A Simple Stochastic Variance Reduced Algorithm with Fast Convergence Rates

Kaiwen Zhou [1] **Fanhua Shang** [2] **James Cheng** [1]

Abstract

Recent years have witnessed exciting progress in the study of stochastic variance reduced gradient methods (e.g., SVRG, SAGA), their accelerated variants (e.g, Katyusha) and their extensions in many different settings (e.g., online, sparse, asynchronous, distributed). Among them, accelerated methods enjoy improved convergence rates but have complex coupling structures, which makes them hard to be extended to more settings (e.g., sparse and asynchronous) due to the existence of perturbation. In this paper, we introduce a simple stochastic variance reduced algorithm (MiG), which enjoys the best-known convergence rates for both strongly convex and non-strongly convex problems. Moreover, we also present its efficient sparse and asynchronous variants, and theoretically analyze its convergence rates in these settings. Finally, extensive experiments for various machine learning problems such as logistic regression are given to illustrate the practical improvement in both serial and asynchronous settings.

1. Introduction

In this paper, we consider the following convex optimization problem with a finite-sum structure, which is prevalent in machine learning and statistics such as regularized empirical risk minimization (ERM):

$$\min_{x\in\mathbb{R}^d} \left\{ F(x) \triangleq f(x) + g(x) \right\}, \qquad (1)$$

where $f(x)=\frac{1}{n}\sum_{i=1}^{n} f_i(x)$ is a finite average of n smooth convex function $f_i(x)$, and $g(x)$ is a relatively simple (but possibly non-differentiable) convex function.

For the strongly convex problem (1), traditional gradient descent (GD) yields a linear convergence rate but with a high per-iteration cost. As an alternative, SGD (Robbins & Monro, 1951) enjoys significantly lower per-iteration complexity than GD, i.e., $O(d)$ vs. $O(nd)$. However, due to the variance of random sampling, standard SGD usually obtains slow convergence and poor performance (Johnson & Zhang, 2013). Recently, many stochastic variance reduced methods (e.g., SAG (Roux et al., 2012), SDCA (Shalev-Shwartz & Zhang, 2013), SVRG (Johnson & Zhang, 2013), SAGA (Defazio et al., 2014), and their proximal variants, such as (Schmidt et al., 2017), (Shalev-Shwartz & Zhang, 2016), (Xiao & Zhang, 2014) and (Konečný et al., 2016)) have been proposed to solve Problem (1). All these methods enjoy low per-iteration complexities comparable with SGD, but with the help of certain variance reduction techniques, they obtain a linear convergence rate as GD. More accurately, these methods achieve an improved oracle complexity $\mathcal{O}((n+\kappa)\log(1/\epsilon))$[1], compared with $\mathcal{O}(n\sqrt{\kappa}\log(1/\epsilon))$ for accelerated deterministic methods (e.g., Nesterov's accelerated gradient descent (Nesterov, 2004)). In summary, these methods dramatically reduce the overall computational cost compared with deterministic methods in theory.

More recently, researchers have proposed accelerated stochastic variance reduced methods for Problem (1), which include Acc-Prox-SVRG (Nitanda, 2014), APCG (Lin et al., 2014), Catalyst (Lin et al., 2015), SPDC (Zhang & Xiao, 2015), point-SAGA (Defazio, 2016), and Katyusha (Allen-Zhu, 2017). For strongly convex problems, both Acc-Prox-SVRG (Nitanda, 2014) and Catalyst (Lin et al., 2015) make good use of the "*Nesterov's momentum*" in (Nesterov, 2004) and attain the corresponding oracle complexities $\mathcal{O}((n+b\sqrt{\kappa})\log(1/\epsilon))$ (with a sufficiently large mini-batch size b) and $\mathcal{O}((n+\sqrt{\kappa n})\log(\kappa)\log(1/\epsilon))$. APCG, SPDC, point-SAGA and Katyusha essentially achieve the best-known oracle complexity $\mathcal{O}((n+\sqrt{\kappa n})\log(1/\epsilon))$.

Inspired by emerging multi-core computer architectures, asynchronous variants of the above stochastic gradient methods have been proposed in recent years, e.g., Hogwild! (Recht et al., 2011), Lock-Free SVRG (Reddi et al., 2015), KroMagnon (Mania et al., 2017) and ASAGA (Leblond et al., 2017). Among them, KroMagnon and ASAGA (as the sparse and asynchronous variants of SVRG and SAGA)

[1]Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong [2]School of Artificial Intelligence, Xidian University, China. Correspondence to: Fanhua Shang <fhshang@xidian.edu.cn>.

Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018. Copyright 2018 by the author(s).

[1]We denote $\kappa \triangleq \frac{L}{\sigma}$ throughout the paper, known as the condition number of an L-smooth and σ-strongly convex function.

enjoy a fast linear convergence rate for strongly convex objectives. However, there still lacks a variant of accelerated algorithms in these settings.

The main issue for those accelerated algorithms is that most of their algorithm designs (e.g., (Allen-Zhu, 2017) and (Hien et al., 2017)) involve tracking at least two highly correlated coupling vectors[2] (in the inner loop). This kind of algorithm structure prevents us from deriving efficient (lock-free) asynchronous sparse variants for those algorithms. More critically, when the number of concurrent threads is large (e.g., 20 threads), the high perturbation (i.e., updates on shared variables from concurrent threads) may even destroy their convergence guarantees. This leads us to the key question we study in this paper:

Can we design an accelerated algorithm that keeps track of only one variable vector?

We answer this question by a simple stochastic variance reduced algorithm (MiG), which has the following features:

- **Simple.** The algorithm construction of MiG requires tracking only one variable vector in the inner loop, which means its computational overhead and memory overhead are exactly the same as SVRG (or Prox-SVRG (Xiao & Zhang, 2014)). This feature allows MiG to be extended to more strict settings such as the sparse and asynchronous settings. We theoretically analyze its variants in Section 4.
- **Theoretically Fast.** MiG achieves the best-known oracle complexity of $\mathcal{O}\big((n+\sqrt{\kappa n})\log{(1/\epsilon)}\big)$ for strongly convex problems. For non-strongly convex problems, MiG also achieves an optimal convergence rate $\mathcal{O}\big(1/T^2\big)$, where T is the total number of stochastic iterations. These rates keep up with those of Katyusha and are consistently faster than non-accelerated algorithms, e.g., SVRG and SAGA.
- **Practically Fast.** Due to its light-weighted algorithm structure, our experiments verify that the running time of MiG is shorter than its counterparts in the serial dense setting. In the sparse and asynchronous settings, MiG achieves significantly better performance than KroMagnon and ASAGA in terms of both gradient evaluations and running time.
- **Implementable.** Unlike many incremental gradient methods (e.g., SAGA), MiG does not require an additional gradient table which is not practical for large-scale problems. Our algorithm layout is similar to SVRG, which means that most existing techniques designed for SVRG (such as a distributed variant) can be modified for MiG without much effort.

