Stochastic Primal-Dual Coordinate Method for Regularized Empirical Risk Minimization

Yuchen Zhang YUCZHANG@EECS.BERKELEY.EDU
University of California Berkeley, Berkeley, CA 94720, USA

Lin Xiao LIN.XIAO@MICROSOFT.COM
Microsoft Research, Redmond, WA 98053, USA

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 method, which alternates between maximizing over one (or more) randomly chosen dual variable and minimizing over the primal variable. We also develop an extension to non-smooth and non-strongly convex loss functions, and an extension with better convergence rate 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.

1. Introduction

We consider a generic convex optimization problem 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:

$$\min_{x \in \mathbb{R}^d} \left\{ P(x) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^{n} \phi_i(a_i^T x) + g(x) \right\}. \qquad (1)$$

Examples of the above formulation include many well-known classification and regression problems. For binary classification, each feature vector a_i is associated with a label $b_i \in \{\pm 1\}$. 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 backgrounds 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 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 expensive, thus incremental methods that operate on a single component function ϕ_i at each iteration can be very attractive. There have been extensive research on incremental (sub)gradient methods (e.g., Tseng, 1998; Nedić & Bertsekas, 2001; Blatt et al., 2007; Bertsekas, 2011) as well as variants of the stochastic gradient method (e.g., Zhang, 2004; Bottou, 2010; Duchi & 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 (it takes many more iterations for them 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 be-*

Proceedings of the 32^{nd} International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37.

ing $(1/\gamma)$-smooth), i.e. for $i = 1, \ldots, n$,

$$|\phi_i'(\alpha) - \phi_i'(\beta)| \leq (1/\gamma)|\alpha - \beta|, \quad \forall\, \alpha, \beta \in \mathbb{R}.$$

In addition, the regularization function g is λ-strongly convex, i.e. for any $x, y \in \mathbb{R}^n$ and any $g'(y) \in \partial g(y)$, we have

$$g(y) \geq g(x) + g'(y)^T(x - y) + \frac{\lambda}{2}\|x - y\|_2^2.$$

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 and the ℓ_1-regularization 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 3.1).

Under Assumption A, the gradient of each component function, $\nabla\phi_i(a_i^T x)$, is also Lipschitz continuous, with constant $L_i = \|a_i\|_2^2/\gamma \leq R^2/\gamma$, where $R = \max_i \|a_i\|_2$. In other words, each $\phi_i(a_i^T x)$ is (R^2/γ)-smooth. We define a *condition number* $\kappa = R^2/(\lambda\gamma)$, and focus on ill-conditioned problems where $\kappa \gg 1$. In the statistical learning context, the regularization parameter λ is usually on the order of $1/\sqrt{n}$ or $1/n$ (e.g. Bousquet & Elisseeff, 2002), thus κ is on the order of $\sqrt{n}$ or n. It can be even larger if the strong convexity in g is added purely for numerical regularization purposes (see Section 3.1).

Let $P^\star = \min_{x\in\mathbb{R}^d} P(x)$ be the optimal value of problem (1). In order to find an approximate solution $\hat{x}$ satisfying $P(\hat{x}) - P^\star \leq \epsilon$, the classical full gradient method and its proximal variants require $\mathcal{O}((1 + \kappa)\log(1/\epsilon))$ iterations (e.g., Nesterov, 2004). Accelerated full gradient (AFG) methods (Nesterov, 2004) enjoy the improved iteration complexity $\mathcal{O}((1 + \sqrt{\kappa})\log(1/\epsilon))$. However, each iteration of these batch methods requires a full pass over the dataset, which cost $\mathcal{O}(nd)$ operations. In contrast, the stochastic gradient method and its 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 $\mathbb{E}[P(\hat{x}) - P^\star] \leq \epsilon$, where the expectation is with respect to the random choices made at all the iterations (e.g., Polyak & Juditsky, 1992; Nemirovski et al., 2009).

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 are 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.

1.2. Our Contribution

In this paper, we present a new algorithm with batch complexity

$$\mathcal{O}\big((1 + \sqrt{\kappa/n})\log(1/\epsilon)\big), \tag{2}$$

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.

