d]$, where $[d] := \{1, \dots, d\}$, the non-negative constraints $w_h \geq 0$ are given for every $h \in \mathcal{I}_\geq$, and for another pre-defined subset $\mathcal{I}_\leq \subseteq ([d] \setminus \mathcal{I}_\geq)$, the non-positive constraints $w_h \leq 0$ are given for every $h \in \mathcal{I}_\leq$.

The sign constraints explicitly avoid violation of the prior knowledge for the directions of correlations between features and class labels. Negative weight coefficients w_h are undesired if positive correlation between the h-th features and the class label is known in advance. Nevertheless, without the sign constraints, a portion of coefficients w_h can be negative, which degrades the generalization performance. Similarly, positive weight coefficients are unfavorable if negative correlation to the class label is known in advance. Posing the sign constraints prevent the coefficients from falling into such an unfavorable region.

In this paper, we extended Tajima et al's work to a wide class of the sign-constrained ERM problems. The proposed algorithms solve a dual problem instead of minimizing the primal objective directly, which enables us to use a clear termination criterion which is the difference between the primal objective and the dual objective values. When the difference between the primal objective and the dual objective values is below a threshold, the primal objective gap is ensured to be smaller than the threshold. Tajima et al employed the Frank-Wolfe algorithm [6] for solving the dual problem, while in this study, the stochastic dual coordinate ascent (SDCA) framework [14] was adopted. This change brought an attractive advantage for the time complexity; the new algorithm is theoretically guaranteed to possess the exponential convergence [11] when a smooth loss such as the log loss is employed, whereas the convergence rate of Tajima et al's algorithm is merely sub-linear.

This paper is organized as follows. Related work is discussed in the next section. In Section 3, the learning problem with the sign constraints is formulated and its dual problem is described. After the general SDCA framework is introduced in Section 4, the implementations of SDCA iterations for the sign constrained learning problem are presented in Section 5. The experimental results for runtime comparison and the application to microbiological water quality analysis are reported in Section 6, followed by the last section concluding this paper.

2 Related work

The sign constraints have been used widely in regression and classification. Readers familiar with machine learning may recognize the sign constrained regression as one of components important to the non-negative matrix factorization [10]. For classification, Tajima et al [15] developed the sign-constrained support vector machines, and Fernandes et al [15] studied other loss functions. For the square error loss function, computational stable and fast optimization algorithms are available [9,8,2]. For the hinge loss function, Tajima et al developed a Frank-Wolfe optimization algorithm [6]. Meanwhile, without sign constraints, there are many stable optimization algorithms for generic empirical risk minimization [12,13,7,16,3]. However, to the best of our knowledge, algorithms for optimizing with generic loss functions under sign constraints have not been studied well so far.

3 Primal and dual problem

The goal of this work is develop an optimization algorithm for the following ERM problem:

$$\min \quad P(\boldsymbol{w}) \quad \text{wrt } \boldsymbol{w} \in \mathcal{S} \subseteq \mathbb{R}^d. \tag{2}$$

Therein, the set $\mathcal{S}$, of which a firm expression shall be given below, is the feasible region narrower than $\mathbb{R}^d$ due to the sign constraints. To express $\mathcal{S}$, we use $\boldsymbol{\sigma} \in \{1, 0\}^d$ where its h-th entry is given by $\sigma_h = 1$ for $h \in \mathcal{I}_{\geq}$ and $\sigma_h = 0$ for $h \in \mathcal{I}_0 := [d] \setminus \mathcal{I}_{\geq}$, where we have here assumed that all the non-positive constraints are in advance transformed to be the non-negative constraints by negating features $x_{h,i}$ for $h \in \mathcal{I}_{\leq}$, where $x_{h,i}$ is the h-th entry in $\boldsymbol{x}_i \in \mathbb{R}^d$. Then, the primal feasible region can be expressed as

$$\mathcal{S} := \{\boldsymbol{w} \in \mathbb{R}^d \mid \forall h \in [d],\, \sigma_h w_h \geq 0\}. \tag{3}$$

The optimization algorithm is based on SDCA framework that maximizes the *Fenchel dual* of the primal objective function. The Fenchel dual [1], say $D : \mathbb{R}^n \to \bar{\mathbb{R}}$, where $\bar{\mathbb{R}} := \mathbb{R} \cup \{\pm\infty\}$, is expressed as

$$D(\boldsymbol{\alpha}) := -\frac{1}{2\lambda n^2}\left\|\boldsymbol{\pi}\left(\sum_{i=1}^{n}\alpha_i \boldsymbol{x}_i\right)\right\|^2 - \frac{1}{n}\sum_{i=1}^{n}\phi_i^*(-\alpha_i), \tag{4}$$

