Accelerated Doubly Stochastic Gradient Algorithm for Large-scale Empirical Risk Minimization

Zebang Shen, Hui Qian*, Tongzhou Mu, and Chao Zhang
College of Computer Science and Technology, Zhejiang University, China
{shenzebang, qianhui, mutongzhou, zczju}@zju.edu.cn

Abstract

Nowadays, algorithms with fast convergence, small memory footprints, and low per-iteration complexity are particularly favorable for artificial intelligence applications. In this paper, we propose a doubly stochastic algorithm with a novel accelerating multi-momentum technique to solve large scale empirical risk minimization problem for learning tasks. While enjoying a provably superior convergence rate, in each iteration, such algorithm only accesses a mini batch of samples and meanwhile updates a small block of variable coordinates, which substantially reduces the amount of memory reference when both the massive sample size and ultra-high dimensionality are involved. Specifically, to obtain an ϵ-accurate solution, our algorithm requires only $\mathcal{O}(\log(1/\epsilon)/\sqrt{\epsilon})$ overall computation for the general convex case and $\mathcal{O}((n+\sqrt{n\kappa})\log(1/\epsilon))$ for the strongly convex case. Empirical studies on huge scale datasets are conducted to illustrate the efficiency of our method in practice.

1 Introduction

In this paper, we consider the following problem:

$$\min_{\mathbf{x}\in\mathbb{R}^d} \mathbf{F}^{\mathbf{P}}(\mathbf{x}) = \mathbf{F}(\mathbf{x}) + \mathbf{P}(\mathbf{x}), \qquad (1)$$

where $\mathbf{F}(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{x})$ is the average of n smooth convex component functions f_i's and $\mathbf{P}(\mathbf{x})$ is a block-separable, possibly non-smooth, convex regularization function. In machine learning applications, many tasks can be naturally phrased as the above problem, e.g. the Empirical Risk Minimization (ERM) problem [Zhang and Xiao, 2015; Friedman *et al.*, 2001]. However, the latest explosive growth of data introduces unprecedented computational challenges of scalability and storage bottleneck. In consequence, algorithms with fast convergence, small memory footprints, and low per-iteration complexity have been ardently pursued in the recent years. Most successful practices adopt stochastic strategies that incorporate randomness into solving procedures. They followed two parallel tracks. That is, in each iteration, gradient is estimated by either a mini batch of *samples* or a small block of *variable coordinates*, commonly designated by a random procedure.

Accessing only a random mini batch of samples in each iteration, classical *Stochastic Gradient Descent* (SGD) and its *Variance Reduction* (VR) variants, such as SVRG [Johnson and Zhang, 2013], SAGA [Defazio *et al.*, 2014], and SAG [Schmidt *et al.*, 2013], have gained increasing attention in the last quinquennium. To further improve the performance, a significant amount of efforts have been made towards reincarnating Nesterov's optimal accelerated convergence rate in SGD type methods [Zhang and Xiao, 2015; Frostig *et al.*, 2015; Lin *et al.*, 2015a; Shalev-Shwartz and Zhang, 2014; Hu *et al.*, 2009; Lan, 2012; Nitanda, 2014]. Recently, a direct accelerated version of SVRG called Katyusha is proposed to obtain optimal convergence results without compromising the low per-iteration sample access [Allen-Zhu, 2016; Woodworth and Srebro, 2016]. However, when datasets are high dimensional, SGD type methods may still suffer from the large memory footprint due to its full vector operation in each iteration, which lefts plenty of scope to push further.

Updating variables only on a randomly selected small block of *variable coordinates* in each iteration is another important strategy that can be adopted solely to reduce the memory reference. The most noteworthy endeavors include the *Randomized Block Coordinate Descent* (RBCD) methods [Nesterov, 2012; Richtárik and Takáč, ; Lee and Sidford, 2013; Wright, 2015]. Now accelerated versions of RBCD type methods, such as APPROX [Fercoq and Richtárik, 2015] and APCG [Lin *et al.*, 2015b], have also made their debuts. A drawback of RBCD type methods is that all samples have to be accessed in each iteration. When the number of samples is huge, they can be still quite inefficient.

Doubly stochastic algorithms, simultaneously utilizing the idea of randomness from *sample choosing* and *coordinate selection* perspective, have emerged more recently. Zhao *et al.* carefully combine ideas from VR and RBCD and propose a method called *Mini-batch Ran-*

[*] Corresponding author.

Table 1: We give the per-iteration Sample Access (S.A.), Vector Operation (V.O.), and overall computational complexities to obtain an ϵ-accurate solution in relative algorithms. Here APG is short for Accelerated Proximal Gradient. L and κ is defined in section 2.1. The mini batch size is set to 1 for simplicity. Ω is the block size.

Method	S.A.	V.O.	General Convex	Strongly Convex
APG	$\mathcal{O}(n)$	$\mathcal{O}(d)$	$\mathcal{O}(dn\sqrt{L/\epsilon})$	$\mathcal{O}(dn\sqrt{\kappa}\log(1/\epsilon))$
RBCD	$\mathcal{O}(n)$	$\mathcal{O}(\Omega)$	$\mathcal{O}(dnL/\epsilon)$	$\mathcal{O}(dn\kappa\log(1/\epsilon))$
APCG	$\mathcal{O}(n)$	$\mathcal{O}(\Omega)$	$\mathcal{O}(dn\sqrt{L/\epsilon})$	$\mathcal{O}(dn\sqrt{\kappa}\log(1/\epsilon))$
SVRG	$\mathcal{O}(1)$	$\mathcal{O}(d)$	$\mathcal{O}(d(n+L/\epsilon)\log\frac{1}{\epsilon})$	$\mathcal{O}(d(n+\kappa)\log(1/\epsilon))$
Katyusha	$\mathcal{O}(1)$	$\mathcal{O}(d)$	$\mathcal{O}(d(n+\sqrt{nL})/\sqrt{\epsilon})$	$\mathcal{O}(d(n+\sqrt{n\kappa})\log(1/\epsilon))$
MRBCD/ASBCD	$\mathcal{O}(1)$	$\mathcal{O}(\Omega)$	$\mathcal{O}(d(n+L/\epsilon)\log(1/\epsilon))$	$\mathcal{O}(d(n+\kappa)\log(1/\epsilon))$
ADSG(this paper)	$\mathcal{O}(1)$	$\mathcal{O}(\Omega)$	$\mathcal{O}(d(n+\sqrt{nL/\epsilon})\log\frac{1}{\epsilon})$	$\mathcal{O}(d(n+\sqrt{n\kappa})\log(1/\epsilon))$

domized Block Coordinate Descent with variance reduction (MRBCD), which achieves linear convergence in strongly convex case [Zhao *et al.*, 2014]. Zhang and Gu propose the *Accelerated Stochastic Block Coordinate Descent* (ASBCD) method, which incorporates RBCD into SAGA and uses a non-uniform probability when sampling data points [Zhang and Gu, 2016]. These methods avoid both full dataset assess and full vector operation in each iteration, and therefore are amenable to solving (1) when n and d are large at the same time. However, none of them meets the optimal convergence rate [Woodworth and Srebro, 2016], and hence can still be accelerated.