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

$$\min_{x\in\mathbb{R}^d} \max_{y\in\mathbb{R}^n} f(x, y), \tag{3}$$

where

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

Under Assumption A, each ϕ_i^* is γ-strongly convex (since ϕ_i is $(1/\gamma)$-smooth (e.g., Hiriart-Urruty & Lemaréchal, 2001, Theorem 4.2.2)) and g is λ-strongly convex. As a consequence, the saddle point problem (3) has a unique solution, which we denote by $(x^\star, y^\star)$. The Stochastic Primal-Dual Coordinate (SPDC) method we propose in this paper achieves the batch complexity in (2) for solving the primal-dual problem (3).

1.3. Comparing with Dual Coordinate Ascent Methods

It is worth comparing our method with the family of dual coordinate ascent methods, which solves the primal problem (1) via its dual:

$$\max_{y\in\mathbb{R}^n} \left\{ \frac{1}{n}\sum_{i=1}^n -\phi_i^*(y_i) - g^*\left(-\frac{1}{n}\sum_{i=1}^n y_i a_i \right) \right\}, \tag{9}$$

where $g^*(u) = \sup_{x\in\mathbb{R}^d}\{x^T u - g(x)\}$ is the conjugate function of g. Recent work show that dual coordinate ascent methods are typically more efficient than primal full gradient methods (e.g., Hsieh et al., 2008; Shalev-Shwartz & 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 ob-

Algorithm 1 The Stochastic Primal-Dual Coordinate (SPDC) method

Input: mini-batch size m, parameters $\tau, \sigma, \theta \in \mathbb{R}_+$, number of iterations T, and $x^{(0)}$ 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 of indices $K \subset \{1, 2, \ldots, n\}$ of size m, such that the probability of each index being picked is equal to m/n. Execute the following updates:

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

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

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

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

end

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

jective value. Shalev-Shwartz & Zhang (2013a) showed that the batch complexity of SDCA is $\widetilde{\mathcal{O}}(1 + \kappa/n)$. The SPDC method, which has batch complexity $\widetilde{\mathcal{O}}(1 + \sqrt{\kappa/n})$, can be much better when $\kappa > n$, i.e., for ill-conditioned problems.

In addition, Shalev-Shwartz & Zhang (2013b) developed an accelerated proximal SDCA method which achieves the same batch complexity in (2). 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 & Zhang, 2013a) is called to solve the dual problem (9) with customized regularization parameter and precision. In contrast, SPDC is a single-loop primal-dual coordinate method.

More recently, Lin et al. (2014) developed an accelerated proximal coordinate gradient (APCG) method for solving a more general class of composite convex optimization problems. When applied to the dual problem (9), APCG enjoys the same batch complexity $\widetilde{\mathcal{O}}\big(1 + \sqrt{\kappa/n}\big)$ as of SPDC. However, it needs an extra primal proximal-gradient step to have theoretical guarantees on the convergence of primal-dual gap (Lin et al., 2014). This in unnecessary for the SPDC method.

2. The SPDC method

In this section, we describe and analyze the 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 m coordinates of y at a time, and maximizing f only with respect to the selected coordinates. Consequently, the computational cost of each iteration is $\mathcal{O}(md)$. Here m is called the mini-batch size; in the simplest case, we have $m = 1$.

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 $\overline{x}^{(t)}$. From the initialization $u^{(0)} = (1/n)\sum_{i=1}^n y_i^{(0)} a_i$ and the update rules (5) and (7), we have $u^{(t)} = \frac{1}{n}\sum_{i=1}^n y_i^{(t)} a_i$. Equation (8) obtains $\overline{x}^{(t+1)}$ based on extrapolation from $x^{(t)}$ and $x^{(t+1)}$. This step is similar to Nesterov's acceleration technique (Nesterov, 2004), and yields faster convergence rate.