where $\boldsymbol{\alpha} := [\alpha_1, \dots, \alpha_n]^\top \in \mathbb{R}^n$ is a dual variable vector, $\phi_i^* : \mathbb{R} \to \bar{\mathbb{R}}$ is the convex conjugate of ϕ_i, and $\boldsymbol{\pi}(\boldsymbol{v}) := \boldsymbol{v} - \max(\boldsymbol{0}, -\boldsymbol{\sigma} \odot \boldsymbol{v})$. The loss function ϕ_i is assumed to be $1/\gamma$-smooth. For example, the log loss is 0.25-smooth. The quadratic hinge loss defined as

$$\phi_i(s) := \frac{1}{2}(\max\{0, 1 - y_i s\})^2 \tag{5}$$

and the smoothed hinge loss defined as

$$\phi_i(s) := \begin{cases} \frac{1-2y_i s}{2} & \text{for } s < 0, \\ \frac{1}{2}(\max\{0, 1 - y_i s\})^2 & \text{if } s \geq 0 \end{cases} \tag{6}$$

are both 1-smooth. The convex conjugates of $(1/\gamma)$-smooth convex functions are a γ-strongly convex function [5]. That is, $\forall \eta \in [0, 1]$,

$$\eta\phi_i^*(-u) + (1-\eta)\phi_i^*(-\alpha) \geq \phi_i^*(-\eta u - (1-\eta)\alpha) + \frac{\gamma}{2}(u-\alpha)^2(1-\eta)\eta. \tag{7}$$

The SDCA framework uses the above inequality rearranged as

$$\phi_i^*(-\alpha) - \phi_i^*(-\alpha - \eta(u-\alpha)) \geq (\phi_i^*(-\alpha) - \phi_i^*(-u))\eta + \frac{\gamma}{2}(u-\alpha)^2(1-\eta)\eta. \tag{8}$$

Once the maximizer of $D(\boldsymbol{\alpha})$, denoted by $\boldsymbol{\alpha}_\star := [\alpha_1^\star, \dots, \alpha_n^\star]^\top$, is found, the optimal solution to the primal problem (2) can be recovered by

$$\boldsymbol{w}_\star = \frac{1}{\lambda n}\boldsymbol{\pi}\left[\sum_{i=1}^{n}\alpha_i^\star \boldsymbol{x}_i\right]. \tag{9}$$

4 SDCA framework

SDCA updates only one randomly selected entry in the dual variable vector $\boldsymbol{\alpha}$ at every iteration. Let i be the index of the selected entry in $\boldsymbol{\alpha}$. Denote by $\Delta\alpha$ the difference of the randomly selected entry from the previous value: $\alpha_i^{(t)} := \alpha_i^{(t-1)} + \Delta\alpha$. Let

$$\bar{\boldsymbol{w}}^{(t)} := \frac{1}{\lambda n}\sum_{i'=1}^{n}\alpha_{i'}^{(t)}\boldsymbol{x}_{i'}. \quad \text{and} \quad \boldsymbol{w}^{(t)} := \boldsymbol{\pi}\left[\bar{\boldsymbol{w}}^{(t)}\right]. \tag{10}$$

Once $\Delta\alpha$ is determined in each iteration, this vector $\bar{\boldsymbol{w}}^{(t)}$ can be updated with $O(d)$ costs as

$$\bar{\boldsymbol{w}}^{(t)} = \bar{\boldsymbol{w}}^{(t-1)} + \frac{\Delta\alpha}{\lambda n}\boldsymbol{x}_i. \tag{11}$$

For the simplicity of notation, we here shall drop the superscript $(t-1)$, to denote

$$\boldsymbol{w} := \boldsymbol{w}^{(t-1)}, \quad \boldsymbol{v}_0 := \bar{\boldsymbol{w}}^{(t-1)}, \quad \text{and} \quad \boldsymbol{\alpha} := \boldsymbol{\alpha}^{(t-1)}. \tag{12}$$

Algorithm 1: SDCA algorithm for maximizing $D(\boldsymbol{\alpha})$.

```
begin
    Choose $\boldsymbol{\alpha}^{(0)}$ s.t. $\boldsymbol{\alpha}^{(0)} \in \mathrm{dom}(-D)$;
    for $t := 1$ to $T$ do
        Pick $i$ randomly from $\{1, \dots, n\}$;
        $\eta_t \in \operatorname{argmax}_{\eta \in [0,1]} J_t^1(\eta)$;
        $\boldsymbol{\alpha}^{(t)} := \boldsymbol{\alpha}^{(t-1)} - (\nabla\phi(\langle \boldsymbol{w}^{(t-1)}, \boldsymbol{x}_i \rangle) + \alpha_i^{(t-1)})\eta_t \boldsymbol{e}_i$;
        Compute $\bar{\boldsymbol{w}}^{(t)}$ and $\boldsymbol{w}^{(t)}$;
    end
end
```