To bridge the gap, we introduce a multi-momentum technique in this paper and devise a method called *Accelerated Doubly Stochastic Gradient algorithm* (ADSG). Our method enjoys an accelerated convergence rate, superior to existing doubly stochastic methods, without compromising the low sample access and small per-iteration complexity. Specifically, our contributions are listed as follows.

1. With two novel *coupling steps*, we incorporate three momenta into ADSG. These two steps enable the acceleration of doubly stochastic optimization procedure. Additionally, we devise an efficient implementation of ADSG for ERM problem.
2. We prove that ADSG has an accelerated convergence rate and the overall computational complexity to obtain an ϵ-accurate solution is $\mathcal{O}(\log(1/\epsilon)/\sqrt{\epsilon})$ in the general convex case and $\mathcal{O}((n+\sqrt{n\kappa})\log(1/\epsilon))$ in the strongly convex case, where κ stands for the condition number.

Further, to show the efficiency of ADSG in practice, we conduct learning tasks on huge datasets with more than 10M samples and 1M features. The results demonstrate superior computational efficiency of our approach compared to the state-of-the-art.

2 Preliminary

2.1 Notation & Assumptions

We assume that the variable $\mathbf{x} \in \mathbb{R}^d$ can be equally partitioned into B blocks for simplicity, and we let $\Omega = d/B$ be the block size. We denote the coordinates in the l^{th} block of $\mathbf{x}$ by $[\mathbf{x}]_l \in \mathbb{R}^\Omega$ and the rest by $[\mathbf{x}]_{\setminus l}$. The regularization $\mathbf{P}(\mathbf{x})$ is assumed to be block separable with respect to the partition of $\mathbf{x}$, i.e. $\mathbf{P}(\mathbf{x}) = \sum_{l=1}^{B} \mathbf{P}_l([\mathbf{x}]_l)$. Many important functions qualify such separability assumption [Tibshirani, 1996; Simon *et al.*, 2013]. The proximal operator of a convex function g is defined as $\text{prox}_g(\mathbf{y}) = \text{argmin}_{\mathbf{x}}\, g(\mathbf{x}) + \frac{1}{2}\|\mathbf{x}-\mathbf{y}\|^2$ where we use $\|\cdot\|$ to denote the Euclidean norm. We say $\mathbf{x}$ is an ϵ-accurate solution of to Problem (1) if $\mathbf{F}^{\mathbf{P}}(\mathbf{x}) - \mathbf{F}^{\mathbf{P}}(\mathbf{x}_*) \le \epsilon$, where $\mathbf{x}_*$ is the optimal. Further, we define ϵ_0 to be $\mathbf{F}^{\mathbf{P}}(\mathbf{x}_0)$, i.e. the objective value at the initial point $\mathbf{x}_0$. We now give some assumptions.

Assumption I Each function f_i is L-smooth, i.e. $\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, $f(\mathbf{y}) \le f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \mathbf{y}-\mathbf{x}\rangle + \frac{L}{2}\|\mathbf{x}-\mathbf{y}\|^2$.
Assumption II The average function $\mathbf{F}$ is L_B-smooth with respect to each block, i.e. for any l and $\forall \mathbf{x}, \mathbf{h} \in \mathbb{R}^d$ such that $[\mathbf{h}]_{\setminus l} \equiv 0$,$f(\mathbf{x}+\mathbf{h}) \le f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \mathbf{h}\rangle + \frac{L_B}{2}\|\mathbf{h}\|^2$.
Assumption III Function $\mathbf{P}$ is μ-strongly convex, i.e. $\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, and $g \in \partial\mathbf{P}(\mathbf{x})$, $\mathbf{P}(\mathbf{y}) \ge \mathbf{P}(\mathbf{x}) + \langle g, \mathbf{y}-\mathbf{x}\rangle + \frac{\mu}{2}\|\mathbf{x}-\mathbf{y}\|^2$.

We define $\kappa = \frac{L+L_B}{\mu}$ as the condition number.

2.2 Doubly Stochastic Algorithms for Convex Composite Optimization

A doubly stochastic method that incorporates both VR and CD technique usually consists of two nested loops. More specifically, at the beginning of each outer loop, the exact full gradient at some snapshot point $\tilde{\mathbf{x}}$ is calculated. Then, each iteration of the following inner loop estimates the partial gradient based on a mini-batch component functions and modifies it by the obtained exact gradient to perform block coordinate descent. Take MRBCD for example. As noted in Table 1, its per-iteration vector operation complexity and sample access are $\mathcal{O}(\Omega)$ and $\mathcal{O}(1)$ respectively, which is amenable to solving large-sample-high-dimension problems. However, its overall computational complexity to achieve an ϵ-accurate solution scales linearly with respect to the condition number κ in strongly convex case and depends on $1/\epsilon$ (up to a log factor) in general convex case. Such complexity leaves much room to improve in doubly stochastic methods, especially when a highly accurate solution to an ill-conditioned problem is sought (small ϵ, large κ).

2.3 Momentum Accelerating Technique

Accelerating the first order algorithm with momentum is by no mean new but has always been considered difficult [Polyak, 1964; Allen-Zhu and Orecchia, 2014]. Nesterov is the first to prove the accelerated convergence rate for deterministic smooth convex optimization [Nesterov, 1983; 1998]. When randomness is involved in data point sampling, noise in the stochastic gradient is further accumulated by the momentum, making the acceleration much harder. Efforts have been made to overcome such difficulty in various ways: with an outer-inner loop manner [Lin *et al.*, 2015a], from a dual perspective [Shalev-Shwartz and Zhang, 2014], or under a primal-dual framework [Zhang and Xiao, 2015]. However, it was not until recently that an optimal method called Katyusha is proposed [Allen-Zhu, 2016; Woodworth and Srebro, 2016]. In a parallel line, at the first attempt to accelerate the RBCD, the momentum step forces full vector operation in each iteration and hence compromises the advantage of RBCD [Nesterov, 2012]. With continuing efforts, Lin *et al.* proposed the APCG method for both strongly convex and general convex problems, avoiding full vector operation completely [Lin *et al.*, 2015b]. While these previous mentioned analyses deal with the randomness only in data point sampling or coordinate choosing, our analysis, due to doubly stochastic nature of ADSG, has to consider how randomness affects the momenta from both sample and feature perspective, and hence is more difficult.

2.4 Empirical Risk Minimization

We focus on Empirical Risk Minimization (ERM) with linear predictor, an important class of smooth convex problems. Specifically, each f_i in Problem (1) is of the form $f_i(\mathbf{x}) = \phi_i(\mathbf{a}_i^\top \mathbf{x})$, where $\mathbf{a}_i$ is the feature vector of the i^{th} sample and $\phi_i(\cdot) : \mathbb{R} \to \mathbb{R}$ is some smooth convex function. Let $\mathbf{A} = [\mathbf{a}_1 \ldots \mathbf{a}_n]^\top$ be the data matrix. In real applications, $\mathbf{A}$ is usually very sparse and we define its sparsity to be $\rho = \frac{nnz(\mathbf{A})}{nd}$. Note that our convergence analysis applies to any convex smooth f_i.

