Journal of Machine Learning Research 18 (2017) 1-42 Submitted 11/16; Revised 5/17; Published 7/17

Stochastic Primal-Dual Coordinate Method for Regularized Empirical Risk Minimization*

Yuchen Zhang ZHANGYUC@CS.STANFORD.EDU
Department of Computer Science
Stanford University
Stanford, CA 94305, USA

Lin Xiao LIN.XIAO@MICROSOFT.COM
Microsoft Research
Redmond, WA 98052, USA

Editor: Leon Bottou

Abstract

We consider a generic convex optimization problem associated with regularized empirical risk minimization of linear predictors. The problem structure allows us to reformulate it as a convex-concave saddle point problem. We propose a stochastic primal-dual coordinate (SPDC) method, which alternates between maximizing over a randomly chosen dual variable and minimizing over the primal variables. An extrapolation step on the primal variables is performed to obtain accelerated convergence rate. We also develop a mini-batch version of the SPDC method which facilitates parallel computing, and an extension with weighted sampling probabilities on the dual variables, which has a better complexity than uniform sampling on unnormalized data. Both theoretically and empirically, we show that the SPDC method has comparable or better performance than several state-of-the-art optimization methods.

Keywords: empirical risk minimization, randomized algorithms, convex-concave saddle point problems, primal-dual algorithms, computational complexity

1. Introduction

We consider a generic convex optimization problem that arises often in machine learning: regularized empirical risk minimization (ERM) of linear predictors. More specifically, let $a_1, \ldots, a_n \in \mathbb{R}^d$ be the feature vectors of n data samples, $\phi_i : \mathbb{R} \to \mathbb{R}$ be a convex loss function associated with the linear prediction $a_i^T x$, for $i = 1, \ldots, n$, and $g : \mathbb{R}^d \to \mathbb{R}$ be a convex regularization function for the predictor $x \in \mathbb{R}^d$. Our goal is to solve the following optimization problem:

$$\underset{x \in \mathbb{R}^d}{\text{minimize}} \quad \Big\{ P(x) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^{n} \phi_i(a_i^T x) + g(x) \Big\}. \tag{1}$$

Examples of this formulation include many well-known classification and regression problems. For binary classification, each feature vector a_i is associated with a label $b_i \in \{\pm 1\}$.

*. A short paper based on a previous version of this manuscript (arXiv:1409.3257) appeared in the Proceedings of The 32nd International Conference on Machine Learning (ICML), Lille, France, July 2015.

We obtain the linear SVM (support vector machine) by setting $\phi_i(z) = \max\{0, 1 - b_i z\}$ (the hinge loss) and $g(x) = (\lambda/2)\|x\|_2^2$, where $\lambda > 0$ is a regularization parameter. Regularized logistic regression is obtained by setting $\phi_i(z) = \log(1 + \exp(-b_i z))$. For linear regression problems, each feature vector a_i is associated with a dependent variable $b_i \in \mathbb{R}$, and $\phi_i(z) = (1/2)(z - b_i)^2$. Then we get ridge regression with $g(x) = (\lambda/2)\|x\|_2^2$, and the Lasso with $g(x) = \lambda\|x\|_1$. Further background on regularized ERM in machine learning and statistics can be found, e.g., in the book by Hastie et al. (2009).

We are especially interested in developing efficient algorithms for solving the problem (1) when the number of samples n is very large. In this case, evaluating the full gradient or subgradient of the function $P(x)$ is very expensive, thus incremental methods that operate on a single component function ϕ_i at each iteration can be very attractive. There has been extensive research on incremental gradient and subgradient methods (e.g., Tseng, 1998; Blatt et al., 2007; Nedić and Bertsekas, 2001; Bertsekas, 2011, 2012) as well as variants of the stochastic gradient method (e.g., Zhang, 2004; Bottou, 2010; Duchi and Singer, 2009; Langford et al., 2009; Xiao, 2010). While the computational cost per iteration of these methods is only a small fraction, say $1/n$, of that of the batch gradient methods, their iteration complexities are much higher (they need a lot more iterations to reach the same precision). In order to better quantify the complexities of various algorithms and position our contributions, we need to make some concrete assumptions and introduce the notion of *condition number* and *batch complexity*.

1.1 Condition Number and Batch Complexity

Let γ and λ be two positive real parameters. We make the following assumption:

Assumption A *Each ϕ_i is convex and differentiable, and its derivative is $(1/\gamma)$-Lipschitz continuous (same as ϕ_i being $(1/\gamma)$-smooth), i.e.,*

$$|\phi_i'(\alpha) - \phi_i'(\beta)| \le (1/\gamma)|\alpha - \beta|, \quad \forall\, \alpha, \beta \in \mathbb{R}, \quad i = 1, \dots, n.$$

In addition, the regularization function g is λ-strongly convex, i.e.,

$$g(x) \ge g(y) + g'(y)^T(x - y) + \frac{\lambda}{2}\|x - y\|_2^2, \quad \forall\, g'(y) \in \partial g(y), \quad x, y \in \mathbb{R}^n.$$

For example, the logistic loss $\phi_i(z) = \log(1 + \exp(-b_i z))$ is $(1/4)$-smooth, the squared error $\phi_i(z) = (1/2)(z - b_i)^2$ is 1-smooth, and the squared ℓ_2-norm $g(x) = (\lambda/2)\|x\|_2^2$ is λ-strongly convex. The hinge loss $\phi_i(z) = \max\{0, 1 - b_i z\}$ and the ℓ_1-regularization $g(x) = \lambda\|x\|_1$ do not satisfy Assumption A. Nevertheless, we can treat them using smoothing and strongly convex perturbations, respectively, so that our algorithm and theoretical framework still apply (see Section 3).

Under Assumption A, the gradient of each component function, $\nabla\phi_i(a_i^T x)$, is also Lipschitz continuous, with Lipschitz constant $L_i = \|a_i\|_2^2/\gamma \le R^2/\gamma$, where $R = \max_i \|a_i\|_2$. In other words, each $\phi_i(a_i^T x)$ is (R^2/γ)-smooth. We define the *condition number*

$$\kappa = R^2/(\lambda\gamma), \tag{2}$$

and focus on ill-conditioned problems where $\kappa \gg 1$. In statistical learning, the regularization parameter λ is usually on the order of $1/\sqrt{n}$ or $1/n$ (e.g., Bousquet and Elisseeff, 2002),

thus the condition number κ is on the order of $\sqrt{n}$ or n. It can be much larger if the strong convexity in g is added purely for numerical regularization purposes (see Section 3). We note that the actual conditioning of problem (1) may be better than κ, if the empirical loss function $(1/n)\sum_{i=1}^n \phi_i(a_i^T x)$ by itself is strongly convex. In those cases, our complexity estimates in terms of κ can be loose (upper bounds), but they are still useful in comparing different algorithms for solving the same problem.

Let $P^\star$ be the optimal value of problem (1), i.e., $P^\star = \min_{x\in\mathbb{R}^d} P(x)$. In order to find an approximate solution $\hat{x}$ satisfying $P(\hat{x}) - P^\star \le \epsilon$, the classical full gradient method and its proximal variants require $\mathcal{O}((1+\kappa)\log(1/\epsilon))$ iterations (e.g., Nesterov, 2004, 2013). Accelerated full gradient methods enjoy the improved iteration complexity $\mathcal{O}((1+\sqrt{\kappa})\log(1/\epsilon))$ (Nesterov, 2004; Tseng, 2008; Beck and Teboulle, 2009; Nesterov, 2013)[1]. However, each iteration of these batch methods requires a full pass over the dataset, computing the gradient of each component function and forming their average, which cost $\mathcal{O}(nd)$ operations (assuming the features vectors $a_i \in \mathbb{R}^d$ are dense). In contrast, the stochastic gradient method and its proximal variants operate on one single component $\phi_i(a_i^T x)$ (chosen randomly) at each iteration, which only costs $\mathcal{O}(d)$. But their iteration complexities are far worse. Under Assumption A, it takes them $\mathcal{O}(\kappa/\epsilon)$ iterations to find an $\hat{x}$ such that $\mathbb{E}[P(\hat{x}) - P^\star] \le \epsilon$, where the expectation is with respect to the random choices made at all the iterations (e.g., Polyak and Juditsky, 1992; Nemirovski et al., 2009; Duchi and Singer, 2009; Langford et al., 2009; Xiao, 2010).

To make fair comparisons with batch methods, we measure the complexity of stochastic or incremental gradient methods in terms of the number of equivalent passes over the dataset required to reach an expected precision ϵ. We call this measure the *batch complexity*, which is usually obtained by dividing their iteration complexities by n. For example, the batch complexity of the stochastic gradient method is $\mathcal{O}(\kappa/(n\epsilon))$. The batch complexities of full gradient methods are the same as their iteration complexities.

By exploiting the finite average structure in (1), several recent work (e.g., Le Roux et al., 2012; Shalev-Shwartz and Zhang, 2013a; Johnson and Zhang, 2013; Xiao and Zhang, 2014; Defazio et al., 2014) proposed new variants of the stochastic gradient and dual coordinate ascent methods which achieve the iteration complexity $\mathcal{O}((n+\kappa)\log(1/\epsilon))$. Since their computational cost per iteration is $\mathcal{O}(d)$, the equivalent batch complexity is $1/n$ of their iteration complexity, i.e., $\mathcal{O}((1+\kappa/n)\log(1/\epsilon))$. This complexity has much weaker dependence on n than the full gradient methods, and also much weaker dependence on ϵ than the stochastic gradient methods.

In this paper, we propose a stochastic primal-dual coordinate (SPDC) method, which has the iteration complexity

$$\mathcal{O}\big((n+\sqrt{\kappa n})\log(1/\epsilon)\big),$$

or equivalently, the batch complexity

$$\mathcal{O}\big((1+\sqrt{\kappa/n})\log(1/\epsilon)\big). \tag{3}$$

When $\kappa > n$, this is lower than the $\mathcal{O}((1+\kappa/n)\log(1/\epsilon))$ batch complexity mentioned above. Indeed, it reaches a lower bound for minimizing finite sums established in Lan and Zhou

1. For the analysis of full gradient methods, we should use $(R^2/\gamma+\lambda)/\lambda = 1+\kappa$ as the condition number of problem (1); see Nesterov (2013, Section 5.1). Here we used the upper bound $\sqrt{1+\kappa} < 1+\sqrt{\kappa}$ for easy comparison. When $\kappa \gg 1$, the additive constant 1 can be dropped.

(2015); see also Agarwal and Bottou (2015) and Woodworth and Srebro (2016). Several other recent work also achieved the same complexity bound, either with a dual or a primal accelerated randomized algorithm (Lin et al., 2015b; Lan and Zhou, 2015; Allen-Zhu, 2017) or through the proximal-point algorithm (Shalev-Shwartz and Zhang, 2015; Frostig et al., 2015; Lin et al., 2015a). We will discuss these related work in Section 5.

1.2 Outline of the Paper

Our approach is based on reformulating problem (1) as a convex-concave saddle point problem, and then devising a primal-dual algorithm to approximate the saddle point. More specifically, we replace each component function $\phi_i(a_i^T x)$ through convex conjugation, i.e.,

$$\phi_i(a_i^T x) = \sup_{y_i \in \mathbb{R}} \left\{ y_i \langle a_i, x \rangle - \phi_i^*(y_i) \right\},$$

where $\phi_i^*(y_i) = \sup_{\alpha \in \mathbb{R}} \{\alpha y_i - \phi_i(\alpha)\}$, and $\langle a_i, x \rangle$ denotes the inner product of a_i and x (which is the same as $a_i^T x$, but is more convenient for later presentation). This leads to a convex-concave saddle point problem

$$\min_{x \in \mathbb{R}^d} \max_{y \in \mathbb{R}^n} \left\{ f(x,y) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n \big(y_i \langle a_i, x \rangle - \phi_i^*(y_i) \big) + g(x) \right\}. \tag{4}$$

Under Assumption A, each ϕ_i is $(1/\gamma)$-smooth, which implies that ϕ_i^* is γ-strongly convex (see, e.g., Hiriart-Urruty and Lemaréchal, 2001, Theorem 4.2.2). In addition, the regularization g is λ-strongly convex. As a consequence, the saddle point problem (4) has a unique solution, which we denote by $(x^\star, y^\star)$.

The above saddle-point formulation allows the regularized ERM problem to be solved by the primal-dual first-order algorithms developed in Chambolle and Pock (2011). Under Assumption A, Algorithm 3 in Chambolle and Pock (2011) have the complexity $\mathcal{O}(1 + \sqrt{\kappa}) \log(1/\epsilon)$, which is that same as that of the accelerated full gradient methods. Our SPDC method can be viewed as a randomized coordinate variant of the primal-dual algorithm in Chambolle and Pock (2011), which achieves a much better complexity by exploiting the finite-sum structure of the regularized ERM problem.

In Section 2, we present the SPDC method as well as its convergence analysis. It alternates between maximizing f over a randomly chosen dual coordinate y_i and minimizing f over the primal variable x. In order to accelerate the convergence, an extrapolation step is applied in updating the primal variable x. We also give a mini-batch SPDC algorithm which is well suited for parallel computing.

In Section 3 and Section 4, we present two extensions of the SPDC method. We first explain how to solve problem (1) when Assumption A does not hold. The idea is to apply small regularizations to the saddle point function so that SPDC can still be applied, which results in accelerated sublinear rates for solving the original problem. The second extension is a SPDC method with non-uniform sampling. The batch complexity of this algorithm has the same form as (3), but with $\kappa = \bar{R}/(\lambda\gamma)$, where $\bar{R} = \frac{1}{n}\sum_{i=1}^n \|a_i\|$, which can be much smaller than $R = \max_i \|a_i\|$ if there is considerable variation in the norms $\|a_i\|$.

In Section 5, we discuss related work. In particular, we explain the connection of SPDC with the primal-dual batch methods in Chambolle and Pock (2011), and several recent work that achieves the same complexity as SPDC.

In Section 6, we discuss efficient implementation of the SPDC method when the feature vectors a_i are sparse. We focus on two popular cases: when g is a squared ℓ_2-norm penalty and when g is an $\ell_1 + \ell_2$ penalty. We show that the computational cost per iteration of SPDC only depends on the number of non-zero elements in the feature vectors.

In Section 7, we present experiment results comparing SPDC with several state-of-the-art optimization methods, including both batch algorithms and randomized incremental and coordinate gradient methods. On all scenarios we tested, SPDC has comparable or better performance.

The SPDC method was first proposed and analyzed in Zhang and Xiao (2015). This manuscript is a significant expansion, which includes several major technical changes, as well as new experiment results. More specifically, the main theoretical result (Theorem 1) has been strengthened to include both iterate convergence and a new result on saddle-point function convergence. This required major changes in the technical proof in Appendix A. Moreover, the convergence rate analysis of the primal-dual gap in Zhang and Xiao (2015) required additional assumptions than those in Theorem 1. In this paper (Section 2.2), we present a new proof that do not require additional assumptions. The SPDC method with non-uniform sampling (Section 4) has been extended with a more general parametrized sampling probability, and we give the optimal parameterization based on the condition number. Additional numerical experiments are presented for both synthetic and real datasets, including new comparisons with the accelerated SDCA algorithm by Shalev-Shwartz and Zhang (2015), and also comparisons between uniform and weighted sampling.

2. The SPDC Method

In this section, we describe and analyze the Stochastic Primal-Dual Coordinate (SPDC) method. The basic idea of SPDC is quite simple: to approach the saddle point of $f(x, y)$ defined in (4), we alternatively maximize f with respect to y, and minimize f with respect to x. Since the dual vector y has n coordinates and each coordinate is associated with a feature vector $a_i \in \mathbb{R}^d$, maximizing f with respect to y takes $\mathcal{O}(nd)$ computation, which can be very expensive if n is large. We reduce the computational cost by randomly picking a single coordinate of y at a time, and maximizing f only with respect to this coordinate. Consequently, the computational cost of each iteration is $\mathcal{O}(d)$.

We give the details of the SPDC method in Algorithm 1. The dual coordinate update and primal vector update are given in equations (5) and (6) respectively. Instead of maximizing f over y_k and minimizing f over x directly, we add two quadratic regularization terms to penalize $y_k^{(t+1)}$ and $x^{(t+1)}$ from deviating from $y_k^{(t)}$ and $x^{(t)}$. The parameters σ and τ control their regularization strength, which we will specify in the convergence analysis (Theorem 1). Moreover, we introduce two auxiliary variables $u^{(t)}$ and $\overline{x}^{(t)}$. Combining the initialization $u^{(0)} = (1/n)\sum_{i=1}^n y_i^{(0)} a_i$ and the update rules (5) and (7), we have

$$u^{(t)} = \frac{1}{n}\sum_{i=1}^n y_i^{(t)} a_i, \qquad t = 0, \ldots, T.$$

Equation (8) obtains $\overline{x}^{(t+1)}$ based on an extrapolation from $x^{(t)}$ and $x^{(t+1)}$. This step is similar to Nesterov's acceleration technique (Nesterov, 2004, Section 2.2), and yields faster convergence rate.

Algorithm 1: The SPDC method

Input: parameters $\tau, \sigma, \theta \in \mathbb{R}_+$, number of iterations T, initial points $x^{(0)}$ and $y^{(0)}$.

Initialize: $\overline{x}^{(0)} = x^{(0)}$, $u^{(0)} = (1/n)\sum_{i=1}^n y_i^{(0)} a_i$.

for $t = 0, 1, 2, \ldots, T-1$ **do**

Pick $k \in \{1, 2, \ldots, n\}$ uniformly at random, and execute the following updates:

$$y_i^{(t+1)} = \begin{cases} \arg\max_{\beta \in \mathbb{R}} \left\{ \beta \langle a_i, \overline{x}^{(t)} \rangle - \phi_i^*(\beta) - \frac{1}{2\sigma}(\beta - y_i^{(t)})^2 \right\} & \text{if } i = k, \\ y_i^{(t)} & \text{if } i \neq k, \end{cases} \tag{5}$$

$$x^{(t+1)} = \arg\min_{x \in \mathbb{R}^d} \left\{ g(x) + \left\langle u^{(t)} + (y_k^{(t+1)} - y_k^{(t)}) a_k,\ x \right\rangle + \frac{\|x - x^{(t)}\|_2^2}{2\tau} \right\}, \tag{6}$$

$$u^{(t+1)} = u^{(t)} + \frac{1}{n}(y_k^{(t+1)} - y_k^{(t)}) a_k, \tag{7}$$

$$\overline{x}^{(t+1)} = x^{(t+1)} + \theta(x^{(t+1)} - x^{(t)}). \tag{8}$$

end

Output: $x^{(T)}$ and $y^{(T)}$

The mini-batch SPDC method in Algorithm 2 is a natural extension of Algorithm 1. The difference between these two algorithms is that, the mini-batch SPDC method may simultaneously select more than one dual coordinates to update. Let m be the mini-batch size. During each iteration, the mini-batch SPDC method randomly picks a subset of indices $K \subset \{1, \ldots, n\}$ of size m, such that the probability of each index being picked is equal to m/n. The following is a simple procedure to achieve this. First, partition the set of indices into m disjoint subsets, so that the cardinality of each subset is equal to n/m (assuming m divides n). Then, during each iteration, randomly select a single index from each subset and add it to K. Other approaches for mini-batch selection are also possible; see the discussions in Richtárik and Takáč (2016).

