# Adaptive Variance Reducing for Stochastic Gradient Descent

**Zebang Shen, Hui Qian*, Tengfei Zhou, Tongzhou Mu**
Zhejiang University, China
{shenzebang, qianhui, zhoutengfei_zju, mutongzhou}@zju.edu.cn

## Abstract

Variance Reducing (VR) stochastic methods are fast-converging alternatives to the classical *Stochastic Gradient Descent* (SGD) for solving large-scale regularized finite sum problems, especially when a highly accurate solution is required. One critical step in VR is the function sampling. State-of-the-art VR algorithms such as SVRG and SAGA, employ either Uniform Probability (UP) or Importance Probability (IP), which is deficient in reducing the variance and hence leads to suboptimal convergence rate. In this paper, we propose a novel sampling scheme that explicitly computes some Adaptive Probability (AP) at each iteration. Analysis shows that, equipped with AP, both SVRG and SAGA yield provably better convergence rate than the ones with UP or IP, which is confirmed in experiments. Additionally, to cut down the per iteration computation load, an efficient variant is proposed by utilizing AP periodically, whose performance is empirically validated.

## 1 Introduction

Minimization of the Regularized Finite Sum (MRFS) is a class of problems often encountered in Artificial Intelligence and Machine Learning applications. Generally, MRFS has the following form:

$$\min_{\mathbf{w}\in\mathbb{R}^d} \{\mathbf{P}(\mathbf{w}) := \mathbf{F}(\mathbf{w}) + \mathbf{G}(\mathbf{w})\}, \tag{1}$$

where $\mathbf{F}(\mathbf{w})$ is the average of finite smooth convex component functions $f_i(\mathbf{w}), i = 1, \ldots, n$, and $\mathbf{G}(\mathbf{w})$ is a regularization function, which is convex but may not be smooth. In practice, many Empirical Risk Minimization (ERM) problems [Hastie *et al.*, 2005], such as ridged regression, regularized logistic regression, SVM with smooth loss, and Lasso, can be formulated as (1).

The standard *Gradient Decent* (GD) method [Chen and Rockafellar, 1997; Nesterov, 1998; Beck and Teboulle, 2009; Nesterov, 2013] for MRFS evaluates the gradients of all $n$ component functions at each iteration, which is computationally prohibitive for large datasets. An alternative strategy is to estimate the full gradient using a solo $\nabla f_{j_k}$ in the $k^{th}$ iteration, with $f_{j_k}$ drawn from some distribution. This is called *Stochastic Gradient Descent* (SGD). On the bright side, since no full gradient computation is involved, SGD has low per iteration cost. However, the variance introduced by the stochastic gradient precludes the using of large step-size and leads to a sublinear convergence rate [Hu *et al.*, 2009; Bottou, 2010; Duchi *et al.*, 2011].

In the past few years, much of the development for stochastic techniques has been driven by the variance reducing (VR) stochastic methods, such as SVRG [Johnson and Zhang, 2013], Mixed Optimization [Mahdavi *et al.*, 2013], S2GD [Konečný and Richtárik, 2013; Konecny *et al.*, 2015], SDCA [Shalev-Shwartz and Zhang, 2013b; 2016; 2013a], SAGA [Defazio *et al.*, 2014], SAG [Schmidt *et al.*, 2013], IProx-SVRG [Xiao and Zhang, 2014], and IProx-SDCA [Zhao and Zhang, 2014]. The difference between VR stochastic methods and the vanilla SGD is that the former ensures the expected distance between some modified stochastic gradient and the full gradient diminishes to zero, while the latter will not significantly decrease the variance of the estimation as the algorithm converges. It has been proved that such diminishing variance enables the constant step size as in GD, and guarantees linear convergence in the strongly convex case, e.g. [Xiao and Zhang, 2014]. Thus VR stochastic methods are more efficient than the slow-converging SGD when a highly accurate solution is required.

Generally, VR stochastic methods include two major components: the strategy to adjust the stochastic gradient at each iteration, and the scheme to sample functions or coordinates. For the sampling scheme, *Uniform probability* (UP) has been widely used in the literature [Johnson and Zhang, 2013; Defazio *et al.*, 2014]. While UP can be implemented efficiently, it is observed that, for many VR algorithms, uniform sampling may result in a suboptimal convergence rate. Several VR methods, e.g. IProx-SVRG [Xiao and Zhang, 2014] and SAGA-NUS [Schmidt *et al.*, 2015], use a non-uniform probability, titled *Importance Probability* (IP), as remedy. Specifically, the probability that $f_i$ is sampled will be proportional to its smoothness parameter.

Recently, Csiba *et al.* (2015) proposed a variant of SDCA named AdaSDCA and its heuristic version AdaSDCA+. The

*Corresponding author

authors prove that AdaSDCA would converge faster than IProx-SDCA, if the sampling probability takes the solution of another (maybe non-convex) optimization problem at each iteration. However, such solution is in closed form only when all $f_i's, i = 1, \dots, n$ are quadratic, and is numerically difficult to approximate otherwise.

By analyzing the variance of the modified stochastic gradient used in SVRG and SAGA, we propose adaptive sampling schemes for these two methods. Our contributions are listed as follows.

i. Our schemes compute the sampling probability in closed form, and can handle any smooth convex loss function, e.g. Logistic Regression, instead of being restricted to quadratic ones.

ii. By utilizing our adaptive sampling schemes, two VR methods, namely AdaSVRG and AdaSAGA, are proposed as alternatives to SVRG and SAGA respectively. We prove that they have better convergence rate than the originals and confirm our analysis with experiments.

iii. Moreover, we devise a hybrid method, named HVRG, with low amortized computational cost per iteration. Our experiments show that HVRG enjoys fast convergence as AdaSVRG and AdaSAGA.