It is ideal to choose the maximizer of the function:

$$\begin{aligned} J_t^0(\Delta\alpha) &:= D(\boldsymbol{\alpha} + \Delta\alpha \boldsymbol{e}_i) - D(\boldsymbol{\alpha}) \\ &= \frac{\lambda}{2} \left\| \boldsymbol{\pi}\left[\boldsymbol{v}_0\right] \right\|^2 - \frac{\lambda}{2} \left\| \boldsymbol{\pi}\left[\boldsymbol{v}_0 + \frac{\Delta\alpha}{\lambda n} \boldsymbol{x}_i \right] \right\|^2 + \frac{1}{n}\left(\phi_i^*(-\alpha) - \phi_i^*(-\alpha - \Delta\alpha)\right). \end{aligned} \tag{13}$$

Since $\Delta\alpha$ is still in the argument of ϕ_i^*, finding the optimal $\Delta\alpha$ is complicated in general. To obtain a closed-form update rule, the range of $\Delta\alpha$ is restricted such that

$$\eta := -\frac{\Delta\alpha}{\alpha_i + \nabla\phi(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle)} \in [0, 1] \tag{14}$$

if $\alpha_i + \nabla\phi(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle) \neq 0$; otherwise $\Delta\alpha := 0$. Hereinafter, we discuss only the non-trivial case of $u := -\nabla\phi(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle) \neq \alpha_i$, where α_i is the i-th entry in $\boldsymbol{\alpha}^{(t-1)}$. Then, $\Delta\alpha = q\eta$ where $q := u - \alpha_i$. Let $\boldsymbol{v}_q := \frac{q}{\lambda n}\boldsymbol{x}_i$. By applying the inequality (8), $J_t^0(q\eta)$ is bounded from below as

$$\begin{aligned} J_t^0(q\eta) &= \frac{\lambda}{2} \left\| \boldsymbol{\pi}\left[\boldsymbol{v}_0\right] \right\|^2 - \frac{\lambda}{2} \left\| \boldsymbol{\pi}\left[\boldsymbol{v}_0 + \eta \boldsymbol{v}_q\right] \right\|^2 + \frac{1}{n}\left(\phi^*(-\alpha_i) - \phi^*(-\alpha_i - q\eta)\right) \\ &\geq \frac{\lambda}{2} \left\| \boldsymbol{\pi}\left[\boldsymbol{v}_0\right] \right\|^2 - \frac{\lambda}{2} \left\| \boldsymbol{\pi}\left[\boldsymbol{v}_0 + \eta \boldsymbol{v}_q\right] \right\|^2 + a_{\mathrm{offs}}\eta^2 + b_{\mathrm{offs}}\eta =: J_t^1(\eta) \end{aligned} \tag{15}$$

where

$$a_{\mathrm{offs}} := -\frac{q^2\gamma}{2n}, \quad \text{and} \quad b_{\mathrm{offs}} := \frac{\phi^*(-\alpha) - \phi^*(-u) + 0.5q^2\gamma}{n}. \tag{16}$$

The lower bound J_t^1 is more amenable than J_t^0 because the variable to be optimized is not in the argument list of an arbitrary loss function. The exponential convergence of SDCA is still guaranteed even if J_t^1 is maximized instead of J_t^0 [14]. The SDCA for learning under sign constraints is summarized in Algorithm 1. In the next section, how to implement Line 5 in Algorithm 1 shall be discussed.

5 Implementations for SDCA iteration

In this section, an algorithm for finding the maximizer of $J_t^1(\eta)$ is presented. A key ingredient found in this study is the fact that J_t^1 is a piecewise concave quadratic function. This finding enabled us to develop efficient algorithms for the update rule. Below, an explicit form of the piecewise quadratic function shall be presented (Subsection 5.1), followed by descriptions of two algorithms to find the maximizer of $J_t^1(\eta)$ (Subsections 5.2 and 5.3).

5.1 Piecewise quadratic form

Denote by $v_{h,0}$ and $v_{h,q}$ the h-th entries of $\boldsymbol{v}_0$ and $\boldsymbol{v}_q$, respectively. Let $\mathcal{I}_0 := \{h \in [d] \mid \sigma_h = 0\}$. Define $\boldsymbol{\theta} := [\theta_1, \dots, \theta_{d_t}, \theta_{d_t+1}]^\top$ such that $0 = \theta_1 < \cdots < \theta_{d_t+1} = 1$ where $\theta_1, \dots, \theta_{d_t}, \theta_{d_t+1}$ are the elements of a set $\Theta \subset \mathbb{R}$ such as $\mathrm{Card}[\Theta] = d_t + 1$ defined as

$$\Theta := \{0,1\} \cup \left\{\theta \in (0,1) \,\middle|\, \exists h \in \mathcal{I}_{\geq} \text{ s.t. } v_{h,0} = -\theta v_{h,q} \neq 0\right\}. \tag{17}$$