In Algorithm 2, we also switched the order of updating $x^{(t+1)}$ and $u^{(t+1)}$ (comparing with Algorithm 1), to better illustrate that $x^{(t+1)}$ is obtained based on an extrapolation from $u^{(t)}$ to $u^{(t+1)}$. However, this form is not recommended in implementation, because $u^{(t)}$ is usually a dense vector even if the feature vectors a_k are sparse. Details on efficient implementation of SPDC are given in Section 6. In the following discussion, we do not make sparseness assumptions.

With a single processor, each iteration of Algorithm 2 takes $\mathcal{O}(md)$ time to accomplish. Since the updates of each coordinate y_k are independent of each other, we can use parallel computing to accelerate the mini-batch SPDC method. Concretely, we can use m processors to update the m coordinates in the subset K in parallel, then aggregate them to update $x^{(t+1)}$. In terms of wall-clock time, each iteration takes $\mathcal{O}(d)$ time, which is the same as running one iteration of the basic SPDC algorithm. Not surprisingly, we will show that the mini-batch SPDC algorithm converges faster than SPDC in terms of the iteration complexity, because it processes multiple dual coordinates in a single iteration.

Algorithm 2: The mini-batch SPDC method

Input: mini-batch size m, parameters $\tau, \sigma, \theta \in \mathbb{R}_+$, number of iterations T, and the initial points $x^{(0)}$ and $y^{(0)}$.

Initialize: $\overline{x}^{(0)} = x^{(0)}$, $u^{(0)} = (1/n)\sum_{i=1}^n y_i^{(0)} a_i$.

for $t = 0, 1, 2, \ldots, T-1$ **do**

Randomly pick a subset $K \subset \{1, 2, \ldots, n\}$ of size m, such that the probability of each index being picked is equal to m/n. Execute the following updates:

$$y_i^{(t+1)} = \begin{cases} \arg\max_{\beta \in \mathbb{R}} \left\{ \beta \langle a_i, \overline{x}^{(t)} \rangle - \phi_i^*(\beta) - \frac{1}{2\sigma}(\beta - y_i^{(t)})^2 \right\} & \text{if } i \in K, \\ y_i^{(t)} & \text{if } i \notin K, \end{cases} \tag{9}$$

$$u^{(t+1)} = u^{(t)} + \frac{1}{n} \sum_{k \in K} (y_k^{(t+1)} - y_k^{(t)}) a_k,$$

$$x^{(t+1)} = \arg\min_{x \in \mathbb{R}^d} \left\{ g(x) + \left\langle u^{(t)} + \frac{n}{m}(u^{(t+1)} - u^{(t)}),\ x \right\rangle + \frac{\|x - x^{(t)}\|_2^2}{2\tau} \right\}, \tag{10}$$

$$\overline{x}^{(t+1)} = x^{(t+1)} + \theta(x^{(t+1)} - x^{(t)}).$$

end

Output: $x^{(T)}$ and $y^{(T)}$

2.1 Convergence Analysis

Since the basic SPDC algorithm is a special case of mini-batch SPDC with $m = 1$, we only present a convergence theorem for the mini-batch version. The expectations in the following results are taken with respect to the random variables $\{K^{(0)}, \ldots, K^{(T-1)}\}$, where $K^{(t)}$ denotes the random subset $K \subset \{1, \ldots, n\}$ picked at the t-th iteration of the mini-batch SPDC method.

Theorem 1 *Suppose Assumption A holds. Let $(x^\star, y^\star)$ be the unique saddle point of f defined in (4), $R = \max\{\|a_1\|_2, \ldots, \|a_n\|_2\}$, and define*

$$\Delta^{(t)} = \left(\frac{1}{2\tau} + \frac{\lambda}{2}\right) \|x^{(t)} - x^\star\|_2^2 + \left(\frac{1}{4\sigma} + \frac{\gamma}{2}\right) \frac{\|y^{(t)} - y^\star\|_2^2}{m} + f(x^{(t)}, y^\star) - f(x^\star, y^\star) + \frac{n}{m}\left(f(x^\star, y^\star) - f(x^\star, y^{(t)})\right). \tag{11}$$

If the parameters τ, σ and θ in Algorithm 2 are chosen such that

$$\tau = \frac{1}{2R}\sqrt{\frac{m\gamma}{n\lambda}}, \qquad \sigma = \frac{1}{2R}\sqrt{\frac{n\lambda}{m\gamma}}, \qquad \theta = 1 - \left(\frac{n}{m} + 2R\sqrt{\frac{n}{m\lambda\gamma}}\right)^{-1}, \tag{12}$$

then for each $t \geq 1$, the mini-batch SPDC algorithm achieves

$$\mathbb{E}[\Delta^{(t)}] \leq \theta^t \left(\Delta^{(0)} + \frac{\|y^{(0)} - y^\star\|_2^2}{4m\sigma}\right).$$

Comparing with Theorem 1 in Zhang and Xiao (2015), our definition of $\Delta^{(t)}$ in (11) includes the additional terms $f(x^{(t)}, y^\star) - f(x^\star, y^\star) + \frac{n}{m}\big(f(x^\star, y^\star) - f(x^\star, y^{(t)})\big)$. This is a weighted sum of the primal and dual gaps for the saddle-point problem. It will help us establish the convergence rate of the objective value for the ERM problem in Section 2.2, which is missing in Zhang and Xiao (2015).

The proof of Theorem 1 is given in Appendix A. The following corollary establishes the expected iteration complexity of mini-batch SPDC for obtaining an ϵ-accurate solution.

Corollary 2 *Suppose Assumption A holds and the parameters τ, σ and θ are set as in* (12). *In order for Algorithm 2 to obtain*

$$\mathbb{E}[\|x^{(T)} - x^\star\|_2^2] \le \epsilon, \qquad \mathbb{E}[\|y^{(T)} - y^\star\|_2^2] \le \epsilon, \tag{13}$$

it suffices to have the number of iterations T satisfy

$$T \ge \left(\frac{n}{m} + 2R\sqrt{\frac{n}{m\lambda\gamma}}\right)\log\left(\frac{C}{\epsilon}\right),$$

where

$$C = \frac{\Delta^{(0)} + \big\|y^{(t)} - y^\star\big\|_2^2/(4m\sigma)}{\min\big\{1/(2\tau) + \lambda/2,\ (1/(4\sigma) + \gamma/2)/m\big\}}.$$

Proof By Theorem 1, we have $\mathbb{E}[\|x^{(t)} - x^\star\|_2^2] \le \theta^t C$ and $\mathbb{E}[\|y^{(t)} - y^\star\|_2^2] \le \theta^t C$ for each $t > 0$. To obtain (13), it suffices to ensure that $\theta^T C \le \epsilon$, which is equivalent to

$$T \ \ge\ \frac{\log(C/\epsilon)}{-\log(\theta)} \ =\ \frac{\log(C/\epsilon)}{-\log\Big(1 - \Big((n/m) + 2R\sqrt{(n/m)/(\lambda\gamma)}\Big)^{-1}\Big)}.$$

Applying the inequality $-\log(1-x) \ge x$ to the denominator above completes the proof. ■

Recall the definition of the condition number $\kappa = R^2/(\lambda\gamma)$ in (2). Corollary 2 establishes that the iteration complexity of the mini-batch SPDC method for achieving (13) is

$$\mathcal{O}\left(\big((n/m) + \sqrt{\kappa(n/m)}\big)\log(1/\epsilon)\right).$$

So a larger batch size m leads to less number of iterations. In the extreme case of $n = m$, we obtain a full batch algorithm, which has iteration or batch complexity $\mathcal{O}((1+\sqrt{\kappa})\log(1/\epsilon))$. This complexity is also shared by the accelerated gradient methods (Nesterov, 2004, 2013), as well as the batch primal-dual algorithm of Chambolle and Pock (2011); see discussions in Section 1.1 and related work in Section 5.

Since an equivalent pass over the dataset corresponds to n/m iterations, the batch complexity (the number of equivalent passes over the data) of mini-batch SPDC is

$$\mathcal{O}\left(\big(1 + \sqrt{\kappa(m/n)}\big)\log(1/\epsilon)\right).$$

The above expression implies that a smaller batch size m leads to less number of passes through the data. In this sense, the basic SPDC method with $m = 1$ is the most efficient one. However, if we prefer the least amount of wall-clock time, then the best choice is to choose a mini-batch size m that matches the number of parallel processors available.

2.2 Convergence Rate of Primal-Dual Gap

In the previous subsection, we established iteration complexity of the mini-batch SPDC method in terms of approximating the saddle point of the minimax problem (4), more specifically, to meet the requirement in (13). Next we show that it has the same order of complexity in reducing the primal-dual objective gap $P(x^{(t)}) - D(y^{(t)})$, where $P(x)$ is defined in (1) and

$$D(y) \overset{\text{def}}{=} \min_{x\in\mathbb{R}^d} f(x,y) = \frac{1}{n}\sum_{i=1}^n -\phi_i^*(y_i) - g^*\bigg(-\frac{1}{n}\sum_{i=1}^n y_i a_i\bigg). \tag{14}$$

where $g^*(u) = \sup_{x\in\mathbb{R}^d}\{x^T u - g(x)\}$ is the conjugate function of g.

Under Assumption A, the function $f(x,y)$ defined in (4) has a unique saddle point $(x^\star, y^\star)$, and

$$P(x^\star) = f(x^\star, y^\star) = D(y^\star).$$

However, in general, for any point $(x,y) \in \mathrm{dom}(g)\times\mathrm{dom}(\phi^*)$, we have

$$P(x) = \max_y f(x,y) \geq f(x,y^\star), \qquad D(y) = \min_x f(x,y) \leq f(x^\star, y).$$

Thus the result in Theorem 1 does not translate directly into a convergence bound on the primal-dual gap. We need to bound $P(x)$ and $D(y)$ by $f(x,y^\star)$ and $f(x^\star,y)$, respectively, in the opposite directions. For this purpose, we need the following result extracted from Yu et al. (2015). We provide the proof in Appendix B for completeness.

Lemma 3 *Suppose Assumption A holds. Let $(x^\star, y^\star)$ be the unique saddle-point of $f(x,y)$, and $R = \max_{1\leq i\leq n}\|a_i\|_2$. Then for any point $(x,y)\in\mathrm{dom}(g)\times\mathrm{dom}(\phi^*)$, we have*

$$P(x) \leq f(x,y^\star) + \frac{R^2}{2\gamma}\|x-x^\star\|_2^2, \qquad D(y) \geq f(x^\star,y) - \frac{R^2}{2\lambda n}\|y-y^\star\|_2^2.$$

Corollary 4 *Suppose Assumption A holds and the parameters τ, σ and θ are set as in (12). Let $\widetilde{\Delta}^{(0)} := \Delta^{(0)} + \frac{\|y^{(0)}-y^\star\|_2^2}{4m\sigma}$. Then for any $\epsilon\geq 0$, the iterates of Algorithm 2 satisfy*

$$\mathbb{E}[P(x^{(T)}) - D(y^{(T)})] \leq \epsilon$$

whenever

$$T \geq \left(\frac{n}{m} + 2R\sqrt{\frac{n}{m\lambda\gamma}}\right)\log\left(\left(1+\frac{R^2}{\lambda\gamma}\right)\frac{\widetilde{\Delta}^{(0)}}{\epsilon}\right).$$

Proof The function $f(x,y^\star)$ is strongly convex in x with parameter λ, and $x^\star$ is the minimizer. Similarly, $-f(x^\star,y)$ is strongly convex in y with parameter γ/n, and is minimized by $y^\star$. Therefore,

$$\frac{\lambda}{2}\|x^{(t)}-x^\star\|_2^2 \leq f(x^{(t)},y^\star) - f(x^\star,y^\star), \quad \frac{\gamma}{2n}\|y^{(t)}-y^\star\|_2^2 \leq f(x^\star,y^\star) - f(x^\star,y^{(t)}). \tag{15}$$

We bound the following weighted primal-dual gap

$$\begin{aligned}
& P(x^{(t)}) - P(x^\star) + \frac{n}{m}\Big(D(y^\star) - D(y^{(t)})\Big) \\
\leq\ & f(x^{(t)}, y^\star) - f(x^\star, y^\star) + \frac{n}{m}\Big(f(x^\star, y^\star) - f(x^\star, y^{(t)})\Big) + \frac{R^2}{2\gamma}\|x^{(t)} - x^\star\|_2^2 + \frac{n}{m}\frac{R^2}{2n\lambda}\|y^{(t)} - y^\star\|_2^2 \\
\leq\ & \Delta^{(t)} + \frac{R^2}{\lambda\gamma}\left(\frac{\lambda}{2}\|x^{(t)} - x^\star\|_2^2 + \frac{n}{m}\frac{\gamma}{2n}\|y^{(t)} - y^\star\|_2^2\right) \\
\leq\ & \Delta^{(t)} + \frac{R^2}{\lambda\gamma}\left(f(x^{(t)}, y^\star) - f(x^\star, y^\star) + \frac{n}{m}\Big(f(x^\star, y^\star) - f(x^\star, y^{(t)})\Big)\right) \\
\leq\ & \left(1 + \frac{R^2}{\lambda\gamma}\right)\Delta^{(t)}.
\end{aligned}$$

The first inequality above is due to Lemma 3, the second and fourth inequalities are due to the definition of $\Delta^{(t)}$, and the third inequality is due to (15). Taking expectations on both sides of the above inequality, then applying Theorem 1, we obtain

$$\mathbb{E}\left[P(x^{(t)}) - P(x^\star) + \frac{n}{m}\Big(D(y^\star) - D(y^{(t)})\Big)\right] \ \leq\ \theta^t\left(1 + \frac{R^2}{\lambda\gamma}\right)\widetilde{\Delta}^{(0)} = (1+\kappa)\widetilde{\Delta}^{(t)}.$$

Since $n \geq m$ and $D(y^\star) - D(y^{(t)})) \geq 0$, this implies the desired result. ■

3. Extensions to Non-Smooth or Non-Strongly Convex Functions

The complexity bounds established in Section 2 require each ϕ_i be $(1/\gamma)$-smooth, and the function g be λ-strongly convex. For general loss functions where either or both of these conditions fail (e.g., the hinge loss and ℓ_1-regularization), we can slightly perturb the saddle-point function $f(x, y)$ so that the SPDC method can still be applied.

To be concise, we only consider the case where neither ϕ_i is smooth nor g is strongly convex. Instead, we only assume that each ϕ_i and g are convex and Lipschitz continuous, and $f(x, y)$ has a saddle point $(x^\star, y^\star)$. We choose a scalar $\delta > 0$ and consider the modified saddle-point function:

$$f_\delta(x, y) \ \stackrel{\text{def}}{=}\ \frac{1}{n}\sum_{i=1}^n\left(y_i\langle a_i, x\rangle - \left(\phi_i^*(y_i) + \frac{\delta y_i^2}{2}\right)\right) + g(x) + \frac{\delta}{2}\|x\|_2^2. \tag{16}$$

Denote the saddle-point of f_δ by $(x_\delta^\star, y_\delta^\star)$. We employ the mini-batch SPDC method in Algorithm 2 to approximate $(x_\delta^\star, y_\delta^\star)$, treating $\phi_i^* + \frac{\delta}{2}(\cdot)^2$ as ϕ_i^* and $g + \frac{\delta}{2}\|\cdot\|_2^2$ as g, which are all δ-strongly convex. We note that adding strongly convex perturbation on ϕ_i^* is equivalent to smoothing ϕ_i, which becomes $(1/\delta)$-smooth (see, e.g., Nesterov, 2005). Letting $\gamma = \lambda = \delta$, the parameters τ, σ and θ in (12) become

$$\tau = \frac{1}{2R}\sqrt{\frac{m}{n}}, \qquad \sigma = \frac{1}{2R}\sqrt{\frac{n}{m}}, \qquad \text{and} \qquad \theta = 1 - \left(\frac{n}{m} + \frac{2R}{\delta}\sqrt{\frac{n}{m}}\right)^{-1}.$$

Although $(x_\delta^\star, y_\delta^\star)$ is not exactly the saddle point of f, the following corollary shows that applying SPDC to the perturbed function f_δ effectively minimizes the original loss function P. Similar results for the convergence of the primal-dual gap can also be established.

Corollary 5 *Assume that each ϕ_i is convex and G_ϕ-Lipschitz continuous, and g is convex and G_g-Lipschitz continuous. In addition, assume that f has a saddle point $(x^\star, y^\star)$ and let the unique saddle point of f_δ be $(x_\delta^\star, y_\delta^\star)$. Define two constants:*

$$C_1 = (\|x^\star\|_2^2 + G_\phi^2), \qquad C_2 = (G_\phi R + G_g)^2 \left(\frac{\Delta_\delta^{(0)} + \left\|y^{(0)} - y_\delta^\star\right\|_2^2 R/(4\sqrt{mn})}{1/(2\tau) + \lambda/2} \right),$$

where $\Delta_\delta^{(0)}$ is evaluated as in (11) but in terms of the perturbed function f_δ. If we choose $\delta \le \epsilon/C_1$, then we have $\mathbb{E}[P(x^{(T)}) - P(x^\star)] \le \epsilon$ whenever

$$T \ge \left(\frac{n}{m} + \frac{2R}{\delta}\sqrt{\frac{n}{m}} \right) \log\left(\frac{4C_2}{\epsilon^2} \right).$$

Proof Let $\widetilde{y} = \arg\max_y f(x_\delta^\star, y)$ be a shorthand notation. We have

$$\begin{aligned} P(x_\delta^\star) &\overset{(i)}{=} f(x_\delta^\star, \widetilde{y}) \overset{(ii)}{\le} f_\delta(x_\delta^\star, \widetilde{y}) + \frac{\delta\|\widetilde{y}\|_2^2}{2n} \overset{(iii)}{\le} f_\delta(x_\delta^\star, y_\delta^\star) + \frac{\delta\|\widetilde{y}\|_2^2}{2n} \overset{(iv)}{\le} f_\delta(x^\star, y_\delta^\star) + \frac{\delta\|\widetilde{y}\|_2^2}{2n} \\ &\overset{(v)}{\le} f(x^\star, y_\delta^\star) + \frac{\delta\|x^\star\|_2^2}{2} + \frac{\delta\|\widetilde{y}\|_2^2}{2n} \overset{(vi)}{\le} f(x^\star, y^\star) + \frac{\delta\|x^\star\|_2^2}{2} + \frac{\delta\|\widetilde{y}\|_2^2}{2n} \\ &\overset{(vii)}{=} P(x^\star) + \frac{\delta\|x^\star\|_2^2}{2} + \frac{\delta\|\widetilde{y}\|_2^2}{2n}. \end{aligned}$$

Here, equations (i) and (vii) use the definition of the function f, inequalities (ii) and (v) use the definition of the function f_δ, inequalities (iii) and (iv) use the fact that $(x_\delta^\star, y_\delta^\star)$ is the saddle point of f_δ, and inequality (vi) is due to the fact that $(x^\star, y^\star)$ is the saddle point of f.