## 2 Preliminaries

### 2.1 Notations and Assumptions

For $\mathbf{x} \in \mathbb{R}^d$, we use $\|\mathbf{x}\|$ to denote the Euclidean norm and $\nabla \mathbf{F}(\mathbf{x})$ to denote the gradient of a function $\mathbf{F}(\cdot)$ at $\mathbf{x}$. The proximal operator $\text{prox}_{\eta,\mathbf{G}}(\cdot) : \mathbb{R}^d \to \mathbb{R}^d$ is defined as

$$\text{prox}_{\eta,\mathbf{G}}(\mathbf{y}) := \operatorname*{argmin}_{\mathbf{x}\in\mathbb{R}^d} \frac{1}{2\eta}\|\mathbf{x}-\mathbf{y}\|^2 + \mathbf{G}(\mathbf{x}).$$

We use $\mathbf{w}^* = \text{argmin}_{\mathbf{w}\in\mathbb{R}^d} \mathbf{P}(\mathbf{w})$ to denote the minimizer of the whole problem. A function $f(\cdot) : \mathbb{R}^d \to \mathbb{R}$ is said to be *$L$-smooth* if for any $\mathbf{w}_1, \mathbf{w}_2 \in \mathbb{R}^d$

$$f(\mathbf{w}_1) \le f(\mathbf{w}_2) + \langle \nabla f(\mathbf{w}_2), \mathbf{w}_1 - \mathbf{w}_2 \rangle + \frac{L}{2}\|\mathbf{w}_1 - \mathbf{w}_2\|^2,$$

and we call $L$ the smoothness parameter of $f$. A function $f(\cdot) : \mathbb{R}^d \to \mathbb{R}$ is said to be *$\mu$-strongly-convex* if for any $\mathbf{w}_1, \mathbf{w}_2 \in \mathbb{R}^d$

$$f(\mathbf{w}_1) \ge f(\mathbf{w}_2) + \langle \nabla f(\mathbf{w}_2), \mathbf{w}_1 - \mathbf{w}_2 \rangle + \frac{\mu}{2}\|\mathbf{w}_1 - \mathbf{w}_2\|^2.$$

In this paper, the expectations are taken with respect to the random variable $j_k$, which indicates the function $f_{j_k}$ is selected in the kth iteration, and are conditioned on all previous iterations, for example

$$\begin{aligned}&\mathbb{E}[\frac{\nabla f_{j_k}(\mathbf{w}^{(k-1)})}{np_{j_k}}] \\ &= \mathbb{E}_{j_k\sim\{p_i\}_{i=1}^n}[\frac{\nabla f_{j_k}(\mathbf{w}^{(k-1)})}{np_{j_k}}|\mathbf{w}^{(1)},\dots,\mathbf{w}^{(k-1)}] \\ &= \sum_{i=1}^n p_i \frac{\nabla f_i(\mathbf{w}^{(k-1)})}{np_i} = \nabla \mathbf{F}(\mathbf{w}^{(k-1)}),\end{aligned}$$

where $\{p_i\}_{i=1}^n$ denotes the distribution of $j_k$, i.e. $\Pr\{j_k = i\} = p_i$. For a random vector $\mathbf{g}$, we define $\mathbb{V}[\mathbf{g}] := \mathbb{E}\|\mathbf{g} - \mathbb{E}[\mathbf{g}]\|^2 = \mathbb{E}[\|\mathbf{g}\|^2] - [\mathbb{E}[\|\mathbf{g}\|]]^2$, which will be critical in our analysis.

### 2.2 Variance Reducing

Recall that in the $k^{th}$ iteration of SGD, some function $f_{j_k}$ is sampled according to $\{p_i\}_{i=1}^n$, and the variable is updated as

$$\mathbf{w}^{(k)} := \text{prox}_{\eta_k,\mathbf{G}}\{\mathbf{w}^{(k-1)} - \eta_k \mathbf{g}^{(k)}\}, \tag{2}$$

where $\eta_k$ is the step size and $\mathbf{g}^{(k)} = \nabla f_{j_k}(\mathbf{w}^{(k-1)})/np_{j_k}$ is the stochastic gradient.

For ease of discussion, we assume in this paragraph that $\mathbf{G}(\mathbf{w}) \equiv 0$. Since $\mathbb{E}[\mathbf{g}^{(k)}] = \mathbf{F}(\mathbf{w}^{(k-1)})$, we have $\mathbb{V}[\mathbf{g}^{(k)}] = \mathbb{E}\|\mathbf{g}^{(k)}\|^2 - \|\nabla\mathbf{F}(\mathbf{w}^{(k-1)})\|^2$. However, such term may not converge to zero when $\mathbf{w}^{(k-1)} \to \mathbf{w}^*$, because while we always have $\nabla\mathbf{F}(\mathbf{w}^{(k-1)}) \to 0$, $\mathbb{E}\|\mathbf{g}^{(k)}\|^2$ does not converge to $0$ since $\nabla f_i(\mathbf{w}^{(k-1)}) \nrightarrow 0$ in general. To counteract the influence of variance, SGD adopts a diminishing step size scheme (usually $\mathcal{O}(\frac{1}{k})$ or $\mathcal{O}(\frac{1}{\sqrt{k}})$) and yields sublinear convergence.

The key to accelerate the convergence is to reduce $\mathbb{V}[\mathbf{g}^{(k)}]$ as much as possible, since it obstructs the utilization of large step size. We describe two important techniques designed towards this goal, namely *gradient adjustment* and *non-uniform sampling*. Note that they are complementary to each other and should be used jointly.

**Gradient Adjustment**

Gradient adjustment is a technique that modifies the stochastic gradient to ensure $\mathbb{V}[\mathbf{g}^{(k)}] \to 0$ when $\mathbf{w}^{(k-1)} \to \mathbf{w}^*$. For example, SVRG maintains a periodically updated variable $\tilde{\mathbf{w}}$ and modifies $\mathbf{g}^{(k)}$ as

$$\mathbf{g}^{(k)} := (\nabla f_{j_k}(\mathbf{w}^{(k-1)}) - \nabla f_{j_k}(\tilde{\mathbf{w}}))/np_{j_k} + \nabla\mathbf{F}(\tilde{\mathbf{w}}). \tag{3}$$

Clearly, $\mathbf{g}^{(k)}$ is an unbiased estimation of $\nabla\mathbf{F}(\mathbf{w}^{(k-1)})$, and it is proved that $\mathbb{V}[\mathbf{g}^{(k)}] \to 0$ when $\mathbf{w}^{(k-1)} \to \mathbf{w}^*$ [Johnson and Zhang, 2013]. Another example is SAGA which memorizes the latest computed gradient of each function $f_i$ in $\alpha_i^{(k-1)}$ and uses their average $\bar{\alpha}^{(k-1)}$ as a surrogate of $\nabla\mathbf{F}(\tilde{\mathbf{w}})$ in SVRG. Thus the updating rule can be written as

$$\mathbf{g}^{(k)} := (\nabla f_{j_k}(\mathbf{w}^{(k-1)}) - \alpha_{j_k}^{(k-1)})/np_{j_k} + \bar{\alpha}^{(k-1)}. \tag{4}$$

Defazio *et al.* (2014) proves that SAGA ensures $\mathbb{V}[\mathbf{g}^{(k)}] \to 0$ when $\mathbf{w}^{(k-1)} \to \mathbf{w}^*$. By substituting the modified stochastic gradient (3) and (4) with $\mathbf{g}^{(k)}$ in SGD, SVRG and SAGA perform the update (2) to finish the iteration.