With a single processor, each iteration of Algorithm 1 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 $x^{(t+1)}$. Such a procedure can be achieved by a single round of communication, for example, using the `Allreduce` operation in MPI or MapReduce. If we ignore the communication delay, then each iteration takes

$\mathcal{O}(d)$ time. Not surprisingly, we will show that the SPDC algorithm converges faster with larger m, because it processes multiple dual coordinates in a single iteration.

2.1. Convergence analysis

We present a convergence theorem for the SPDC algorithm.

Theorem 1. *Assume that each ϕ_i is $(1/\gamma)$-smooth and g is λ-strongly convex (Assumption A). Let $R = \max\{\|a_i\|_2 : i = 1, \dots, n\}$. If the parameters τ, σ and θ in Algorithm 1 are chosen such that*

$$\tau = \frac{1}{2R}\sqrt{\frac{m\gamma}{n\lambda}}, \quad \sigma = \frac{1}{2R}\sqrt{\frac{n\lambda}{m\gamma}} \quad \text{and}$$
$$\theta = 1 - \frac{1}{(n/m) + R\sqrt{(n/m)/(\lambda\gamma)}}, \tag{10}$$

then for each $t \geq 1$, the Mini-Batch SPDC algorithm achieves

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

The proof of Theorem 1 is given in the long version of this paper (Zhang & Xiao, 2014). The following corollary establishes the expected iteration complexity for obtaining an ϵ-accurate solution.

Corollary 1. *Suppose Assumption A holds and the parameters τ, σ and θ are set as in (10). In order for Algorithm 1 to obtain*

$$\mathbb{E}[\|x^{(T)} - x^\star\|_2^2] \leq \epsilon, \qquad \mathbb{E}[\|y^{(T)} - y^\star\|_2^2] \leq \epsilon, \tag{11}$$

it suffices to have the number of iterations T satisfy

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

where

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

Proof. By Theorem 1, we have $\mathbb{E}[\|x^{(T)} - x^\star\|_2^2] \leq \theta^T C$ and $\mathbb{E}[\|y^{(T)} - y^\star\|_2^2] \leq \theta^T C$. To obtain (11), it suffices to ensure that $\theta^T C \leq \epsilon$, which is equivalent to

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

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

Recall the definition of the condition number $\kappa = R^2/(\lambda\gamma)$ in Section 1.1. Corollary 1 establishes that the iteration complexity of the SPDC method for achieving (11) 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 shared by the AFG methods (Nesterov, 2004), as well as the batch primal-dual algorithm of Chambolle & Pock (2011).

Since an equivalent pass over the dataset corresponds to n/m iterations, the batch complexity of SPDC is

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

This 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.

3. Extensions of SPDC

In this section, we derive two extensions of the SPDC method. The first one handles problems for which Assumption A does not hold. The second one employs a non-uniform sampling scheme to improve the iteration complexity when the feature vectors a_i are unnormalized.

3.1. Non-smooth or non-strongly convex functions

The complexity bounds established in Section 2 require each ϕ_i^* to be γ-strongly convex, which corresponds to the condition that the first derivative of ϕ_i is $(1/\gamma)$-Lipschitz continuous. In addition, the function g needs to 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 SPDC can still be applied.

For simplicity, here we consider the case where neither ϕ_i is smooth nor g is strongly convex. Formally, we 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:

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

Denote by $(x_\delta^\star, y_\delta^\star)$ the saddle-point of f_δ. We employ the SPDC method (Algorithm 1) 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 now are all δ-strongly convex. We note that adding strongly convex

Algorithm 2 SPDC method with weighted sampling

Input: parameters $\tau, \sigma, \theta \in \mathbb{R}_+$, number of iterations T, and 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 $p_k = \frac{1}{2n} + \frac{\|a_k\|_2}{2\sum_{i=1}^n \|a_i\|_2}$. Execute the following updates:

$$y_i^{(t+1)} = \begin{cases} \arg\max_{\beta \in \mathbb{R}} \left\{ \beta \langle a_i, \overline{x}^{(t)} \rangle - \phi_i^*(\beta) - \frac{p_i n}{2\sigma}(\beta - y_i^{(t)})^2 \right\} & i = k, \\ y_i^{(t)} & i \neq k, \end{cases} \tag{12}$$

$$x^{(t+1)} = \arg\min_{x \in \mathbb{R}^d} \left\{ g(x) + \left\langle u^{(t)} + \frac{1}{p_k n}(y_k^{(t+1)} - y_k^{(t)}) a_k,\ x \right\rangle + \frac{\|x - x^{(t)}\|_2^2}{2\tau} \right\}, \tag{13}$$