3 Methodology

We present the proposed ADSG in algorithm 1, and discuss about some crucial details in this section.

Input: The input of ADSG varies for strongly convex and general convex problems. In the former case, μ should be the strongly convex parameter, and we set $\alpha_{2,s} = \frac{1}{2B}\min\{1, \sqrt{\frac{n}{\kappa}}\}$ and $\alpha_{3,s} = \frac{1}{2B}$. In the latter case, μ is set to 0, and we set $\alpha_{2,s} = \frac{2}{s+4B}$ and $\alpha_{3,s} = \frac{1}{2B}$. When $\mu = 0$, line 14 adopts uniform probability for sampling.

Main Body: Our algorithm is divided into epochs. Four variables $\mathbf{x}_k$, $\mathbf{y}_k$, $\mathbf{z}_k$, and $\tilde{\mathbf{x}}^s$ are maintained throughout. At the beginning of each epoch, the full gradient at the snapshot point $\tilde{\mathbf{x}}^s$ is computed. Updating steps are taken in the follow-up m inner loops, where we randomly select a mini-batch I of size b and a feature block l to construct a mixed stochastic gradient at point $\mathbf{y}_k$ and perform proximal coordinate descent on the auxiliary variable $\mathbf{z}_k$.

Algorithm 1 ADSG I

```
Input: $n, \mathbf{x}_0, S, b, B, \mu, \{\alpha_{2,s}, \alpha_{3,s}\}_{s=0}^{S}$
 1: $\mathbf{z}_0 = \tilde{\mathbf{x}}^0 \leftarrow \mathbf{x}_0, m \leftarrow Bn, \alpha_{1,s} \leftarrow 1 - \alpha_{2,s} - \alpha_{3,s}$;
 2: for $s \leftarrow 0$ to $S$ do
 3:     $\bar{L}_s \leftarrow \frac{L}{B\alpha_{3,s}} + L_B$;
 4:     $\eta_s \leftarrow \frac{1}{\bar{L}_s \alpha_{2,s} B}, \theta_s \leftarrow 1 + \frac{\mu}{\bar{L}_s B^2 \alpha_{2,s} + (B-1)\mu}$;
 5:     $\tilde{\nabla}^s \leftarrow \nabla \mathbf{F}(\tilde{\mathbf{x}}^s)$;
 6:     for $j \leftarrow 1$ to $m$ do
 7:         $k \leftarrow sm + j$;
 8:         $\mathbf{y}_k \leftarrow \alpha_{1,s}\mathbf{x}_{k-1} + \alpha_{2,s}\mathbf{z}_{k-1} + \alpha_{3,s}\tilde{\mathbf{x}}^s$;
 9:         sample mini batch $I$ of size $b$ and feature block $l$;
10:         $[\mathbf{v}_k]_l \leftarrow [\tilde{\nabla}^s]_l + \frac{1}{b}\sum_{i\in I}\left([\nabla f_i(\mathbf{y}_k)]_l - [\nabla f_i(\tilde{\mathbf{x}}^s)]_l\right)$;
11:         $[\mathbf{z}_k]_l \leftarrow \text{prox}_{\eta_s \mathbf{P}_l}([\mathbf{z}_{k-1}]_l - \eta_s[\mathbf{v}_k]_l)$, $[\mathbf{z}_k]_{\backslash l} \leftarrow [\mathbf{z}_{k-1}]_{\backslash l}$;
12:         $\mathbf{x}_k \leftarrow \mathbf{y}_k + \alpha_{2,s}B(\mathbf{z}_k - \mathbf{z}_{k-1})$;
13:     end for
14:     sample $\sigma$ from $\{1, \ldots, m\}$ with probability $\frac{\theta_s^{\sigma-1}}{\sum_{i=0}^{m-1}\theta_s^{i-1}}$;
15:     $\tilde{\mathbf{x}}^{s+1} \leftarrow \mathbf{x}_{sm+\sigma}$;
16: end for
```

Momenta: In sharp contrast to existing doubly stochastic algorithms, *two coupling steps* are added in ADSG to accelerate the convergence. In line 8, $\mathbf{y}_k$ is constructed as the convex combination of $\mathbf{x}_{k-1}$, $\mathbf{z}_{k-1}$, and the snapshot point $\tilde{\mathbf{x}}^s$. Here, $\mathbf{z}_{k-1}$ acts as a *historical momentum* that adds weight to the previous stochastic gradient $\{\mathbf{v}_t\}_{t\leq k}$ (note that $\mathbf{z}_k$ is simply the linear combination of all $\{\mathbf{v}_t\}_{t\leq k}$ when $\mathbf{P} \equiv 0$). This historical momentum is used in many deterministic accelerated methods, e.g. [Beck and Teboulle, 2009]. $\tilde{\mathbf{x}}^s$ serves as a *negative momentum* that ensures the "gradient" variable $\mathbf{y}_k$ not drifting away from $\tilde{\mathbf{x}}^s$ and prevents the variance introduced by the randomness from surging. Such negative momentum is recently proposed in [Allen-Zhu, 2016], but with a different weight $\alpha_{3,s} = \frac{1}{2}$. Note that such weight is crucial to the convergence, see Theorem 1 and 2. In line 12, we have $\mathbb{E}_l \mathbf{x}_k = \alpha_1 \mathbf{x}_{k-1} + \alpha_2 \mathbf{z}_{k-1} + \alpha_3 \tilde{\mathbf{x}}^s + \alpha_2(\tilde{\mathbf{z}}_k - \mathbf{z}_{k-1})$, where $\tilde{\mathbf{z}}_k = \text{prox}_{\eta_s \mathbf{P}}(\mathbf{z}_{k-1} - \eta_s \mathbf{v}_k)$. Hence, $\alpha_{2,s}B(\mathbf{z}_k - \mathbf{z}_{k-1})$, in expectation, adds extra weight on the most recent progress $\tilde{\mathbf{z}}_k - \mathbf{z}_{k-1}$ and is called *momentum in expectation* for this reason. These three momenta are the key to the accelerated convergence of ADSG.

4 Convergence Analysis

In this section, we present the convergence rate of ADSG in both strongly convex case and general convex case.

4.1 Strongly Convex Case

If $\kappa = \mathcal{O}(n)$, we can see from Table 1 that existing doubly stochastic algorithm like MRBCD requires only $\mathcal{O}(\log 1/\epsilon)$ passes over the whole dataset to achieve an ϵ-accurate solution. However, facing ill-conditioned problems where $\kappa > n^2$, the performance of such method decays faster than the deterministic APG method due to its linear dependence on the condition number κ. The following theorem shows that ADSG enjoys an accelerated convergence rate and depends only on $\sqrt{\kappa}$.