**Non-Uniform Sampling**

While the asymptotic property of $\mathbb{V}[\mathbf{g}^{(k)}]$ is improved by gradient adjustment, its value also depends on the sampling probability, see (8). To improve upon UP, Zhao and Zhang (2014) propose to use the **Importance Probability (IP)** for sampling in SGD and SDCA, which is defined as

$$p_i^I := \frac{L_i}{\sum_{i=1}^n L_i}, i = 1, \dots, n, \tag{5}$$

where $L_i$ is the smoothness parameter of each function $f_i$. Such probability can be used in SVRG and SAGA to improve their bounds on $\mathbb{V}[\mathbf{g}^{(k)}]$ [Xiao and Zhang, 2014; Defazio *et al.*, 2014]. Note that IP is left unchanged throughout the iterations and only minimizes some loose upper bound on $\mathbb{V}[\mathbf{g}^{(k)}]$. In the next section, we present an adaptive sampling scheme that directly minimizes $\mathbb{V}[\mathbf{g}^{(k)}]$ to reduce the variance to a larger extent.

**Algorithm 1** AdaSVRG

```
Input: $\mathbf{w}^{(0)}, \eta, m, K$
Output: $\mathbf{w}^{(K)}$
 1: Initialize $\tilde{\mathbf{w}} = \mathbf{w}^{(0)}, \alpha_i^{(0)} := \nabla f_i(\mathbf{w}^{(0)}), i = 1, \dots, n$
 2: $\bar{\alpha}^{(0)} := \nabla \mathbf{F}(\tilde{\mathbf{w}})$
 3: for $k := 1$ to $K$ do
 4:     Compute the probability $\{p_i^{(k)}\}_{i=1}^n$ according to (7)
 5:     Sampling index $j_k$ according to $\{p_i^{(k)}\}_{i=1}^n$
 6:     $\mathbf{g}^{(k)} := (\nabla f_{j_k}(\mathbf{w}^{(k-1)}) - \nabla f_{j_k}(\tilde{\mathbf{w}}))/np_{j_k}^{(k)} + \bar{\alpha}^{(k-1)}$
 7:     $\mathbf{v}^{(k)} := \mathbf{w}^{(k-1)} - \eta \mathbf{g}^{(k)}$
 8:     $\mathbf{w}^{(k)} := \text{Prox}_{\eta,\mathbf{G}}(\mathbf{v}^{(k)})$
 9:     if $k \bmod m = 0$ then
10:         $\tilde{\mathbf{w}} := \frac{1}{m}\sum_{i=0}^{m-1} \mathbf{w}^{(k-i)}$
11:         $\mathbf{w}^{(k)} := \tilde{\mathbf{w}}$
12:         $\bar{\alpha}^{(k)} := \nabla \mathbf{F}(\tilde{\mathbf{w}})$
13:     else
14:         $\bar{\alpha}^{(k)} := \bar{\alpha}^{(k-1)}$
15:     end if
16: end for
```

**Algorithm 2** AdaSAGA

```
Input: $\mathbf{w}^{(0)}, \eta, K$
Output: $\mathbf{w}^{(K)}$
 1: Initialize $\alpha_i^{(0)} := \nabla f_i(\mathbf{w}^{(0)}), i = 1, \dots, n$
 2: $\bar{\alpha}^{(0)} := \frac{1}{n}\sum_{i=1}^n \alpha_i^{(0)}$
 3: for $k := 1$ to $K$ do
 4:     Compute the probability $\{p_i^{(k)}\}_{i=1}^n$ according to (7)
 5:     Sampling index $j_k$ according to $\{p_i^{(k)}\}_{i=1}^n$
 6:     $\mathbf{g}^{(k)} := (\nabla f_{j_k}(\mathbf{w}^{(k-1)}) - \alpha_{j_k}^{(k-1)})/np_{j_k}^{(k)} + \bar{\alpha}^{(k-1)}$
 7:     $\mathbf{v}^{(k)} := \mathbf{w}^{(k-1)} - \eta \mathbf{g}^{(k)}$
 8:     $\mathbf{w}^{(k)} := \text{Prox}_{\eta,\mathbf{G}}(\mathbf{v}^{(k)})$
 9:     $\alpha_i^{(k)} := \begin{cases} \nabla f_{j_k}(\mathbf{w}^{(k)}) & \text{if } i = j_k \\ \alpha_i^{(k-1)} & \text{otherwise} \end{cases}$
10:     $\bar{\alpha}^{(k)} := \frac{1}{n}\sum_{i=1}^n \alpha_i^{(k)}$
11: end for
```

## 3 Methodology

In the $k^{th}$ iteration, denoting $\nabla f_{j_k}(\tilde{\mathbf{w}})$ with $\alpha_{j_k}^{(k-1)}$ and $\nabla \mathbf{F}(\tilde{\mathbf{w}})$ with $\bar{\alpha}^{(k-1)}$, we can write the stochastic gradient $\mathbf{g}^{(k)}$ of SVRG and SAGA in a uniform way:

$$\mathbf{g}^{(k)} := \beta_{j_k}^{(k)}/np_{j_k} + \bar{\alpha}^{(k-1)}, \tag{6}$$

where $\beta_{j_k}^{(k)} := \nabla f_{j_k}(\mathbf{w}^{(k-1)}) - \alpha_{j_k}^{(k-1)}$ and $\beta_{j_k}^{(k)}/np_{j_k}$ can be regarded as a calibration to $\bar{\alpha}^{(k-1)}$ to produce an up-to-date stochastic gradient. We define the *Adaptive Probability* (AP) as

$$p_i^{(k)} = \frac{\|\beta_i^{(k)}\|}{\sum_{i=1}^n \|\beta_i^{(k)}\|}, i = 1, \dots, n, \tag{7}$$

i.e. $f_i$ is sampled with probability proportional to $\|\beta_i^{(k)}\|$. The next lemma shows that AP is the minimizer of $\mathbb{V}[\mathbf{g}^{(k)}]$.