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

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

end

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

perturbation on ϕ_i^* is equivalent to smoothing ϕ_i, which becomes $(1/\delta)$-smooth. Letting $\gamma = \lambda = \delta$, the parameters τ, σ and θ in (10) become

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

Although $(x_\delta^\star, y_\delta^\star)$ is not exactly the saddle point of f, the following corollary shows that applying the SPDC method to the perturbed function f_δ effectively minimizes the original loss function P. See the long version of this paper (Zhang & Xiao, 2014) for the proof.

Corollary 2. *Assume that each ϕ_i is convex and G_ϕ-Lipschitz continuous, and g is convex and G_g-Lipschitz continuous. Define two constants:*

$$C_1 = (\|x^\star\|_2^2 + G_\phi^2), \quad \textit{and}$$

$$C_2 = (G_\phi R + G_g)^2 \left(\|x^{(0)} - x_\delta^\star\|_2^2 + \frac{\frac{1}{2\sigma} + \delta}{\frac{1}{2\tau} + \delta} \frac{\|y^{(0)} - y_\delta^\star\|_2^2}{m} \right).$$

If we choose $\delta = \epsilon / C_1$, and run the SPDC algorithm for T iterations where

$$T \geq \left(\frac{n}{m} + \frac{C_1 R}{\epsilon}\sqrt{\frac{n}{m}} \right) \log\left(\frac{4 C_2}{\epsilon^2} \right),$$

then $\mathbb{E}[P(x^{(T)}) - P(x^\star)] \leq \epsilon$.

When $m = 1$, the corresponding batch complexity is $\widetilde{\mathcal{O}}(1 + (\epsilon^2 n)^{-1/2})$. Under the same condition, the batch complexity of full accelerated gradient method and that of stochastic gradient descent are $\mathcal{O}(1 + \epsilon^{-1})$ and $\widetilde{\mathcal{O}}(1 + (\epsilon^2 n)^{-1})$ respectively (Nesterov, 2005; Shamir & Zhang, 2012), both slower than the SPDC method.

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. In Table 1, we list the batch complexities of the SPDC method for finding an ϵ-optimal solution of problem (1) under various assumptions.

3.2. 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 2. For simplicity of presentation, we described in Algorithm 2 with single dual coordinate update, i.e., the case of $m = 1$.

The basic idea is to use non-uniform sampling in picking the dual coordinate to update at each iteration. In particular, instances with large feature norms should be sampled more frequently. Simultaneously, we adopt an adaptive regularization in step (12), imposing stronger regularization on such instances. In addition, we adjust the weight of a_k in (13) for updating the primal variable. As a consequence, the convergence rate of Algorithm 2 depends on the aver-

ϕ_i	g	batch complexity
$(1/\gamma)$-smooth	λ-strongly convex	$1 + \sqrt{\frac{m}{\lambda\gamma n}} \log(1/\epsilon)$
$(1/\gamma)$-smooth	non-strongly convex	$1 + \sqrt{\frac{m}{\epsilon\gamma n}}$
non-smooth	λ-strongly convex	$1 + \sqrt{\frac{m}{\epsilon\lambda n}}$
non-smooth	non-strongly convex	$1 + \sqrt{\frac{m}{\epsilon^2 n}}$

Table 1. Batch 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 $\delta = \epsilon / C_1$.

age norm of feature vectors.