[2]Here we refer to the number of variable vectors involved in one update.

Table 1. Comparison of different stochastic variance reduced algorithms. ("Complexity" is for strongly-convex problems. "Memory" is those used to store variables."S&A" refers to efficient (lock-free) Sparse & Asynchronous variant.)

Algorithm	Complexity	Memory	S&A
SVRG	$\mathcal{O}\big((n+\kappa)\log\frac{1}{\epsilon}\big)$	1 Vec.	✓
SAGA	$\mathcal{O}\big((n+\kappa)\log\frac{1}{\epsilon}\big)$	1 Vec. 1∇ Table.	✓
Katyusha	$\mathcal{O}\big((n+\sqrt{\kappa n})\log\frac{1}{\epsilon}\big)$	2 Vec.	×
MiG	$\mathcal{O}\big((n+\sqrt{\kappa n})\log\frac{1}{\epsilon}\big)$	1 Vec.	✓

We summarize some properties of the existing methods and MiG in Table 1.

2. Notations

We mainly consider Problem (1) in standard Euclidean space with the Euclidean norm denoted by $\|\cdot\|$. We use $\mathbb{E}$ to denote that the expectation is taken with respect to all randomness in one epoch. To further categorize the objective functions, we define that a convex function $f:\mathbb{R}^n\rightarrow\mathbb{R}$ is said to be L-smooth if for all $x,y\in\mathbb{R}^d$, it holds that

$$f(x)\leq f(y)+\langle\nabla f(y),x-y\rangle+\frac{L}{2}\|x-y\|^2, \qquad (2)$$

and σ-strongly convex if for all $x,y\in\mathbb{R}^d$,

$$f(x)\geq f(y)+\langle\mathcal{G},x-y\rangle+\frac{\sigma}{2}\|x-y\|^2, \qquad (3)$$

where $\mathcal{G}\in\partial f(y)$, the set of sub-gradient of f at y. If f is differentiable, we replace $\mathcal{G}\in\partial f(y)$ with $\mathcal{G}=\nabla f(y)$. Then we make the following assumptions to categorize Problem (1):

Assumption 1 (Strongly Convex)**.** *In Problem (1), each $f_i(\cdot)$[3] is L-smooth and convex, $g(\cdot)$ is σ-strongly convex.*

Assumption 2 (Non-strongly Convex)**.** *In Problem (1), each $f_i(\cdot)$ is L-smooth and convex, and $g(\cdot)$ is convex.*

3. A Simple Accelerated Algorithm

In this section, we introduce a simple accelerated stochastic algorithm (MiG) for both strongly convex and non-strongly convex problems.

3.1. MiG for Strongly Convex Objectives

We first consider Problem (1) that satisfies Assumption 1.

Now we formally introduce MiG in Algorithm 1. In order to

[3]In fact, if each $f_i(\cdot)$ is L-smooth, the averaged function $f(\cdot)$ is itself L-smooth — but probably with a smaller L. We keep using L as the smoothness constant for a consistent analysis.

Algorithm 1 MiG

```
1:  Input: Initial vector $x_0$, epoch length $m$, learning rate $\eta$, parameter $\theta$.
2:  $\tilde{x}_0 = x_0^1 = x_0, \omega = 1 + \eta\sigma$;
3:  for $s = 1 \ldots \mathcal{S}$ do
4:      $\mu_s = \nabla f(\tilde{x}_{s-1})$;
5:      for $j = 1 \ldots m$ do
6:          Sample $i_j$ uniformly in $\{1 \ldots n\}$;
7:          $y_{j-1} = \theta x_{j-1}^s + (1-\theta)\tilde{x}_{s-1}$; //temp variable $y$
8:          $\tilde{\nabla} = \nabla f_{i_j}(y_{j-1}) - \nabla f_{i_j}(\tilde{x}_{s-1}) + \mu_s$;
9:          $x_j^s = \arg\min_x \left\{\frac{1}{2\eta}\|x - x_{j-1}^s\|^2 + \langle\tilde{\nabla}, x\rangle + g(x)\right\}$;
10:     end for
11:     $\tilde{x}_s = \theta\big(\sum_{j=0}^{m-1}\omega^j\big)^{-1}\sum_{j=0}^{m-1}\omega^j x_{j+1}^s + (1-\theta)\tilde{x}_{s-1}$;
12:     $x_0^{s+1} = x_m^s$;
13: end for
14: return $\tilde{x}_{\mathcal{S}}$.
```

further illustrate some ideas behind the algorithm structure, we make the following remarks:

- *Temp variable y.* As we can see in Algorithm 1, y is a convex combination of x and $\tilde{x}$ with the parameter θ. So for implementation, we do not need to keep track of y in the whole inner loop. For the purpose of giving a clean proof, we mark y with iteration number j.
- *Fancy update for $\tilde{x}_s$.* One can easily verify that this update for $\tilde{x}_s$ is equivalent to using ω^j weighted averaged y_{j+1} to update $\tilde{x}_s$, which is written as: $\tilde{x}_s = \big(\sum_{j=0}^{m-1}\omega^j\big)^{-1}\sum_{j=0}^{m-1}\omega^j y_{j+1}$. Since we only keep track of x, we adopt this expended fancy update for $\tilde{x}_s$ — but it is still quite simple in implementation.
- *Choice of x_0^{s+1}.* In recent years, some existing stochastic algorithms such as (Zhang et al., 2013; Xiao & Zhang, 2014) choose to use $\tilde{x}_s$ as the initial vector for new epoch. For MiG, when using $\tilde{x}_s$, the overall oracle complexity will degenerate to a non-accelerated one for some ill-conditioned problems, which is $\mathcal{O}\big((n+\kappa)\log(1/\epsilon)\big)$. It is reported that even in practice, using the last iterate yields a better performance as discussed in (Allen-Zhu & Hazan, 2016b).