The element θ_k for $k \in \{2, \dots, d_t\}$ is the position at which for some $h \in \mathcal{I}_{\geq}$ the affine function $\eta \mapsto v_{h,0} + \eta v_{h,q}$ crosses the horizontal axis. Figure 1 shows a numerical example of the affine functions where $d = 3$, $\mathcal{I}_{\geq} = \{1,2,3\}$, $\boldsymbol{v}_0 = [0.5, 0.75, -0.5]^\top$, and $\boldsymbol{v}_q = [0.5, -1, 1]^\top$. From the definition of Θ, we have $d_t = 3$, $\theta_1 = 0$, $\theta_2 = 0.5$, $\theta_3 = 0.75$, and $\theta_4 = 1$. It is observed that the affine function $\eta \mapsto v_{3,0} + \eta v_{3,q}$ crosses the horizontal axis at $\eta = \theta_2$, and the affine function $\eta \mapsto v_{2,0} + \eta v_{2,q}$ crosses the horizontal axis at $\eta = \theta_3$.

Let us define index sets, for $k \in [d_t]$,

$$\mathcal{H}_k := \mathcal{I}_0 \cup \left\{h \in \mathcal{I}_{\geq} \,\middle|\, 2v_{h,0} + (\theta_k + \theta_{k+1})v_{h,q} > 0\right\}. \tag{18}$$

In the case of the example depicted in Figure 1, the index sets are $\mathcal{H}_1 = \{1,2\}$, $\mathcal{H}_2 = \{1,2,3\}$, and $\mathcal{H}_3 = \{1,3\}$, from the definition (18). For $h \in \mathcal{H}_k \cap \mathcal{I}_{\geq}$, the affine functions $\eta \mapsto v_{h,0} + \eta v_{h,q}$ are over the horizontal axis. Namely, it holds that

$$\forall \eta \in (\theta_k, \theta_{k+1}), \quad \forall h \in \mathcal{H}_k, \quad v_{h,0} + \eta v_{h,q} > 0, \tag{19}$$

which leads to $\forall \eta \in (\theta_k, \theta_{k+1})$,

$$[\boldsymbol{\pi}(\boldsymbol{v}_0 + \eta \boldsymbol{v}_q)]_h = \begin{cases} v_{h,0} + \eta v_{h,q} & \text{for } h \in \mathcal{H}_k, \\ 0 & \text{for } h \notin \mathcal{H}_k \end{cases} \tag{20}$$

where $[\boldsymbol{\pi}(\boldsymbol{v}_0 + \eta \boldsymbol{v}_q)]_h$ is the h-th entry in the d-dimensional vector $\boldsymbol{\pi}(\boldsymbol{v}_0 + \eta \boldsymbol{v}_q)$. The vector $\boldsymbol{\theta}$ and the sets $\mathcal{H}_k$ for $k \in [d_t]$ result in a piecewise quadratic expression for the function J_t^1:

$$\forall \eta \in [\theta_k, \theta_{k+1}], \quad J_t^1(\eta) = a_k \eta^2 + b_k \eta \tag{21}$$

where a_k and b_k are given by

$$a_k = a_{\text{offs}} - \frac{\lambda}{2} \sum_{h \in \mathcal{H}_k} v_{h,q}^2, \quad \text{and} \quad b_k = b_{\text{offs}} - \lambda \sum_{h \in \mathcal{H}_k} v_{h,q} v_{h,0}. \tag{22}$$

5.2 $O(d^2)$ implementation

Due to the concavity and the differentiability of J_t^1, one of the maximizers of $J_t^1(\eta)$, denoted by $\eta_\star$, can be found as follows.

- If $\nabla J_t^1(0) = b_1 \leq 0$, then $\eta_\star = 0$;
- if $\nabla J_t^1(1) = 2a_{d_t+1} + b_{d_t+1} \geq 0$, then $\eta_\star = 1$;
- otherwise, there exists $k_\star \in [d_t]$ such that the interval $[\theta_{k_\star}, \theta_{k_\star+1}]$ contains a maximizer $\eta_\star = -0.5 b_{k_\star} / a_{k_\star}$.

The interval index $k_\star$ in the third case (i.e. $2a_{d_t+1} + b_{d_t+1} < 0 < b_1$) can be found by checking every interval, because it holds that $\nabla J_t^1(\theta_{k_\star}) \geq 0 \geq \nabla J_t^1(\theta_{k_\star+1})$ due to the differentiability of J_t^1. Combining this discussion and the aforementioned observations, each iteration of SDCA can be implemented as follows.