Theorem 2. *Suppose Assumption A holds. Let* $\bar{R} = \frac{1}{n}\sum_{i=1}^n \|a_i\|_2$. *If the parameters* τ, σ, θ *in Algorithm 2 are chosen such that*

$$\tau = \frac{1}{4\bar{R}}\sqrt{\frac{\gamma}{n\lambda}}, \quad \sigma = \frac{1}{4\bar{R}}\sqrt{\frac{n\lambda}{\gamma}}, \quad \theta = 1 - \left(2n + 2\bar{R}\sqrt{\frac{n}{\lambda\gamma}}\right)^{-1},$$

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

$$\left(\frac{1}{2\tau} + \lambda\right)\mathbb{E}[\|x^{(t)} - x^\star\|_2^2] + \left(\frac{7}{16\sigma} + \frac{2\gamma}{n}\right)\mathbb{E}[\|y^{(t)} - y^\star\|_2^2]$$
$$\leq \theta^t\left(\left(\frac{1}{2\tau} + \lambda\right)\|x^{(0)} - x^\star\|_2^2 + \left(\frac{1}{2\sigma} + 2\gamma\right)\|y^{(0)} - y^\star\|_2^2\right).$$

Comparing the constant θ in Theorem 2 to that of Theorem 1, the constant $\bar{R}$ here is determined by the average norm of features, instead of the largest one. It makes the algorithm 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 $\mathbb{E}[\|a_i\|_2]$.

4. Efficient Implementation with Sparse Data

During each iteration of the SPDC method, the updates of primal variables (i.e., computing $x^{(t+1)}$) require full d-dimensional vector operations; see the step (6) of Algorithm 1 and the step (13) of Algorithm 2. 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.

4.1. 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:

$$x_j^{(t+1)} = \frac{1}{1+\lambda\tau}(x_j^{(t)} - \tau u_j^{(t)} - \tau\Delta u_j), \qquad (14)$$

where Δu denotes $\frac{1}{m}\sum_{k\in K}(y_k^{(t+1)} - y_k^{(t)})a_k$ in Algorithm 1, or $(y_k^{(t+1)} - y_k^{(t)})a_k/(p_k n)$ in Algorithm 2, 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) and (12), 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 $\Delta u_j = 0$, for $t = t_0 + 1, t_0 + 2, \ldots, t_1 - 1$ we have

$$x_j^{t+1} = \frac{1}{1+\lambda\tau}(x_j^{(t)} - \tau u_j^{(t)}). \qquad (15)$$

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 formula (15), we obtain

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

This update 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.

4.2. $(\ell_1 + \ell_2)$-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,

$$x_j^{(t+1)} = \arg\min_{\alpha\in\mathbb{R}}\Big\{\lambda_1|\alpha| + \frac{\lambda_2\alpha^2}{2} + (u_j^{(t)} + \Delta u_j)\alpha + \frac{(\alpha - x_j^{(t)})^2}{2\tau}\Big\}, \qquad (16)$$

where Δu_j follows the definition in Section 4.1. If $j \notin J^{(t)}$, then $\Delta u_j = 0$ and equation (16) can be simplified as

$$x_j^{(t+1)} = \begin{cases} \frac{1}{1+\lambda_2\tau}(x_j^{(t)} - \tau u_j^{(t)} - \tau\lambda_1) & \text{if } x_j^{(t)} - \tau u_j^{(t)} > \tau\lambda_1, \\ \frac{1}{1+\lambda_2\tau}(x_j^{(t)} - \tau u_j^{(t)} + \tau\lambda_1) & \text{if } x_j^{(t)} - \tau u_j^{(t)} < -\tau\lambda_1, \\ 0 & \text{otherwise.} \end{cases}$$

Similar to the approach of Section 4.1, 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