Theorem 1. *Suppose Assumption I-III are satisfied. Set* $\alpha_{2,s} = \frac{1}{2B}\min\{1, \sqrt{\frac{n}{\kappa}}\}, \alpha_{3,s} = \frac{1}{2B}$ *and set the mini batch size to be* 1. *If* $\kappa > 8B$, *for* $s \geq 0$, *we have*

$$\mathbb{E}\mathbf{F}^{\mathbf{P}}(\tilde{\mathbf{x}}^s) - \mathbf{F}^{\mathbf{P}}(\mathbf{x}_*) \leq \mathcal{O}(1)\min\{\frac{9}{8}, (1+\sqrt{\frac{n}{\kappa}})\}^{-s}(\mathbf{F}^{\mathbf{P}}(\mathbf{x}_0) - \mathbf{F}^{\mathbf{P}}(\mathbf{x}_*)),$$

and therefore ADSG takes $\mathcal{O}((1+\sqrt{\frac{\kappa}{n}})\log\frac{\epsilon_0}{\epsilon})$ *outer loops to achieve an* ϵ*-accurate solution.*

4.2 General Convex Case

We present both direct and indirect methods to solve Problem (1) in the general convex case. We make the assumption that the optimal $\mathbf{x}_*$ is bounded $\|\mathbf{x}_*\|^2 \leq D$, which is common in the literature.

Indirect Method: One option is to obtain an $\epsilon/2$-accurate solution $\bar{\mathbf{x}}$ to the following strongly convex, but perturbed stand-in of Problem (1):

$$\min_{\mathbf{x}\in\mathbb{R}^d} \mathbf{F}^{\mathbf{P}}_{\epsilon}(\mathbf{x}) \overset{\text{def}}{=} \mathbf{F}(\mathbf{x}) + \mathbf{P}(\mathbf{x}) + \frac{\epsilon}{2D}\|\mathbf{x}\|^2. \tag{2}$$

Such strategy is reasonable in that $\frac{\epsilon}{2} \geq \mathbf{F}^{\mathbf{P}}_{\epsilon}(\bar{\mathbf{x}}) - \mathbf{F}^{\mathbf{P}}_{\epsilon}(\mathbf{x}_*) = \mathbf{F}^{\mathbf{P}}(\bar{\mathbf{x}}) - \mathbf{F}^{\mathbf{P}}(\mathbf{x}_*) + \frac{\epsilon}{2D}\|\bar{\mathbf{x}}\|^2 - \frac{\epsilon}{2D}\|\mathbf{x}_*\|^2 \Rightarrow \mathbf{F}^{\mathbf{P}}(\bar{\mathbf{x}}) - \mathbf{F}^{\mathbf{P}}(\mathbf{x}_*) \leq \epsilon$, and hence $\bar{\mathbf{x}}$ is an ϵ-accurate solution to Problem (1). Furthermore, since the condition number of Problem (2) is $\frac{DL}{\epsilon}$, Theorem 1 implies the following corollary.

Corollary 1. *Suppose the optimal point* $\mathbf{x}_*$ *is bounded* $\|\mathbf{x}_*\|^2 \leq D$ *and suppose* $\bar{\mathbf{x}}$ *is an* $\epsilon/2$*-accurate solution to Problem (2). Then* $\bar{\mathbf{x}}$ *is an* ϵ*-accurate solution to Problem (1). Further, if Assumption I and II are satisfied, then it takes* $\mathcal{O}((1+\sqrt{\frac{LD}{n\epsilon}})\log\frac{\epsilon_0}{\epsilon})$ *outer loops to compute* $\bar{\mathbf{x}}$ *in ADSG.*

Direct Method: With different parameter setting, ADSG solves Problem (1) explicitly in general convex case, yielding a direct method with accelerated convergence rate. In this way, we avoid the extra $\log\frac{\epsilon_0}{\epsilon}$ factor in the indirect method.

Theorem 2. *Suppose Assumption I and II are satisfied. Set* $\alpha_{2,s} = \frac{2}{s+4B}, \alpha_{3,s} = \frac{1}{2B}$, *for* $s \geq 0$, *we have*

$$\mathbf{F}^{\mathbf{P}}(\tilde{\mathbf{x}}^s) - \mathbf{F}^{\mathbf{P}}(\mathbf{x}_*) \leq \mathcal{O}(1)\frac{B^2}{s^2}(\mathbf{F}^{\mathbf{P}}(\mathbf{x}_0) - \mathbf{F}^{\mathbf{P}}(\mathbf{x}_*) + \frac{L}{n}\|\mathbf{x}_0 - \mathbf{x}_*\|^2),$$

and therefore ADSG takes $\mathcal{O}(B\sqrt{\frac{\epsilon_0}{\epsilon} + \frac{LD}{n\epsilon}})$ *outer loops to achieve an* ϵ*-accurate solution.*

The extra B in Theorem 2 leaves room for improvement. The following theorem shows the dependence on B can be alleviated when $\mathbf{P} \equiv 0$.

Theorem 3. *Suppose Assumption I and II are satisfied and* $\mathbf{P} \equiv 0$. *Denote* $L' = L + BL_B$. *Setting* $\alpha_{2,s} = \frac{2}{s+4}, \alpha_{3,s} = \frac{1}{2}$, *and* $\alpha_{1,s} = 1 - \alpha_{2,s} - \alpha_{3,s}$, *then for* $s \geq 0$,

$$\mathbf{F}^{\mathbf{P}}(\tilde{\mathbf{x}}^s) - \mathbf{F}^{\mathbf{P}}(\mathbf{x}_*) \leq \mathcal{O}(1)\frac{1}{s^2}(\mathbf{F}^{\mathbf{P}}(\mathbf{x}_0) - \mathbf{F}^{\mathbf{P}}(\mathbf{x}_*) + \frac{L'}{n}\|\mathbf{x}_0 - \mathbf{x}_*\|^2).$$

Therefore, ADSG takes $\mathcal{O}(\sqrt{\frac{\epsilon_0}{\epsilon} + \frac{L'D}{n\epsilon}})$ *outer loops to achieve an* ϵ*-accurate solution.*

4.3 Proof for Strongly Convex Case

In this section, we give two key lemmas for our analysis. The full proofs of Theorem (1-3) are deferred to the long version of this paper.

The idea of the first lemma is to express $\mathbf{x}_k$ as the convex combination of $\{\tilde{\mathbf{x}}^i\}_{i=0}^s$ and $\{\mathbf{z}_l\}_{l=0}^k$.