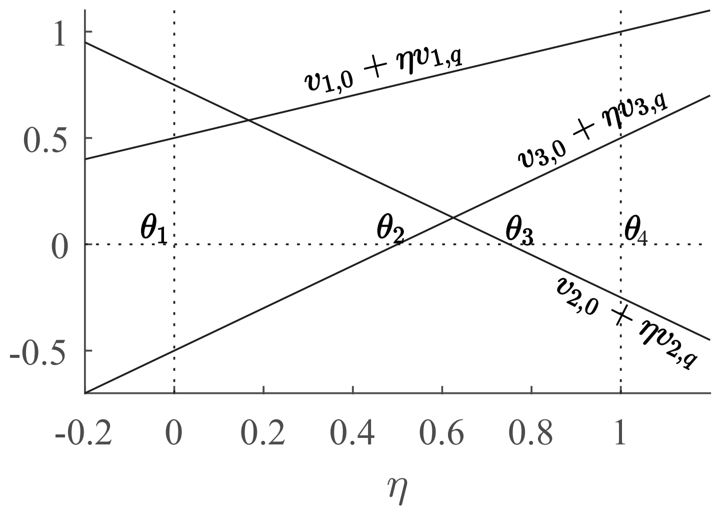

Fig. 1. Example of affine functions $\eta \mapsto v_{h,0} + \eta v_{h,q}$.

1. Pick $i \in [n]$ at random;	$O(1)$.
2. Compute $\boldsymbol{v}_0$ and $\boldsymbol{v}_q$;	$O(d)$.
3. Determine Θ;	$O(d)$.
4. Sort the elements in Θ;	$O(d \log d)$.
5. Compute $\mathcal{H}_k$ for $k \in [d_t]$;	$O(d^2)$.
6. Compute coefficients (a_k, b_k) for $k \in [d_t]$;	$O(d^2)$.
7. Find the maximizer $\eta_\star$;	$O(d)$.
8. $\Delta\alpha = q\eta_\star$;	$O(1)$.
9. Compute $\bar{\boldsymbol{w}}^{(t)}$ by (11);	$O(d)$.

This implementation enables each iteration to run within $O(d^2)$ computational cost. The most heavy steps in this implementation are the step computing index sets $\mathcal{H}_k$ (i.e. Step 5) and the step computing coefficients a_k, b_k (i.e. Step 6), both of which pays $O(d^2)$ cost. These time complexities are derived as follows. Observe that the number of pieces of the piecewise quadratic function is bounded as $d_t \le \mathrm{Card}(\mathcal{I}_\ge) + 2 \le d + 2 = O(d)$. For $k \in [d_t]$, each $\mathcal{H}_k$ is computed with $O(d)$ time since $\mathcal{H}_k \subseteq [d]$. Hence, it is proved that the time complexity of Step 5 is $O(d^2)$. Since $\mathrm{Card}(\mathcal{H}_k) = O(d)$, computation of $2d_t (= O(d))$ coefficients, $a_1, b_1, \dots, a_{d_t}, b_{d_t}$, using (22) consumes $O(d^2)$ cost in total.

Meanwhile, we found another algorithm that cut down the time complexity to a linear cost if ignoring the logarithmic term. The linear-time algorithm shall be presented below.

5.3 $O(d \log d)$ implementation

Here, another algorithm that exactly maximizes $J_t^1(\eta)$ with respect to $\eta \in [0,1]$ is presented. The theoretical time complexity of the algorithm given in Subsection 5.2 is $O(d^2)$, whereas the time complexity of the algorithm presented below is reduced to $O(d \log d)$. Defining $\mathcal{H}_{k,\mathrm{in}} := \mathcal{H}_k \setminus \mathcal{H}_{k-1}$ and $\mathcal{H}_{k,\mathrm{out}} := \mathcal{H}_{k-1} \setminus \mathcal{H}_k$ allows us to recursively express the coefficients of the piecewise quadratic functions as $\forall k \ge 2$,

$$\begin{aligned} a_k &:= a_{k-1} - \frac{\lambda}{2} \sum_{h \in \mathcal{H}_{k,\mathrm{out}}} v_{h,q}^2 + \frac{\lambda}{2} \sum_{h \in \mathcal{H}_{k,\mathrm{in}}} v_{h,q}^2, \\ b_k &:= b_{k-1} - \lambda \sum_{h \in \mathcal{H}_{k,\mathrm{out}}} v_{h,q} v_{h,0} + \lambda \sum_{h \in \mathcal{H}_{k,\mathrm{in}}} v_{h,q} v_{h,0}. \end{aligned} \tag{23}$$

To use (23) to compute a_k and b_k, the sets $\mathcal{H}_{k,\text{in}}$ and $\mathcal{H}_{k,\text{out}}$ as well as $\mathcal{H}_1$ are required beforehand. The set $\mathcal{H}_1$ can be obtained within $O(d)$ by checking whether one of the following three conditions is satisfied: (i) $h \in \mathcal{I}_0$; (ii) $v_{h,0} > 0$; (iii) $v_{h,0} = 0$ and $v_{h,q} > 0$. Namely, if $h \in [d]$ satisfies one of the three above conditions, then $h \in \mathcal{H}_1$; otherwise $h \notin \mathcal{H}_1$. We now discuss how to compute $\mathcal{H}_{k,\text{in}}$ and $\mathcal{H}_{k,\text{out}}$. To this end, we first compute $\tilde{\theta}^{\circ}_h := -\frac{v_{h,0}}{v_{h,q}}$ for all $h \in \mathcal{I}_{\geq}$. The $(d_t - 1)$ end points $\theta_2, \dots, \theta_{d_t}$ are then obtained by sorting the values of $\tilde{\theta}^{\circ}_h$, eliminating the values outside the open interval $(0, 1)$, and excluding duplicate values. During the process for computing $\boldsymbol{\theta}$, the sets $\mathcal{H}_{k,\text{in}}$ and $\mathcal{H}_{k,\text{out}}$ for $k \in \{2, \dots, d_t\}$ can be computed simultaneously as