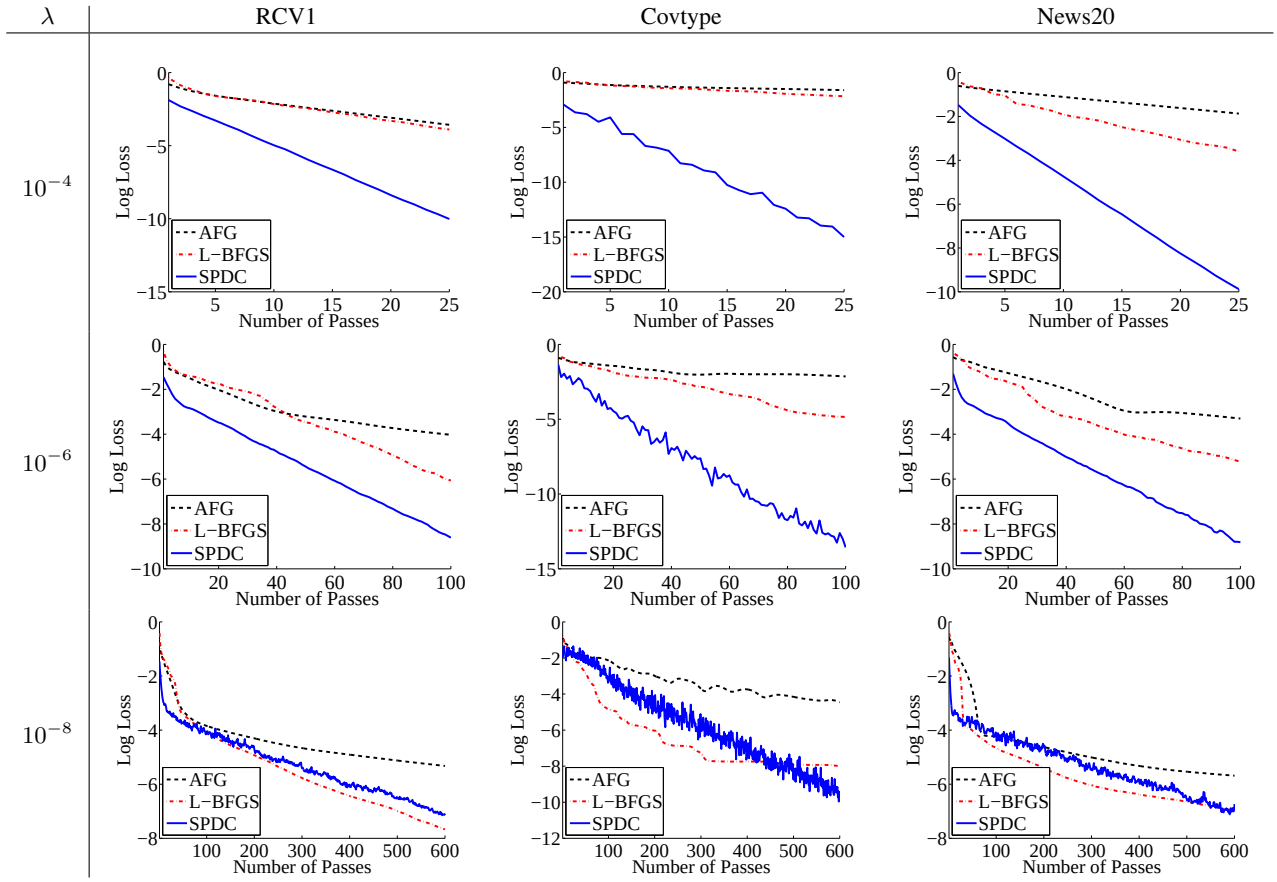

Figure 1. Comparing SPDC with AFG and L-BFGS on three real datasets. The horizontal axis is the number of passes through the entire dataset, and the vertical axis is the logarithmic optimality gap $\log(P(x^{(t)}) - P(x^\star))$.

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 the above equation 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 $x_j^{(t_1)}$. See Zhang & Xiao (2014, Appendix C) for the algorithm details. 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.

5. Experiments

In this section, we compare the SPDC method (Algorithm 1 with $m = 1$) with several state-of-the-art optimization algorithms for solving problem (1). They include two batch-update algorithms: the accelerated full gradient (FAG) method (Nesterov, 2004), and the limited-memory quasi-Newton method L-BFGS (e.g., Nocedal & Wright, 2006). 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 & Wright (2006). We also compare SPDC with three stochastic algorithms: the stochastic average gradient (SAG) method (Roux et al., 2012), the stochastic dual coordinate descent (SDCA) method (Shalev-Shwartz & Zhang, 2013a) and the accelerated stochastic dual coordinate descent (ASDCA) method (Shalev-Shwartz & Zhang, 2013b).