Since ϕ_i is G_ϕ-Lipschitz continuous, the domain of ϕ_i^* is in the interval $[-G_\phi, G_\phi]$, which implies $\|\widetilde{y}\|_2^2 \le nG_\phi^2$ (see, e.g., (Shalev-Shwartz and Zhang, 2015, Lemma 1)). Thus, we have

$$P(x_\delta^\star) - P(x^\star) \le \frac{\delta}{2}(\|x^\star\|_2^2 + G_\phi^2) = \frac{\delta}{2}C_1. \tag{17}$$

On the other hand, since P is $(G_\phi R + G_g)$-Lipschitz continuous, Theorem 1 implies

$$\begin{aligned} \mathbb{E}[P(x^{(T)}) - P(x_\delta^\star)] &\le (G_\phi R + G_g)\mathbb{E}[\|x^{(T)} - x_\delta^\star\|_2] \\ &\le \sqrt{C_2}\left(1 - \left(\frac{n}{m} + \frac{2R}{\delta}\sqrt{\frac{n}{m}}\right)^{-1}\right)^{T/2}. \end{aligned} \tag{18}$$

Combining (17) and (18), in order to obtain $\mathbb{E}[P(x^{(T)}) - P(x^\star)] \le \epsilon$, it suffices to have $C_1\delta \le \epsilon$ and

$$\sqrt{C_2}\left(1 - \left(\frac{n}{m} + \frac{2R}{\delta}\sqrt{\frac{n}{m}}\right)^{-1}\right)^{T/2} \le \frac{\epsilon}{2}. \tag{19}$$

ϕ_i	g	iteration complexity $\widetilde{\mathcal{O}}(\cdot)$
$(1/\gamma)$-smooth	λ-strongly convex	$n/m + \sqrt{(n/m)/(\lambda\gamma)}$
$(1/\gamma)$-smooth	non-strongly convex	$n/m + \sqrt{(n/m)/(\epsilon\gamma)}$
non-smooth	λ-strongly convex	$n/m + \sqrt{(n/m)/(\epsilon\lambda)}$
non-smooth	non-strongly convex	$n/m + \sqrt{n/m}/\epsilon$

Table 1: Iteration complexities of the SPDC method under different assumptions on the functions ϕ_i and g. For the last three cases, we solve the perturbed saddle-point problem with $\delta = \epsilon/C_1$.

The corollary is established by finding the smallest T that satisfies inequality (19). ■

There are two other cases that can be considered: when ϕ_i is not smooth but g is strongly convex, and when ϕ_i is smooth but g is not strongly convex. They can be handled with the same technique described above, and we omit the details here. Alternatively, it is possible to use the techniques described in Chambolle and Pock (2011, Section 5.1) to obtain accelerated sublinear convergence rates without using strongly convex perturbations. In Table 1, we list the complexities of the mini-batch SPDC method for finding an ϵ-optimal solution of problem (1) under various assumptions. Similar results are also obtained in Shalev-Shwartz and Zhang (2015).

4. SPDC with Non-Uniform Sampling

One potential drawback of the SPDC algorithm is that, its convergence rate depends on a problem-specific constant R, which is the largest ℓ_2-norm of the feature vectors a_i. As a consequence, the algorithm may perform badly on unnormalized data, especially if the ℓ_2-norms of some feature vectors are substantially larger than others. In this section, we propose an extension of the SPDC method to mitigate this problem, which is given in Algorithm 3.

The basic idea is to use non-uniform sampling in picking the dual coordinate to update at each iteration. In Algorithm 3, we pick coordinate k with the probability

$$p_k = (1-\alpha)\frac{1}{n} + \alpha\,\frac{\|a_k\|_2}{\sum_{i=1}^n \|a_i\|_2}, \qquad k = 1, \ldots, n, \tag{20}$$

where $\alpha \in (0,1)$ is a parameter. In other words, this distribution is a (strict) convex combination of the uniform distribution and the distribution that is proportional to the feature norms. Therefore, instances with large feature norms are sampled more frequently, controlled by α. Simultaneously, we adopt an adaptive regularization in step (21), imposing stronger regularization on such instances. In addition, we adjust the weight of a_k in (23) for updating the primal variable. As a consequence, the convergence rate of Algorithm 3 depends on the average norm of feature vectors, as well as the parameter α. This is summarized in the following theorem, whose proof is given in Appendix C.

Algorithm 3: SPDC method with weighted sampling

Input: parameters $\tau, \sigma, \theta \in \mathbb{R}_+$, number of iterations T, initial points $x^{(0)}$ and $y^{(0)}$.

Initialize: $\overline{x}^{(0)} = x^{(0)}$, $u^{(0)} = (1/n)\sum_{i=1}^n y_i^{(0)} a_i$.

for $t = 0, 1, 2, \ldots, T-1$ **do**

Randomly pick $k \in \{1, 2, \ldots, n\}$, with probability p_k given in (20).

Execute the following updates:

$$y_i^{(t+1)} = \begin{cases} \arg\max_{\beta \in \mathbb{R}} \left\{ \beta \langle a_i, \overline{x}^{(t)} \rangle - \phi_i^*(\beta) - \frac{p_i n}{2\sigma}(\beta - y_i^{(t)})^2 \right\} & i = k, \\ y_i^{(t)} & i \neq k, \end{cases} \tag{21}$$

$$u^{(t+1)} = u^{(t)} + \frac{1}{n}(y_k^{(t+1)} - y_k^{(t)}) a_k, \tag{22}$$

$$x^{(t+1)} = \arg\min_{x \in \mathbb{R}^d} \left\{ g(x) + \left\langle u^{(t)} + \frac{1}{p_k}(u^{(t+1)} - u^{(t)}),\ x \right\rangle + \frac{\|x - x^{(t)}\|_2^2}{2\tau} \right\}, \tag{23}$$

$$\overline{x}^{(t+1)} = x^{(t+1)} + \theta(x^{(t+1)} - x^{(t)}).$$

end

Output: $x^{(T)}$ and $y^{(T)}$.

Theorem 6 *Suppose Assumption A holds. Let* $R := \max_i \|a_i\|_2$, $\overline{R} := \frac{1}{n}\sum_{i=1}^n \|a_i\|_2$ *and* $R_\alpha := \big((1-\alpha)/R + \alpha/\overline{R}\big)^{-1}$. *If the parameters* τ, σ, θ *in Algorithm 3 are chosen such that*

$$\tau = \frac{1}{2R_\alpha}\sqrt{\frac{\gamma}{n\lambda}}, \qquad \sigma = \frac{1}{2R_\alpha}\sqrt{\frac{n\lambda}{\gamma}}, \qquad \theta = 1 - \left(\frac{n}{1-\alpha} + R_\alpha\sqrt{\frac{n}{\lambda\gamma}}\right)^{-1}, \tag{24}$$

then for each $t \geq 1$, *we have*

$$\begin{aligned} &\Big(\frac{1}{2\tau} + \lambda\Big)\mathbb{E}\big[\|x^{(t)} - x^\star\|_2^2\big] + \Big(\frac{1}{4\sigma} + \frac{\gamma}{n}\Big)\mathbb{E}\big[\|y^{(t)} - y^\star\|_2^2\big] \\ \leq\ & \theta^t \left(\Big(\frac{1}{2\tau} + \lambda\Big)\|x^{(0)} - x^\star\|_2^2 + \Big(\frac{1}{2\sigma} + \frac{\gamma}{1-\alpha}\Big)\|y^{(0)} - y^\star\|_2^2 \right). \end{aligned}$$

Note that $R_\alpha \leq \overline{R}/\alpha$ always holds. If we choose $\alpha = 1/2$, then the contraction ratio θ is bounded by $1 - \left(\frac{n}{1-\alpha} + 2\overline{R}\sqrt{\frac{n}{\lambda\gamma}}\right)^{-1}$. Comparing this bound with Theorem 1 with $m = 1$, the convergence rate of Theorem 6 is determined by the average norm of the features, $\overline{R} = \frac{1}{n}\sum_{i=1}^n \|a_i\|_2$, instead of the largest one $R = \max_i \|a_i\|_2$. This difference makes Algorithm 3 more robust to unnormalized feature vectors. For example, if the a_i's are sampled i.i.d. from a multivariate normal distribution, then $\max_i\{\|a_i\|_2\}$ almost surely goes to infinity as $n \to \infty$, but the average norm $\frac{1}{n}\sum_{i=1}^n \|a_i\|_2$ converges to $\mathbb{E}[\|a_i\|_2]$.

Since θ is a bound on the convergence factor, we would like to make it as small as possible. Let $\rho := R/\overline{R} - 1$. The expression of θ in (24) can be minimized by choosing

$$\alpha^\star = \begin{cases} 0 & \text{if } \rho \leq \sqrt{n/\kappa}, \\ \frac{\rho^{1/2}(\kappa/n)^{1/4} - 1}{\rho^{1/2}(\kappa/n)^{1/4} + \rho} & \text{if } \rho > \sqrt{n/\kappa}. \end{cases} \tag{25}$$

where $\kappa = R^2/(\lambda\gamma)$ is the condition number. The value of $\alpha^\star$ will be equal to zero if the condition number is large enough, and increases slowly to one as the condition number increases. Thus, we choose a (more conservative) uniform distribution for ill-conditioned problems, but a more aggressively weighted distribution for well-conditioned problems.

For simplicity of presentation, we described in Algorithm 3 a weighted sampling SPDC method with single dual coordinate update, i.e., the case of $m = 1$. In fact, the non-uniform sampling scheme can also be extended to mini-batch SPDC. For mini-batch size $m > 1$, we randomly pick a subset of indices $K \subset \{1, 2, \ldots, n\}$ of size m. The probability of $i \in K$ is denoted by p_i and should satisfy the constraint:

$$\min\Big\{1, m\Big(\frac{1-\alpha}{n} + \frac{\alpha\|a_i\|_2}{n\overline{R}}\Big)\Big\} \;\le\; p_i \;\le\; 1 \tag{26}$$

for $i = 1, \ldots, n$, and

$$\sum_{i=1}^{n} p_i = m.$$

This constraint can be satisfied by first adding all indices $\{i : p_i = 1\}$ to the set K, then sampling without replacement from the remaining indices in order to make $|K| = m$. More concretely, there is an efficient sampling-without-replacement algorithm (Chao, 1982) which adds each remaining index i to the set K with probability proportional to $\frac{1-\alpha}{n} + \frac{\alpha\|a_i\|_2}{n\overline{R}}$. It can be verified that the lower bound in (26) holds with such a procedure.

For the mini-batch extension, we replace the updates (21)-(23) by the following updates:

$$y_i^{(t+1)} = \begin{cases} \arg\max_{\beta\in\mathbb{R}} \Big\{\beta\langle a_i, \overline{x}^{(t)}\rangle - \phi_i^*(\beta) - \frac{p_i n}{2\sigma m}(\beta - y_i^{(t)})^2\Big\} & i \in K, \\ y_i^{(t)} & i \notin K, \end{cases}$$

$$u^{(t+1)} = u^{(t)} + \frac{1}{n}\sum_{k\in K}(y_k^{(t+1)} - y_k^{(t)})a_k,$$

$$x^{(t+1)} = \arg\min_{x\in\mathbb{R}^d}\left\{g(x) + \left\langle u^{(t)} + \frac{1}{n}\sum_{k\in K}\frac{y_k^{(t+1)} - y_k^{(t)}}{p_k}a_k,\; x\right\rangle + \frac{\|x - x^{(t)}\|_2^2}{2\tau}\right\},$$

which resembles the updates of Algorithm 2. Similar to the mini-batch SPDC, we are able to show that by increasing the batch size m, the convergence rate of the algorithm will be improved. On the other hand, the lower bound on p_i given by constraint (26) implies that with a proper choice of α (e.g. $\alpha = 1/2$), the convergence rate will depend on

$$\max\;\Big\{\overline{R},\; \frac{m}{n}R\Big\},$$

instead of the maximum norm R. For $m = 1$, we have $\max\{\overline{R}, \frac{m}{n}R\} = \overline{R}$, so that it captures the theoretical guarantee for Algorithm 3 as a special case. We omit the proof details.

5. Related Work

Chambolle and Pock (2011) considered a class of convex optimization problems with the following saddle-point structure:

$$\min_{x\in\mathbb{R}^d}\;\max_{y\in\mathbb{R}^n}\;\big\{\langle Kx, y\rangle + G(x) - F^*(y)\big\}, \tag{27}$$

algorithm	τ	σ	θ	batch complexity
Chambolle-Pock	$\frac{\sqrt{n}}{\|A\|_2}\sqrt{\frac{\gamma}{\lambda}}$	$\frac{n\sqrt{n}}{\|A\|_2}\sqrt{\frac{\lambda}{\gamma}}$	$1-\frac{1}{1+\|A\|_2/(2\sqrt{n\lambda\gamma})}$	$\left(1+\frac{\|A\|_2}{2\sqrt{n\lambda\gamma}}\right)\log(1/\epsilon)$
SPDC with $m=n$	$\frac{1}{R}\sqrt{\frac{\gamma}{\lambda}}$	$\frac{1}{R}\sqrt{\frac{\lambda}{\gamma}}$	$1-\frac{1}{1+R/\sqrt{\lambda\gamma}}$	$\left(1+\frac{R}{\sqrt{\lambda\gamma}}\right)\log(1/\epsilon)$
SPDC with $m=1$	$\frac{1}{R}\sqrt{\frac{\gamma}{n\lambda}}$	$\frac{1}{R}\sqrt{\frac{n\lambda}{\gamma}}$	$1-\frac{1}{n+R\sqrt{n/\lambda\gamma}}$	$\left(1+\frac{R}{\sqrt{n\lambda\gamma}}\right)\log(1/\epsilon)$

Table 2: Comparing step sizes and complexity of SPDC with Chambolle and Pock (2011, Algorithm 3, Theorem 3). Here $A\in\mathbb{R}^{n\times d}$ and its spectral norm $\|A\|_2$ usually grows with n, but always bounded by $\sqrt{n}R$.

where $K\in\mathbb{R}^{m\times d}$, G and F^* are proper closed convex functions, with F^* itself being the conjugate of a convex function F. They developed the following first-order primal-dual algorithm:

$$y^{(t+1)}=\arg\max_{y\in\mathbb{R}^n}\left\{\langle K\overline{x}^{(t)},y\rangle-F^*(y)-\frac{1}{2\sigma}\|y-y^{(t)}\|_2^2\right\}, \tag{28}$$

$$x^{(t+1)}=\arg\min_{x\in\mathbb{R}^d}\left\{\langle K^Ty^{(t+1)},x\rangle+G(x)+\frac{1}{2\tau}\|x-x^{(t)}\|_2^2\right\}, \tag{29}$$

$$\overline{x}^{(t+1)}=x^{(t+1)}+\theta(x^{(t+1)}-x^{(t)}). \tag{30}$$

When both F^* and G are strongly convex and the parameters τ, σ and θ are chosen appropriately, this algorithm obtains accelerated linear convergence rate (Chambolle and Pock, 2011, Theorem 3).

We can map the saddle-point problem (4) into the form of (27) by letting $A=[a_1,\ldots,a_n]^T$ and

$$K=\frac{1}{n}A,\qquad G(x)=g(x),\qquad F^*(y)=\frac{1}{n}\sum_{i=1}^n\phi_i^*(y_i). \tag{31}$$

The SPDC method developed in this paper can be viewed as an extension of the batch method (28)-(30), where the dual update step (28) is replaced by a single coordinate update (5) or a mini-batch update (9). However, in order to obtain accelerated convergence rate, more subtle changes are necessary in the primal update step. More specifically, we introduced the auxiliary variable $u^{(t)}=\frac{1}{n}\sum_{i=1}^n y_i^{(t)}a_i=K^Ty^{(t)}$, and replaced the primal update step (29) by (6) and (10). The primal extrapolation step (30) stays the same.

To compare the batch complexity of SPDC with that of (28)-(30), we use the following facts implied by Assumption A and the relations in (31):

$$\|K\|_2=\frac{1}{n}\|A\|_2,\quad G(x)\text{ is }\lambda\text{-strongly convex},\quad\text{and }F^*(y)\text{ is }(\gamma/n)\text{-strongly convex}.$$

Based on these conditions, we list in Table 2 the equivalent parameters used by Algorithm 3 in Chambolle and Pock (2011) and the batch complexity obtained in Theorem 3 of that paper, and compare them with SPDC.

The batch complexity of the Chambolle-Pock algorithm is $\widetilde{\mathcal{O}}(1+\|A\|_2/(2\sqrt{n\lambda\gamma}))$, where the $\widetilde{\mathcal{O}}(\cdot)$ notation hides the $\log(1/\epsilon)$ factor. We can bound the spectral norm $\|A\|_2$ by the Frobenius norm $\|A\|_F$ and obtain

$$\|A\|_2 \le \|A\|_F \le \sqrt{n}\max_i\{\|a_i\|_2\} = \sqrt{n}R.$$

(Note that the second inequality above would be an equality if the columns of A are normalized.) So in the worst case, the batch complexity of the Chambolle-Pock algorithm becomes

$$\widetilde{\mathcal{O}}\left(1 + R/\sqrt{\lambda\gamma}\right) = \widetilde{\mathcal{O}}\left(1+\sqrt{\kappa}\right), \qquad \text{where } \kappa = R^2/(\lambda\gamma),$$

which matches the worst-case complexity of the accelerated gradient methods (Nesterov, 2004, 2013); see Section 1.1 and also the discussions in Lin et al. (2015b, Section 5). This is also of the same order as the complexity of SPDC with $m = n$ (see Section 2.1). When the condition number $\kappa \gg 1$, they can be $\sqrt{n}$ worse than the batch complexity of SPDC with $m = 1$, which is $\widetilde{\mathcal{O}}(1+\sqrt{\kappa/n})$.

If either $G(x)$ or $F^*(y)$ in (27) is not strongly convex, Chambolle and Pock (2011, Section 5.1) proposed variants of the primal-dual batch algorithm to achieve accelerated sublinear convergence rates. It is also possible to extend them to coordinate update methods for solving problem (1) when either ϕ_i^* or g is not strongly convex. Their complexities would be similar to those in Table 1.

Our algorithms and theory can be readily generalized to solve the problem of

$$\underset{x\in\mathbb{R}^d}{\text{minimize}} \quad \frac{1}{n}\sum_{i=1}^n \phi_i(A_i^T x) + g(x),$$

where each A_i is an $d_i \times d$ matrix, and $\phi_i : \mathbb{R}^{d_i} \to \mathbb{R}$ is a smooth convex function. This more general formulation is used, e.g., in Shalev-Shwartz and Zhang (2015). Most recently, Lan and Zhou (2015) considered the case with $d_i = d$ and $A_i = I_d$, which corresponding to a general class of problems with the finite-sum (or finite-average) structure. He extended the primal-dual algorithm by replacing the quadratic penalty terms in (5) and (21) with the Bregman divergence associated with the loss functions themselves. This led to an algorithm that does not rely on computing the proximal mapping of the conjugate $\phi_i^\star$, but only requires computing the primal gradient $\nabla\phi_i$ at a particular sequence of the primal variables. As a result, the algorithm in Lan and Zhou (2015) can be considered as a (primal-only or dual-free) randomized incremental gradient algorithm, which share the same order of iteration complexity as SPDC.