Next we give the convergence rate of MiG in terms of oracle complexity as follows (the proofs to all theorems in this paper are given in the Supplementary Material):

Theorem 1 (Strongly Convex)**.** *Let x^* be the optimal solution of Problem (1). If Assumption 1 holds, then by choosing $m = \Theta(n)$,* MiG *achieves an ϵ-additive error with the following oracle complexities in expectation:*

$$\begin{cases}\mathcal{O}\big(\sqrt{\kappa n}\log\frac{F(x_0)-F(x^*)}{\epsilon}\big), & \textit{if } \frac{m}{\kappa} \le \frac{3}{4},\\ \mathcal{O}\big(n\log\frac{F(x_0)-F(x^*)}{\epsilon}\big), & \textit{if } \frac{m}{\kappa} > \frac{3}{4}.\end{cases}$$

In other words, the overall oracle complexity of MiG *is* $\mathcal{O}\big((n+\sqrt{\kappa n})\log\frac{F(x_0)-F(x^*)}{\epsilon}\big)$.

Table 2. The theoretical settings of the parameters η and θ.

CONDITION	LEARNING RATE η	PARAMETER θ
$\frac{m}{\kappa} \le \frac{3}{4}$	$\sqrt{\frac{1}{3\sigma m L}}$	$\sqrt{\frac{m}{3\kappa}}$
$\frac{m}{\kappa} > \frac{3}{4}$	$\frac{2}{3L}$	$\frac{1}{2}$

This result implies that in the strongly convex setting, MiG enjoys the best-known oracle complexity for stochastic first-order algorithms, e.g., APCG, SPDC, and Katyusha. The theoretical suggestions[4] of the learning rate η and the parameter θ are shown in Table 2.

3.1.1. Comparison Between MiG And Related Methods

We carefully compare the algorithm structure of MiG with Katyusha, and find that MiG corresponds to a case of Katyusha, when $1-\tau_1-\tau_2=0$, and Option II in Katyusha is used. However, this setting is neither suggested nor analyzed in (Allen-Zhu, 2017), and thus without a convergence guarantee. In some sense, MiG can be regarded as a "simplified Katyusha", while this simplification does not hurt its oracle complexity. Since this simplification discards all the proximal gradient steps in Katyusha, MiG enjoys a lower memory overhead in practice and a cleaner proof in theory. Detailed comparison with Katyusha can be found in the Supplementary Material B.1.1.

MiG does not use any kind of "*Nesterov's Momentum*", which is used in some accelerated algorithms, e.g., Acc-Prox-SVRG (Nitanda, 2014) and Catalyst (Lin et al., 2015).

3.2. MiG for Non-strongly Convex Objectives

In this part, we consider Problem (1) when Assumption 2 holds. Since non-strongly convex optimization problems (e.g., LASSO) are becoming prevalent these days, making a direct variant of MiG for these problems is of interest.

In this setting, we summarize MiG$^{\text{NSC}}$ with the optimal $\mathcal{O}\big(1/T^2\big)$ convergence rate in Algorithm 2.

Theorem 2 (Non-strongly Convex)**.** *If Assumption 2 holds, then by choosing $m=\Theta(n)$,* MiG$^{\text{NSC}}$ *achieves the following oracle complexity in expectation:*

$$\mathcal{O}\left(n\sqrt{\frac{F(x_0)-F(x^*)}{\epsilon}} + \sqrt{\frac{nL\|x_0-x^*\|^2}{\epsilon}}\right).$$

This result implies that MiG$^{\text{NSC}}$ *attains the optimal con-*

[4]We recommend users to tune these two parameters for better performance in practice, or to use the tuning criteria mentioned in Table 3 with only tuning θ.

Algorithm 2 MiG$^{\text{NSC}}$

```
1: Input: Initial vector $x_0$, epoch length $m$, learning rate $\eta$, parameter $\theta$.
2: $\tilde{x}_0 = x_0^1 = x_0$;
3: for $s = 1 \dots \mathcal{S}$ do
4:    $\mu_s = \nabla f(\tilde{x}_{s-1}), \theta = \frac{2}{s+4}, \eta = \frac{1}{4L\theta}$;
5:    for $j = 1 \dots m$ do
6:       Sample $i_j$ uniformly in $\{1 \dots n\}$;
7:       $y_{j-1} = \theta x_{j-1}^s + (1-\theta)\tilde{x}_{s-1}$; //temp variable $y$
8:       $\tilde{\nabla} = \nabla f_{i_j}(y_{j-1}) - \nabla f_{i_j}(\tilde{x}_{s-1}) + \mu_s$;
9:       $x_j^s = \arg\min_x \left\{\frac{1}{2\eta}\|x - x_{j-1}^s\|^2 + \langle\tilde{\nabla}, x\rangle + g(x)\right\}$;
10:   end for
11:   $\tilde{x}_s = \frac{\theta}{m}\sum_{j=1}^m x_j^s + (1-\theta)\tilde{x}_{s-1}$;
12:   $x_0^{s+1} = x_m^s$;
13: end for
14: return $\tilde{x}_{\mathcal{S}}$.
```

vergence rate $\mathcal{O}(1/T^2)$, where $T = \mathcal{S}(m+n)$ is the total number of stochastic iterations.

The result in Theorem 2 shows that MiG$^{\text{NSC}}$ enjoys the same oracle complexity as Katyusha$^{\text{ns}}$ (Allen-Zhu, 2017), which is close to the best-known complexity in this case[5]. If the reduction techniques in (Allen-Zhu & Hazan, 2016a; Xu et al., 2016) are used in our algorithm, our algorithm can obtain the best-known oracle complexity.

3.3. Extensions

It is common to apply reductions to extend the algorithms designed for L-smooth and σ-strongly convex objectives to other cases (e.g., non-strongly convex or non-smooth). For example, Allen-Zhu & Hazan (2016a) proposed several reductions for the algorithms that satisfy *homogeneous objective decrease* (HOOD), which is defined as follows.

Definition 1 (Allen-Zhu & Hazan (2016a)). *An algorithm $\mathcal{A}$ that solves Problem (1) with Assumption 1 satisfies* HOOD *if, for every starting vector x_0, $\mathcal{A}$ produces an output x' satisfying*[6] $F(x') - F(x^*) \leq \frac{F(x_0)-F(x^*)}{4}$ *in* Time(L, σ).

Based on Theorem 1, it is a direct corollary that MiG (refers to Algorithm 1) satisfies HOOD.

Corollary 1. MiG *satisfies the* HOOD *property in* Time($n + \sqrt{\kappa n}$).