The datasets are obtained from LIBSVM data (Fan & Lin, 2011) and summarized in Table 2. The 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). For all tasks, the data points take the form of (a_i, b_i), where

Dataset name	# samples n	# features d
Covtype	581,012	54
RCV1	20,242	47,236
News20	19,996	1,355,191

Table 2. Characteristics of three real datasets.

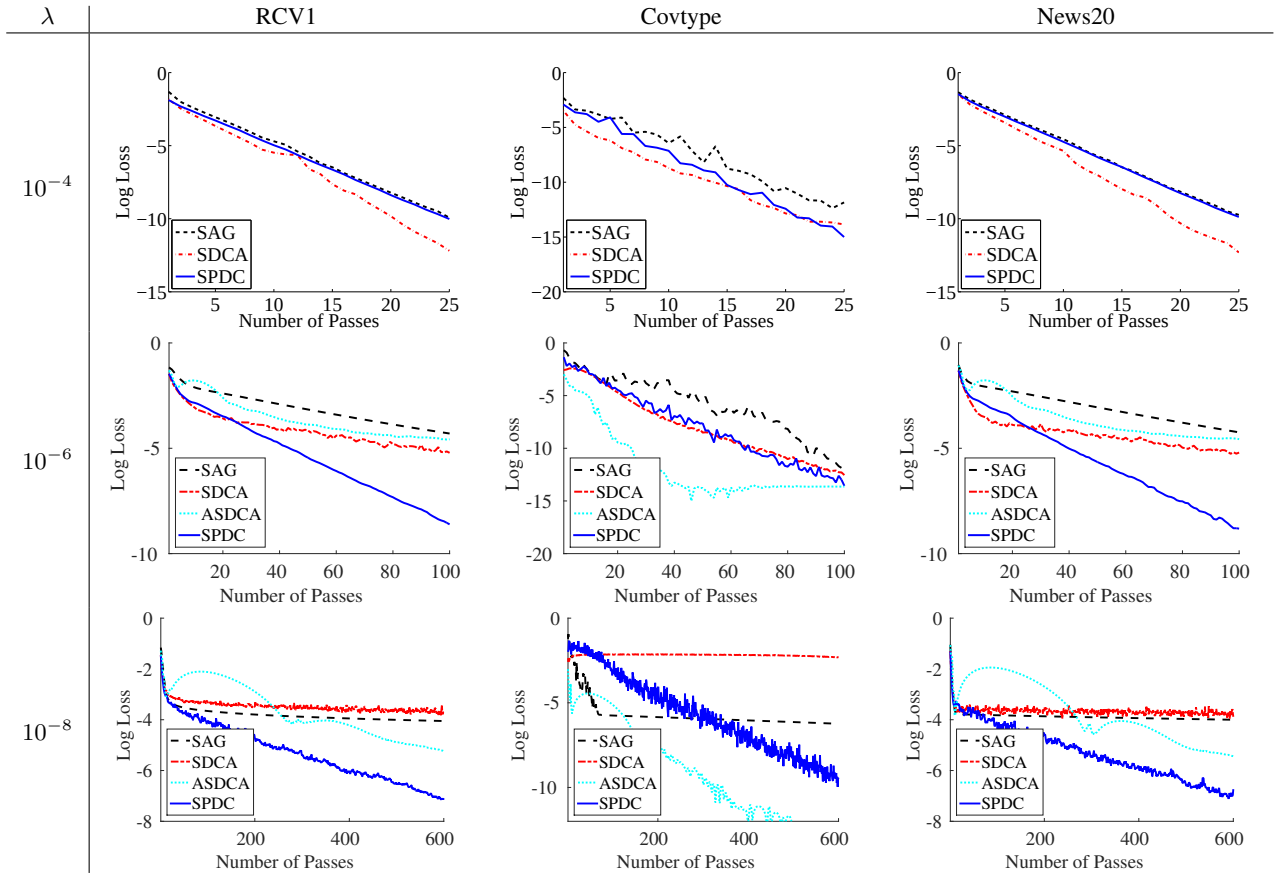

Figure 2. Comparing SPDC with SAG, SDCA and ASDCA on three real datasets. The horizontal axis is the number of passes through the entire dataset, and the vertical axis is the logarithmic optimality gap $\log(P(x^{(T)}) - P(x^\star))$.