5.1 Dual Coordinate Ascent Methods

We can also solve the primal problem (1) via its dual:

$$\underset{y\in\mathbb{R}^n}{\text{maximize}} \ \left\{ D(y) \ \stackrel{\text{def}}{=} \ \frac{1}{n}\sum_{i=1}^n -\phi_i^*(y_i) - g^*\Big(-\frac{1}{n}\sum_{i=1}^n y_i a_i\Big)\right\}, \tag{32}$$

where $g^*(u) = \sup_{x\in\mathbb{R}^d}\{x^T u - g(x)\}$ is the convex conjugate of g. Due to the problem structure, it is well-known that coordinate ascent methods can be more efficient than full

gradient methods for solving this problem (e.g., Platt, 1999; Chang et al., 2008; Hsieh et al., 2008; Shalev-Shwartz and Zhang, 2013a). In the stochastic dual coordinate ascent (SDCA) method a dual coordinate y_i is picked at random during each iteration and updated to increase the dual objective value. Shalev-Shwartz and Zhang (2013a) showed that the iteration complexity of SDCA is $O\left((n+\kappa)\log(1/\epsilon)\right)$, which corresponds to the batch complexity $\mathcal{O}\left((1+\kappa/n)\log(1/\epsilon)\right)$.

For more general convex optimization problems, there is a vast literature on coordinate descent methods; see, e.g., the recent overview by Wright (2015). In particular, the work of Nesterov (2012) on randomized coordinate descent sparked a lot of recent activities on this topic. Richtárik and Takáč (2014) extended the algorithm and analysis to composite convex optimization. When applied to the dual problem (32), it becomes one variant of the SDCA algorithm studied in Shalev-Shwartz and Zhang (2013a). Mini-batch and distributed versions of SDCA have been proposed and analyzed in Takáč et al. (2013) and Yang (2013) respectively. Non-uniform sampling schemes similar to the one used in Algorithm 3 have been studied for both stochastic gradient and dual coordinate ascent methods (e.g., Needell et al., 2016; Xiao and Zhang, 2014; Zhao and Zhang, 2015; Qu et al., 2015).

Shalev-Shwartz and Zhang (2013b) proposed an accelerated mini-batch SDCA method which incorporates additional primal updates than SDCA, and bears some similarity to our mini-batch SPDC method. They showed that its complexity interpolates between that of SDCA and accelerated gradient methods by varying the mini-batch size m. In particular, for $m = n$, it matches that of the accelerated gradient methods (as SPDC does). But for $m = 1$, the complexity of their method is the same as SDCA, which is worse than SPDC for ill-conditioned problems.

In addition, Shalev-Shwartz and Zhang (2015) developed an accelerated proximal SDCA method which achieves the same batch complexity $\widetilde{\mathcal{O}}\big(1+\sqrt{\kappa/n}\big)$ as SPDC. Their method is an inner-outer iteration procedure, where the outer loop is a full-dimensional accelerated gradient method in the primal space $x \in \mathbb{R}^d$. At each iteration of the outer loop, the SDCA method (Shalev-Shwartz and Zhang, 2013a) is called to solve the dual problem (32) with customized regularization parameter and precision. In contrast, SPDC is a straightforward single-loop coordinate optimization methods. Two recent works extended the inner-outer iteration method to derive more general accelerated proximal-point algorithms: Frostig et al. (2015) and Lin et al. (2015a). Basically, one can replace the inner-loop SDCA algorithm by other efficient algorithms such as Prox-SVRG (Xiao and Zhang, 2014) or SAGA (Defazio et al., 2014) to obtain the same overall complexity.

More recently, Lin et al. (2015b) developed an accelerated proximal coordinate gradient (APCG) method for solving a more general class of composite convex optimization problems. When applied to solve the dual problem (32), APCG enjoys the same batch complexity $\widetilde{\mathcal{O}}\big(1+\sqrt{\kappa/n}\big)$ for reducing the dual objective gap. In order to obtain the same complexity for the primal-dual gap, One needs an extra primal proximal-gradient step at the end after applying the APCG algorithm. The computational cost of this additional step is equivalent to one pass of the dataset, thus it does not affect the overall complexity.

5.2 Other Related Work

Another way to approach problem (1) is to reformulate it as a constrained optimization problem

$$\begin{aligned} \text{minimize} \quad & \frac{1}{n}\sum_{i=1}^{n}\phi_i(z_i) + g(x) \\ \text{subject to} \quad & a_i^T x = z_i, \quad i = 1, \ldots, n, \end{aligned} \tag{33}$$

and solve it by ADMM type of operator-splitting methods (e.g., Lions and Mercier, 1979; Boyd et al., 2010). In fact, as shown in Chambolle and Pock (2011), the batch primal-dual algorithm (28)-(30) is equivalent to a pre-conditioned ADMM or an inexact Uzawa method (see, e.g., Zhang et al., 2011). Several authors (Wang and Banerjee, 2012; Ouyang et al., 2013; Suzuki, 2013; Zhong and Kwok, 2014) have considered a more general formulation than (33), where each ϕ_i is a function of the whole vector $z \in \mathbb{R}^n$. They proposed online or stochastic versions of ADMM which operate on only one ϕ_i in each iteration, and obtained sublinear convergence rates. However, their cost per iteration is $\mathcal{O}(nd)$ instead of $\mathcal{O}(d)$.

Suzuki (2014) considered a problem similar to (1), but with more complex regularization function g, meaning that g does not have a simple proximal mapping. Thus primal updates such as step (6) or (10) in SPDC and similar steps in SDCA cannot be computed efficiently. He proposed an algorithm that combines SDCA (Shalev-Shwartz and Zhang, 2013a) and ADMM, and showed that it has linear rate of convergence under similar conditions as Assumption A. It would be interesting to see if the SPDC method can be extended to their setting to obtain accelerated linear convergence rate.

6. Efficient Implementation with Sparse Data

During each iteration of the SPDC method, the update of primal variables (i.e., computing $x^{(t+1)}$) requires full d-dimensional vector operations; see the step (6) of Algorithm 1, the step (10) of Algorithm 2 and the step (23) of Algorithm 3. So the computational cost per iteration is $\mathcal{O}(d)$, and this can be too expensive if the dimension d is very high. In this section, we show how to exploit problem structure to avoid high-dimensional vector operations when the feature vectors a_i are sparse. We illustrate the efficient implementation for two popular cases: when g is an squared-ℓ_2 penalty and when g is an $\ell_1+\ell_2$ penalty. For both cases, we show that the computation cost per iteration only depends on the number of non-zero components of the feature vector.

6.1 Squared ℓ_2-Norm Penalty

Suppose that $g(x) = \frac{\lambda}{2}\|x\|_2^2$. For this case, the updates for each coordinate of x are independent of each other. More specifically, $x^{(t+1)}$ can be computed coordinate-wise in closed form:

$$x_j^{(t+1)} = \frac{1}{1+\lambda\tau}(x_j^{(t)} - \tau u_j^{(t)} - \tau\Delta u_j), \quad j = 1, \ldots, n, \tag{34}$$

where Δu denotes $(y_k^{(t+1)} - y_k^{(t)})a_k$ in Algorithm 1, or $\frac{1}{m}\sum_{k\in K}(y_k^{(t+1)} - y_k^{(t)})a_k$ in Algorithm 2, or $(y_k^{(t+1)} - y_k^{(t)})a_k/(p_k n)$ in Algorithm 3, and Δu_j represents the j-th coordinate of Δu.

Although the dimension d can be very large, we assume that each feature vector a_k is sparse. We denote by $J^{(t)}$ the set of non-zero coordinates at iteration t, that is, if for some index $k \in K$ picked at iteration t we have $a_{kj} \neq 0$, then $j \in J^{(t)}$. If $j \notin J^{(t)}$, then the SPDC algorithm (and its variants) updates $y^{(t+1)}$ without using the value of $x_j^{(t)}$ or $\overline{x}_j^{(t)}$. This can be seen from the updates in (5), (9) and (21), where the value of the inner product $\langle a_k, \overline{x}^{(t)} \rangle$ does not depend on the value of $\overline{x}_j^{(t)}$. As a consequence, we can delay the updates on x_j and $\overline{x}_j$ whenever $j \notin J^{(t)}$ without affecting the updates on $y^{(t)}$, and process all the missing updates at the next time when $j \in J^{(t)}$.

Such a delayed update can be carried out very efficiently. We assume that t_0 is the last time when $j \in J^{(t)}$, and t_1 is the current iteration where we want to update x_j and $\overline{x}_j$. Since $j \notin J^{(t)}$ implies $\Delta u_j = 0$, we have

$$x_j^{t+1} = \frac{1}{1+\lambda\tau}(x_j^{(t)} - \tau u_j^{(t)}), \qquad t = t_0+1, t_0+2, \ldots, t_1-1. \tag{35}$$

Notice that $u_j^{(t)}$ is updated only at iterations where $j \in J^{(t)}$. The value of $u_j^{(t)}$ doesn't change during iterations $[t_0+1, t_1]$, so we have $u_j^{(t)} \equiv u_j^{(t_0+1)}$ for $t \in [t_0+1, t_1]$. Substituting this equation into the recursive formula (35), we obtain

$$x_j^{(t_1)} = \frac{1}{(1+\lambda\tau)^{t_1-t_0-1}} \left(x_j^{(t_0+1)} + \frac{u_j^{(t_0+1)}}{\lambda} \right) - \frac{u_j^{(t_0+1)}}{\lambda}. \tag{36}$$

The update (36) takes $\mathcal{O}(1)$ time to compute. Using the same formula, we can compute $x_j^{(t_1-1)}$ and subsequently compute $\overline{x}_j^{(t_1)} = x_j^{(t_1)} + \theta(x_j^{(t_1)} - x_j^{(t_1-1)})$. Thus, the computational complexity of a single iteration in SPDC is proportional to $|J^{(t)}|$, independent of the dimension d.

We note that similar tricks of delayed updates, or "just-in-time" updates, have been derived and used for the SAG algorithm (Schmidt et al., 2013). For SPDC, the delayed updates become more complex due to the full vector extrapolation required for Nesterov-type acceleration.

6.2 $(\ell_1 + \ell_2)$-Norm Penalty

Suppose that $g(x) = \lambda_1 \|x\|_1 + \frac{\lambda_2}{2}\|x\|_2^2$. Since both the ℓ_1-norm and the squared ℓ_2-norm are decomposable, the updates for each coordinate of $x^{(t+1)}$ are independent. More specifically,

$$x_j^{(t+1)} = \arg\min_{\alpha \in \mathbb{R}} \left\{ \lambda_1 |\alpha| + \frac{\lambda_2 \alpha^2}{2} + (u_j^{(t)} + \Delta u_j)\alpha + \frac{(\alpha - x_j^{(t)})^2}{2\tau} \right\}, \tag{37}$$

where Δu_j follows the definition in Section 6.1. If $j \notin J^{(t)}$, then $\Delta u_j = 0$ and equation (37) can be simplified as

$$x_j^{(t+1)} = \begin{cases} \frac{1}{1+\lambda_2\tau}(x_j^{(t)} - \tau u_j^{(t)} - \tau\lambda_1) & \text{if } x_j^{(t)} - \tau u_j^{(t)} > \tau\lambda_1, \\ \frac{1}{1+\lambda_2\tau}(x_j^{(t)} - \tau u_j^{(t)} + \tau\lambda_1) & \text{if } x_j^{(t)} - \tau u_j^{(t)} < -\tau\lambda_1, \\ 0 & \text{otherwise,} \end{cases} \qquad t \in [t_0+1, t_1]. \tag{38}$$

Similar to the approach of Section 6.1, we delay the update of x_j until $j \in J^{(t)}$. We assume t_0 to be the last iteration when $j \in J^{(t)}$, and let t_1 be the current iteration when we want to update x_j. During iterations $[t_0+1, t_1]$, the value of $u_j^{(t)}$ doesn't change, so we have $u_j^{(t)} \equiv u_j^{(t_0+1)}$ for $t \in [t_0+1, t_1]$. Using equation (38) and the invariance of $u_j^{(t)}$ for $t \in [t_0+1, t_1]$, we have an $\mathcal{O}(1)$ time algorithm to calculate $x_j^{(t_1)}$. More specifically, given $x_j^{(t_0+1)}$ at iteration t_0, we present an efficient algorithm for calculating $x_j^{(t_1)}$. We begin by examining the sign of $x_j^{(t_0+1)}$.

Case I ($x_j^{(t_0+1)} = 0$): If $-u_j^{(t_0+1)} > \lambda_1$, then equation (38) implies $x_j^{(t)} > 0$ for all $t > t_0+1$. Consequently, we have a closed-form formula for $x_j^{(t_1)}$:

$$x_j^{(t_1)} = \frac{1}{(1+\lambda_2\tau)^{t_1-t_0-1}}\Big(x_j^{(t_0+1)} + \frac{u_j^{(t_0+1)} + \lambda_1}{\lambda_2}\Big) - \frac{u_j^{(t_0+1)} + \lambda_1}{\lambda_2}. \tag{39}$$

If $-u_j^{(t_0+1)} < -\lambda_1$, then equation (38) implies $x_j^{(t)} < 0$ for all $t > t_0+1$. Therefore, we have the closed-form formula:

$$x_j^{(t_1)} = \frac{1}{(1+\lambda_2\tau)^{t_1-t_0-1}}\Big(x_j^{(t_0+1)} + \frac{u_j^{(t_0+1)} - \lambda_1}{\lambda_2}\Big) - \frac{u_j^{(t_0+1)} - \lambda_1}{\lambda_2}. \tag{40}$$

Finally, if $-u_j^{(t_0+1)} \in [-\lambda_1, \lambda_1]$, then equation (38) implies $x_j^{(t_1)} = 0$.

Case II ($x_j^{(t_0+1)} > 0$): If $-u_j^{(t_0+1)} \geq \lambda_1$, then it is easy to verify that $x_j^{(t_1)}$ is obtained by equation (39). Otherwise, We use the recursive formula (38) to derive the latest time $t^+ \in [t_0+1, t_1]$ such that $x_j^{t^+} > 0$ is true. Indeed, since $x_j^{(t)} > 0$ for all $t \in [t_0+1, t^+]$, we have a closed-form formula for $x_j^{t^+}$:

$$x_j^{t^+} = \frac{1}{(1+\lambda_2\tau)^{t^+-t_0-1}}\Big(x_j^{(t_0+1)} + \frac{u_j^{(t_0+1)} + \lambda_1}{\lambda_2}\Big) - \frac{u_j^{(t_0+1)} + \lambda_1}{\lambda_2}. \tag{41}$$

We look for the largest t^+ such that the right-hand side of equation (41) is positive, which is equivalent of

$$t^+ - t_0 - 1 < \log\Big(1 + \frac{\lambda_2 x_j^{(t_0+1)}}{u_j^{(t_0+1)} + \lambda_1}\Big)/\log(1+\lambda_2\tau). \tag{42}$$

Thus, t^+ is the largest integer in $[t_0+1, t_1]$ such that inequality (42) holds. If $t^+ = t_1$, then $x_j^{(t_1)}$ is obtained by (41). Otherwise, we can calculate $x_j^{t^++1}$ by formula (38), then resort to Case I or Case III, treating t^+ as t_0.

Case III ($x_j^{(t_0+1)} < 0$): If $-u_j^{(t_0+1)} \leq -\lambda_1$, then $x_j^{(t_1)}$ is obtained by equation (40). Otherwise, we calculate the largest integer $t^- \in [t_0+1, t_1]$ such that $x_j^{t^-} < 0$ is true. Using the same argument as for Case II, we have the closed-form expression

$$x_j^{t^-} = \frac{1}{(1+\lambda_2\tau)^{t^--t_0-1}}\Big(x_j^{(t_0+1)} + \frac{u_j^{(t_0+1)} - \lambda_1}{\lambda_2}\Big) - \frac{u_j^{(t_0+1)} - \lambda_1}{\lambda_2}. \tag{43}$$

where t^- is the largest integer in $[t_0+1, t_1]$ such that the following inequality holds:

$$t^- - t_0 - 1 < \log\Big(1 + \frac{\lambda_2 x_j^{(t_0+1)}}{u_j^{(t_0+1)} - \lambda_1}\Big)/\log(1+\lambda_2\tau). \tag{44}$$

If $t^- = t_1$, then $x_j^{(t_1)}$ is obtained by (43). Otherwise, we can calculate $x_j^{t^-+1}$ by formula (38), then resort to Case I or Case II, treating t^- as t_0.

Finally, we note that formula (38) implies the monotonicity of $x_j^{(t)}$ ($t = t_0+1, t_0+2, \dots$). As a consequence, the procedure of either Case I, Case II or Case III is executed for at most once. Hence, the algorithm for calculating $x_j^{(t_1)}$ has $\mathcal{O}(1)$ time complexity.

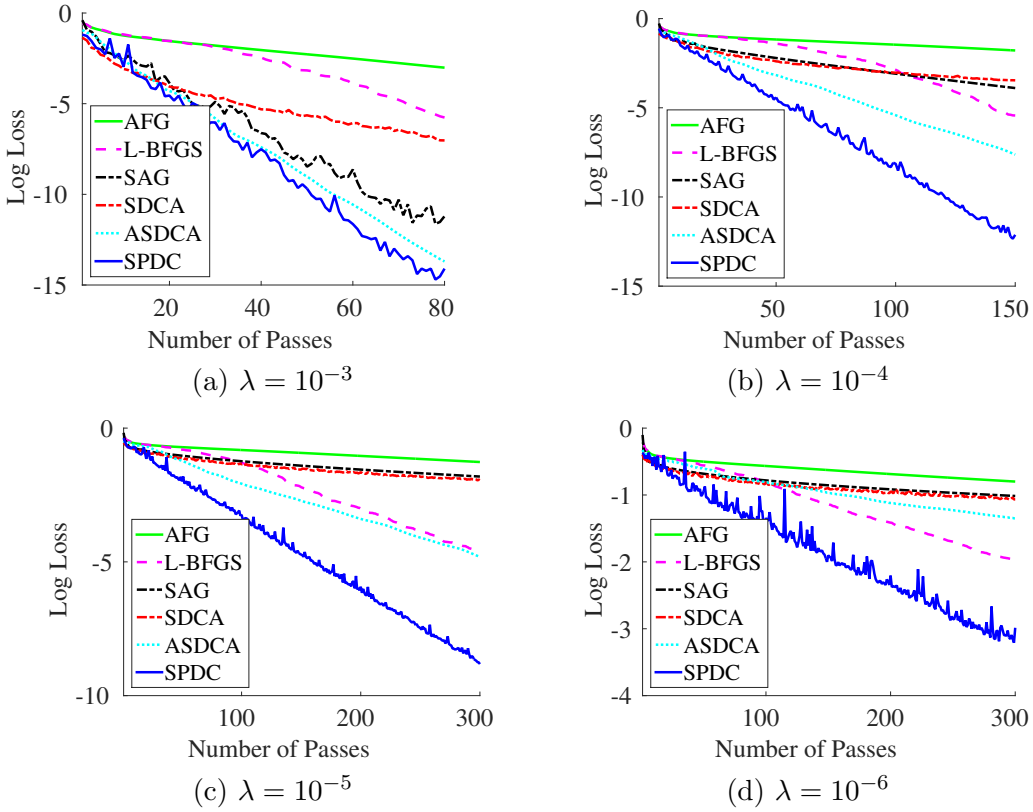
The vector $\overline{x}_j^{(t_1)}$ can be updated by the same algorithm since it is a linear combination of $x_j^{(t_1)}$ and $x_j^{(t_1-1)}$. As a consequence, the computational complexity of each iteration in SPDC is proportional to $|J^{(t)}|$, independent of the dimension d.