Lemma 1. *In Algorithm I, by setting* $\alpha_{2,0} = \alpha_{3,0} = 1/2B$, *for* $k = sm + j \geq 1$, *we have*

$$\mathbf{x}_k = \sum_{i=0}^{s-1}\lambda_k^i\tilde{\mathbf{x}}^i + \beta_j^s\tilde{\mathbf{x}}^s + \sum_{l=0}^{k}\gamma_k^l\mathbf{z}_l, \tag{3}$$

where $\gamma_0^0 = 1$, $\gamma_1^0 = \frac{1}{2} - \frac{1}{2B}$, $\gamma_1^1 = \frac{1}{2}$, $\beta_0^0 = 0$, $\beta_0^s = \alpha_{3,s}$, $\lambda_{(s+1)m}^s = \beta_m^s$, $\lambda_{k+1}^i = \alpha_{1,s}\lambda_k^i$,

$$\gamma_{k+1}^l = \begin{cases} \alpha_{1,s}\gamma_k^l, & l = 0,\dots,k-1, \\ B\alpha_{1,s}\alpha_{2,s} + (1-B)\alpha_{2,s}, & l = k, \\ B\alpha_{2,s}, & l = k+1, \end{cases} \tag{4}$$

and

$$\beta_{j+1}^s = \alpha_{1,s}\beta_j^s + \alpha_{3,s}. \tag{5}$$

Additionally, we have $\sum_{i=0}^{s-1}\lambda_k^i + \beta_j^s + \sum_{l=0}^{k}\gamma_k^l = 1$. *If all* $\alpha_{1,s} \geq \frac{B-1}{B}$, *then each entry in this sum is non-negative for all* $k \geq 1$, *i.e.* $\mathbf{x}_k$ *is a convex combination of* $\{\tilde{\mathbf{x}}^i\}_{i=0}^s$ *and* $\{\mathbf{z}_l\}_{l=0}^k$.

The second lemma analyzes ADSG in one iteration. Before proceeding, we first define a few terms: (1) $\hat{\mathbf{P}}(\mathbf{x}_k) \overset{\text{def}}{=} \sum_{i=0}^{s-1}\lambda_k^i\mathbf{P}(\tilde{\mathbf{x}}^i) + \beta_j^s\mathbf{P}(\tilde{\mathbf{x}}^s) + \sum_{l=0}^{k}\gamma_k^l\mathbf{P}(\mathbf{z}_l) \geq \mathbf{P}(\mathbf{x}_k)$, where the inequality uses the convexity of $\mathbf{P}$; (2) $d(\mathbf{x}_k) \overset{\text{def}}{=} \mathbf{F}^{\mathbf{P}}(\mathbf{x}_k) - \mathbf{F}^{\mathbf{P}}(\mathbf{x}_*)$; (3) $\tilde{d}^s \overset{\text{def}}{=} \mathbf{F}^{\mathbf{P}}(\tilde{\mathbf{x}}^s) - \mathbf{F}^{\mathbf{P}}(\mathbf{x}_*)$; and (4) $d_k \overset{\text{def}}{=} (\mathbf{F}(\mathbf{x}_k) + \hat{\mathbf{P}}(\mathbf{x}_k)) - \mathbf{F}^{\mathbf{P}}(\mathbf{x}_*)$. Additionally, we have $0 \leq d(\mathbf{x}_k) \leq d_k$ and $d(\mathbf{x}_0) = d_0$.

Lemma 2. *In the* s^{th} *epoch of Algorithm 1, we have*

$$\mathbb{E}_{i,l}\hat{d}_j + \theta_s\mathbb{E}_{i,l}A_j \leq \alpha_{3,s}\tilde{d}^s + \alpha_{1,s}\hat{d}_{j-1} + A_{j-1}, \tag{6}$$

with $\rho_s = \alpha_{2,s}^2B^2\bar{L}_s + (B-1)\mu\alpha_{2,s}$, $\theta_s = 1 + \frac{\mu\alpha_{2,s}}{\rho_s}$, $A_j = \rho_s\|\mathbf{x}^* - \mathbf{z}_{sm+j}\|^2/2$, *and* $\hat{d}_j = d_{sm+j}$.

5 Efficient Implementation

While ADSG has an accelerated convergence rate, naively implementing Algorithm 1 requires $\mathcal{O}(d)$ computation in each inner loop due to the two *coupling steps*, which compromises the low per-iteration complexity enjoyed by RBCD type methods. Such quandary strikes all existing accelerated RBCD algorithm [Lin *et al.*, 2015b; Nesterov, 2012; Fercoq and Richtárik, 2015; Lee and Sidford, 2013]. To bypass this dilemma, we cast Algorithm 1 in an equivalent but more practical form, ADSG II, with the inner loop complexity reduced to $\mathcal{O}(\Omega)$. ADSG II uses three auxiliary functions $\{\bar{\mathbf{x}}_k, \bar{\mathbf{y}}_k, \bar{\mathbf{z}}_k\}$ which are defined as

$$\begin{aligned} \bar{\mathbf{y}}_k &= \beta_{j-1}\mathbf{u}_{j-1}^s + \gamma_s\hat{\mathbf{z}}_{j-1}^s + \dot{\mathbf{x}}^s, && \text{Line 10,} \\ \bar{\mathbf{z}}_k &= \hat{\mathbf{z}}_j^s + \dot{\mathbf{x}}^s, && \text{Line 11 and 16,} \\ \bar{\mathbf{x}}_k &= \beta_{j-1}\mathbf{u}_j^s + \gamma_s\hat{\mathbf{z}}_j^s + \dot{\mathbf{x}}^s, && \text{Line 15 and 16,} \end{aligned}$$

with $k = sm + j$. The following proposition shows the equivalence between Algorithm 1 and 2.