**Lemma 1.** *Using the notation in (6), we have that*

$$\mathbb{V}[\mathbf{g}^{(k)}] = \mathbb{E}\|\beta_j^{(k)}/np_j\|^2 - \|\nabla \mathbf{F}(\mathbf{w}^{(k-1)}) - \bar{\alpha}^{(k-1)}\|^2 \tag{8}$$

*is minimized by taking $p_i = p_i^{(k)}$, which is defined in (7).*

### 3.1 Algorithms

Two new algorithms, AdaSVRG and AdaSAGA, are devised by employing AP in SVRG and SAGA respectively. We summarize them in Algorithm 1 and Algorithm 2. Note that when $\mathrm{mod}(k, n) = 1$ in AdaSVRG or $k = 1$ in AdaSAGA, (7) is not well defined since both the numerator and the denominator are zero. However, in such situation, we will have $\mathbf{g}^{(k)} = \bar{\alpha}^{(k-1)}$ for any $j_k$, and thus we directly set $\mathbf{g}^{(k)} = \bar{\alpha}^{(k-1)}$.

The following two lemmas bound $\mathbb{V}[\mathbf{g}^{(k)}]$ in AdaSVRG and AdaSAGA respectively. They will be used in the convergence analyses of our algorithms in Section 4.

**Lemma 2.** *Assume that each function $f_i$ is $L_i$-smooth and define $\bar{L} := \frac{1}{n}\sum_{i=1}^n L_i$. In AdaSVRG, we have*

$$\mathbb{V}[\mathbf{g}^{(k)}] \le 4L_A^{(k)}[\mathbf{P}(\mathbf{w}^{(k-1)}) - \mathbf{P}(\mathbf{w}^*) + \mathbf{P}(\tilde{\mathbf{w}}) - \mathbf{P}(\mathbf{w}^*)]$$

*where $L_A^{(k)} \le \bar{L}$ for each $k \in \{1 \dots m\}$.*

**Remark 1.** *When UP or IP is used for sampling, similar bound can be obtained, but with $L_A^{(k)}$ replaced with $\max_i L_i$ and $\bar{L}$ respectively. Since we have $L_A^{(k)} \le \bar{L} \le \max_i L_i$, our bound is tighter and will lead to faster convergence. Note that $L_A^{(k)}$, $\bar{L}$, and $\max_i L_i$ will appear in the convergence results when the corresponding distribution is used for sampling.*

**Remark 2.** *$L_A^{(k)}$ is potentially equal to $\bar{L}$ as AP could be identical to IP, e.g. all $f_i$'s are identical. However, we find $L_A^{(k)}$ much smaller than $\bar{L}$ in experiments.*

**Lemma 3.** *Assume each function $f_i$ is $L_i$-smooth and define $\bar{L} := \frac{1}{n}\sum_{i=1}^n L_i$. In AdaSAGA, we have that, in the $k^{th}$ iteration,*

$$\frac{n}{2L_A^{(k)}\eta^2}\mathbb{E}\|\mathbf{v}^{(k)} - \mathbf{w}^{(k-1)} - \eta\nabla \mathbf{F}(\mathbf{w}^*)\|^2$$
$$\le \sum_{i=1}^n \frac{1}{L_i}\left[\|f_i'(\mathbf{x}^{(k-1)}) - f_i'(\mathbf{x}^*)\|^2 + \|\alpha_i^{(k-1)} - f_i'(\mathbf{x}^*)\|^2\right]$$

*for some constant $L_A^{(k)}$, where $L_A^{(k)} \le \bar{L}$.*

**Remark 3.** *We use this lemma to bound the Lyapunov function (11) when proving the convergence of SAGA. Similar to the previous lemma, when IP or UP is used for sampling, $L_A^{(k)}$ degenerates to $\bar{L}$ and $\max_i L_i$ respectively, with $L_A^{(k)} \le \bar{L} \le \max_i L_i$.*

### 3.2 AP vs. IP in SVRG

In this section, we discuss the relation between AP and IP in SVRG and present a concrete example to demonstrate their disparity. As for SAGA, the discussion is similar. Here we assume that every $f_i$ is of the form $f_i(\mathbf{w}) := \phi_i(\mathbf{X}_i^\top \mathbf{w})$, where $\mathbf{X}_i$ is the non-zero feature vector of the $i^{th}$ sample. We have $\nabla f_i(\mathbf{w}) = \phi_i'(\mathbf{X}_i^\top \mathbf{w})\mathbf{X}_i$ and $\nabla^2 f_i(\mathbf{w}) = \phi_i''(\mathbf{X}_i^\top \mathbf{w})\mathbf{X}_i\mathbf{X}_i^\top$.

If we assume each $\phi_i(\cdot)$ to be $\alpha$-smooth, we can compute the smoothness parameter $L_i$ of $f_i$

$$L_i = \max_{\mathbf{w}} \|\nabla^2 f_i(\mathbf{w})\| = \alpha\|\mathbf{X}_i\|^2$$

and hence IP:

$$p_i^I = \frac{\|\mathbf{X}_i\|^2}{\sum_{i=1}^n \|\mathbf{X}_i\|^2}. \tag{9}$$

Now we compute AP

$$p_i^{(k)} = \frac{|\phi_i'(\mathbf{X}_i^\top \mathbf{w}^{(k)}) - \phi_i'(\mathbf{X}_i^\top \tilde{\mathbf{w}})|\|\mathbf{X}_i\|}{\sum_{i=1}^n |\phi_i'(\mathbf{X}_i^\top \mathbf{w}^{(k)}) - \phi_i'(\mathbf{X}_i^\top \tilde{\mathbf{w}})|\|\mathbf{X}_i\|}. \tag{10}$$

Note that AP degenerates to IP when we substitute each $|\phi_i'(\mathbf{X}_i^\top \mathbf{w}^{(k)}) - \phi_i'(\mathbf{X}_i^\top \tilde{\mathbf{w}})|$ with its upper bound

$$|\phi_i'(\mathbf{X}_i^\top \mathbf{w}^{(k)}) - \phi_i'(\mathbf{X}_i^\top \tilde{\mathbf{w}})| \leq \alpha \|\mathbf{X}_i\| \|\mathbf{w}^{(k)} - \tilde{\mathbf{w}}\|.$$