$$\begin{aligned}\mathcal{H}_{k,\text{in}} &= \left\{ h \in \mathcal{I}_{\geq} \,\middle|\, \theta_k = \tilde{\theta}^{\circ}_h,\ v_{h,q} > 0 \right\}, \quad \text{and} \\ \mathcal{H}_{k,\text{out}} &= \left\{ h \in \mathcal{I}_{\geq} \,\middle|\, \theta_k = \tilde{\theta}^{\circ}_h,\ v_{h,q} < 0 \right\}.\end{aligned} \tag{24}$$

From these discussions, the $O(d^2)$ implementation given in Subsection 5.2 can be modified as follows.

1. Pick $i \in [n]$ at random;	$O(1)$.
2. Compute $\boldsymbol{v}_0$ and $\boldsymbol{v}_q$;	$O(d)$.
3. Compute $\mathcal{H}_1$;	$O(d)$.
4. Compute $\tilde{\theta}^{\circ}_h$ for $h \in \mathcal{I}_{\geq}$;	$O(d)$.
5. Compute $(\mathcal{H}_{k,\text{in}}, \mathcal{H}_{k,\text{out}})$ and θ_k for $k \in [d_t]$;	$O(d \log d)$.
6. Compute coefficients (a_k, b_k) for $k \in [d_t]$;	$O(d)$.
7. Find the maximizer $\eta_\star$;	$O(d)$.
8. $\Delta\alpha = q\eta_\star$;	$O(1)$.
9. Compute $\bar{\boldsymbol{w}}^{(t)}$ by (11);	$O(d)$.

Step 5 takes $O(d \log d)$ cost for sorting $\tilde{\theta}^{\circ}_h$ because the number of values to be sorted is $\text{Card}(\mathcal{I}_{\geq}) = O(d)$. The computational cost for Line 6 is $O(d)$ since the relationship

$$\bigcup_{k=2}^{d_t} \mathcal{H}_{k,\text{in}} \subseteq \mathcal{I}_{+} \subseteq [d] \quad \text{and} \quad \bigcup_{k=2}^{d_t} \mathcal{H}_{k,\text{out}} \subseteq \mathcal{I}_{+} \subseteq [d] \tag{25}$$

leads to the fact that an upper bound of the number of the total terms in (23) for all $k \in \{2, \dots, d_t\}$ is $4d$. Thus, it can be shown that each iteration of SDCA can be done within $O(d \log d)$ computation.

6 Experiments

6.1 Pattern recognition performance

We conducted experiments to demonstrate the effects of the sign constraints on the pattern recognition performance. For a pattern recognition task, we selected the microbiological water quality analysis. We used three water quality datasets named *NY top*, *NY bottom*, and *Indian* provided by kaggle.com. The three datasets contain 534, 523, and 896 examples, respectively. Each example has a target variate representing the fecal coliform (FC) and five other feature variates. The positive and negative class variates, respectively, were given to FC over and below the median, to pose a binary classification problem. Three loss functions, the *smoothed hinge*, the *quadratic hinge*, the *logistic hinge*, were examined. For each loss function, the conventional learning and the sign-constrained learning were performed. Then, six linear predictors were obtained in total. Accuracy (i.e. the sum of true