Algorithm 2 ADSG II

```
Input: $n, \mathbf{x}_0, S, b, B, \mu, \{\alpha_{2,s}, \alpha_{3,s}\}_{s=0}^{S}$
 1: $\hat{\mathbf{z}}_0^0 \leftarrow \vec{0}, \xi = \dot{\mathbf{x}}^0 \leftarrow \mathbf{x}_0, k \leftarrow 0, m \leftarrow Bn, \alpha_{1,s} \leftarrow 1 - \alpha_{2,s} - \alpha_{3,s}$;
 2: for $s \leftarrow 0$ to $S$ do
 3:     $\gamma_s \leftarrow \frac{\alpha_{2,s}}{\alpha_{2,s}+\alpha_{3,s}}, \beta_{-1}^s \leftarrow 1, \beta_0^s \leftarrow \alpha_{1,s}, \bar{L}_s \leftarrow \frac{L}{B\alpha_{3,s}} + L_B$;
 4:     $\eta_s \leftarrow \frac{1}{\bar{L}_s \alpha_{2,s} B}, \theta_s \leftarrow 1 + \frac{\mu}{\bar{L}_s B^2 \alpha_{2,s} + (B-1)\mu}$;
 5:     $\dot{\nabla}^s \leftarrow \nabla f(\dot{\mathbf{x}}^s)$;
 6:     $\mathbf{u}_0^s \leftarrow \xi - \gamma_s \hat{\mathbf{z}}_0^s - \dot{\mathbf{x}}^s$;
 7:     for $j \leftarrow 1$ to $m$ do
 8:         $k \leftarrow sm + j$;
 9:         sample mini batch $I$ of size $b$ and feature block $l$;
10:         $[\dot{\mathbf{v}}_k]_l \leftarrow [\dot{\nabla}^s]_l + \frac{1}{b}\sum_{i \in I} \left([\nabla f_i(\bar{\mathbf{y}}_k)]_l - [\nabla f_i(\tilde{\mathbf{x}}^s)]_l\right)$;
11:         $[\hat{\mathbf{z}}_j^s]_l \leftarrow \mathrm{prox}_{\eta_s \mathbf{P}_l}([\bar{\mathbf{z}}_k]_l - \eta_s [\dot{\mathbf{v}}_k]_l) - [\dot{\mathbf{x}}^s]_l$, $[\hat{\mathbf{z}}_j^s]_{\backslash l} \leftarrow [\hat{\mathbf{z}}_{j-1}^s]_{\backslash l}$;
12:         $\mathbf{u}_j^s \leftarrow \mathbf{u}_{j-1}^s + \frac{\alpha_{2,s}B - \gamma_s}{\beta_{j-1}^s}(\hat{\mathbf{z}}_j^s - \hat{\mathbf{z}}_{j-1}^s)$; $\beta_j^s \leftarrow \alpha_{1,s}\beta_{j-1}^s$;
13:     end for
14:     sample $\sigma$ from $\{1, \ldots, m\}$ with probability $\frac{\theta_s^{\sigma-1}}{\sum_{i=0}^{m-1} \theta_s^{i-1}}$;
15:     $\dot{\mathbf{x}}^{s+1} \leftarrow \bar{\mathbf{x}}_{sm+\sigma}$;
16:     $\hat{\mathbf{z}}_0^{s+1} \leftarrow \bar{\mathbf{z}}_k - \dot{\mathbf{x}}^{s+1}$, $\xi = \bar{\mathbf{x}}_k$;
17: end for
```

Proposition 1. *If Algorithm 1 has the same input as Algorithm 2, its iterates $\mathbf{x}_k$, $\mathbf{y}_k$, and $\mathbf{z}_k$ equal to $\bar{\mathbf{x}}_k$, $\bar{\mathbf{y}}_k$, $\bar{\mathbf{z}}_k$ respectively for all k.*

Proof. First, we prove that if at the beginning of the s^{th} epoch, $\bar{\mathbf{z}}_{sm} = \mathbf{z}_{sm}$, $\bar{\mathbf{x}}_{sm} = \mathbf{x}_{sm}$, and $\dot{\mathbf{x}}^s = \tilde{\mathbf{x}}^s$ stand, then the proposition stand in the following iterations in that epoch. We prove with induction. Assume that the equivalence holds till the $(k-1)^{th}$ iteration. In the k^{th} iteration, for $\bar{\mathbf{y}}_k$ we have

$$\begin{aligned}
\mathbf{y}_k &= \alpha_{1,s}\mathbf{x}_{k-1} + \alpha_{2,s}\mathbf{z}_{k-1} + \alpha_{3,s}\tilde{\mathbf{x}}^s \\
&= \alpha_{1,s}\bar{\mathbf{x}}_{k-1} + \alpha_{2,s}\bar{\mathbf{z}}_{k-1} + \alpha_{3,s}\dot{\mathbf{x}}^s \\
&= \alpha_{1,s}(\beta_{j-2}^s \mathbf{u}_{j-1}^s + \gamma_s \hat{\mathbf{z}}_{j-1}^s) + \alpha_{2,s}\hat{\mathbf{z}}_{j-1}^s + \dot{\mathbf{x}}^s \\
&= \beta_{j-1}^s \mathbf{u}_{j-1}^s + \gamma_s \hat{\mathbf{z}}_{j-1}^s + \dot{\mathbf{x}}^s = \bar{\mathbf{y}}_k.
\end{aligned}$$

since $\alpha_{1,s}\beta_{j-2}^s = \beta_{j-1}^s$ and $\alpha_{1,s}\gamma_s + \alpha_{2,s} = \gamma_s$. For $\bar{\mathbf{z}}_k$, by induction we have $[\bar{\mathbf{z}}_k]_{\backslash l} = [\bar{\mathbf{z}}_{k-1}]_{\backslash l} = [\mathbf{z}_{k-1}]_{\backslash l} = [\mathbf{z}_k]_{\backslash l}$. Additionally, since $\dot{\nabla}^s = \tilde{\nabla}^s$ and $\mathbf{y}_k = \bar{\mathbf{y}}_k$, we have $[\mathbf{v}_k]_l = [\dot{\mathbf{v}}_k]_l$ and thus $[\bar{\mathbf{z}}_k]_l = [\hat{\mathbf{z}}_j^s + \dot{\mathbf{x}}^s]_l = \mathrm{prox}_{\eta \mathbf{P}_l}([\bar{\mathbf{z}}_{k-1} - \eta \dot{\mathbf{v}}_k]_l) = \mathrm{prox}_{\eta \mathbf{P}_l}([\mathbf{z}_{k-1} - \eta \mathbf{v}_k]_l) = [\mathbf{z}_k]_l$. For $\bar{\mathbf{x}}_k$, we have

$$\begin{aligned}
\mathbf{x}_k &= \bar{\mathbf{y}}_k + \alpha_{2,s}B(\hat{\mathbf{z}}_j^s - \hat{\mathbf{z}}_{j-1}^s) \\
&= \beta_{j-1}^s \mathbf{u}_{j-1}^s + \gamma_s \hat{\mathbf{z}}_{j-1}^s + \alpha_{2,s}B(\hat{\mathbf{z}}_j^s - \hat{\mathbf{z}}_{j-1}^s) + \dot{\mathbf{x}}^s \\
&= \beta_{j-1}^s \mathbf{u}_j^s + \gamma_s \hat{\mathbf{z}}_j^s + \dot{\mathbf{x}}^s = \bar{\mathbf{x}}_k
\end{aligned}$$

by the updating rule of $\mathbf{u}_k$ in line 12 in ADSG II.