The reductions in (Allen-Zhu & Hazan, 2016a) use either decaying regularization (AdaptReg) or certain smoothing trick (AdaptSmooth) to achieve optimal reductions that shave off a non-optimal log factor comparing to other reduction techniques. Thus, we can apply AdaptReg to MiG and get an improved $\mathcal{O}(n\log(1/\epsilon) + \sqrt{nL/\epsilon})$ rate for non-strongly convex problems. Moreover, we can also apply AdaptSmooth to MiG to tackle non-smooth optimization problems, e.g., SVM.

[5]Note that the best-known oracle complexity for non-strongly convex problems is $\mathcal{O}(n\log(1/\epsilon) + \sqrt{nL/\epsilon})$.

[6]This definition can be extended to the probabilistic guarantee, which is $\mathbb{E}\big[F(x')\big] - F(x^*) \leq \frac{F(x_0)-F(x^*)}{4}$ (Allen-Zhu & Hazan, 2016a).

4. Sparse and Asynchronous Variants

In order to further elaborate the importance of keeping track of only one variable vector (in the inner loop), in this section we propose the variants of MiG for both the serial sparse and asynchronous sparse settings.

Adopting the sparse update technique in (Mania et al., 2017) for sparse datasets is a very practical choice to reduce collisions between threads. However, due to additional sparse approximating variance and asynchronous perturbation, we need to compensate it with a slower theoretical speed. On the other hand, asynchrony (in the lock-free style) may even destroy convergence guarantees if the algorithm requires tracking many highly correlated vectors. In practice, it is reported that maintaining more atomic[7] variables also degrades the performance. Thus, in this section, we mainly focus on practical issues and experimental performance.

As we can see, MiG has only one variable vector. This feature gives us convenience in both theoretical analysis and practical implementation. In order to give a clean proof, we first make a simpler assumption on the objective function, which is identical to those in (Recht et al., 2011; Mania et al., 2017; Leblond et al., 2017):

$$\min_{x\in\mathbb{R}^d} F(x) \triangleq \frac{1}{n}\sum_{i=1}^{n} f_i(x). \tag{4}$$

Assumption 3 (Sparse and Asynchronous Settings). *In Problem (4), each $f_i(\cdot)$ is L-smooth, and the averaged function $F(\cdot)$ is σ-strongly convex.*

Next we start with analyzing MiG in the serial sparse setting and then extend it to a sparse and asynchronous one.

4.1. Serial Sparse MiG

The sparse variant (as shown in Algorithm 3) of MiG is slightly different from MiG in the dense case. We explain these differences by making the following remarks:

- *Sparse approximate gradient $\tilde{\nabla}_S$.* In order to perform fully sparse updates, following (Mania et al., 2017), we use a diagonal matrix D to re-weigh the dense vector

[7]Atomic write of some necessary variables is a requirement to achieve high precision in practice (Leblond et al., 2017).

Algorithm 3 Serial Sparse MiG

```
1: Input: Initial vector $x_0$, epoch length $m$, learning rate $\eta$, parameter $\theta$.
2: $\tilde{x}_0 = x_0^1 = x_0$;
3: for $s = 1 \ldots \mathcal{S}$ do
4:     $\mu_s = \nabla F(\tilde{x}_{s-1})$;
5:     for $j = 1 \ldots m$ do
6:         $T_{i_j}$ := support of sample $i_j$;
7:         $[y_{j-1}]_{T_{i_j}} = \theta \cdot [x_{j-1}^s]_{T_{i_j}} + (1-\theta) \cdot [\tilde{x}_{s-1}]_{T_{i_j}}$;
8:         $\tilde{\nabla}_S = \nabla f_{i_j}([y_{j-1}]_{T_{i_j}}) - \nabla f_{i_j}([\tilde{x}_{s-1}]_{T_{i_j}}) + D_{i_j}\mu_s$;
9:         $[x_j^s]_{T_{i_j}} = [x_{j-1}^s]_{T_{i_j}} - \eta \cdot \tilde{\nabla}_S$;
10:    end for
11:    Option I: $x_0^{s+1} = \tilde{x}_s = \frac{\theta}{m}\sum_{j=0}^{m-1} x_j^s + (1-\theta)\tilde{x}_{s-1}$;
12:    Option II: $x_0^{s+1} = x_m^s$, $\tilde{x}_s = \frac{\theta}{m}\sum_{j=1}^{m} x_j^s + (1-\theta)\tilde{x}_{s-1}$;
13: end for
14: return $\tilde{x}_{\mathcal{S}}$.
```

μ_s, whose entries are the inverse probabilities $\{p_k^{-1}\}$ of the corresponding coordinates $\{k \mid k = 1, \ldots, d\}$ belonging to a uniformly sampled support T_{i_j} of sample i_j. P_{i_j} is the projection matrix for the support T_{i_j}. We define $D_{i_j} = P_{i_j} D$, which ensures the unbiasedness $\mathbb{E}_{i_j}\left[D_{i_j}\mu_s\right] = \mu_s$. Here we also define $D_m = \max_{k=1\ldots d} p_k^{-1}$ for future usage. Note that we only need to compute y on the support of sample i_j, and hence the entire inner loop updates sparsely.

- *Update $\tilde{x}$ with uniform average.* In the sparse and asynchronous setting, a weighted average in Algorithm 1 is not effective due to the perturbation both in theory and in practice. Thus, we choose a simple uniform average scheme for a better practical performance.

We now consider the convergence property of Algorithm 3.

Theorem 3 (Option I). *Let x^* be the optimal solution of Problem (4). If Assumption 3 holds, then by choosing $\eta = 1/L$, $\theta = 1/10$, $m = 25\kappa$, Algorithm 3 with Option I satisfies the following inequality in one epoch s:*

$$\mathbb{E}\left[\left(F(\tilde{x}_s) - F(x^*)\right)\right] \le 0.75 \cdot \left(F(\tilde{x}_{s-1}) - F(x^*)\right),$$

which means that the total oracle complexity of the serial sparse MiG *is $\mathcal{O}\left((n+\kappa)\log\frac{F(x_0)-F(x^*)}{\epsilon}\right)$.*

Since it is natural to ask whether we can get an improved bound for the Serial Sparse MiG, we analyze Algorithm 3 with Option II and a somewhat intriguing restart scheme. The convergence result is given as follows:

Theorem 4 (Option II). *If Assumption 3 holds, then by executing Algorithm 3 with Option II and restarting[8] the algorithm every $\mathcal{S} = \left\lceil 2 \cdot \frac{(1-\theta)\cdot(1+\zeta) + \frac{\theta}{\eta m \sigma}}{\theta + \zeta\theta - \zeta} \right\rceil$ epochs, where $\zeta = D_m^2 - D_m$, the oracle complexity of the entire procedure is divided into the cases,*

$$\begin{cases} \mathcal{O}\left(\sqrt{\kappa n}\log\frac{F(x_0)-F(x^*)}{\epsilon}\right), & \text{if } \frac{m}{\kappa} \le \frac{3}{4} \text{ with } \zeta \le \sqrt{\frac{m}{4\kappa}}, \\ \mathcal{O}\left(n\log\frac{F(x_0)-F(x^*)}{\epsilon}\right), & \text{if } \frac{m}{\kappa} > \frac{3}{4} \text{ with } \zeta \le C_\zeta, \end{cases}$$

where C_ζ is a constant for the sparse estimator variance. Detailed parameter settings are given in the Supplementary Material C.2.1.

Remark: This result indeed imposes some strong assumptions on D_m, which may not be true for real world datasets, because the variance bound used for Option II highly correlates to D_m, and D_m can be as large as n for extreme datasets. Detailed discussion is given in the Supplementary Material C.2.2.

The result of Theorem 4 shows that under some constraints on sparse variance, Serial Sparse MiG attains a faster convergence rate than Sparse SVRG (Mania et al., 2017) and Sparse SAGA (Leblond et al., 2017). Although these constraints are strong and the restart scheme is not quite practical, we keep the result here as a reference for both the sparse ($\zeta > 0$) and dense ($\zeta = 0$) cases.

4.2. Asynchronous Sparse MiG

In this part, we extend the Serial Sparse MiG to the Asynchronous Sparse MiG.

Our algorithm is given in Algorithm 4. Notice that Option I and II correspond to the update options mentioned in Algorithm 3. The difference is that Option II corresponds to averaging "fake" iterates defined at (5), while Option I is the average of inconsistent read[9] of x. Since the averaging scheme in Option II is not proposed before, we refer to it as "*fake average*". Just like the analysis in the serial sparse case, Option I leads to a direct and clean proof while Option II may require restart and leads to troublesome theoretical analysis. So we only analyze Option I in this setting.

However, Option II is shown to be highly practical since the "*fake average*" scheme only requires updates on the support of samples and enjoys strong robustness when the actual number of inner loops does not equal to m[10]. Thus, Option II leads to a very practical implementation.

Following (Mania et al., 2017), our analysis is based on the "fake" iterates x and y, which are defined as:

$$x_j = x_0 - \eta\sum_{i=0}^{j-1}\tilde{\nabla}(\hat{y}_i),\ y_j = \theta x_j + (1-\theta)\tilde{x}_{s-1}, \quad (5)$$

where the "perturbed" iterates $\hat{y}$, $\hat{x}$ with perturbation ξ are

[8] We set $x_0 = \frac{1}{\mathcal{S}}\sum_{s=1}^{\mathcal{S}}\tilde{x}_s$ as the initial vector after each restart.

[9] We could use "*fake average*" in Option I, but it leads to a complex proof and a worse convergence rate with factor ($\propto \kappa^{-2}$).

[10] This phenomenon is prevalent in the asynchronous setting.

Algorithm 4 Asynchronous Sparse MiG

```
1: **Input:** Initial vector $x_0$, epoch length $m$, learning rate $\eta$, parameter $\theta$.
2: $x :=$ shared variable, $\bar{x} :=$ average of $x$;
3: $\tilde{x}_0 = x = x_0$;
4: **for** $s = 1 \dots \mathcal{S}$ **do**
5:     Compute $\mu_s = \nabla F(\tilde{x}_{s-1})$ in parallel;
6:     Option I: $\bar{x} = \mathbf{0}$;
7:     Option II: $\bar{x} = x$;
8:     $j = 0$; {inner loop counter}
9:     **while** $j < m$ **do** {in parallel}
10:        $j = j + 1$;   // atomic increase counter $j$
11:        Sample $i_j$ uniformly in $\{1 \dots n\}$;
12:        $T_{i_j} :=$ support of sample $i_j$;
13:        $[\hat{x}]_{T_{i_j}} :=$ inconsistent read of $[x]_{T_{i_j}}$;
14:        $[\hat{y}]_{T_{i_j}} = \theta \cdot [\hat{x}]_{T_{i_j}} + (1-\theta) \cdot [\tilde{x}_{s-1}]_{T_{i_j}}$;
15:        $\tilde{\nabla}(\hat{y}) = \nabla f_{i_j}([\hat{y}]_{T_{i_j}}) - \nabla f_{i_j}([\tilde{x}_{s-1}]_{T_{i_j}}) + D_{i_j}\mu_s$;
16:        $[u]_{T_{i_j}} = -\eta \cdot \tilde{\nabla}(\hat{y})$;
17:        // atomic write $x$, $\bar{x}$ for each coordinate
18:        $[x]_{T_{i_j}} = [x]_{T_{i_j}} + [u]_{T_{i_j}}$;
19:        Option I: $\bar{x} = \bar{x} + \frac{1}{m} \cdot \hat{x}$;
20:        Option II: $[\bar{x}]_{T_{i_j}} = [\bar{x}]_{T_{i_j}} + [u]_{T_{i_j}} \cdot \frac{(m+1-j)_+}{m}$;
21:    **end while**
22:    $\tilde{x}_s = \theta\bar{x} + (1-\theta)\tilde{x}_{s-1}$;
23:    Option I: $x = \tilde{x}_s$;
24:    Option II: keep $x$ unchanged;
25: **end for**
26: **return** $\tilde{x}_{\mathcal{S}}$.
```

defined as

$$\hat{y}_j = \theta\hat{x}_j + (1-\theta)\tilde{x}_{s-1}, \hat{x}_j = x_j + \xi_j. \tag{6}$$

The labeling order and detailed analysis framework are given in the Supplementary Material C.3.

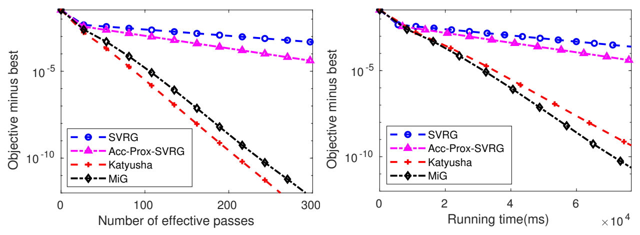
Note that y is a temp variable, so the only source of perturbation comes from x. This is the benefit of keeping track of only one variable vector since it controls the perturbation and allows us to give a smooth analysis in asynchrony.