$a_i \in \mathbb{R}^d$ is the feature vector, and $b_i \in \{-1, 1\}$ is the binary class label. Our goal is to minimize the regularized empirical risk:

$$P(x) = \frac{1}{n}\sum_{i=1}^{n} \phi_i(a_i^T x) + \frac{\lambda}{2}\|x\|_2^2.$$

Here, ϕ_i is the smoothed hinge loss (see e.g., Shalev-Shwartz & Zhang, 2013a):

$$\phi_i(z) = \begin{cases} 0 & \text{if } b_i z \geq 1 \\ \frac{1}{2} - b_i z & \text{if } b_i z \leq 0 \\ \frac{1}{2}(1 - b_i z)^2 & \text{otherwise.} \end{cases}$$

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 $+\infty$ otherwise.

The performance of the five algorithms are plotted in Figure 1 and Figure 2. In Figure 1, 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 dropping to 10^{-8}, the batch methods (especially L-BFGS) become comparable to SPDC.

In Figure 2, we compare SPDC with the three stochastic methods: SAG, SDCA and ASDCA. The ASDCA specification (Shalev-Shwartz & Zhang, 2013b) 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 $\lambda \in \{10^{-6}, 10^{-8}\}$. Here, the observations are just the opposite to that of Figure 1. All stochastic algorithms have comparable performance on relatively large λ, but the two accelerated algorithms SPDC and ASDCA becomes substantially faster when λ gets closer to zero.

Among these the two accelerated algorithms, ASDCA converges faster than SPDC on the Covtype dataset (which has a very small feature dimension) and slower on the remaining two datasets. In addition, due to the outer-inner loop structure of the ASDCA algorithm, its objective gap my increase at the beginning of each outer iteration. This oscillation can be undesirable, especially at early iterations. In contrast, the convergence of SPDC is almost linear and more stable than ASDCA.

References

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

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

Bottou, L. Large-scale machine learning with stochastic gradient descent. In Lechevallier, Y. and Saporta, G. (eds.), *Proceedings of the 19th International Conference on Computational Statistics*, pp. 177–187, Paris, France, August 2010. Springer.

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

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

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

Fan, R.-E. and Lin, C.-J. LIBSVM data: Classification, regression and multi-label. URL: http://www.csie.ntu.edu.tw/˜cjlin/libsvmtools/datasets, 2011.

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

Hiriart-Urruty, J.-B. and Lemaréchal, C. *Fundamentals of Convex Analysis*. Springer, 2001.

Hsieh, C.-J., Chang, K.-W., Lin, C.-J., Keerthi, S., and Sundararajan, S. A dual coordinate descent method for large-scale linear svm. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, pp. 408–415, 2008.

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

Lin, Q., Lu, Z., and Xiao, L. An accelerated proximal coordinate gradient method and its application to regularized empirical risk minimization. Technical Report MSR-TR-2014-94, Microsoft Research, 2014.

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

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

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

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

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

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

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

Roux, N. L., Schmidt, M., and Bach, F. A stochastic gradient method with an exponential convergence rate for finite training sets. In *Advances in Neural Information Processing Systems 25*, pp. 2672–2680. 2012.

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

Shalev-Shwartz, S. and Zhang, T. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. arXiv:1309.2375, 2013b.

Shamir, O. and Zhang, T. Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. *arXiv:1212.1824*, 2012.

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

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

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

Zhang, Y. and Xiao, L. Stochastic primal-dual coordinate method for regularized empirical risk minimization. *arXiv preprint arXiv:1409.3257*, 2014.