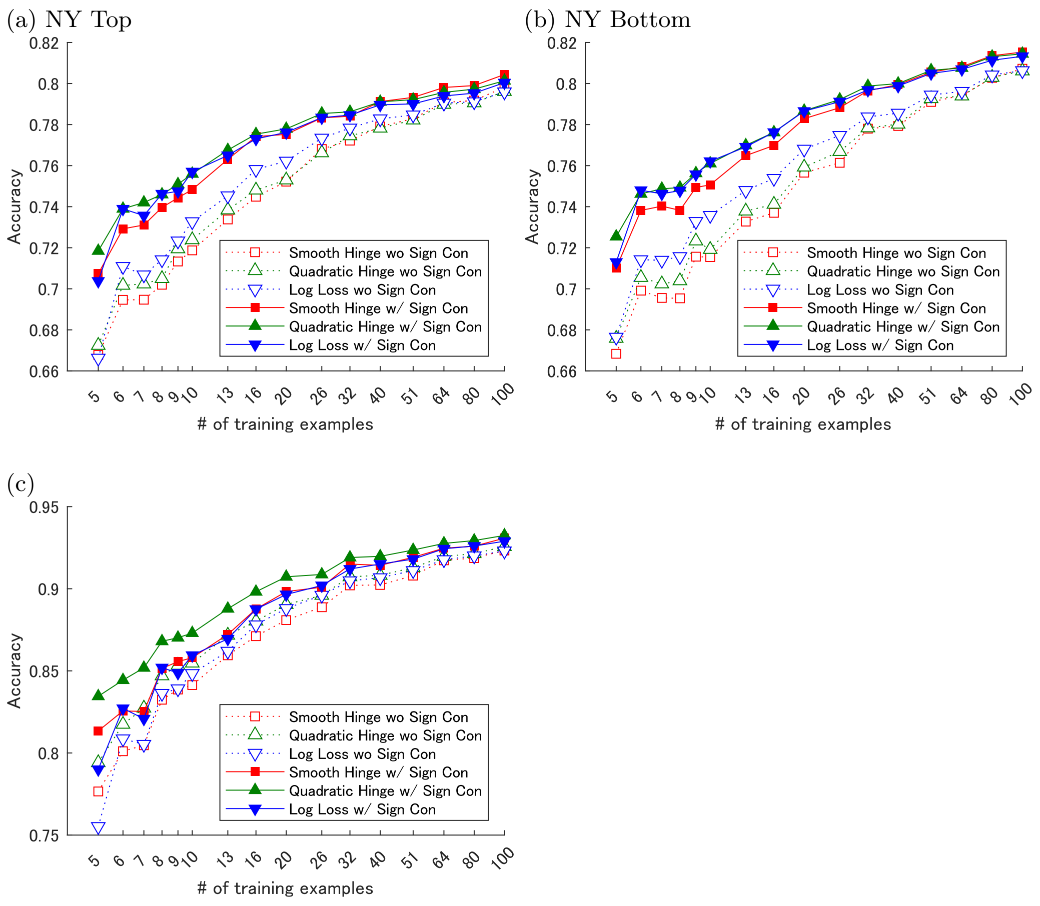

Fig. 2. Prediction performances on three datasets. (a) NY Top; (b) NY Bottom; (c) Indian.

positives and true negatives over the size of testing dataset) was used for the performance criterion. The number of training examples, n, was varied from 5 to 100. The n examples were picked at random from each dataset in the stratified manner. The n examples were fed to the six learning methods to get six predictors. The remaining $(n_{\text{tot}} - n)$ examples were used for testing. This procedure was repeated 200 times.

The averages of the 200 accuracies obtained for the three water quality datasets were plotted against the size of the training dataset, say n, in Figure 2 For all the three datasets and all the three loss functions, the sign constraints improved the prediction performance. In particular, the improvement was more significant when training examples were fewer. Sign constraints represent a sort of the domain-specific prior knowledge, and explicitly prevent the learning from violating the prior knowledge. Without sign constraints, when the sample size is small, the signs of weights in linear predictors may often be flipped from the true signs of correlations between the features and the class label. The improvement of the generalization performance must be the effect of the sign constraints that avoided the inversion of the weight signs.

6.2 Runtime

The two algorithms, presented in Subsections 5.2 and 5.3, were implemented with Cython 3.0.0a11 and run on a Linux machine equipped with Core i7-12700K and two 16GB DDR4 SDRAM. Feature vectors were generated with uniform distribution $U(-1, 1)$ and normal-

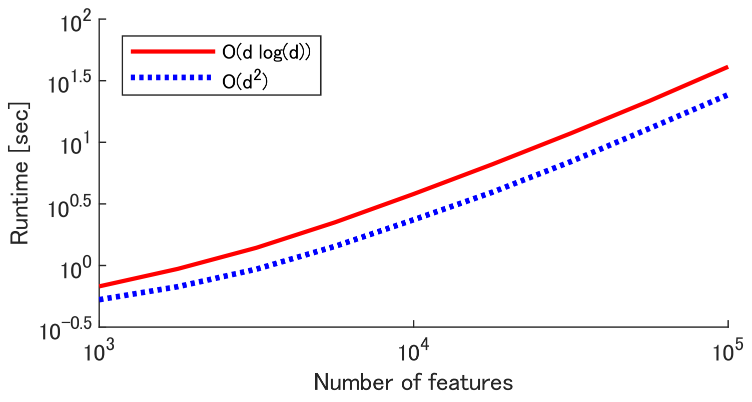

Fig. 3. Runtime for one epoch.

ized. Binary class labels were generated at random with equal probabilities. The size of training examples was fixed to $n = 500$. The number of features was varied from $d = 10^3$ to 10^5.