Next we give our convergence result as follows:

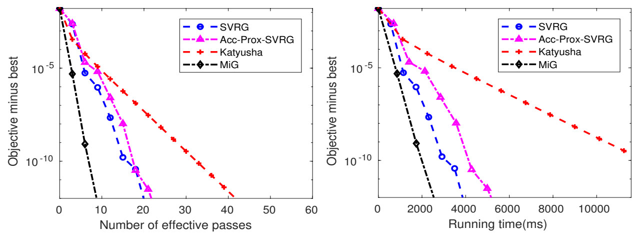
Theorem 5. *If Assumption 3 holds, then by choosing $m = 60\kappa$, $\eta = 1/(5L)$, $\theta = 1/6$, suppose τ satisfies $\tau \leq \min\{\frac{5}{4\sqrt{\Delta}}, 2\kappa, \sqrt{\frac{2\kappa}{\sqrt{\Delta}}}\}$ (the linear speed-up condition), Algorithm 4 with Option I has the following oracle complexity:*

$$\mathcal{O}\left((n+\kappa)\log\frac{F(x_0)-F(x^*)}{\epsilon}\right),$$

where τ represents the maximum number of overlaps between concurrent threads (Mania et al., 2017) and $\Delta = \max_{k=1\dots d} p_k$, which is a measure of sparsity (Leblond et al., 2017).



(a) Relatively small regularization parameter $\lambda = 10^{-8}$



(b) Relatively large regularization parameter $\lambda = 10^{-4}$

Figure 1. Comparison of the effect of the momentum techniques used in Acc-Prox-SVRG (Nitanda, 2014), Katyusha (Allen-Zhu, 2017) and MiG for $\ell 2$-logistic regression on covtype.

This result is better than that of KroMagnon, which correlates to κ^2 (Mania et al., 2017), and keeps up with ASAGA (Leblond et al., 2017). Although without significant improvement on theoretical bounds due to the existence of perturbation, the coupling step of MiG can still be regarded as a simple add-on boosting and stabilizing the performance of SVRG variants. We show this improvement by empirical evaluations in Section 5.3.

5. Experiments

In this section, we evaluate the performance of MiG on real-world datasets for both serial dense and asynchronous sparse[11] cases. All the algorithms were implemented in C++ and executed through MATLAB interface for a fair comparison. Detailed experimental setup is given in the Supplementary Material D.

We first give a detailed comparison between MiG and other algorithms in the sequential dense setting.

5.1. Comparison of Momentums

The parameter θ in MiG, similar to the parameter τ_2 in Katyusha, is referred as the parameter for "*Katyusha Momentum*" in (Allen-Zhu, 2017). So intuitively, MiG can be regarded as adding "*Katyusha Momentum*" on top of SVRG. Katyusha equipped with the linear coupling framework in (Allen-Zhu & Orecchia, 2017) and thus can be re-

[11]Experiments for the serial sparse variant are omitted since it corresponds to the asynchronous sparse variant with 1 thread.

Table 3. Compared algorithms and parameter tuning criteria.

	Momentum	Parameter Tuning
SVRG	None	learning rate η
Acc-Prox-SVRG	Nestrv.	same η, tune momentum β
Katyusha	Nestrv.&Katyu.	$\tau_2=\frac{1}{2}$, $\alpha=\frac{1}{3\tau_1 L}$, tune τ_1
MiG	Katyu.	$\eta=\frac{1}{3\theta L}$, tune θ

garded as the combination of "*Nesterov's Momentum*" with "*Katyusha Momentum*".

We empirically evaluate the effect of the two kinds of momentums. Moreover, we also examine the performance of Acc-Prox-SVRG (Nitanda, 2014), which can be regarded as SVRG with pure "*Nesterov's Momentum*". Note that the mini-batch size used in Acc-Prox-SVRG is set to 1. The algorithms and parameter settings are listed in Table 3 (we use the same notations as in their original papers).

The results in Figure 1 correspond to the two typical conditions with relatively large $\lambda=10^{-4}$ and relatively small $\lambda=10^{-8}$. One can verify that with the epoch length $m=2n$ for all algorithms, these two conditions fall into the two regions correspondingly in Table 2.

For the case of $\frac{m}{\kappa}\leq\frac{3}{4}$ (see the first row in Figure 1), we set the parameters for MiG and Katyusha[12] with their theoretical suggestions (e.g., $\theta=\tau_1=\sqrt{\frac{m}{3\kappa}}$). For fair comparison, we set the learning rate $\eta=\frac{1}{4L}$ for SVRG and Acc-Prox-SVRG, which is theoretically reasonable. The results imply that MiG and Katyusha have close convergence results and outperform SVRG and Acc-Prox-SVRG. This justifies the improvement of $\sqrt{\kappa n}$ convergence rate in theory.

We notice that Katyusha is slightly faster than MiG in terms of the number of oracle calls, which is reasonable since Katyusha has one more "*Nesterov's Momentum*". From the result of Acc-Prox-SVRG, we see that "*Nesterov's Momentum*" is effective in this case, but without significant improvement. As analyzed in (Nitanda, 2014), using large enough mini-batch is a requirement to make Acc-Prox-SVRG improve its convergence rate in theory (see Table 1 in (Nitanda, 2014)), which also explains the limited difference between MiG and Katyusha.

When comparing running time (millisecond, ms), MiG outperforms other algorithms. Using "*Nesterov's Momentum*" requires tracking of at least two variable vectors, which increases both memory consumption and computational overhead. More severely, it prevents the algorithms with this trick to have an efficient sparse and asynchronous variant.

[12]We choose to implement Katyusha with Option I, which is analyzed theoretically in (Allen-Zhu, 2017).

