Noname manuscript No.
(will be inserted by the editor)

Variance Reduction on General Adaptive Stochastic Mirror Descent

Wenjie Li · Zhanyu Wang · Yichen Zhang · Guang Cheng

arXiv:2012.13760v3 [stat.ML] 29 Aug 2021

Received: date / Accepted: date

Abstract In this work, we investigate the idea of variance reduction by studying its properties with general adaptive mirror descent algorithms in the nonsmooth nonconvex finite-sum optimization problems. We propose a simple yet generalized framework for variance reduced adaptive mirror descent algorithms named SVRAMD and provide its convergence analysis in both the nonsmooth nonconvex problem and the P-L conditioned problem. We prove that variance reduction reduces the SFO complexity of adaptive mirror descent algorithms and thus accelerates their convergence. In particular, our general theory implies that variance reduction can be applied to algorithms using time-varying step sizes and self-adaptive algorithms such as AdaGrad and RMSProp. Moreover, the convergence rates of SVRAMD recover the best existing rates of non-adaptive variance reduced mirror descent algorithms without complicated algorithmic components. Extensive experiments in deep learning validate our theoretical findings.

Keywords Variance Reduction · Adaptive Mirror Descent · Nonconvex Nonsmooth Optimization · General Framework · Convergence Analysis

1 Introduction

In this work, we study the nonsmooth nonconvex finite sum problem

$$\min_{x\in\mathcal{X}} F(x) := f(x) + h(x)$$

Correspondence to Wenjie Li

Wenjie Li
E-mail: li3549@purdue.edu Department of Statistics, Purdue University

Zhanyu Wang
E-mail: wang4094@purdue.edu Department of Statistics, Purdue University

Yichen Zhang
E-mail: zhang@purdue.edu Krannert School of Management, Purdue University

Guang Cheng
E-mail: chengg@purdue.edu Department of Statistics, Purdue University

where $f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x)$ and each f_i is a smooth but possibly nonconvex function, and $h(x)$ is a nonsmooth but convex function, for example, the L_1 regularization. Recently, the smooth version of the problem has been thoroughly studied, i.e., when $h(x) = 0$. Since it is difficult to determine the global minimum of a nonconvex function, the convergence analyses of different algorithms have focused on the gradient complexity of finding the first order stationary point of the loss $F(x)$, i.e., $\|\nabla F(x)\|_2^2 \leq \epsilon$. To reduce the gradient complexity of gradient descent and stochastic gradient descent (SGD), the famous Stochastic Variance Reduced Gradient method (SVRG) (Johnson and Zhang, 2013) and its popular variants have been proposed, such as SAGA (Defazio et al., 2014), SCSG (Lei et al., 2017), SNVRG (Zhou et al., 2018b), SPIDER (Fang et al., 2018), stablized SVRG (Ge et al., 2019), and Natasha momentum variants (Allen-Zhu, 2017a,b). These algorithms are proven to accelerate the convergence of SGD substantially.

When it comes to the nonsmooth case, a few algorithms based on the mirror descent algorithm (Beck and Teboulle, 2003; Duchi et al., 2010) have been studied recently. For example, Ghadimi et al. (2016) provided the convergence rate of Proximal Gradient Descent (ProxGD), Proximal SGD(ProxSGD), and Stochastic Mirror Descent (SMD) when the sample size was sufficiently large. The gradient complexity of these algorithms were shown to match the original algorithms without the proximal operation (Ghadimi et al., 2016). Reddi et al. (2016b) proposed ProxSVRG and ProxSAGA and analyzed their convergence rates, which were the proximal variants of SVRG and SAGA respectively. Li and Li (2018) proposed ProxSVRG+ and obtained even faster convergence than ProxSVRG with favorable constant minibatch sizes.

However, all the above extensions of SGD and SVRG, both for the smooth and the nonsmooth optimization problems, do not consider the case when the algorithm becomes adaptive, for example, when the step sizes are not fixed or even when the proximal functions in mirror descent are not fixed. Adaptive step sizes, such as linear decay, warm up (Goyal et al., 2017), and restart (Loshchilov and Hutter, 2016) are frequently used in the training of neural networks in practice, but recent papers of variance reduced algorithms only use constant step sizes in their analyses (Li and Li, 2018; Zhou et al., 2018b). Moreover, adaptive algorithms such as AdaGrad (Duchi et al., 2011), Adam (Kingma and Ba, 2015) and AMSGrad (Reddi et al., 2018) have become popular in application, but their proximal functions change with time. No theoretical guarantees of applying variance reduction to such adaptive algorithms have been shown before.

Therefore, instead of trying to create even faster algorithms in the nonsmooth setting with special components, this work addresses a more important and interesting question: *Can the simplest variance reduction technique accelerate the convergence of all or most of the adaptive stochastic mirror descent algorithms?* We give an affirmative answer to this question, with the only additional mild requirement that there is a lower bound for the strong convexity of the proximal functions in the mirror descent algorithms.