We then show that at the beginning of each epoch $\bar{\mathbf{z}}_{sm} = \mathbf{z}_{sm}$, $\bar{\mathbf{x}}_{sm} = \mathbf{x}_{sm}$, and $\dot{\mathbf{x}}^s = \tilde{\mathbf{x}}^s$ stand. For $\mathbf{z}_0$, we have $\bar{\mathbf{z}}_0 = \mathbf{z}_0$ from the initialization. For $\mathbf{z}_{sm}, s \geq 1$, we have $\mathbf{z}_{sm} = \bar{\mathbf{z}}_{sm} = \hat{\mathbf{z}}_0^s + \dot{\mathbf{x}}^s = \bar{\mathbf{z}}_{sm}$, where the first equation is from the induction in previous epoch, and the second equation is from the definition of $\hat{\mathbf{z}}_0^s$ in line 16 in ADSG II. For $\mathbf{x}_0$, we clearly have $\mathbf{x}_0 = \bar{\mathbf{x}}_0$ by the initialization. For $\mathbf{x}_{sm}, s \geq 1$, we have $\mathbf{x}_{sm} = \xi = \beta_{-1}^s \mathbf{u}_0^s + \gamma_s \hat{\mathbf{z}}_0^s + \dot{\mathbf{x}}^s = \bar{\mathbf{x}}_{sm}$, where the first equation is from the induction in previous epoch, and the second equation is from line 6 in ADSG II. $\dot{\mathbf{x}}^s = \tilde{\mathbf{x}}^s$ because $\mathbf{x}_k = \bar{\mathbf{x}}_k$ for all k in that epoch. Thus we have the result. □

5.1 Overall Computational Complexity

We discuss the detailed implementation of ADSG II when solving ERM problems. For simplicity, the mini batch size b is set to 1. In line 10, $[\nabla f_i(\bar{\mathbf{y}}_k)]_l = \nabla \phi_i(\mathbf{a}_i^\top \bar{\mathbf{y}}_k)[\mathbf{a}_i]_l$, where we compute each term in $\mathbf{a}_i^\top \bar{\mathbf{y}}_k = \beta_{j-1}\mathbf{a}_i^\top \mathbf{u}_{j-1}^s + \gamma_s \mathbf{a}_i^\top \hat{\mathbf{z}}_{j-1}^s + \mathbf{a}_i^\top \dot{\mathbf{x}}^s$ separately. This can be done with $\mathcal{O}(\rho d)$. Line 12 takes $\mathcal{O}(\Omega)$ because only the l^{th} block is updated. Consequently, we have $\mathcal{O}(\rho d + \Omega)$ from every inner loop and the per-epoch complexity of Algorithm 2 is $\mathcal{O}(Bn\Omega + Bn\rho d)$. When ρ is small, Ω dominates ρd and it becomes $\mathcal{O}(dn)$.

Combining the above per-epoch complexity analysis and the convergence rate, the overall computational complexity of ADSG is $\mathcal{O}(d(n + \sqrt{n/\kappa}) \log 1/\epsilon)$ in strongly convex case, and $\mathcal{O}(d(n + \sqrt{nLD/\epsilon}) \log 1/\epsilon)$ in general convex case.

6 Experiments

In this section, we conduct several experiments to show the efficiency of ADSG on huge-scale real problems. We use $\frac{\mathbf{F}^{\mathbf{P}}(\mathbf{x}^t) - \mathbf{F}^{\mathbf{P}}(\mathbf{x}^*)}{\mathbf{F}^{\mathbf{P}}(\mathbf{x}_0) - \mathbf{F}^{\mathbf{P}}(\mathbf{x}^*)}$ to measure the suboptimality.

Problems We conduct experiments of three ERM problems, namely l_1-logistic regression, $l_1 l_2$-logistic regression, and ridge regression. Four large scale datasets from LibSVM [Chang and Lin, 2011] are used: kdd2010-raw, avazu-app, new20.binary, and url-combined. Their statistics are given in Table 2.

Algorithms Katyusha [Allen-Zhu, 2016], MRBCD [Zhao *et al.*, 2014] (ASBCD has similar performance), and SVRG [Johnson and Zhang, 2013] are included for comparison. MRBCD and ADSG adopts the same block parameter B. All methods use the same mini batch size b. We use the default inner loop count described in the original paper for SVRG, MRBCD, and Katyusha. We tune the step size to give the best performance. For SVRG and MRBCD, it is usually $1/L$.

6.1 l_1-Logistic Regression

In this problem, we set $f_i(\mathbf{x}) = \log(1 + \exp(-y_i \mathbf{a}_i^\top \mathbf{x}))$ and set $\mathbf{P}(\mathbf{x}) = \lambda\|\mathbf{x}\|_1$, where $\{\mathbf{a}_i, y_i\}_{i=1}^n$ are data points. We present suboptimality vs. Evaluated Partial Gradients (EPG) and suboptimality vs. time in the first two rows of Figure 1, along with the parameter λ used in experiments. The result shows that (i) Katyusha and ADSG have superior convergence rate over non-accelerated SVRG and MRBCD, (ii) ADSG and MRBCD, as doubly stochastic methods, enjoy a better time

Table 2: Statistics of datasets.

Dataset	n	d	sparsity
news20-binary	19,996	1,355,191	0.0336%
kdd2010-raw	19,264,097	1,129,522	0.0008%
avazu-app	14,596,137	999,990	0.0015%
url-combined	2,396,130	3,231,961	0.0036%