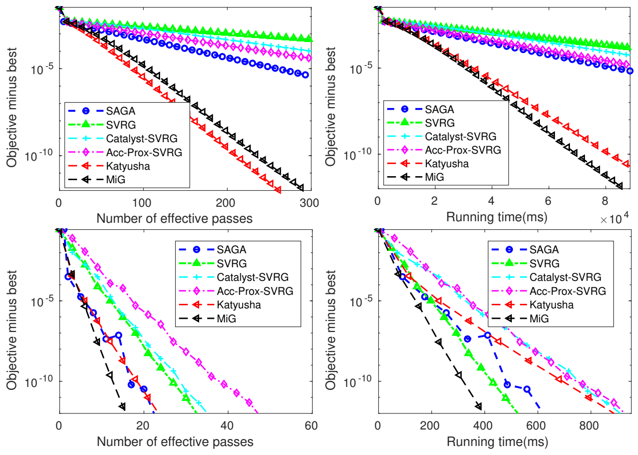


Figure 2. First row: Theoretical evaluation of MiG and state-of-the-art algorithms for $\ell2$-logistic regression ($\lambda=10^{-8}$) on covtype. Second row: Practical evaluation of MiG and the state-of-the-art algorithms for ridge regression ($\lambda=10^{-4}$) on a9a.

For the case of $\frac{m}{\kappa}>\frac{3}{4}$ (see the second row in Figure 1), we tuned all the parameters in Table 3. From the parameter tuning of Acc-Prox-SVRG, we found that using a smaller momentum parameter β yields a better performance, but still worse than the original SVRG. This result seems to indicate that the "*Nesterov's Momentum*" is not effective in this case. Katyusha yields a poor performance in this case because the parameter suggestion limits $\tau_1\leq\frac{1}{2}$. When tuning both τ_2 and τ_1, Katyusha performs much better, but with increasing difficulty of parameter tuning. MiG performs quite well with tuning only one parameter θ, which further verifies its simplicity and efficiency.

5.2. Comparison with state-of-the-art Algorithms

We compare MiG with many state-of-the-art accelerated algorithms (e.g., Acc-Prox-SVRG (Nitanda, 2014), Catalyst (based on SVRG) (Lin et al., 2015), and Katyusha (Allen-Zhu, 2017)) and non-accelerated algorithms (e.g., SVRG (Johnson & Zhang, 2013) or Prox-SVRG (Xiao & Zhang, 2014) for non-smooth regularizer, and SAGA (Defazio et al., 2014)), as shown in Figure 2.

In order to give clear comparisons, we designed two different types of experiments. One is called "theoretical evaluation" with a relatively small λ[13], where most of the parameter settings follow the corresponding theoretical recommendations[14] to justify the improvement of $\sqrt{\kappa n}$ convergence rate. Another is "practical evaluation" for a relatively large

[13]Note that we normalize data vectors to ensure a uniform L.

[14]Except for Acc-Prox-SVRG and Catalyst, we carefully tuned the parameters for them, and the detailed parameter settings are given in the Supplementary Material D.1.

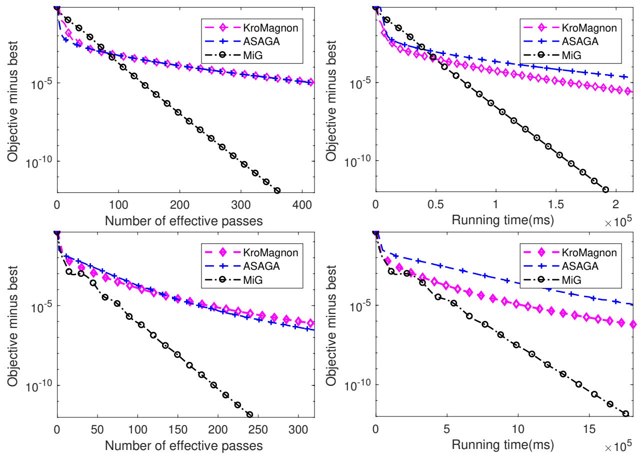


Figure 3. Comparison of KroMagnon (Mania et al., 2017), ASAGA (Leblond et al., 2017), and MiG with 16 threads. First row: RCV1, ℓ_2-logistic regression with $\lambda = 10^{-9}$. Second row: KDD2010, ℓ_2-logistic regression with $\lambda = 10^{-10}$.

Table 4. Summary of the two sparse data sets.

Dataset	# Data	# Features	Density
RCV1	697,641	47,236	1.5×10^{-3}
KDD2010	19,264,097	1,163,024	10^{-6}

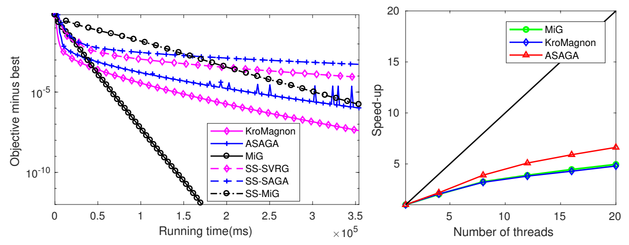


Figure 4. Speed-up evaluation on RCV1. Left: Evaluation of sub-optimality in terms of running time for asynchronous versions (20 threads) and SS (Serial Sparse) versions. Right: Speed-up of achieving 10^{-5} sub-optimality in terms of the number of threads.

λ, where we carefully tuned the parameters for all the algorithms since in this condition, all the algorithms have similar convergence rates.

For a relatively large λ, the results (see the Supplementary Material D.1 for more results) show that MiG performs consistently better than Katyusha in terms of both oracle calls and running time. In other words, we see that MiG achieves satisfactory performance in both conditions. Moreover, experimental results for non-strongly convex objectives are also given in the Supplementary Material D.1.

5.3. In Sparse and Asynchronous Settings

To further stress the simplicity and implementability of MiG, we make some experiments to assess the performance of its asynchronous variant. We also compare MiG (i.e., Algorithm 4 with Option II[15]) with KroMagnon (Mania et al., 2017) and ASAGA (Leblond et al., 2017).

Unlike in the serial dense case where we have strong theoretical guarantees, in these settings, we mainly focus on practical performance and stability. So we carefully tuned the parameter(s) for each algorithm to achieve a best-tuned performance (detailed setting and parameter tuning criteria is given in the Supplementary Material D.2). We measure the performance on the two sparse datasets listed in Table 4.

When comparing performance in terms of oracle calls, MiG significantly outperforms other algorithms, as shown in Figure 3. When considering running time, the difference is narrowed due to the high simplicity of KroMagnon (which only uses one atomic vector) compared with ASAGA (which uses atomic gradient table and atomic gradient average vector) and MiG (which only uses atomic "*fake average*").

We then examine the speed-up gained from more parallel threads on RCV1. We evaluate the improvement of using asynchronous variants (20 threads) and the speed-up ratio as a function of the number of threads as shown in Figure 4. For the latter evaluation, the running time is recorded when the algorithms achieve 10^{-5} sub-optimality. The speed-up ratio is calculated based on the running time of a single core.