In particular, we highlight the following contributions:

1. We propose a simple and general variance reduction algorithm framework for most adaptive mirror descent algorithms named SVRAMD. We prove that in both the nonsmooth nonconvex problems and the gradient dominated problems (P-L condition, see Section 4.4), the SVRAMD algorithm converges faster than

the original adaptive mirror descent algorithms with a mild assumption on the convexity of the proximal functions.
2. Our general theory implies many interesting results. For example, we claim that time-varying step sizes are allowed for ProxSVRG+ (and many other variance reduced algorithms). As long as the step sizes are upper bounded by the inverse of the smoothness constant $1/L$, ProxSVRG+ still converges faster than ProxSGD under the same step size schedule. Also, we claim that variance reduction can work well with self-adaptive algorithms, such as AdaGrad (Duchi et al., 2011) and RMSProp (Tieleman and Hinton, 2012) (zero momentum version of Adam) and make their convergence faster. Moreover, our choices of the hyper-parameters in the theorem provide a general intuition that larger batch sizes are needed when using variance reduction on adaptive mirror descent algorithms with weaker convexity, which can be helpful when tuning the batch sizes in practice.
3. We examine the correctness of our claims thoroughly using popular deep learning models on the MNIST (Schölkopf and Smola, 2002), the CIFAR-10 and the CIFAR-100 (Krizhevsky et al., 2009) datasets. All the experimental results are consistent with our theoretical findings.

2 Additional Related Work

Due to the huge amount of work in optimization algorithms and their analyses, we are not able to review all of them in this paper. Instead, we choose to review the two additional lines of research that are related to this paper.

2.1 Self-adaptive Algorithms and Their Analysis

Since the creation of AdaGrad (Duchi et al., 2011), self-adaptive algorithms have become popular choices for optimization, especially in deep learning topics such as Natural Language Processing (NLP) and training generative adversarial models (Ghadimi et al., 2016). Kingma and Ba (2015) proposed Adam that combined momentum with RMSProp (Tieleman and Hinton, 2012) to further improve the convergence speed of AdaGrad, but Reddi et al. (2016b) proved that Adam could diverge even in convex problems. Wilson et al. (2017) also showed that the generalization performance of adaptive algorithms was even worse than SGD. Therefore, several complicated variants of Adam have been proposed recently, such as AdaBound (Luo et al., 2019), NosAdam (Huang et al., 2019), Radam (Liu et al., 2019), AdaBelief (Zhuang et al., 2020) and AdaX (Li et al., 2020) to further improve the convergence and the generalization performance of Adam. However, all the aforementioned papers only proved their fast convergence in convex problems and with sparse gradients. Recently, Chen et al. (2019) and Zhou et al. (2018a) proved the convergence rates of some self-adaptive algorithms (AMSGrad, AdaGrad) in the nonconvex smooth problem, but their convergence rates are still the same as that of SGD.

2.2 Combining Variance Reduction with Mirror Descent Algorithms

Combining variance reduction with mirror descent algorithms has become another heated topic. Recently, Lei and Jordan (2019) further extended SCSG to the general mirror descent form with some additional assumptions on the (fixed) proximal function, but they only considered a convex problem setting. Liu et al. (2020) proposed Adam+ and claimed that variance reduction could improve the convergence rate of a special variant of Adam. Dubois-Taine et al. (2021) proposed to use AdaGrad in the inner loop of SVRG to make it robust to different step size choices.

3 Problem Setup and Notations

In this section, we present the preliminary notations and the concepts used for the convergence analysis throughout this paper.

3.1 Notations

We first present the notations we use. For two matrices $A, B \in \mathbb{R}^{d\times d}$, we use $A \succeq B$ to denote that the matrix $A - B$ is positive semi-definite. For two real numbers $a, b \in \mathbb{R}$, we use $a \wedge b, a \vee b$ as short-hands for $\min(a, b)$ and $\max(a, b)$. We use $\lfloor a \rfloor$ to denote the largest integer that is smaller than a. We use $\widetilde{O}(\cdot)$ to hide logarithm factors in big-O notations. Moreover, for an integer $n \in \mathbb{N}$, we frequently use the notation $[n]$ to represent the set $\{1, 2, \cdots, n\}$. For the loss function $F(x)$, we denote the global minimum value of $F(x)$ to be F^*, and define $\Delta_F = F(x_1) - F^*$, where x_1 is the initialization point of the algorithm.