7. Experiments

In this section, we compare the basic SPDC method (Algorithm 1) with several state-of-the-art algorithms for solving problem (1). They include two batch-update algorithms: the accelerated full gradient (AFG) method (Nesterov, 2004, Section 2.2), and the limited-memory quasi-Newton method L-BFGS (Nocedal and Wright, 2006, Section 7.2)). For the AFG method, we adopt an adaptive line search scheme (Nesterov, 2013) to improve its efficiency. For the L-BFGS method, we use the memory size 30 as suggested by Nocedal and Wright (2006, Section 7.2).

We also compare SPDC with three stochastic algorithms: the stochastic average gradient (SAG) method (Le Roux et al., 2012; Schmidt et al., 2013), the stochastic dual coordinate descent (SDCA) method (Shalev-Shwartz and Zhang, 2013a) and the accelerated stochastic dual coordinate descent (ASDCA) method(Shalev-Shwartz and Zhang, 2015). We conduct experiments on a synthetic dataset and three real datasets. The hyper-parameters τ, σ, θ of the SPDC algorithm are chosen by their theoretical values given in (12). For SDCA, we use their default parameter settings given in Shalev-Shwartz and Zhang (2013a), which can be determined from the Lipschitz constant $1/\gamma$ of the loss functions and the strongly convex regularization parameter λ. For SAG, we choose the learning rate $\alpha = \gamma/R$ as recommended by Schmidt et al. (2013).

We caution the readers that our numerical experiments in this section focus on comparing the *optimization* performance of various algorithms, i.e., how fast they can reduce the objective function to its minimum value. In order to illustrate their differences for very ill-conditioned problems, we often run the algorithms with hundreds of passes over the datasets in order to reach a small optimization error. Such large numbers of passes over datasets and the small optimization errors may not be appropriate for machine learning problems, especially from the point of view of *generalization* (reducing testing error). See Bottou and Bousquet (2008) for discussions on the fundamental tradeoffs of large scale learning problems.



(a) $\lambda = 10^{-3}$ (b) $\lambda = 10^{-4}$

(c) $\lambda = 10^{-5}$ (d) $\lambda = 10^{-6}$

Figure 1: Comparing SPDC with other methods for ridge regression on synthetic data, with the regularization coefficient $\lambda \in \{10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$. The horizontal axis is the number of passes through the entire dataset, and the vertical axis is the logarithmic gap $\log(P(x^{(T)}) - P(x^\star))$.

7.1 Ridge Regression with Synthetic Data

We first compare SPDC with other algorithms on a simple quadratic problem using synthetic data. We generate $n = 500$ i.i.d. training examples $\{a_i, b_i\}_{i=1}^n$ according to the model

$$b = \langle a, x^* \rangle + \varepsilon, \quad a \sim \mathcal{N}(0, \Sigma), \quad \varepsilon \sim \mathcal{N}(0, 1),$$

where $a \in \mathbb{R}^d$ and $d = 500$, and x^* is the all-ones vector. To make the problem ill-conditioned, the covariance matrix Σ is set to be diagonal with $\Sigma_{jj} = j^{-2}$, for $j = 1, \ldots, d$. Given the set of examples $\{a_i, b_i\}_{i=1}^n$, we then solved a standard ridge regression problem

$$\underset{x \in \mathbb{R}^d}{\text{minimize}} \quad \left\{ P(x) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n \frac{1}{2}(a_i^T x - b_i)^2 + \frac{\lambda}{2}\|x\|_2^2 \right\}.$$

In the form of problem (1), we have $\phi_i(z) = z^2/2$ and $g(x) = (1/2)\|x\|_2^2$. As a consequence, the derivative of ϕ_i is 1-Lipschitz continuous and g is λ-strongly convex.

Dataset name	# of samples n	# of features d	sparsity	size
Covtype	581,012	54	22%	71 MB
RCV1	20,242	47,236	0.16%	37 MB
News20	19,996	1,355,191	0.04%	140 MB
RCV1-test	677,399	47,236	0.15%	1.2 GB
URL	2,396,130	3,231,961	0.004%	2.2 GB

Table 3: Characteristics of real datasets from LIBSVM data (Fan and Lin, 2011).

We evaluate the algorithms by the logarithmic optimality gap $\log(P(x^{(t)}) - P(x^\star))$, where $x^{(t)}$ is the output of the algorithms after t passes over the entire dataset, and $x^\star$ is the global minimum. When the regularization coefficient is relatively large, e.g., $\lambda = 10^{-1}$ or 10^{-2}, the problem is well-conditioned and we observe fast convergence of the stochastic algorithms SAG, SDCA, ASDCA and SPDC, which are substantially faster than the two batch methods AFG and L-BFGS.

Figure 1 shows the convergence of the five different algorithms when we varied λ from 10^{-3} to 10^{-6}. As the plot shows, when the condition number is greater than n, the SPDC algorithm also converges substantially faster than the other two stochastic methods SAG and SDCA. It is also notably faster than L-BFGS. These results support our theory that SPDC enjoys a faster convergence rate on ill-conditioned problems. In terms of their batch complexities, SPDC is up to $\sqrt{n}$ times faster than AFG, and $(\lambda n)^{-1/2}$ times faster than SAG and SDCA.

Theoretically, ASDCA enjoys the same batch complexity as SPDC up to a multiplicative constant factor. Figure 1 shows that the empirical performance of SPDC is substantially faster that of ASDCA for small λ. This may due to the fact that ASDCA follows an inner-outer iteration procedure, which requires careful selection of the regularization parameter and accuracy to reach for each call of SDCA. SPDC is a single-loop algorithm that needs less parameters to set up, thus it can be empirically more efficient.

7.2 Binary Classification with Real Data

Finally we show the results of solving the binary classification problem on real datasets. The datasets are obtained from the LIBSVM data collection (Fan and Lin, 2011) and summarized in Table 3. The first three datasets are selected to reflect different relations between the sample size n and the feature dimensionality d, which cover $n \gg d$ (Covtype), $n \approx d$ (RCV1) and $n \ll d$ (News20). The remaining two are relatively larger datasets (RCV1-test and URL) that we did not test in previous experiments conducted in Zhang and Xiao (2015). For all tasks, the data points take the form of (a_i, b_i), where $a_i \in \mathbb{R}^d$ is the feature vector, and $b_i \in \{-1, 1\}$ is the binary class label. As a preprocessing step, the feature vectors are normalized to the unit ℓ_2-norm (meaning $R = 1$).

Our goal is to minimize the regularized empirical risk:

$$P(x) = \frac{1}{n}\sum_{i=1}^{n} \phi_i(a_i^T x) + \frac{\lambda}{2}\|x\|_2^2, \quad \text{where} \quad \phi_i(z) = \begin{cases} 0 & \text{if } b_i z \geq 1 \\ \frac{1}{2} - b_i z & \text{if } b_i z \leq 0 \\ \frac{1}{2}(1 - b_i z)^2 & \text{otherwise.} \end{cases}$$

Here, ϕ_i is the smoothed hinge loss (see, e.g., Shalev-Shwartz and Zhang, 2013a). It is easy to verify that the conjugate function of ϕ_i is $\phi_i^*(\beta) = b_i\beta + \frac{1}{2}\beta^2$ for $b_i\beta \in [-1, 0]$ and ∞ otherwise.

The performance of the five algorithms on the three smaller datasets are plotted in Figure 2 and Figure 3. In Figure 2, we compare SPDC with the two batch methods: AFG and L-BFGS. The results show that SPDC is substantially faster than AFG and L-BFGS for relatively large λ, illustrating the advantage of stochastic methods over batch methods on well-conditioned problems. As λ decreases to 10^{-8}, the batch methods (especially L-BFGS) become comparable to SPDC.

In Figure 3, we compare SPDC with the three stochastic methods: SAG, SDCA and ASDCA. Note that the specification of ASDCA (Shalev-Shwartz and Zhang, 2015) requires the regularization coefficient λ satisfies $\lambda \leq \frac{R^2}{10n}$ where R is the maximum ℓ_2-norm of feature vectors. To satisfy this constraint, we run ASDCA with $\lambda \in \{10^{-6}, 10^{-7}, 10^{-8}\}$. In Figure 3, the observations are just the opposite to that of Figure 2. All stochastic algorithms have comparable performances on relatively large λ, but SPDC and ASDCA becomes substantially faster when λ gets closer to zero. In particular, ASDCA converges faster than SPDC on the Covtype dataset, but SPDC is faster on the remaining two datasets. In addition, due to the outer-inner loop structure of the ASDCA algorithm, its error rate oscillates and might be bad at early iterations. In contrast, the curve of SPDC is almost linear and it is more stable than ASDCA.

Figure 4 plots the convergence results on the last two datasets, where both the sample size n and the dimension d are big. If the regularization parameter λ is also relatively large, then the stochastic algorithms (SPDC, SAG and SDCA) will guarantee to converge very quickly, making it an easy optimization problem. We report experiments on small regularization values $\lambda = 10^{-6}$ and $\lambda = 10^{-8}$. The ASDCA algorithm is reported on $\lambda = 10^{-8}$ because it satisfies the constraint $\lambda \leq \frac{R^2}{10n}$. Comparing results on the RCV1 and RCV1-test datasets, the SPDC algorithm has a more significant advantage over the batch methods (AFG and L-BFGS) on the bigger dataset, because the stochastic algorithm converges faster with a larger sample size. On the other hand, the performance gaps between SPDC and the two other stochastic methods (SAG and SDCA) are less significant on the bigger dataset. We observe the same phenomenon on the URL dataset.

Summarizing Figure 2, Figure 3 and Figure 4, the performance of the SPDC algorithm are always comparable or better than the other methods, for various of relations between the sample size n and the dimension d, and on both small and large datasets.

7.3 Uniform Sampling versus Non-Uniform Sampling

In this subsection, we compare the uniform sampling strategy (Algorithm 1) and the non-uniform sampling strategy (Algorithm 3) for SPDC. We repeat the binary classification experiments on the Covtype, RCV1 and News20 datasets, but this time without performing feature normalization. More precisely, for each data point taking the form of (a_i, b_i), we don't normalize the feature vector a_i to the unit ℓ_2-norm. But instead, we multiply a constant number to every feature vector so that the average ℓ_2-norm $\overline{R}$ is equal to one. It ensures that the overall loss function is 1-smooth.

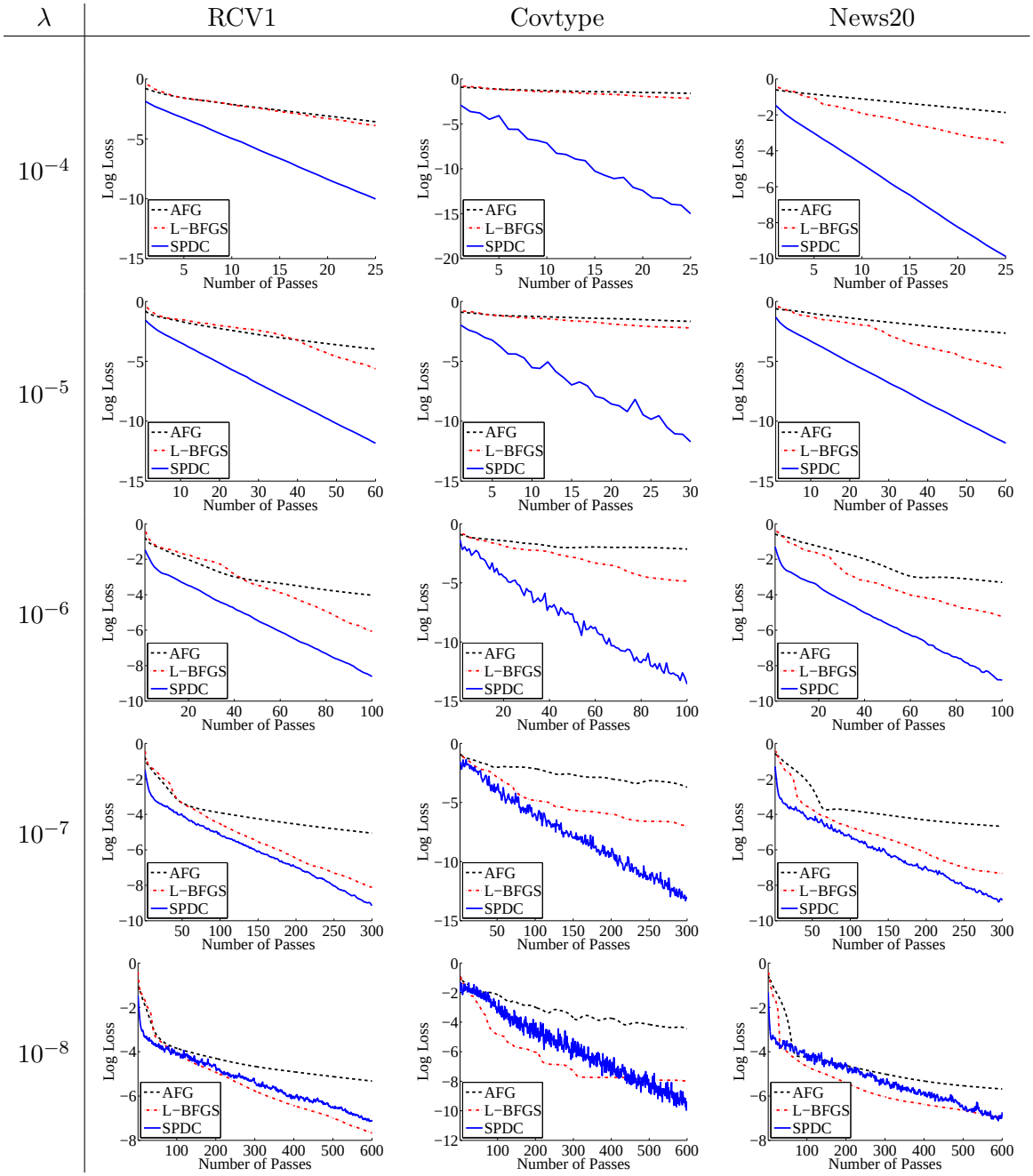


Figure 2: Comparing SPDC with AFG and L-BFGS on three real datasets with smoothed hinge loss. The horizontal axis is the number of passes through the entire dataset, and the vertical axis is the logarithmic optimality gap $\log(P(x^{(t)}) - P(x^\star))$. The SPDC algorithm is faster than the two batch methods when λ is relatively large.

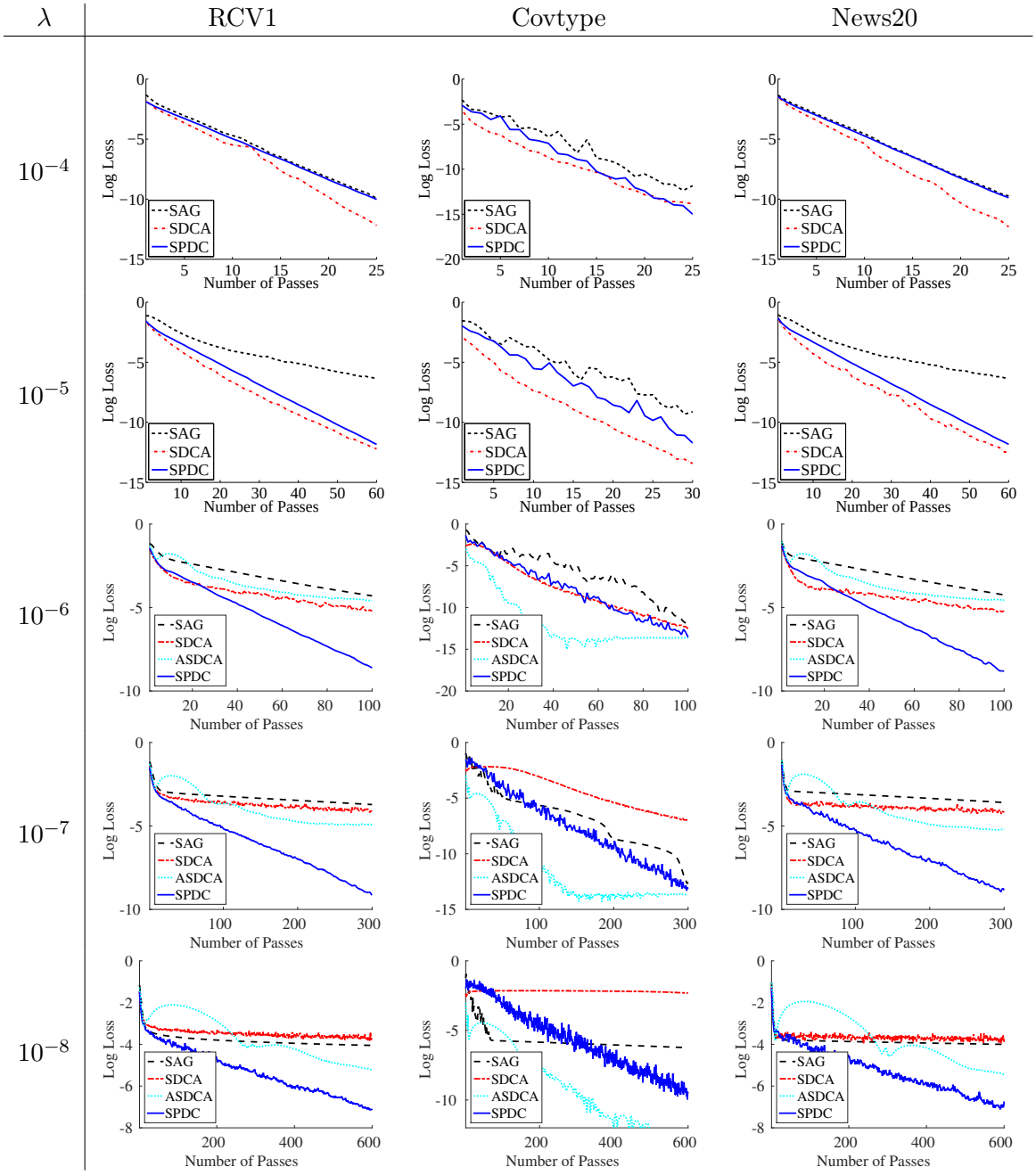
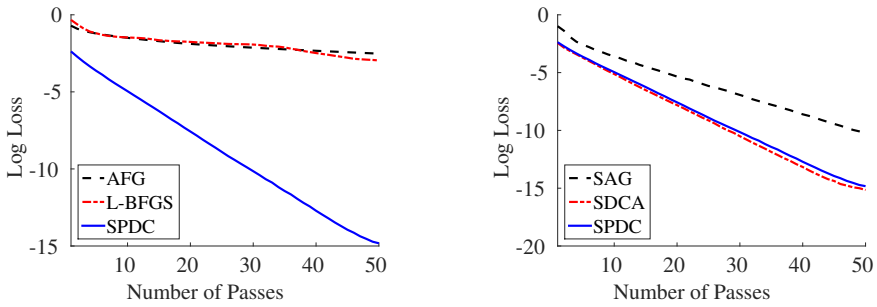
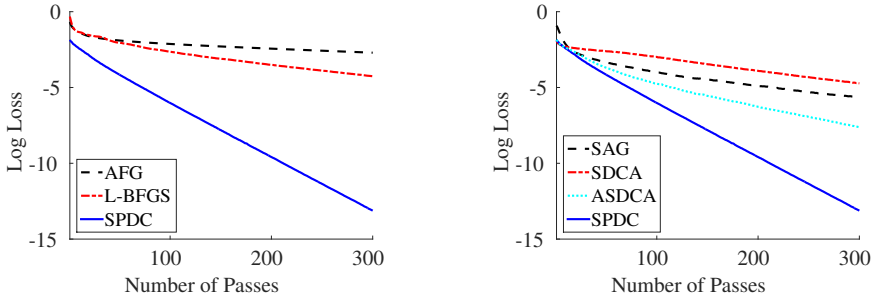


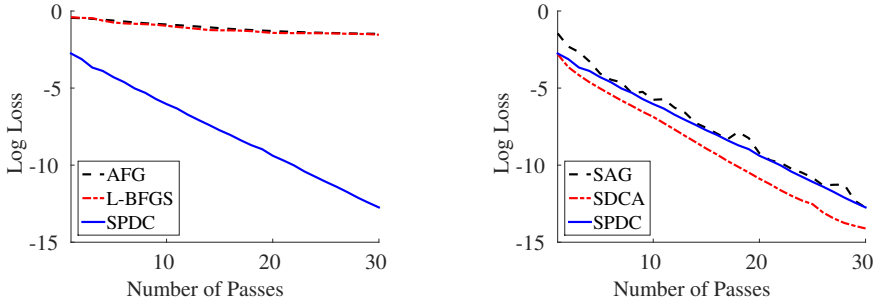
Figure 3: Comparing SPDC with SAG, SDCA and ASDCA on three real datasets with smoothed hinge loss. The horizontal axis is the number of passes through the entire dataset, and the vertical axis is the logarithmic optimality gap $\log(P(x^{(T)}) - P(x^\star))$. The SPDC algorithm is faster than SAG and SDCA when λ is small. It is faster than ASDCA on datasets RCV1 and News20.