However, such upper bound can be rather loose, and AP therefore can be quite different from IP. Consider the case where $\mathbf{X}_{i_1} = \mathbf{w}^*$, $\|\mathbf{X}_{i_1}\| = \|\mathbf{X}_{i_2}\|$, but $\mathbf{X}_{i_2} \perp \mathbf{w}^*$, and $\phi_{i_2} = \phi_{i_1}$. Additionally, assume that both $\mathbf{w}^{(k)}$ and $\tilde{\mathbf{w}}$ are close to $\mathbf{w}^*$, which happens as the algorithm converges. In such case, both $\mathbf{X}_{i_1}^\top \mathbf{w}^{(k)}$ and $\mathbf{X}_{i_1}^\top \tilde{\mathbf{w}}$ are close to $\|\mathbf{w}^*\|^2$, but $\mathbf{X}_{i_2}^\top \mathbf{w}^{(k)}$ and $\mathbf{X}_{i_2}^\top \tilde{\mathbf{w}}$ are close to 0. According to (9) and (10), the sampling probabilities of $f_{i_1}$ and $f_{i_2}$ are equal in IP, but can be far from each other in AP.

## 4 Analysis

In this section, we show that both AdaSVRG and AdaSAGA converge faster than the originals with IP. The proof utilizes the tightened variance bound we derived in the previous section and is left to the long version of this paper due to limitation of space.

### 4.1 Convergence Analysis of AdaSVRG

To present the convergence result, let us first define

$$\rho_\mu(\eta, L, m) := \frac{1}{\mu\eta(1-4L\eta)m} + \frac{4L\eta(m+1)}{(1-4L\eta)m},$$

where $\mu$ is some fixed constant. A useful fact about $\rho_\mu$ is that, $\rho_\mu(\eta, L_1, m) \leq \rho_\mu(\eta, L_2, m)$ for any $0 < \eta < \frac{1}{4L_2}$ and positive $m$, if $L_1 \leq L_2$. This can be easily checked by computing the derivative of $\rho_\mu$ with respect to $L$. Additionally, auxiliary variables $\{\tilde{\mathbf{w}}^{(t)}\}_{t=0}^T$ are defined as: $\tilde{\mathbf{w}}^{(0)} := \mathbf{w}^{(0)}$, and $\tilde{\mathbf{w}}^{(t)} := \tilde{\mathbf{w}}$ when $\tilde{\mathbf{w}}$ is updated for the $t^{th}$ time.

**Theorem 1.** *Assume that $\mathbf{F}(\mathbf{w})$ is $\mu$-strongly convex and each $f_i$ is $L_i$-smooth. Define $\tilde{L}_A^{(t)} := \max_{i\in\{1,\ldots,m\}} L_A^{(tm-i)}$, where $L_A^{(k)}$ is defined in Lemma 2. Taking $m$ and $\eta_t \in (0, \frac{1}{4\tilde{L}_A^{(t)}})$ such that $\rho_\mu(\eta_t, \tilde{L}_A^{(t)}, m) < 1$, we have*

$$\mathbb{E}\mathbf{P}(\tilde{\mathbf{w}}^{(T)}) - \mathbf{P}(\mathbf{w}^*) \leq \prod_{t=1}^T \rho_\mu(\eta_t, \tilde{L}_A^{(t)}, m)[\mathbf{P}(\tilde{\mathbf{w}}^{(0)}) - \mathbf{P}(\mathbf{w}^*)].$$

**Remark 4.** *Xiao and Zhang (2014) have shown that IProx-SVRG has the following convergence result*

$$\mathbb{E}\mathbf{P}(\tilde{\mathbf{w}}^{(T)}) - \mathbf{P}(\mathbf{w}^*) \leq \prod_{t=1}^T \rho_\mu(\bar{\eta}, \bar{L}, m)[\mathbf{P}(\tilde{\mathbf{w}}^{(0)}) - \mathbf{P}(\mathbf{w}^*)]$$

*where $\bar{\eta} \in (0, \frac{1}{4\bar{L}})$ is some fixed step-size. If we use the same fixed step-size in AdaSVRG, i.e. $\eta_t = \bar{\eta}$ for each $t \in \{1 \ldots T\}$, then we have $\rho_\mu(\bar{\eta}, \tilde{L}_A^{(t)}, m) \leq \rho_\mu(\bar{\eta}, \bar{L}, m)$ and thus Theorem 1 entails that AdaSVRG enjoys a faster convergence. Additionally, it indicates that we can choose a larger step-size in AdaSVRG, e.g. $\eta_t = \frac{L^I}{\tilde{L}_A^{(t)}}\bar{\eta} \geq \bar{\eta}$, which leads to faster convergence in practice. Note that in such case, $\rho_\mu(\eta_t, \tilde{L}_A^{(t)}, m)$ is still smaller than $\rho_\mu(\bar{\eta}, \tilde{L}^I, m)$.*

### 4.2 Convergence Analysis of AdaSAGA

Before diving into the convergence theorem of AdaSAGA, we first define $\phi_i^{(k)} := \mathbf{w}^{(k_i'-1)}$ for $i = 1, \ldots, n$, where $k_i' \leq k$ is the last iteration when $f_i$ is sampled. We define $\phi_i^{(0)} = \mathbf{w}^{(0)}$. Following this definition, we have $\alpha_i^{(k-1)} = f_i'(\phi_i^{(k-1)})$ for all $i = 1, \ldots, n$. Note that $\phi_i^{(k)}$ is only used for analysis and it is not memorized over iterations. We then define the Lyapunov function $T$, which is an upper bound of $\frac{1}{2n\eta}\|\mathbf{w} - \mathbf{w}^*\|^2$ due to the convexity of $f_i$:

$$\begin{aligned} T(\mathbf{w}, \{\phi_i\}_{i=1}^n) &:= \frac{1}{2n\eta}\|\mathbf{w} - \mathbf{w}^*\|^2 \\ &+ \frac{1}{n}\sum_{i=1}^n [f_i(\phi_i) - f_i(\mathbf{w}^*) - \langle f_i'(\mathbf{w}^*), \phi_i - \mathbf{w}^*\rangle] \end{aligned} \tag{11}$$

where $\eta$ is some positive constant.