Figure 3 shows the runtimes consumed in one epoch. In conflict with the theoretical analysis, the two algorithm seemed to be a same time complexity, and the $O(d^2)$ algorithm always ran faster than the $O(d \log d)$ algorithm. To analyze why the inconsistency between the theory and the actual runtime happened, the numbers of pieces in the piecewise quadratic functions $J_t^1(\eta)$, say d_t, were counted, where we set $d_t = 0$ after convergence $(P(\boldsymbol{w}(\boldsymbol{\alpha}^{(t)})) - D(\boldsymbol{\alpha}^{(t)}) < 10^{-6})$. It was observed that the average numbers of pieces were around 1% of the number of features at the first epoch, and were less than 2.5 after the first epoch. It suggested that the actual number of pieces was much smaller than the number of features. Nevertheless, in our theoretical analysis, we used $d_t = O(d)$ which comes from a loose bound $d_t \leq \text{Card}(\mathcal{I}_{\geq}) \leq d$, resulting in the theoretical time complexity $O(d^2)$ for the implementation presented in Subsection 5.2. The difference of the upper bound from the actual number of pieces yielded the inconsistency between the theoretical runtime bound and the actual runtime.

7 Conclusions

In this paper, new algorithms for ERM under the sign constraints were presented. Tajima et al developed the Frank-Wolfe optimization algorithm for learning SVM under sign constraints. The algorithm developed in this study extends the class of ERM problems so that an arbitrary smooth and convex loss function can be employed. The optimization algorithm is based on the SDCA framework, which inherits a favorable property that guarantees the exponential convergence which is superior to the convergence rate of the Frank-Wolfe algorithm. The effects of the sign constraints on the pattern recognition were demonstrated with simulation experiments on microbiological water quality analysis using real-world data. Actual runtimes of the two SDCA algorithms developed in this study were compared, which suggested that the simpler $O(d^2)$ algorithm runs fast enough compared to the $O(d \log d)$ algorithm. Although pathogenic water quality analysis was illustrated as an application of the sign constraints in this paper, we believe the existence of a wide range of other applications in which the sign constraints work effectively. Exploring the other applications is left to future work.

References

1. Bertsekas, D.: Nonlinear Programming. Athena Scientific (1999)
2. Bierlaire, M., Toint, P., Tuyttens, D.: On iterative algorithms for linear least squares problems with bound constraints. Linear Algebra and its Applications **143**, 111–143 (jan 1991), doi: 10.1016/0024-3795(91)90009-l
3. Defazio, A., Bach, F., Lacoste-julien, S.: Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., Weinberger, K. (eds.) Advances in Neural Information Processing Systems 27. pp. 1646–1654. Curran Associates, Inc. (2014)
4. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning – Data Mining, Inference, and Prediction. Springer, 2nd edn. (2009)
5. Hiriart-Urruty, J.B.: Fundamentals of Convex Analysis. Springer (2001)
6. Jaggi, M.: Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In: Dasgupta, S., McAllester, D. (eds.) Proceedings of the 30th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 28, pp. 427–435. PMLR, Atlanta, Georgia, USA (17–19 Jun 2013)
7. Johnson, R., Zhang, T.: Accelerating stochastic gradient descent using predictive variance reduction. In: Advances in Neural Information Processing Systems 26: Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States. pp. 315–323 (2013)
8. Kim, D., Sra, S., Dhillon, I.S.: Tackling box-constrained optimization via a new projected quasi-newton approach. SIAM Journal on Scientific Computing **32**(6), 3548–3563 (jan 2010), doi:10.1137/08073812x
9. Lawson, C.L., Hanson, R.J.: Solving Least Squares Problems. Society for Industrial and Applied Mathematics (jan 1995), doi:10.1137/1.9781611971217
10. Lee, D.D., Seung, H.S.: Algorithms for non-negative matrix factorization. In: Advances in neural information processing systems. pp. 556–562 (2001)
11. Nesterov, Y.E.: Introductory Lectures on Convex Optimization: A Basic Course. Kluwer Academic Publishers (2003)
12. Roux, N.L., Schmidt, M., Bach, F.R.: A stochastic gradient method with an exponential convergence rate for finite training sets. In: Pereira, F., Burges, C., Bottou, L., Weinberger, K. (eds.) Advances in Neural Information Processing Systems 25. pp. 2663–2671. Curran Associates, Inc. (2012)
13. Schmidt, M., Roux, N.L., Bach, F.: Minimizing finite sums with the stochastic average gradient. Mathematical Programming **162**(1-2), 83–112 (jun 2016), doi:10.1007/s10107-016-1030-6
14. Shalev-Shwartz, S., Zhang, T.: Stochastic dual coordinate ascent methods for regularized loss. J. Mach. Learn. Res. **14**(1), 567–599 (Feb 2013)
15. Tajima, K., Tsuchida, K., Zara, E.R.R., Ohta, N., Kato, T.: Learning sign-constrained support vector machines. In: 2020 25th International Conference on Pattern Recognition (ICPR). IEEE (Jan 2021), doi:10.1109/icpr48806.2021.9412786
16. Xiao, L., Zhang, T.: A proximal stochastic gradient method with progressive variance reduction. SIAM Journal on Optimization **24**(4), 2057–2075 (jan 2014), doi:10.1137/140961791