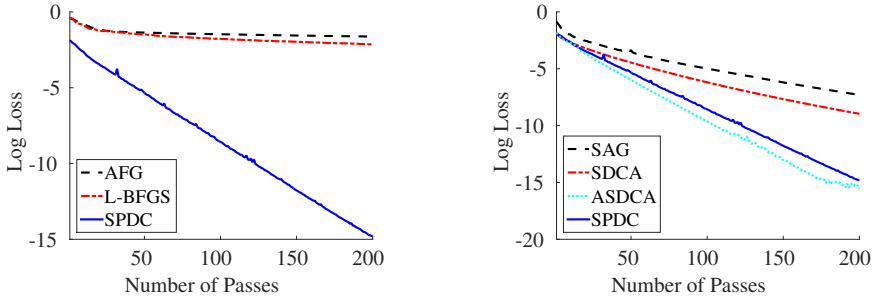
(a) RCV1-test ($\lambda = 10^{-6}$)



(b) RCV1-test ($\lambda = 10^{-8}$)

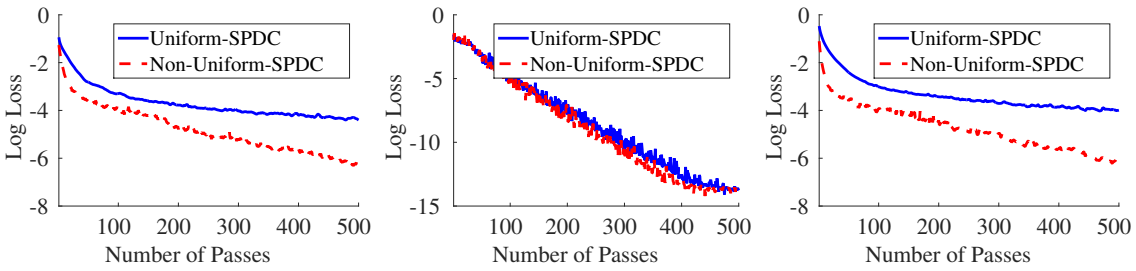


(c) URL ($\lambda = 10^{-6}$)



(d) URL ($\lambda = 10^{-8}$)

Figure 4: Comparing SPDC with other methods on the two larger datasets. The vertical axis is the logarithmic optimality gap $\log(P(x^{(t)}) - P(x^\star))$.



(a) RCV1 ($R/\overline{R} = 3.87$) (b) Covtype ($R/\overline{R} = 1.25$) (c) News20 ($R/\overline{R} = 6.71$)

Figure 5: Comparing the uniform sampling and non-uniform sampling strategies for SPDC, where the vertical axis is the logarithmic optimality gap $\log(P(x^{(t)}) - P(x^\star))$. The quantity $R/\overline{R}$ represents the ratio between the largest feature norm and the average feature norm. When this quantity is large, the non-uniform sampling algorithm converges significantly faster.

For this experiment, we use a small regularization parameter $\lambda = 10^{-8}$, so that the problem has a large condition number. The hyper-parameters τ, σ, θ are chosen by their theoretical values in (12) and (24), respectively for the two sampling strategies. For non-uniform sampling, the hyper-parameter α is chosen by the optimal value given in (25). Figure 5 plots their convergence profiles. On all of the three datasets, the non-uniform sampling algorithm converges faster. The margin is quite significant when the ratio between the largest feature norm R and the average feature norm $\overline{R}$ is relatively large. This is consistent with our analysis in Theorem 1 and Theorem 6, which state that the convergence rate of the uniform sampling algorithm depends R, while that of the non-uniform sampling algorithm depends on $\overline{R}$.

Acknowledgments

We are grateful to Qihang Lin for helpful discussions, especially on the proof of Lemma 3.

Appendix A. Proof of Theorem 1

We focus on characterizing the values of x and y after the t-th update in Algorithm 2. For any $i \in \{1, \ldots, n\}$, let $\widetilde{y}_i$ be the value of $y_i^{(t+1)}$ if $i \in K$, i.e.,

$$\widetilde{y}_i = \arg\max_{\beta \in \mathbb{R}} \left\{ \beta \langle a_i, \overline{x}^{(t)} \rangle - \phi_i^*(\beta) - \frac{(\beta - y_i^{(t)})^2}{2\sigma} \right\}.$$

Since by assumption ϕ_i is $(1/\gamma)$-smooth, its convex conjugate ϕ_i^* is γ-strongly convex (see, e.g., Hiriart-Urruty and Lemaréchal, 2001, Theorem 4.2.2). Thus the function being maxi-

mized above is $(1/\sigma+\gamma)$-strongly concave. Therefore,

$$-y_i^\star\langle a_i,\overline{x}^{(t)}\rangle+\phi_i^*(y_i^\star)+\frac{(y_i^\star-y_i^{(t)})^2}{2\sigma}\geq-\widetilde{y}_i\langle a_i,\overline{x}^{(t)}\rangle+\phi_i^*(\widetilde{y}_i)+\frac{(\widetilde{y}_i-y_i^{(t)})^2}{2\sigma}+\Big(\frac{1}{\sigma}+\gamma\Big)\frac{(\widetilde{y}_i-y_i^\star)^2}{2}.$$

Multiplying both sides of the above inequality by m/n and re-arrange terms, we have

$$\frac{m}{2\sigma n}(y_i^{(t)}-y_i^\star)^2\geq\Big(\frac{1}{\sigma}+\gamma\Big)\frac{m}{2n}(\widetilde{y}_i-y_i^\star)^2+\frac{m}{2\sigma n}(\widetilde{y}_i-y_i^{(t)})^2-\frac{m}{n}(\widetilde{y}_i-y_i^\star)\langle a_i,\overline{x}^{(t)}\rangle+\frac{m}{n}\big(\phi_i^*(\widetilde{y}_i)-\phi_i^*(y_i^\star)\big).\tag{45}$$

According to Algorithm 2, the set K of indices to be updated are chosen randomly. For every specific index i, the event $i\in K$ happens with probability m/n. If $i\in K$, then $y_i^{(t+1)}$ is updated to the value $\widetilde{y}_i$, which satisfies inequality (45). Otherwise, $y_i^{(t+1)}$ is assigned by its old value $y_i^{(t)}$. Let $\mathcal{F}_t$ be the sigma field generated by all random variables defined before round t, and taking expectation conditioned on $\mathcal{F}_t$, we have

$$\begin{aligned}\mathbb{E}[(y_i^{(t+1)}-y_i^\star)^2|\mathcal{F}_t]&=\frac{m(\widetilde{y}_i-y_i^\star)^2}{n}+\frac{(n-m)(y_i^{(t)}-y_i^\star)^2}{n},\\ \mathbb{E}[(y_i^{(t+1)}-y_i^{(t)})^2|\mathcal{F}_t]&=\frac{m(\widetilde{y}_i-y_i^{(t)})^2}{n},\\ \mathbb{E}[y_i^{(t+1)}|\mathcal{F}_t]&=\frac{m\widetilde{y}_i}{n}+\frac{(n-m)y_i^{(t)}}{n}\\ \mathbb{E}[\phi_i^*(y_i^{(t+1)})|\mathcal{F}_t]&=\frac{m}{n}\phi_i^*(\widetilde{y}_i)+\frac{n-m}{n}\phi_i^*(y_i^{(t)}).\end{aligned}$$

As a result, we can represent $(\widetilde{y}_i-y_i^\star)^2$, $(\widetilde{y}_i-y_i^{(t)})^2$, $\widetilde{y}_i$ and $\phi_i^*(\widetilde{y}_i)$ in terms of the conditional expectations on $(y_i^{(t+1)}-y_i^\star)^2$, $(y_i^{(t+1)}-y_i^{(t)})^2$, $y_i^{(t+1)}$ and $\phi_i^*(y_i^{(t+1)})$, respectively. Plugging these representations into inequality (45) and re-arranging terms, we obtain

$$\begin{aligned}\left(\frac{1}{2\sigma}+\frac{(n-m)\gamma}{2n}\right)(y_i^{(t)}-y_i^\star)^2\geq\left(\frac{1}{2\sigma}+\frac{\gamma}{2}\right)\mathbb{E}[(y_i^{(t+1)}-y_i^\star)^2|\mathcal{F}_t]+\frac{1}{2\sigma}\mathbb{E}[(y_i^{(t+1)}-y_i^{(t)})^2|\mathcal{F}_t]\\-\Big(\frac{m}{n}(y_i^{(t)}-y_i^\star)+\mathbb{E}[y_i^{(t+1)}-y_i^{(t)}|\mathcal{F}_t]\Big)\langle a_i,\overline{x}^{(t)}\rangle\\+\mathbb{E}[\phi_i^*(y_i^{(t+1)})|\mathcal{F}_t]-\phi_i^*(y_i^{(t)})+\frac{m}{n}\big(\phi_i^*(y_i^{(t)})-\phi_i^*(y_i^\star)\big).\end{aligned}\tag{46}$$

Then summing over all indices $i=1,2,\ldots,n$ and dividing both sides of the resulting inequality by m, we have

$$\begin{aligned}\left(\frac{1}{2\sigma}+\frac{(n-m)\gamma}{2n}\right)\frac{\|y^{(t)}-y^\star\|_2^2}{m}\geq\left(\frac{1}{2\sigma}+\frac{\gamma}{2}\right)\frac{\mathbb{E}[\|y^{(t+1)}-y^\star\|_2^2|\mathcal{F}_t]}{m}+\frac{1}{2\sigma}\frac{\mathbb{E}[\|y^{(t+1)}-y^{(t)}\|_2^2|\mathcal{F}_t]}{m}\\+\mathbb{E}\Big[\frac{1}{m}\sum_{k\in K}\big(\phi_k^*(y_k^{(t+1)})-\phi_k^*(y_k^{(t)})\big)\Big|\mathcal{F}_t\Big]+\frac{1}{n}\sum_{i=1}^n\big(\phi_i^*(y_i^{(t)})-\phi_i^*(y_i^\star)\big)\\-\mathbb{E}\Big[\Big\langle u^{(t)}-u^\star+\frac{n}{m}(u^{(t+1)}-u^{(t)}),\,\overline{x}^{(t)}\Big\rangle\Big|\mathcal{F}_t\Big],\end{aligned}\tag{47}$$

where we used the shorthand notations (appeared in Algorithm 2)

$$u^{(t)} = \frac{1}{n}\sum_{i=1}^{n} y_i^{(t)} a_i, \qquad u^{(t+1)} = \frac{1}{n}\sum_{i=1}^{n} y_i^{(t+1)} a_i, \qquad \text{and} \qquad u^\star = \frac{1}{n}\sum_{i=1}^{n} y_i^\star a_i. \tag{48}$$

Since only the dual coordinates with indices in K are updated, we have

$$\frac{n}{m}(u^{(t+1)} - u^{(t)}) = \frac{1}{m}\sum_{i=1}^{n}(y_i^{(t+1)} - y_i^{(t)})a_i = \frac{1}{m}\sum_{k\in K}(y_k^{(t+1)} - y_k^{(t)})a_k.$$

We also derive an inequality characterizing the relation between $x^{(t+1)}$ and $x^{(t)}$. Since the function being minimized on the right-hand side of (10) has strong convexity parameter $1/\tau + \lambda$ and $x^{(t+1)}$ is the minimizer, we have

$$\begin{aligned} & g(x^\star) + \left\langle u^{(t)} + \frac{n}{m}(u^{(t+1)} - u^{(t)}),\ x^\star \right\rangle + \frac{\|x^{(t)} - x^\star\|_2^2}{2\tau} \\ \geq\ & g(x^{(t+1)}) + \left\langle u^{(t)} + \frac{n}{m}(u^{(t+1)} - u^{(t)}),\ x^{(t+1)} \right\rangle + \left(\frac{1}{2\tau} + \frac{\lambda}{2}\right)\|x^{(t+1)} - x^\star\|_2^2 \\ & + \frac{\|x^{(t+1)} - x^{(t)}\|_2^2}{2\tau}. \end{aligned} \tag{49}$$

Rearranging terms and taking expectation conditioned on $\mathcal{F}_t$, we have

$$\begin{aligned} \frac{\|x^{(t)} - x^\star\|_2^2}{2\tau} \geq \left(\frac{1}{2\tau} + \frac{\lambda}{2}\right)\mathbb{E}[\|x^{(t+1)} - x^\star\|_2^2|\mathcal{F}_t] + \frac{\mathbb{E}[\|x^{(t+1)} - x^{(t)}\|_2^2|\mathcal{F}_t]}{2\tau} \\ + \mathbb{E}\left[g(x^{(t+1)}) - g(x^\star)|\mathcal{F}_t\right] \\ + \mathbb{E}\Big[\Big\langle u^{(t)} + \frac{n}{m}(u^{(t+1)} - u^{(t)}),\ x^{(t+1)} - x^\star\Big\rangle\Big|\mathcal{F}_t\Big]. \end{aligned} \tag{50}$$

In addition, we consider a particular combination of the saddle-point function values at different points. By the definition of $f(x, y)$ in (4) and the notations in (48), we have

$$\begin{aligned} & f(x^{(t+1)}, y^\star) - f(x^\star, y^\star) + \frac{n}{m}\left(f(x^\star, y^\star) - f(x^\star, y^{(t+1)})\right) - \frac{n-m}{m}\left(f(x^\star, y^\star) - f(x^\star, y^{(t)})\right) \\ =\ & f(x^{(t+1)}, y^\star) - f(x^\star, y^{(t)}) + \frac{n}{m}\left(f(x^\star, y^{(t)}) - f(x^\star, y^{(t+1)})\right) \\ =\ & \langle u^\star, x^{(t+1)}\rangle - \frac{1}{n}\sum_{i=1}^{n}\phi_i^*(y_i^\star) + g(x^{(t+1)}) - \langle u^{(t)}, x^\star\rangle + \frac{1}{n}\sum_{i=1}^{n}\phi_i^*(y_i^{(t)}) - g(x^\star) \\ & + \frac{n}{m}\left(\langle u^{(t)}, x^\star\rangle - \frac{1}{n}\sum_{i=1}^{n}\phi_i^*(y_i^{(t)}) + g(x^\star) - \langle u^{(t+1)}, x^\star\rangle + \frac{1}{n}\sum_{i=1}^{n}\phi_i^*(y_i^{(t+1)}) - g(x^\star)\right) \\ =\ & \frac{1}{n}\sum_{i=1}^{n}\left(\phi_i^*(y_i^{(t)}) - \phi_i^*(y_i^\star)\right) + \frac{1}{m}\sum_{k\in K}\left(\phi_i^*(y_k^{(t+1)}) - \phi_i^*(y_k^{(t)})\right) + g(x^{(t+1)}) - g(x^\star) \\ & + \langle u^\star, x^{(t+1)}\rangle - \langle u^{(t)}, x^\star\rangle + \frac{n}{m}\langle u^{(t)} - u^{(t+1)}, x^\star\rangle. \end{aligned} \tag{51}$$

Next we add both sides of the inequalities (47) and (50) together, and then subtract equality (51) after taking expectation with respect to $\mathcal{F}_t$. This leads to the following inequality:

$$\begin{aligned}
&\frac{\|x^{(t)}-x^\star\|_2^2}{2\tau}+\left(\frac{1}{2\sigma}+\frac{(n-m)\gamma}{2n}\right)\frac{\|y^{(t)}-y^\star\|_2^2}{m}+\frac{n-m}{m}\left(f(x^\star,y^\star)-f(x^\star,y^{(t)})\right)\\
\geq&\left(\frac{1}{2\tau}+\frac{\lambda}{2}\right)\mathbb{E}[\|x^{(t+1)}-x^\star\|_2^2|\mathcal{F}_t]+\left(\frac{1}{2\sigma}+\frac{\gamma}{2}\right)\frac{\mathbb{E}[\|y^{(t+1)}-y^\star\|_2^2|\mathcal{F}_t]}{m}+\frac{\mathbb{E}[\|x^{(t+1)}-x^{(t)}\|_2^2|\mathcal{F}_t]}{2\tau}\\
&+\frac{\mathbb{E}[\|y^{(t+1)}-y^{(t)}\|_2^2|\mathcal{F}_t]}{2\sigma m}+\mathbb{E}\left[f(x^{(t+1)},y^\star)-f(x^\star,y^\star)+\frac{n}{m}\left(f(x^\star,y^\star)-f(x^\star,y^{(t+1)})\right)\Big|\mathcal{F}_t\right]\\
&+\mathbb{E}\left[\left\langle u^{(t)}-u^\star+\frac{n}{m}(u^{(t+1)}-u^{(t)}),\ x^{(t+1)}-\overline{x}^{(t)}\right\rangle\Big|\mathcal{F}_t\right]. \qquad (52)
\end{aligned}$$