**Theorem 2.** *Assume that $\mathbf{F}(\mathbf{w})$ is $\mu$-strongly convex and each $f_i$ is $L_i$-smooth, and define $\bar{L} = \frac{1}{n}\sum_{i=1}^n L_i$. For AdaSAGA[1], let $T^{(k)}$ be:*

$$T^{(k)} := T(\mathbf{w}^{(k)}, \{\phi_i^{(k)}\}_{i=1}^n)$$

*with $\eta = \frac{1}{4\bar{L}}$ being the step size. We have*

$$\mathbb{E}T^{(k+1)} \leq (1 - \kappa_k)T^{(k)}$$

*where $\kappa_k = \min\{\frac{1}{2n}, \frac{\mu}{8L_A^{(k)}}\}$ and $L_A^{(k)}$ is defined in Lemma 3.*

**Remark 5.** *If there is no regularization, i.e. $\mathbf{G}(\mathbf{w}) \equiv 0$, and IP is used for sampling, Schmidt* et al. *(2015) prove that*

$$\mathbb{E}T^{(k+1)} \leq (1 - \kappa)T^{(k)},$$

*where $\kappa = \min\{\frac{1}{3n}, \frac{\mu}{8\bar{L}}\} < 1$. Our result is better since $(1 - \kappa_k) \leq (1 - \kappa)$ for all $k = 1, \ldots, m$. Besides, it is valid even if $\mathbf{G}(\mathbf{w}) \neq 0$. From this theorem, it is clear that $\mathbf{w}^{(k)}$ converges to the optimal $\mathbf{w}^*$ linearly:*

$$\mathbb{E}\|\mathbf{w}^{(m)} - \mathbf{w}^*\|^2 \leq 2n\eta T^{(0)}[\prod_{k=1}^m (1 - \kappa_k)]. \tag{12}$$

## 5 Efficient Variant

Theorem 1 and 2 suggest that AdaSVRG and AdaSAGA converge faster than SVRG and SAGA respectively. However, exactly computing (7) takes $\mathcal{O}(nd)$ operations, which

[1]Though unnecessary in practice, some additional uniform samples are needed in our analysis, similar to [Schmidt *et al.*, 2015]

**Algorithm 3** HVRG

```
Input: $\mathbf{w}_0, \eta, K, c, \rho$
Output: $\mathbf{w}_K$
1:  for $k := 1$ to $K$ do
2:      if $k \bmod cn = 1$ then
3:          Synchronous Update of $\alpha_i$'s:
4:          $\alpha_i^{(k-1)} := \nabla f_i(\mathbf{w}^{(k-1)}), i = 1, \ldots, n$
5:          $\bar{\alpha}^{(k-1)} := \frac{1}{n}\sum_{i=1}^n \alpha_i^{(k-1)}$
6:      end if
7:      if $k \bmod cn = 2$ then
8:          Compute $\{p_i\}_{i=1}^n$ according to (7)
9:      end if
10:     Sampling index $j_k$ according to $\{p_i\}_{i=1}^n$
11:     $\mathbf{g}^{(k)} := (\nabla f_{j_k}(\mathbf{w}^{(k-1)}) - \alpha_{j_k}^{(k-1)})/np_{j_k} + \bar{\alpha}^{(k-1)}$
12:     $\mathbf{v}^{(k)} := \mathbf{w}^{(k-1)} - \eta\mathbf{g}^{(k)}$
13:     $\mathbf{w}^{(k)} := \mathrm{Prox}_{\eta,\mathbf{G}}(\mathbf{v}^{(k)})$
14:     $\alpha_i^{(k)} := \begin{cases} \nabla f_{j_k}(\mathbf{w}^{(k)}) & \text{if } i = j_k \\ \alpha_i^{(k-1)} & \text{otherwise} \end{cases}$
15:     $\bar{\alpha}^{(k)} := \frac{1}{n}\sum_{i=1}^n \alpha_i^{(k)}$
16:     Probability Shrinkage:
17:     $p_{j_k} := p_{j_k}/\rho$; for $i \neq j_k$, $p_i$ are left unchanged.
18:     $p_j := p_j / \sum_{i=1}^n p_i, j = 1, \ldots, n$
19: end for
```

makes our algorithms less practical when handling large scale problems. To ameliorate this shortcoming, we present a variant called Hybrid Variance Reducing Gradient descent method (HVRG), which computes AP exactly every $c$ epochs ($cn$ iterations). For ease of notation, we suppress the superscripts of $\alpha_i$'s in this section .

Like AdaSAGA, HVRG keeps auxiliary variables $\alpha_i, i = 1, \ldots, n$ to store the latest computed gradient for every $f_i$. And like AdaSVRG, it updates all $\alpha_i$ to $\nabla f_i(\mathbf{w}^{(k)})$ synchronously every $c$ epochs. However, unlike both methods, HVRG exactly computes AP only after updating all $\alpha_i$'s synchronously, and shrink $p_{j_k}$ for the sampled index $j_k$ in the $k^{th}$ iteration. We summarize HVRG in Algorithm 3 and discuss the details as follows.

**Synchronous Update of $\alpha_i$'s:** When uniform probability is used for sampling, we know from the coupon collector's problem that it takes $\Theta(n \log n)$ iterations in expectation for every function $f_i$ to be sampled at least once. This means that some function $f_j$ will not be visited in $\Theta(\log n)$ epochs, nor will the corresponding $\alpha_j$ be updated. As $\bar{\alpha}$ is kept as the average of all $\alpha_j$'s, leaving some $\alpha_j$ out of date potentially undermines the convergence of algorithm. Such situation will get even more severe when non-uniform probability is employed, because the minimum time at which every function is sampled will be further postponed in expectation. To circumvent this pitfall, a periodically full update of all $\alpha_j$'s is needed.

**Shrinkage of $p_{j_k}$:** After $k^{th}$ iteration, $\alpha_{j_k}$ is updated, and the information of $\nabla f_{j_k}$ has been incorporated into $\bar{\alpha}$ for estimating of the full gradient in the future iterations. Therefore, the randomness related to $f_{j_k}$ is reduced. So we shrink its sampling probability $p_{j_k}$ by $\rho$, in order to allow other unexplored components to be sampled with higher probability. Csiba *et al.* (2015) also use similar adaptive sampling method for AdaSDCA+ in dual space.