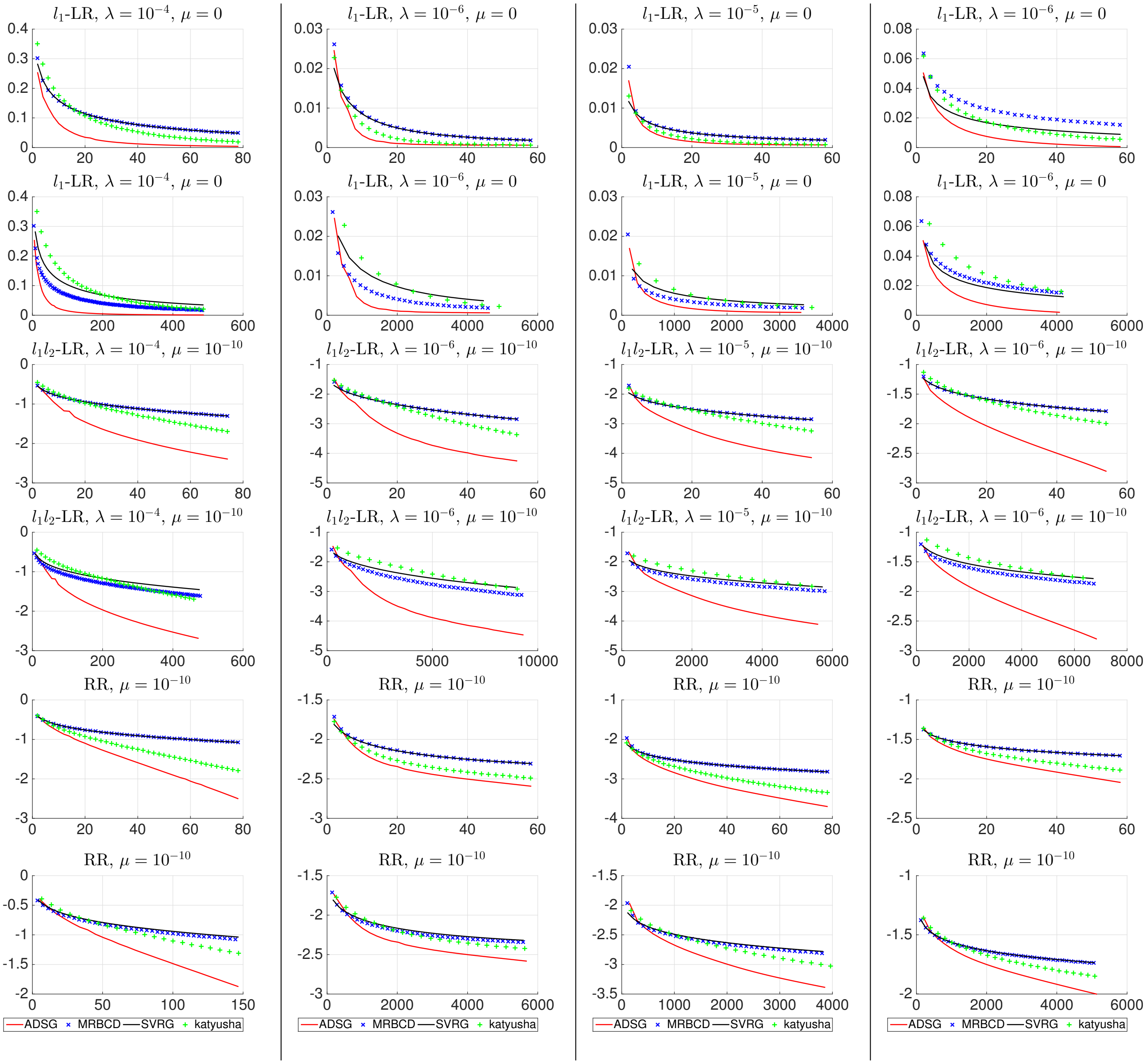

Figure 1: Columns, from left to right, are the results on news20-binary, kddb-raw, avazu-app, and url-combined. The y-axis is the (log) suboptimality. The x-axes in the odd rows are EPG/dn and the x-axes in the even rows are time.

efficiency than Katyusha and SVRG, and (iii) ADSG has the best performance among all competitors.

6.2 l_1l_2-Logistic Regression

$f_i(\mathbf{x})$ is $\log(1+\exp(-y_i\mathbf{a}_i^\top\mathbf{x}))$ and $\mathbf{P}(\mathbf{x})$ is $\frac{\mu}{2}\|\mathbf{x}\|^2+\lambda\|\mathbf{x}\|_1$ in l_1l_2-logistic regression. The log suboptimality vs. EPG and log suboptimality vs. time plots are given in the third and fourth rows of Figure 1. We observe similar phenomenon here as in the previous experiment.

6.3 Ridge Regression

We set $f_i(\mathbf{x}) = \frac{1}{2}\|\mathbf{a}_i^\top\mathbf{x} - \mathbf{y}_i\|^2$ and $\mathbf{P}(\mathbf{x}) = \frac{\mu}{2}\|\mathbf{x}\|^2$. In the last two rows of Figure 1, we give the log suboptimality vs. EPG and log suboptimality vs. time plots. ADSG shows exceptional computational efficiency.

7 Conclusion

An accelerated doubly stochastic algorithm called ADSG is proposed in this paper. We give its convergence analyses, and compare our algorithm to the state-of-the-art in large scale ERM problems. The result is promising.

Acknowledgments

This work is partially supported by National Natural Science Foundation of China (Grant No: 61472347, 61672376, and 61672449)

References

[Allen-Zhu and Orecchia, 2014] Zeyuan Allen-Zhu and Lorenzo Orecchia. Linear coupling: An ultimate unification of gradient and mirror descent. *arXiv preprint arXiv:1407.1537*, 2014.

[Allen-Zhu, 2016] Zeyuan Allen-Zhu. Katyusha: Accelerated variance reduction for faster sgd. *arXiv preprint arXiv:1603.05953*, 2016.

[Beck and Teboulle, 2009] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2009.

[Chang and Lin, 2011] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2011.

[Defazio *et al.*, 2014] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *NIPS*, 2014.

[Fercoq and Richtárik, 2015] Olivier Fercoq and Peter Richtárik. Accelerated, parallel, and proximal coordinate descent. *SIAM Journal on Optimization*, 2015.

[Friedman *et al.*, 2001] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*. 2001.

[Frostig *et al.*, 2015] Roy Frostig, Rong Ge, Sham M Kakade, and Aaron Sidford. Un-regularizing: approximate proximal point and faster stochastic algorithms for empirical risk minimization. In *ICML*, 2015.

[Hu *et al.*, 2009] Chonghai Hu, Weike Pan, and James T Kwok. Accelerated gradient methods for stochastic optimization and online learning. In *Advances in Neural Information Processing Systems*, 2009.

[Johnson and Zhang, 2013] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *NIPS*, 2013.

[Lan, 2012] Guanghui Lan. An optimal method for stochastic composite optimization. *Mathematical Programming*, 2012.

[Lee and Sidford, 2013] Yin Tat Lee and Aaron Sidford. Efficient accelerated coordinate descent methods and faster algorithms for solving linear systems. In *FOCS*, 2013.

[Lin *et al.*, 2015a] Hongzhou Lin, Julien Mairal, and Zaid Harchaoui. A universal catalyst for first-order optimization. In *NIPS*, 2015.

[Lin *et al.*, 2015b] 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*, 2015.

[Nesterov, 1983] Yurii Nesterov. A method of solving a convex programming problem with convergence rate o (1/k2). In *Soviet Mathematics Doklady*, 1983.

[Nesterov, 1998] Yu Nesterov. Introductory lectures on convex programming volume i: Basic course. 1998.

[Nesterov, 2012] Yu Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 2012.

[Nitanda, 2014] Atsushi Nitanda. Stochastic proximal gradient descent with acceleration techniques. In *NIPS*, 2014.

[Polyak, 1964] Boris T Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 1964.

[Richtárik and Takáč,] Peter Richtárik and Martin Takáč. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Mathematical Programming*.

[Schmidt *et al.*, 2013] Mark Schmidt, Nicolas Le Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient. *arXiv preprint arXiv:1309.2388*, 2013.

[Shalev-Shwartz and Zhang, 2014] Shai Shalev-Shwartz and Tong Zhang. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. In *ICML*, 2014.

[Simon *et al.*, 2013] Noah Simon, Jerome Friedman, Trevor Hastie, and Robert Tibshirani. A sparse-group lasso. *Journal of Computational and Graphical Statistics*, 2013.

[Tibshirani, 1996] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 1996.

[Woodworth and Srebro, 2016] Blake E Woodworth and Nati Srebro. Tight complexity bounds for optimizing composite objectives. In *NIPS*, 2016.

[Wright, 2015] Stephen J Wright. Coordinate descent algorithms. *Mathematical Programming*, 2015.

[Zhang and Gu, 2016] Aston Zhang and Quanquan Gu. Accelerated stochastic block coordinate descent with optimal sampling. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[Zhang and Xiao, 2015] Yuchen Zhang and Lin Xiao. Stochastic primal-dual coordinate method for regularized empirical risk minimization. In *ICML*, 2015.

[Zhao *et al.*, 2014] Tuo Zhao, Mo Yu, Yiming Wang, Raman Arora, and Han Liu. Accelerated mini-batch randomized block coordinate descent method. In *NIPS*, 2014.