We need to lower bound the last term on the right-hand-side of the above inequality. To this end, we have

$$\begin{aligned}
&\left\langle u^{(t)}-u^\star+\frac{n}{m}(u^{(t+1)}-u^{(t)}),\ x^{(t+1)}-\overline{x}^{(t)}\right\rangle\\
=&\left(\frac{y^{(t)}-y^\star}{n}+\frac{y^{(t+1)}-y^{(t)}}{m}\right)^T A(x^{(t+1)}-x^{(t)}-\theta(x^{(t)}-x^{(t-1)}))\\
=&\frac{(y^{(t+1)}-y^\star)^TA(x^{(t+1)}-x^{(t)})}{n}-\frac{\theta(y^{(t)}-y^\star)^TA(x^{(t)}-x^{(t-1)})}{n}\\
&+\frac{n-m}{mn}(y^{(t+1)}-y^{(t)})^TA(x^{(t+1)}-x^{(t)})-\frac{\theta}{m}(y^{(t+1)}-y^{(t)})^TA(x^{(t)}-x^{(t-1)}). \qquad (53)
\end{aligned}$$

Recall that $\|a_k\|_2\leq R$ and, according to (12), $1/\tau=4\sigma R^2$. Therefore,

$$\begin{aligned}
|(y^{(t+1)}-y^{(t)})^TA(x^{(t+1)}-x^{(t)})|&\leq\frac{\|x^{(t+1)}-x^{(t)}\|_2^2}{4\tau/m}+\frac{\|(y^{(t+1)}-y^{(t)})^TA\|_2^2}{m/\tau}\\
&\leq\frac{\|x^{(t+1)}-x^{(t)}\|_2^2}{4\tau/m}+\frac{(\sum_{k\in K}|y_k^{(t+1)}-y_k^{(t)}|\cdot\|a_k\|_2)^2}{4m\sigma R^2}\\
&\leq\frac{m\|x^{(t+1)}-x^{(t)}\|_2^2}{4\tau}+\frac{\|y^{(t+1)}-y^{(t)}\|_2^2}{4\sigma},
\end{aligned}$$

Similarly, we have

$$|(y^{(t+1)}-y^{(t)})^TA(x^{(t)}-x^{(t-1)})|\leq\frac{m\|x^{(t)}-x^{(t-1)}\|_2^2}{4\tau}+\frac{\|y^{(t+1)}-y^{(t)}\|_2^2}{4\sigma}.$$

The above upper bounds on the absolute values imply

$$\begin{aligned}
(y^{(t+1)}-y^{(t)})^TA(x^{(t+1)}-x^{(t)})&\geq-\frac{m\|x^{(t+1)}-x^{(t)}\|_2^2}{4\tau}-\frac{\|y^{(t+1)}-y^{(t)}\|_2^2}{4\sigma},\\
(y^{(t+1)}-y^{(t)})^TA(x^{(t)}-x^{(t-1)})&\geq-\frac{m\|x^{(t)}-x^{(t-1)}\|_2^2}{4\tau}-\frac{\|y^{(t+1)}-y^{(t)}\|_2^2}{4\sigma}.
\end{aligned}$$

Combining the above two inequalities with (52) and (53), we obtain

$$\begin{aligned}
&\frac{\|x^{(t)}-x^\star\|_2^2}{2\tau}+\left(\frac{1}{2\sigma}+\frac{(n-m)\gamma}{2n}\right)\frac{\|y^{(t)}-y^\star\|_2^2}{m}\\
&+\theta\big(f(x^{(t)},y^\star)-f(x^\star,y^\star)\big)+\frac{n-m}{m}\Big(f(x^\star,y^\star)-f(x^\star,y^{(t)})\Big)\\
&+\theta\frac{\|x^{(t)}-x^{(t-1)}\|_2^2}{4\tau}+\theta\frac{(y^{(t)}-y^\star)^TA(x^{(t)}-x^{(t-1)})}{n}\\
\geq\ &\left(\frac{1}{2\tau}+\frac{\lambda}{2}\right)\mathbb{E}[\|x^{(t+1)}-x^\star\|_2^2|\mathcal{F}_t]+\left(\frac{1}{2\sigma}+\frac{\gamma}{2}\right)\frac{\mathbb{E}[\|y^{(t+1)}-y^\star\|_2^2|\mathcal{F}_t]}{m}\\
&+\mathbb{E}\left[f(x^{(t+1)},y^\star)-f(x^\star,y^\star)+\frac{n}{m}\Big(f(x^\star,y^\star)-f(x^\star,y^{(t+1)})\Big)\,\middle|\,\mathcal{F}_t\right]\\
&+\frac{\mathbb{E}[\|x^{(t+1)}-x^{(t)}\|_2^2|\mathcal{F}_t]}{4\tau}+\frac{\mathbb{E}[(y^{(t+1)}-y^\star)^TA(x^{(t+1)}-x^{(t)})|\mathcal{F}_t]}{n}. \qquad (54)
\end{aligned}$$

Note that we have added the nonnegative term $\theta\big(f(x^{(t)},y^\star)-f(x^\star,y^\star)\big)$ to the left-hand side in (54) to ensure that each term on one side of the inequality has a corresponding term on the other side.

If the parameters τ, σ, and θ are chosen as in (12), that is,

$$\tau=\frac{1}{2R}\sqrt{\frac{m\gamma}{n\lambda}},\quad \sigma=\frac{1}{2R}\sqrt{\frac{n\lambda}{m\gamma}},\quad \text{and}\quad \theta=1-\frac{1}{(n/m)+2R\sqrt{(n/m)/(\lambda\gamma)}},$$

Then the ratios between the coefficients of the corresponding terms on both sides of the inequality (54) are either equal to θ or bounded by θ. More specifically,

$$\begin{aligned}
\frac{n-m}{m}\Big/\frac{n}{m}=1-\frac{m}{n}\ &\leq\ \theta,\\
\frac{1}{2\tau}\Big/\left(\frac{1}{2\tau}+\frac{\lambda}{2}\right)=1-\frac{1}{1+2R\sqrt{(n/m)/(\lambda\gamma)}}\ &\leq\ \theta,\\
\left(\frac{1}{2\sigma}+\frac{(n-m)\gamma}{2n}\right)\Big/\left(\frac{1}{2\sigma}+\frac{\gamma}{2}\right)=1-\frac{1}{n/m+2R\sqrt{(n/m)/(\lambda\gamma)}}\ &=\ \theta.
\end{aligned}$$

Therefore, if we define the following sequence,

$$\begin{aligned}
\widetilde{\Delta}^{(t)}=\ &\left(\frac{1}{2\tau}+\frac{\lambda}{2}\right)\|x^{(t)}-x^\star\|_2^2+\left(\frac{1}{2\sigma}+\frac{\gamma}{2}\right)\frac{\|y^{(t)}-y^\star\|_2^2}{m}\\
&+f(x^{(t)},y^\star)-f(x^\star,y^\star)+\frac{n}{m}\Big(f(x^\star,y^\star)-f(x^\star,y^{(t)})\Big)\\
&+\frac{\|x^{(t)}-x^{(t-1)}\|_2^2}{4\tau}+\frac{(y^{(t)}-y^\star)^TA(x^{(t)}-x^{(t-1)})}{n},
\end{aligned}$$

then inequality (54) implies $\mathbb{E}\big[\widetilde{\Delta}^{(t+1)}|\mathcal{F}_t\big]\ \leq\ \theta\,\widetilde{\Delta}^{(t)}$. Apply this relation recursively and taking expectation with respect to all random variables up to time t, we have

$$\mathbb{E}\big[\widetilde{\Delta}^{(t)}\big]\ \leq\ \theta^t\,\widetilde{\Delta}^{(0)}. \qquad (55)$$

Comparing the definition of $\Delta^{(t)}$ in (11), we have

$$\widetilde{\Delta}^{(t)} = \Delta^{(t)} + \frac{\|y^{(t)} - y^\star\|_2^2}{4\sigma m} + \frac{\|x^{(t)} - x^{(t-1)}\|_2^2}{4\tau} + \frac{(y^{(t)} - y^\star)^T A(x^{(t)} - x^{(t-1)})}{n}. \tag{56}$$

For $t = 0$, by letting $x^{(-1)} = x^{(0)}$, the last two terms in (56) for $\widetilde{\Delta}^{(0)}$ disappears. Moreover, we can show that the sum of the last three terms in (56) are nonnegative, and therefore we can replace $\widetilde{\Delta}^{(t)}$ with $\Delta^{(t)}$ on the left-hand side of (55). To see this, we bound the absolute value of the last term:

$$\begin{aligned}
\frac{\left|(y^{(t)} - y^\star)^T A(x^{(t)} - x^{(t-1)})\right|}{n} &\leq \frac{\|x^{(t)} - x^{(t-1)}\|_2^2}{4\tau} + \frac{\|A\|_2^2 \, \|y^{(t)} - y^\star\|_2^2}{n^2/\tau} \\
&\leq \frac{\|x^{(t)} - x^{(t-1)}\|_2^2}{4\tau} + \frac{nR^2\|y^{(t)} - y^\star\|_2^2}{n^2/\tau} \\
&= \frac{\|x^{(t)} - x^{(t-1)}\|_2^2}{4\tau} + \frac{\|y^{(t)} - y^\star\|_2^2}{4n\sigma} \\
&\leq \frac{\|x^{(t)} - x^{(t-1)}\|_2^2}{4\tau} + \frac{\|y^{(t)} - y^\star\|_2^2}{4m\sigma},
\end{aligned}$$

where in the second inequality we used $\|A\|_2^2 \leq \|A\|_F^2 \leq nR^2$, in the equality we used $\tau\sigma = 1/(4R^2)$, and in the last inequality we used $m \leq n$. The above upper bound on absolute value implies

$$\frac{(y^{(t)} - y^\star)^T A(x^{(t)} - x^{(t-1)})}{n} \geq -\frac{\|x^{(t)} - x^{(t-1)}\|_2^2}{4\tau} - \frac{\|y^{(t)} - y^\star\|_2^2}{4m\sigma}.$$

To summarize, we have proved

$$\mathbb{E}\left[\Delta^{(t)}\right] \leq \theta^t \left(\Delta^{(0)} + \frac{\|y^{(0)} - y^\star\|_2^2}{4m\sigma}\right),$$

which is the desired result.

Appendix B. Proof of Lemma 3

We can write $P(x) = F(x) + g(x)$ where

$$F(x) = \frac{1}{n}\sum_{i=1}^n \phi_i(a_i^T x) = \max_{y \in \mathbb{R}^n}\left\{\frac{1}{n} y^T A x - \frac{1}{n}\sum_{i=1}^n \phi_i^*(y_i)\right\}.$$

Assumption A implies that $F(x)$ is smooth and $\nabla F(x)$ is Lipschitz continuous with constant $\|A\|_2^2/(n\gamma)$. We can bound the spectral norm with the Frobenius norm, i.e., $\|A\|_2^2 \leq \|A\|_F^2 \leq nR^2$, which results in $\|A\|_2^2/(n\gamma) \leq nR^2/(n\gamma) = R^2/\gamma$. By definition of the saddle point,

the gradient of F at $x^\star$ is $\nabla F(x^\star) = (1/n)A^T y^\star$. Therefore, we have

$$\begin{aligned}
F(x) &\leq F(x^\star) + \langle \nabla F(x^\star), x - x^\star \rangle + \frac{R^2}{2\gamma}\|x - x^\star\|_2^2 \\
&= \max_{y \in \mathbb{R}^n} \left\{ \frac{1}{n} y^T A x^\star - \frac{1}{n}\sum_{i=1}^n \phi_i^*(y_i) \right\} + \frac{1}{n}(y^\star)^T A(x - x^\star) + \frac{R^2}{2\gamma}\|x - x^\star\|_2^2 \\
&= \left\{ \frac{1}{n}(y^\star)^T A x^\star - \frac{1}{n}\sum_{i=1}^n \phi_i^*(y_i^\star) \right\} + \frac{1}{n}(y^\star)^T A(x - x^\star) + \frac{R^2}{2\gamma}\|x - x^\star\|_2^2 \\
&= \frac{1}{n}(y^\star)^T A x - \frac{1}{n}\sum_{i=1}^n \phi_i^*(y_i^\star) + \frac{R^2}{2\gamma}\|x - x^\star\|_2^2.
\end{aligned}$$

Combining the above inequality with $P(x) = F(x) + g(x)$, we have

$$P(x) \leq \frac{1}{n}(y^\star)^T A x - \frac{1}{n}\sum_{i=1}^n \phi_i^*(y_i^\star) + \frac{R^2}{2\gamma}\|x - x^\star\|_2^2 + g(x) = f(x, y^\star) + \frac{R^2}{2\gamma}\|x - x^\star\|_2^2,$$

which is the first desired inequality.

Similarly, the second inequality can be shown by first writing

$$D(y) = -\frac{1}{n}\sum_{i=1}^n \phi_i^*(y_i) - G^*(y),$$

where

$$G^*(y) = g^*\left(-\frac{1}{n}A^T y\right) = \max_{x \in \mathbb{R}^d} \left\{ -\frac{1}{n} x^T A^T y - g(x) \right\}.$$

In this case, $\nabla G^*(y)$ is Lipschitz continuous with constant

$$\|A\|_2^2/(n^2\lambda) \leq nR^2/(n^2\lambda) = R^2/(n\lambda).$$

Again by definition of the saddle-point, we have $\nabla G^*(y^\star) = -(1/n)Ax^\star$. Therefore,

$$\begin{aligned}
G^*(y) &\leq G^*(y^\star) + \langle \nabla G^*(y^\star), y - y^\star \rangle + \frac{R^2}{2n\lambda}\|y - y^\star\|_2^2 \\
&= \max_{x \in \mathbb{R}^d} \left\{ -\frac{1}{n} x^T A^T y^\star - g(x) \right\} - \frac{1}{n}(y - y^\star)^T A x^\star + \frac{R^2}{2n\lambda}\|y - y^\star\|_2^2 \\
&= \left\{ -\frac{1}{n}(x^\star)^T A^T y^\star - g(x^\star) \right\} - \frac{1}{n}(y - y^\star)^T A x^\star + \frac{R^2}{2n\lambda}\|y - y^\star\|_2^2 \\
&= -\frac{1}{n} y^T A x^\star - g(x^\star) + \frac{R^2}{2n\lambda}\|y - y^\star\|_2^2.
\end{aligned}$$

Recalling that $D(y) = -\frac{1}{n}\sum_{i=1}^n \phi_i^*(y_i) - G^*(y)$, we conclude with

$$D(y) \geq -\frac{1}{n}\phi_i^*(y_i) + \frac{1}{n} y^T A x^\star + g(x^\star) - \frac{R^2}{2n\lambda}\|y - y^\star\|_2^2 = f(x^\star, y) - \frac{R^2}{2n\lambda}\|y - y^\star\|_2^2.$$

This finishes the proof.

Appendix C. Proof of Theorem 6

The proof of Theorem 6 follows similar steps for proving Theorem 1. We start by establishing relation between $(y^{(t)}, y^{(t+1)})$ and between $(x^{(t)}, x^{(t+1)})$. Suppose that the quantity $\widetilde{y}_i$ minimizes the function $\phi_i^*(\beta) - \beta\langle a_i, \overline{x}^{(t)}\rangle + \frac{p_i n}{2\sigma}(\beta - y_i^{(t)})^2$. Also notice that $\phi_i^*(\beta) - \beta\langle a_i, x^*\rangle$ is a γ-strongly convex function minimized by y_i^*, which implies

$$\phi_i^*(\widetilde{y}_i) - \widetilde{y}_i\langle a_i, x^*\rangle \geq \phi_i^*(y_i^*) - y_i^*\langle a_i, x^*\rangle + \frac{\gamma}{2}(\widetilde{y}_i - y_i^*)^2. \tag{57}$$

Then, following the same argument for establishing inequality (45) and plugging in inequality (57), we obtain

$$\frac{p_i n}{2\sigma}(y_i^{(t)} - y_i^\star)^2 \geq \left(\frac{p_i n}{2\sigma} + \gamma\right)(\widetilde{y}_i - y_i^\star)^2 + \frac{p_i n(\widetilde{y}_i - y_i^{(t)})^2}{2\sigma} + \langle a_i, x^\star - \overline{x}^{(t)}\rangle(\widetilde{y}_i - y_i^\star). \tag{58}$$

Note that $i = k$ with probability p_i. Therefore, we have

$$\begin{aligned}
(\widetilde{y}_i - y_i^\star)^2 &= \frac{1}{p_i}\mathbb{E}[(y_i^{(t+1)} - y_i^\star)^2|\mathcal{F}_t] - \frac{1-p_i}{p_i}(y_i^{(t)} - y_i^\star)^2,\\
(\widetilde{y}_i - y_i^{(t)})^2 &= \frac{1}{p_i}\mathbb{E}[(y_i^{(t+1)} - y_i^{(t)})^2|\mathcal{F}_t],\\
\widetilde{y}_i &= \frac{1}{p_i}\mathbb{E}[y_i^{(t+1)}|\mathcal{F}_t] - \frac{1-p_i}{p_i}y_i^{(t)},
\end{aligned}$$

where $\mathcal{F}_t$ represents the sigma field generated by all random variables defined before iteration t. Substituting the above equations into inequality (58), and averaging over $i = 1, 2, \ldots, n$, we have

$$\begin{aligned}
&\sum_{i=1}^n \left(\frac{1}{2\sigma} + \frac{(1-p_i)\gamma}{p_i n}\right)(y_i^{(t)} - y_i^\star)^2\\
\geq\ &\sum_{i=1}^n \left(\frac{1}{2\sigma} + \frac{\gamma}{p_i n}\right)\mathbb{E}[(y_i^{(t+1)} - y_i^\star)^2|\mathcal{F}_t] + \frac{\mathbb{E}[(y_k^{(t+1)} - y_k^{(t)})^2|\mathcal{F}_t]}{2\sigma}\\
&+ \mathbb{E}\Big[\Big\langle (u^{(t)} - u^\star) + \frac{1}{p_k}(u^{(t+1)} - u^{(t)}),\ x^\star - \overline{x}^{(t)}\Big\rangle\Big|\mathcal{F}_t\Big],
\end{aligned} \tag{59}$$

where $u^\star = \frac{1}{n}\sum_{i=1}^n y_i^\star a_i$ and $u^{(t)} = \frac{1}{n}\sum_{i=1}^n y_i^{(t)} a_i$ have the same definition as in the proof of Theorem 1.