Table 1: Statistics of datasets.

| Dataset | n | p | $\lambda_2$ |
|---|---|---|---|
| w7a | $24,692$ | $300$ | $1/n$ |
| ijcnn1 | $49,990$ | $22$ | $1/n$ |
| a9a | $32,561$ | $123$ | $1/n$ |
| covtype | $581,012$ | $54$ | $1$ |
| rcv1 | $20,242$ | $47,236$ | $10^{-4}$ |
| YearPredictionMSD | $463,715$ | $90$ | $10^5/n$ |

**Maintenance of Cumulative Distribution Function:** In the last line of Algorithm 3, we normalize $\{p_i\}_{i=1}^n$ to make it a distribution. However, such operation takes $\mathcal{O}(n)$ operations, which is quite expensive. To alleviate such high cost, we use the Vitter algorithm [Vitter, 1987] to maintain the Cumulative Distribution Function (CDF) of the sampling probability $\{p_i\}_{i=1}^n$ to avoid such normalization procedure. Since we update only one probability per iteration, the maintain operation can be done in $\mathcal{O}(\log n)$ by this algorithm. Additionally, the sampling operation can be done with the same complexity. An alternative strategy is [Nesterov, 2012] .

**Amortized Per Iteration Complexity** Updating all $\alpha_i$'s, exactly computing AP, and constructing the Hoffman tree for CDF take $\mathcal{O}(nd+n \log n)$ operations, which add $\mathcal{O}(d+\log n)$ operation to per iteration of HVRG in average, since they are performed only every $cn$ iterations. Addtionally, it takes $\mathcal{O}(\log n)$ operations to maintain and sample from the CDF as we discussed above. Thus the amortized per iteration complexity is $\mathcal{O}(d + \log n)$.

## 6 Experiment

In this section, we present results of several numerical experiments to validate our theoretical analyses of AdaSVRG and AdaSAGA and to show the empirical efficiency of HVRG. Experiments on $l_2$-Logistic Regression, $l_1l_2$-Logistic Regression, and Ridge Regression are conducted. We use datasets from LIBSVM [Chang and Lin, 2011] and list their statistics in Table 1. The parameters of the $l_2$-regularization are also included in that table. In our experiments, no normalization is performed on the data. We added an additional feature to all samples to represent the bias term, which is a common practice. SVRG, IProx-SVRG, SAGA[2], and AdaSDCA+ are included in our experiments for comparison. Since AdaSVRG and AdaSAGA have high per iteration computation load, we only test them on small problems (w7a, ijcnn1, and a9a) to validate our theoretical analysis. The parameter $m$ in SVRG, IProx-SVRG, and AdaSVRG is set to $2n$ uniformly, as suggested in [Xiao and Zhang, 2014]. We tune the step size (typically from $\frac{1}{4L}$ to $\frac{1}{L}$) for different methods so that they give the best performance. The parameters $c$ and $\rho$ in HVRG are fixed to 5 and 1.5 respectively. As for initialization, $\mathbf{w}_0$ is set to zero in all experiments. We define the *log-suboptimality* at

[2]SAGA-NUS [Schmidt *et al.*, 2015] is omitted for its poor performance in our experiments

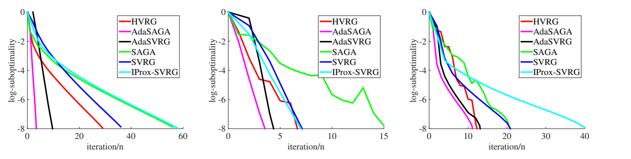

Figure 1: $L_2$ Logistic Regression. Left to Right, w7a, ijcnn1, and a9a. X-axis is iteration.

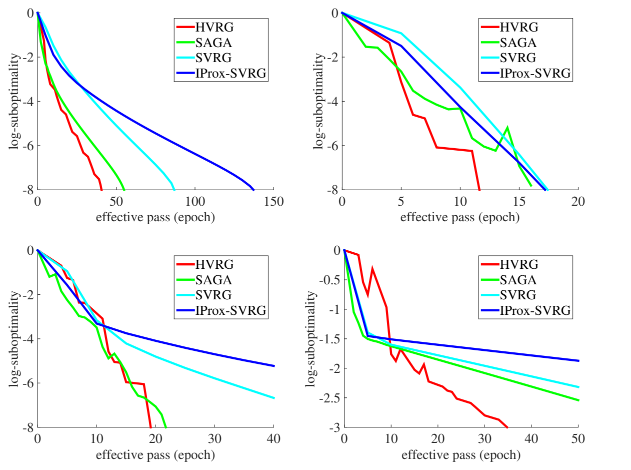

Figure 2: $L_2$ Logistic Regression. From left to right, up to down, we have w7a, ijcnn1, a9a, and covtype. X-axis is effective pass over dataset.

$\mathbf{w}$ as $\log_{10}\frac{\mathbf{P}(\mathbf{w})-\mathbf{P}(\mathbf{w}^*)}{\mathbf{P}(\mathbf{w}_0)-\mathbf{P}(\mathbf{w}^*)}$ and the *effective pass* as the evaluation of $n$ component gradients. These quantities are used to evaluate the performance of algorithms. Due to the randomness of the algorithms, the reported results are the average of 10 independent trials.

### 6.1 $l_2$-Logistic Regression

Four datasets are used in $l_2$-Logistic Regression, namely w7a, ijcnn1, a9a, and covtype. We set $f_i(\mathbf{w}) = \log(1 + \exp(-y_i\mathbf{X}_i^\top\mathbf{w})) + \frac{\lambda_2}{2}\|\mathbf{w}\|^2$ and $\mathbf{G}(\mathbf{w}) = 0$, the same as that in [Xiao and Zhang, 2014]. First, we compare the convergence rate of different algorithms in Figure 1. The results show that AdaSAGA and AdaSVRG are the best, conforming to our analysis. We also report the log-suboptimality over the effective pass over the dataset in Figure 2. HVRG either matches or outperforms all the other methods.