Since we used MiG with Option II for the above experiments which only has a theoretical analysis (with restart) in the serial case, we further designed an experiment to evaluate the effectiveness of θ. The results in the Supplementary Material D.2.2 indicate the effectiveness of our acceleration trick.

6. Conclusion

We proposed a simple stochastic variance reduction algorithm (MiG) with the best-known oracle complexities for stochastic first-order algorithms. These elegant results further reveal the mystery of acceleration tricks in stochastic first-order optimization. Moreover, the high simplicity of MiG allows us to derive the variants for the asynchronous and sparse settings, which shows its potential to be applied to other cases (e.g., online (Borsos et al., 2018), distributed (Lee et al., 2017; Lian et al., 2017)) as well as to tackle more complex problems (e.g., structured prediction). In general, our approach can be implemented to boost the speed of large-scale real-world optimization problems.

[15] We omit the tricky restart scheme required in theory to examine the most practical variant.

Acknowledgements

We thank the reviewers for their valuable comments. This work was supported in part by Grants (CUHK 14206715 & 14222816) from the Hong Kong RGC, the Major Research Plan of the National Natural Science Foundation of China (Nos. 91438201 and 91438103), Project supported the Foundation for Innovative Research Groups of the National Natural Science Foundation of China (No. 61621005), the National Natural Science Foundation of China (Nos. U1701267, 61573267, 61502369 and 61473215), the Program for Cheung Kong Scholars and Innovative Research Team in University(No. IRT_15R53) and the Fund for Foreign Scholars in University Research and Teaching Programs (the 111 Project) (No. B07048).

References

Allen-Zhu, Z. Katyusha: The first direct acceleration of stochastic gradient methods. In *STOC*, 2017.

Allen-Zhu, Z. and Hazan, E. Optimal black-box reductions between optimization objectives. In *NIPS*, pp. 1606–1614, 2016a.

Allen-Zhu, Z. and Hazan, E. Variance reduction for faster non-convex optimization. In *ICML*, pp. 699–707, 2016b.

Allen-Zhu, Z. and Orecchia, L. Linear coupling: An ultimate unification of gradient and mirror descent. In *ITCS*, 2017.

Borsos, Z., Krause, A., and Levy, K. Y. Online variance reduction for stochastic optimization. *arXiv:1802.04715v2*, 2018.

Defazio, A. A simple practical accelerated method for finite sums. In *NIPS*, pp. 676–684, 2016.

Defazio, A., Bach, F., and Lacoste-Julien, S. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In *NIPS*, pp. 1646–1654, 2014.

Hien, L., Lu, C., Xu, H., and Feng, J. Accelerated stochastic mirror descent algorithms for composite non-strongly convex optimization. *arXiv:1605.06892v4*, 2017.

Johnson, R. and Zhang, T. Accelerating stochastic gradient descent using predictive variance reduction. In *NIPS*, pp. 315–323, 2013.

Konečný, J., Liu, J., Richtárik, P., , and Takáč, M. Mini-batch semi-stochastic gradient descent in the proximal setting. *IEEE J. Sel. Top. Sign. Proces.*, 10(2):242–255, 2016.

Leblond, R., Pedregosa, F., and Lacoste-Julien, S. ASAGA: Asynchronous parallel SAGA. In *AISTATS*, pp. 46–54, 2017.

Lee, J. D., Lin, Q., Ma, T., and Yang, T. Distributed stochastic variance reduced gradient methods by sampling extra data with replacement. *J. Mach. Learn. Res.*, 18:1–43, 2017.

Lian, X., Zhang, C., Zhang, H., Hsieh, C.-J., Zhang, W., and Liu, J. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. In *NIPS*, pp. 5336–5346, 2017.

Lin, H., Mairal, J., and Harchaoui, Z. A universal catalyst for first-order optimization. In *NIPS*, pp. 3366–3374, 2015.

Lin, Q., Lu, Z., and Xiao, L. An accelerated proximal coordinate gradient method. In *NIPS*, pp. 3059–3067, 2014.

Mania, H., Pan, X., Papailiopoulos, D., Recht, B., Ramchandran, K., and Jordan, M. I. Perturbed iterate analysis for asynchronous stochastic optimization. *SIAM J. Optim.*, 27(4):2202–2229, 2017.

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

Nitanda, A. Stochastic proximal gradient descent with acceleration techniques. In *NIPS*, pp. 1574–1582, 2014.

Pedregosa, F., Leblond, R., and Lacoste-Julien, S. Breaking the nonsmooth barrier: A scalable parallel method for composite optimization. In *NIPS*, pp. 55–64, 2017.

Recht, B., Re, C., Wright, S., and Niu, F. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *NIPS*, pp. 693–701, 2011.

Reddi, S., Hefny, A., Sra, S., Poczos, B., and Smola, A. On variance reduction in stochastic gradient descent and its asynchronous variants. In *NIPS*, pp. 2629–2637, 2015.

Robbins, H. and Monro, S. A stochastic approximation method. *Ann. Math. Statist.*, 22(3):400–407, 1951.

Roux, N. L., Schmidt, M., and Bach, F. A stochastic gradient method with an exponential convergence rate for finite training sets. In *NIPS*, pp. 2672–2680, 2012.

Schmidt, M., Roux, N. L., and Bach, F. Minimizing finite sums with the stochastic average gradient. *Math. Program.*, 162:83–112, 2017.

Shalev-Shwartz, S. and Zhang, T. Stochastic dual coordinate ascent methods for regularized loss minimization. *J. Mach. Learn. Res.*, 14:567–599, 2013.

Shalev-Shwartz, S. and Zhang, T. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. *Math. Program.*, 155:105–145, 2016.

Xiao, L. and Zhang, T. A proximal stochastic gradient method with progressive variance reduction. *SIAM J. Optim.*, 24(4):2057–2075, 2014.

Xu, Y., Yan, Y., Lin, Q., and Yang, T. Homotopy smoothing for non-smooth problems with lower complexity than $O(1/\epsilon)$. In *NIPS*, pp. 1208–1216, 2016.

Zhang, L., Mahdavi, M., and Jin, R. Linear convergence with condition number independent access of full gradients. In *NIPS*, pp. 980–988, 2013.

Zhang, Y. and Xiao, L. Stochastic primal-dual coordinate method for regularized empirical risk minimization. In *ICML*, pp. 353–361, 2015.