For the relation between $x^{(t)}$ and $x^{(t+1)}$, we first notice that $\langle u^*, x\rangle + g(x)$ is a λ-strongly convex function minimized by x^*, which implies

$$\langle u^*, x^{(t+1)}\rangle + g(x^{(t+1)}) \geq \langle u^*, x^*\rangle + g(x^*) + \frac{\lambda}{2}(x^{(t+1)} - x^*)^2. \tag{60}$$

Following the same argument for establishing inequality (49) and plugging in inequality (60), we obtain

$$\begin{aligned}
\frac{\|x^{(t)} - x^\star\|_2^2}{2\tau} \geq \left(\frac{1}{2\tau} + \lambda\right)\|x^{(t+1)} - x^\star\|_2^2 + \frac{\|x^{(t+1)} - x^{(t)}\|_2^2}{2\tau}\\
+ \Big\langle (u^{(t)} - u^\star) + \frac{1}{p_k}(u^{(t+1)} - u^{(t)}),\ x^{(t+1)} - x^\star\Big\rangle.
\end{aligned} \tag{61}$$

Taking expectation over both sides of inequality (61) and adding it to inequality (59) yields

$$\begin{aligned}\frac{\|x^{(t)}-x^\star\|_2^2}{2\tau}+\sum_{i=1}^n\left(\frac{1}{2\sigma}+\frac{(1-p_i)\gamma}{p_in}\right)(y_i^{(t)}-y_i^\star)^2 &\geq \left(\frac{1}{2\tau}+\lambda\right)\mathbb{E}[\|x^{(t+1)}-x^\star\|_2^2|\mathcal{F}_t]\\ &+\sum_{i=1}^n\left(\frac{1}{2\sigma}+\frac{\gamma}{p_in}\right)\mathbb{E}[(y_i^{(t+1)}-y_i^\star)^2|\mathcal{F}_t]+\frac{\|x^{(t+1)}-x^{(t)}\|_2^2}{2\tau}+\frac{\mathbb{E}[(y_k^{(t+1)}-y_k^{(t)})^2|\mathcal{F}_t]}{2\sigma}\\ &+\mathbb{E}\Big[\underbrace{\Big(\frac{(y^{(t)}-y^\star)^TA}{n}+\frac{(y_k^{(t+1)}-y_k^{(t)})a_k^T}{p_kn}\Big)((x^{(t+1)}-x^{(t)})-\theta(x^{(t)}-x^{(t-1)}))}_{v}\Big|\mathcal{F}_t\Big],\end{aligned} \tag{62}$$

where the matrix A is a n-by-d matrix, whose i-th row is equal to the vector a_i^T.

Next, we lower bound the last term on the right-hand side of inequality (62). Indeed, it can be expanded as

$$\begin{aligned}v=&\frac{(y^{(t+1)}-y^\star)^TA(x^{(t+1)}-x^{(t)})}{n}-\frac{\theta(y^{(t)}-y^\star)^TA(x^{(t)}-x^{(t-1)})}{n}\\ &+\frac{1-p_k}{p_kn}(y_k^{(t+1)}-y_k^{(t)})a_k^T(x^{(t+1)}-x^{(t)})-\frac{\theta}{p_kn}(y_k^{(t+1)}-y_k^{(t)})a_k^T(x^{(t)}-x^{(t-1)}).\end{aligned} \tag{63}$$

Note that the probability p_k given in (20) satisfies

$$p_k=\frac{1-\alpha}{n}+\frac{\alpha\|a_k\|_2}{\sum_{i=1}^n\|a_i\|_2}\geq\frac{(1-\alpha)\|a_k\|_2}{nR}+\frac{\alpha\|a_k\|_2}{\sum_{i=1}^n\|a_i\|_2}=\frac{\|a_k\|_2}{nR_\alpha},\qquad k=1,\ldots,n.$$

Since the parameters τ and σ satisfies $\sigma\tau R_\alpha^2=1/4$, we have $p_k^2n^2/\tau\geq 4\sigma\|a_k\|_2^2$ and consequently

$$\begin{aligned}\frac{|(y_k^{(t+1)}-y_k^{(t)})a_k^T(x^{(t+1)}-x^{(t)})|}{p_kn}&\leq\frac{\|x^{(t+1)}-x^{(t)}\|_2^2}{4\tau}+\frac{\|(y_k^{(t+1)}-y_k^{(t)})a_k\|_2^2}{p_k^2n^2/\tau}\\ &\leq\frac{\|x^{(t+1)}-x^{(t)}\|_2^2}{4\tau}+\frac{(y_k^{(t+1)}-y_k^{(t)})^2}{4\sigma}.\end{aligned}$$

Similarly, we have

$$\frac{|(y_k^{(t+1)}-y_k^{(t)})a_k^T(x^{(t)}-x^{(t-1)})|}{p_kn}\leq\frac{\|x^{(t)}-x^{(t-1)}\|_2^2}{4\tau}+\frac{(y_k^{(t+1)}-y_k^{(t)})^2}{4\sigma}.$$

Combining the above two inequalities with lower bounds (62) and (63), we obtain

$$\begin{aligned}\frac{\|x^{(t)}-x^\star\|_2^2}{2\tau}+\sum_{i=1}^n\left(\frac{1}{2\sigma}+\frac{(1-p_i)\gamma}{p_in}\right)(y_i^{(t)}-y_i^\star)^2 &\geq \left(\frac{1}{2\tau}+\lambda\right)\mathbb{E}[\|x^{(t+1)}-x^\star\|_2^2|\mathcal{F}_t]\\ &+\sum_{i=1}^n\left(\frac{1}{2\sigma}+\frac{\gamma}{p_in}\right)\mathbb{E}[(y_i^{(t+1)}-y_i^\star)^2|\mathcal{F}_t]+\frac{\mathbb{E}[\|x^{(t+1)}-x^{(t)}\|_2^2|\mathcal{F}_t]-\theta\|x^{(t)}-x^{(t-1)}\|_2^2}{4\tau}\\ &+\frac{\mathbb{E}[(y^{(t+1)}-y^\star)^TA(x^{(t+1)}-x^{(t)})|\mathcal{F}_t]-\theta(y^{(t)}-y^\star)A(x^{(t)}-x^{(t-1)})}{n}.\end{aligned} \tag{64}$$

Recall that the parameters τ, σ, and θ are chosen to be

$$\tau = \frac{1}{2R_\alpha}\sqrt{\frac{\gamma}{n\lambda}}, \quad \sigma = \frac{1}{2R_\alpha}\sqrt{\frac{n\lambda}{\gamma}}, \quad \text{and} \quad \theta = 1 - \left(\frac{n}{1-\alpha} + R_\alpha\sqrt{\frac{n}{\lambda\gamma}}\right)^{-1}.$$

Plugging in these assignments and using the fact that $p_i \geq \frac{1-\alpha}{n}$, we find that

$$\frac{1/(2\tau)}{1/(2\tau)+\lambda} = 1 - \left(1 + \frac{1}{2\tau\lambda}\right)^{-1} = 1 - \left(1 + R_\alpha\sqrt{\frac{n}{\lambda\gamma}}\right)^{-1} \leq \theta,$$

and for $i = 1, 2, \ldots, n$,

$$\frac{1/(2\sigma) + (1-p_i)\gamma/(p_i n)}{1/(2\sigma)+\gamma/(p_i n)} = 1 - \left(\frac{1}{p_i} + \frac{n}{2\sigma}\right)^{-1} \leq 1 - \left(\frac{n}{1-\alpha} + \frac{n}{2\sigma\gamma}\right)^{-1} = \theta.$$

Therefore, if we define a sequence $\Delta^{(t)}$ such that

$$\begin{aligned}\Delta^{(t)} = &\left(\frac{1}{2\tau}+\lambda\right)\mathbb{E}[\|x^{(t)}-x^\star\|_2^2] + \sum_{i=1}^n\left(\frac{1}{2\sigma}+\frac{\gamma}{p_i n}\right)\mathbb{E}[(y_i^{(t)}-y_i^\star)^2] \\ &+ \frac{\mathbb{E}[\|x^{(t)}-x^{(t-1)}\|_2^2]}{4\tau} + \frac{\mathbb{E}[(y^{(t)}-y^\star)^T A(x^{(t)}-x^{(t-1)})]}{n},\end{aligned}$$

then inequality (64) implies the recursive relation $\Delta^{(t+1)} \leq \theta\cdot\Delta^{(t)}$, which implies

$$\begin{aligned}&\left(\frac{1}{2\tau}+\lambda\right)\mathbb{E}[\|x^{(t)}-x^\star\|_2^2] + \left(\frac{1}{2\sigma}+\frac{\gamma}{n}\right)\mathbb{E}[\|y^{(t)}-y^\star\|_2^2] \\ &+ \frac{\mathbb{E}[\|x^{(t)}-x^{(t-1)}\|_2^2]}{4\tau} + \frac{\mathbb{E}[(y^{(t)}-y^\star)^T A(x^{(t)}-x^{(t-1)})]}{n} \;\leq\; \theta^t\Delta^{(0)}, \qquad (65)\end{aligned}$$

where

$$\begin{aligned}\Delta^{(0)} &= \left(\frac{1}{2\tau}+\lambda\right)\|x^{(0)}-x^\star\|_2^2 + \sum_{i=1}^n\left(\frac{1}{2\sigma}+\frac{\gamma}{p_i n}\right)(y_i^{(0)}-y_i^\star)^2 \\ &\leq \left(\frac{1}{2\tau}+\lambda\right)\|x^{(0)}-x^\star\|_2^2 + \left(\frac{1}{2\sigma}+\frac{\gamma}{1-\alpha}\right)\|y^{(0)}-y^\star\|_2^2.\end{aligned}$$

To eliminate the last two terms on the left-hand side of inequality (65), we notice that

$$\begin{aligned}\frac{|(y^{(t)}-y^\star)^T A(x^{(t)}-x^{(t-1)})|}{n} &\leq \frac{\|x^{(t)}-x^{(t-1)}\|_2^2}{4\tau} + \frac{\|y^{(t)}-y^\star\|_2^2\,\|A\|_2^2}{n^2/\tau} \\ &\leq \frac{\|x^{(t)}-x^{(t-1)}\|_2^2}{4\tau} + \frac{\|y^{(t)}-y^\star\|_2^2\,\|A\|_F^2}{n^2/\tau} \\ &= \frac{\|x^{(t)}-x^{(t-1)}\|_2^2}{4\tau} + \frac{\|y^{(t)}-y^\star\|_2^2\,\sum_{i=1}^n\|a_i\|_2^2}{4\sigma(\sum_{i=1}^n\|a_i\|_2)^2} \\ &\leq \frac{\|x^{(t)}-x^{(t-1)}\|_2^2}{4\tau} + \frac{\|y^{(t)}-y^\star\|_2^2}{4\sigma},\end{aligned}$$

where in the equality we used $n^2/\tau = 4\sigma n^2 R_\alpha^2 \geq 4\sigma n^2\overline{R} = 4\sigma\left(\sum_{i=1}^n\|a_i\|_2\right)^2$. This implies

$$\frac{(y^{(t)}-y^\star)^T A(x^{(t)}-x^{(t-1)})}{n} \;\geq\; -\frac{\|x^{(t)}-x^{(t-1)}\|_2^2}{4\tau} - \frac{\|y^{(t)}-y^\star\|_2^2}{4\sigma}.$$

Substituting the above inequality into inequality (65) completes the proof.

References

Alekh Agarwal and Léon Bottou. A lower bound for the optimization of finite sums. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pages 78–86, Lille, France, 2015.

Zeyuan Allen-Zhu. Katyusha: The first direct acceleration of stochastic gradient methods. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 1200–1205, Montreal, Canada, June 2017.

Amir Beck and Marc Teboulle. A fast iterative shrinkage-threshold algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

Dimitri P. Bertsekas. Incremental proximal methods for large scale convex optimization. *Mathematical Programming, Ser. B*, 129:163–195, 2011.

Dimitri P. Bertsekas. Incremental gradient, subgradient, and proximal methods for convex optimization: a survey. In S. Sra, S. Nowozin, and S. J. Wright, editors, *Optimization for Machine Learning*, chapter 4. The MIT Press, 2012.

Doron Blatt, Alfred Hero, and Hillel Gauchman. A convergent incremental gradient method with a constant step size. *SIAM Journal on Optimization*, 18(1):29–51, 2007.

Léon Bottou. Large-scale machine learning with stochastic gradient descent. In Yves Lechevallier and Gilbert Saporta, editors, *Proceedings of the 19th International Conference on Computational Statistics (COMPSTAT'2010)*, pages 177–187, Paris, France, August 2010. Springer.

Léon Bottou and Olivier Bousquet. The tradeoffs of large scale learning. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 161–168. MIT Press, Cambridge, MA, 2008.

Olivier Bousquet and André Elisseeff. Stability and generalization. *Journal of Machine Learning Research*, 2:499–526, 2002.

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2010.

Antonin Chambolle and Thomas Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision*, 40 (1):120–145, 2011.

Kai-Wei Chang, Cho-Jui Hsieh, and Chih-Jen Lin. Coordinate descent method for large-scale l_2-loss linear support vector machines. *Journal of Machine Learning Research*, 9: 1369–1398, 2008.

Min-Te Chao. A general purpose unequal probability sampling plan. *Biometrika*, 69(3): 653–656, 1982.

Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems 27*, pages 1646–1654. 2014.

John Duchi and Yoram Singer. Efficient online and batch learning using forward backward splitting. *Journal of Machine Learning Research*, 10:2873–2898, 2009.

Rong-En Fan and Chih-Jen Lin. LIBSVM data: Classification, regression and multi-label. URL: http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets, 2011.

Roy Frostig, Rong Ge, Sham Kakade, and Aaron Sidford. Un-regularizing: approximate proximal point and faster stochastic algorithms for empirical risk minimization. In *Proceedings of The 32nd International Conference on Machine Learning (ICML)*, pages 2540–2548. 2015.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, New York, 2nd edition, 2009.

Jean-Baptiste Hiriart-Urruty and Claude Lemaréchal. *Fundamentals of Convex Analysis*. Springer, 2001.

Cho-Jui Hsieh, Kai-Wei Chang, Chih-Jen Lin, S. Sathiya Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear svm. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, pages 408–415, 2008.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems 26*, pages 315–323. 2013.

Guanghui Lan and Yi Zhou. An optimal randomized incremental gradient method. Technical report, Department of Industrial and System Engineering, University of Florida, July 2015.

John Langford, Lihong Li, and Tong Zhang. Sparse online learning via truncated gradient. *Journal of Machine Learning Research*, 10:777–801, 2009.

Nicolas Le Roux, Mark Schmidt, and Francis Bach. A stochastic gradient method with an exponential convergence rate for finite training sets. In *Advances in Neural Information Processing Systems 25*, pages 2672–2680. 2012.

Hongzhou Lin, Julien Mairal, and Zaid Harchaoui. A universal catalyst for first-order optimization. In *Advances in Neural Information Processing Systems 28*, pages 3384–3392. 2015a.

Qihang Lin, Zhaosong Lu, and Lin Xiao. An accelerated randomized proximal coordinate gradient method and its application to regularized empirical risk minimization. *SIAM Journal on Optimization*, 25(4):2244–2273, 2015b.

P. L. Lions and B. Mercier. Splitting algorithms for the sum of two nonlinear operators. *SIAM Journal on Numerical Analysis*, 16(6):964–979, December 1979.

Angelia Nedić and Dimitri P. Bertsekas. Incremental subgradient methods for nondifferentiable optimization. *SIAM Journal on Optimization*, 12(1):109–138, 2001.

Deanna Needell, Nathan Srebro, and Rachel Ward. Stochastic gradient descent, weighted sampling, and the randomized Kaczmarz algorithm. *Mathematical Programming*, 155 (1-2):549–573, 2016.

Arkadi Nemirovski, Anatoli Juditsky, Guanghui Lan, and Alexander Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, 19(4):1574–1609, 2009.

Yurii Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Kluwer, Boston, 2004.

Yurii Nesterov. Smooth minimization of nonsmooth functions. *Mathematical Programming*, 103:127–152, 2005.

Yurii Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.

Yurii Nesterov. Gradient methods for minimizing composite functions. *Mathematical Programming, Ser. B*, 140:125–161, 2013.

Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, New York, 2nd edition, 2006.

Hua Ouyang, Niao He, Long Tran, and Alexander Gray. Stochastic alternating direction method of multipliers. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, Atlanta, GA, USA, 2013.

John Platt. Fast training of support vector machine using sequential minimal optimization. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 185–208. MIT Press, Cambridge, MA, USA, 1999.

Boris T. Polyak and Anatoli Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30:838–855, 1992.

Zheng Qu, Peter Richtárik, and Tong Zhang. Quartz: Randomized dual coordinate ascent with arbitrary sampling. In *Advances in Neural Information Processing Systems 28*, pages 865–873. 2015.

Peter Richtárik and Martin Takáč. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Mathematical Programming*, 144 (1):1–38, 2014.

Peter Richtárik and Martin Takáč. Parallel coordinate descent methods for big data optimization. *Mathematical Programming*, 156(1):433–484, 2016.

Mark Schmidt, Nicolas Le Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient. Technical Report HAL 00860051, INRIA, Paris, France, 2013.

Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research*, 14:567–599, 2013a.

Shai Shalev-Shwartz and Tong Zhang. Accelerated mini-batch stochastic dual coordinate ascent. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 378–385. 2013b.

Shai Shalev-Shwartz and Tong Zhang. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. *Mathematical Programming*, 155(1):105–145, 2015.

Taiji Suzuki. Dual averaging and proximal gradient descent for online alternating direction multiplier method. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, pages 392–400, Atlanta, GA, USA, 2013.

Taiji Suzuki. Stochastic dual coordinate ascent with alternating direction method of multipliers. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, pages 736–744, Beijing, 2014.

Martin Takáč, Avleen Bijral, Peter Richtárik, and Nathan Srebro. Mini-batch primal and dual methods for SVMs. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, 2013.

Paul Tseng. An incremental gradient(-projection) method with momentum term and adaptive stepsiz rule. *SIAM Journal on Optimization*, 8(2):506–531, 1998.

Paul Tseng. On accelerated proximal gradient methods for convex-concave optimization. Unpublished manuscript, 2008.

Huahua Wang and Arindam Banerjee. Online alternating direction method. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*, pages 1119–1126, Edinburgh, Scotland, UK, 2012.

Blake Woodworth and Nathan Srebro. Tight complexity bounds for optimizing composite objectives. arXiv:1605.08003, 2016.

Stephen J. Wright. Coordinate descent algorithms. *Mathematical Programming, Series B*, 151(1):3–34, 2015.

Lin Xiao. Dual averaging methods for regularized stochastic learning and online optimization. *Journal of Machine Learning Research*, 11:2534–2596, 2010.

Lin Xiao and Tong Zhang. A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization*, 24(4):2057–2075, 2014.

Tianbao Yang. Trading computation for communication: Distributed stochastic dual coordinate ascent. In *Advances in Neural Information Processing Systems 26*, pages 629–637. 2013.

Adams Wei Yu, Qihang Lin, and Tianbao Yang. Double stochastic primal-dual coordinate method for regularized empirical risk minimization with factorized data. arXiv:1508.03390, 2015.

Tong Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the 21st International Conference on Machine Learning (ICML)*, pages 116–123, Banff, Alberta, Canada, 2004.

Xiaoqun Zhang, Martin Burger, and Stanley Osher. A unifoed primal-dual algorithm framework based on Bregman iteration. *Journal of Scientific Computing*, 46(1):20–46, January 2011.

Yuchen Zhang and Lin Xiao. Stochastic primal-dual coordinate method for regularized empirical risk minimization. In *Proceedings of The 32nd International Conference on Machine Learning (ICML)*, pages 353–361. 2015.

Peilin Zhao and Tong Zhang. Stochastic optimization with importance sampling for regularized loss minimization. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *JMLR Proceedings*, pages 1–9. JMLR.org, 2015.

Leon Wenliang Zhong and James T. Kwok. Fast stochastic alternating direction method of multipliers. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, pages 46–54, Beijing, China, 2014.