### 6.2 $l_1l_2$-Logistic Regression

The rcv1 dataset [Lewis *et al.*, 2004] is used to test the performance of HVRG in $l_1l_2$-Logistic Regression, which is a problem with non-zero regularization. We set $f_i(\mathbf{w}) = \log(1 + \exp(-y_i\mathbf{X}_i^\top\mathbf{w})) + \frac{\lambda_2}{2}\|\mathbf{w}\|^2$, $\mathbf{G}(\mathbf{w}) = \lambda_1\|\mathbf{w}\|$, and $\lambda_1 = 10^{-5}$, as suggested in [Xiao and Zhang, 2014]. A critical criterion in such problem is the number of non-zero entries in the variable. We include this quantity in comparison in addition to the log-suboptimality (see Figure 3). HVRG

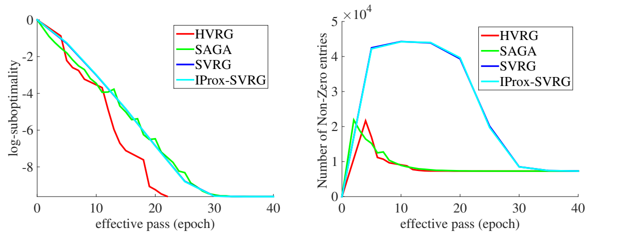

Figure 3: $l_1l_2$-Logistic Regression. X-axis is effective pass over dataset.

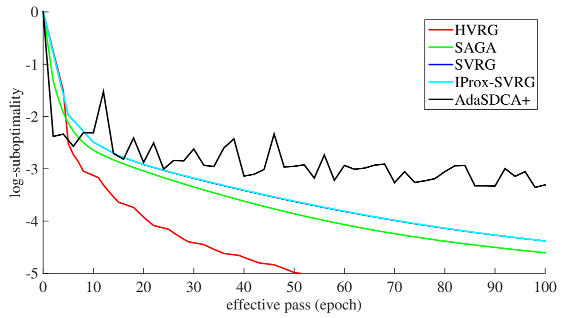

Figure 4: Ridge Regression. X-axis is effective pass over dataset.

uses the least effective passes to obtain a high accuracy solution and to correctly identify the support of $\mathbf{w}^*$.

### 6.3 Ridge Regression

Since AdaSDCA+ can only handle quadratic loss, we include it in comparison by conducting Ridge Regression on the YearPredictionMSD dataset. We set $f_i(\mathbf{w}) = \frac{1}{2}\|\mathbf{X}_i^\top\mathbf{w} - \mathbf{y}_i\|^2 + \frac{\lambda_2}{2}\|\mathbf{w}\|^2$ and $\mathbf{G}(\mathbf{w}) = 0$. The results are provided in Figure 4. Since the condition number is large in this problem, all algorithms converge slowly. AdaSDCA+ does not seem to converge in primal, although we do find its dual value ascending monotonically. We believe that this has to do with the large condition number. It is clear that HVRG outperforms all the other methods substantially.

## 7 Conclusion

In this paper, we propose a new adaptive sampling scheme called AP for SVRG and SAGA, and devise two adaptive alternatives, namely AdaSVRG and AdaSAGA, with provably better convergence rate. We validate our analysis with experiments. Additionally, to cut down the per iteration computation load, an efficient variant called HVRG is proposed, whose performance is empirically demonstrated.

## Acknowledgment

This work is partially supported by National Natural Science Foundation of China (Grant No: 61472347).

## References

[Beck and Teboulle, 2009] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

[Bottou, 2010] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[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*, 2:27:1–27:27, 2011.

[Chen and Rockafellar, 1997] George HG Chen and RT Rockafellar. Convergence rates in forward–backward splitting. *SIAM Journal on Optimization*, 7(2):421–444, 1997.

[Csiba *et al.*, 2015] Dominik Csiba, Zheng Qu, and Peter Richtárik. Stochastic dual coordinate ascent with adaptive probabilities. *arXiv preprint arXiv:1502.08053*, 2015.

[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 *Advances in Neural Information Processing Systems*, pages 1646–1654, 2014.

[Duchi *et al.*, 2011] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159, 2011.

[Hastie *et al.*, 2005] Trevor Hastie, Robert Tibshirani, Jerome Friedman, and James Franklin. The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer*, 27(2):83–85, 2005.

[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*, pages 781–789, 2009.

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

[Konečnỳ and Richtárik, 2013] Jakub Konečnỳ and Peter Richtárik. Semi-stochastic gradient descent methods. *arXiv preprint arXiv:1312.1666*, 2013.

[Konecny *et al.*, 2015] Jakub Konecny, Jie Liu, Peter Richtárik, and Martin Takác. Mini-batch semi-stochastic gradient descent in the proximal setting. 2015.

[Lewis *et al.*, 2004] David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. Rcv1: A new benchmark collection for text categorization research. *The Journal of Machine Learning Research*, 5:361–397, 2004.

[Mahdavi *et al.*, 2013] Mehrdad Mahdavi, Lijun Zhang, and Rong Jin. Mixed optimization for smooth functions. In *Advances in Neural Information Processing Systems*, pages 674–682, 2013.

[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*, 22(2):341–362, 2012.

[Nesterov, 2013] Yu Nesterov. Gradient methods for minimizing composite functions. *Mathematical Programming*, 140(1):125–161, 2013.

[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.

[Schmidt *et al.*, 2015] Mark Schmidt, Reza Babanezhad, Mohamed Osama Ahmed, Aaron Defazio, Ann Clifton, and Anoop Sarkar. Non-uniform stochastic average gradient method for training conditional random fields. *arXiv preprint arXiv:1504.04406*, 2015.

[Shalev-Shwartz and Zhang, 2013a] Shai Shalev-Shwartz and Tong Zhang. Accelerated mini-batch stochastic dual coordinate ascent. In *Advances in Neural Information Processing Systems*, pages 378–385, 2013.

[Shalev-Shwartz and Zhang, 2013b] Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for regularized loss. *The Journal of Machine Learning Research*, 14(1):567–599, 2013.

[Shalev-Shwartz and Zhang, 2016] Shai Shalev-Shwartz and Tong Zhang. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. *Mathematical Programming*, 155(1-2):105–145, 2016.

[Vitter, 1987] Jeffrey Scott Vitter. Design and analysis of dynamic huffman codes. *Journal of the ACM (JACM)*, 34(4):825–845, 1987.

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

[Zhao and Zhang, 2014] Peilin Zhao and Tong Zhang. Stochastic optimization with importance sampling. *arXiv preprint arXiv:1401.2753*, 2014.