3.2 Assumptions and Definitions for Nonconvex Nonsmooth Analysis

We recall the update rule of the general stochastic mirror descent (SMD) algorithm with *adaptive proximal functions* $\psi_t(x)$ as follows

$$x_{t+1} = \text{argmin}_x \left\{\alpha_t \langle g_t, x\rangle + \alpha_t h(x) + B_{\psi_t}(x, x_t)\right\} \tag{1}$$

where α_t is the step size, $g_t = \nabla f_{\mathcal{I}_j}(x_t)$ is the gradient from a random data batch $\mathcal{I}_j$, and $h(x)$ is the regularization function on the dual space. $B_{\psi_t}(x, x_t)$ is the Bregman divergence with respect to the proximal function $\psi_t(x)$, defined as

$$B_{\psi_t}(x, y) = \psi_t(x) - \psi_t(y) - \langle \nabla\psi_t(y), x - y\rangle$$

Different $\psi_t(x)$ would generate different Bregman divergences. For example, the update rule for ProxSGD is

$$x_{t+1} = \text{argmin}_x \left\{\alpha_t \langle g_t, x\rangle + \alpha_t h(x) + \frac{1}{2}\|x - x_t\|_2^2)\right\}$$

In this case, $B_{\psi_t}(x, y) = \frac{1}{2}\|x - y\|_2^2$ and it is generated by the proximal function $\psi_t(x) = \frac{1}{2}\|x\|_2^2$. A very important property for Bregman divergence is that $\psi_t(x)$ is m-strongly convex if and only if $B_{\psi_t}(x, y) \geq \frac{m}{2}\|y - x\|_2^2$. In this work, we

consider general adaptive mirror descent algorithms whose proximal functions are undetermined and can vary with respect to time. However, we require that they are all m-strongly convex for some real constant $m > 0$ (Assumption **1**) ,

Assumption 1 *The proximal functions $\psi_t(x)$ are all m-strongly convex with respect to $\|\cdot\|_2$, i.e.,*

$$\psi_t(y) \geq \psi_t(x) + \langle \nabla\psi_t(x), y - x\rangle + \frac{m}{2}\|y - x\|_2^2, \forall t > 0$$

The constant m can be viewed as a lower bound of the strong convexity of all the proximal functions $\{\psi_t(x)\}_{t=1}^T$, where T is the total number of iterations, and therefore Assumption **1** is mild. Here we provide a few examples of different proximal functions that satisfy Assumption **1** and the corresponding lower bound to show the cases where our general theory can be applied.

Example 1 $\psi_t(x) = \phi_t(x) + \frac{c}{2}\|x\|_2^2, c > 0$, where each $\phi_t(x)$ is an arbitrary convex differentiable function, then $m = c$. Note that this case also reduces to the ProxSGD algorithm when $\phi_t(x) = 0$ and $c = 1$.

Example 2 $\psi_t(x) = \frac{c_t}{2}\|x\|_2^2$, where $c_t \geq c > 0, \forall t \in [T]$, then $m = c$. This case is equivalent to using time-varying step sizes in ProxSGD even when α_t is fixed, as we can divide all terms in Eqn. (1) by c_t simultaneously. In other words, we only require the time-varying step sizes (α_t/c_t) to be upper bounded to satisfy Assumption **1**.

Example 3 $\psi_t(x) = \frac{1}{2}\langle x, H_t x\rangle$, where $H_t \in \mathbb{R}^{d\times d}$ and $H_t \succeq cI, \forall t \in [T]$, then $m = c$. This case covers all the adaptive algorithms with a lower bound for the matrix H_t. Such a lower bound can be achieved by adding mI to H_t when it is a non-negative diagonal matrix or by assuming a lower bound of the gradient sizes. More details will be discussed in Section 4.

For the functions $\{f_i\}_{i=1}^n$, we assume that their gradients are all L-smooth, unbiased with mean ∇f, and have bounded variance σ^2, which are all standard assumptions in the nonconvex optimization analysis literature (Ghadimi et al., 2016; Reddi et al., 2016b; Li and Li, 2018).

Assumption 2 *Each function f_i is L-smooth, i.e.,*

$$\|\nabla f_i(x) - \nabla f_i(y)\|_2 \leq L\|x - y\|_2$$

Assumption 3 *Each $f_i(x)$ has unbiased stochastic gradients with bounded variance σ^2, i.e.,*

$$\mathbb{E}_{i\sim[n]}\left[\nabla f_i(x)\right] = \nabla f(x), \mathbb{E}_{i\sim[n]}\left[\|\nabla f_i(x) - \nabla f(x)\|_2^2\right] \leq \sigma^2$$

The convergence rates of different algorithms in the nonconvex optimization problem is usually measured by the stationarity of the gradient $\nabla f(x)$, i.e., $\mathbb{E}[\|\nabla f(x)\|^2] \leq \epsilon$ [1]. However, due to the existence of $h(x)$ in the nonsmooth setting, such a definition is no longer intuitive as it does not indicate that the algorithm is stationary anymore. Instead, we use a more general definition of the generalized gradient and the related convergence criterion.

[1] Some recent works such as Zhou et al. (2018b) use $\mathbb{E}[\|\nabla f(x)\|^2] \leq \epsilon^2$ instead of our choice. Simply replacing all the ϵ in our results by ϵ^2 can generate the conclusions in their settings.

Definition 1 Given the x_t generated by Eqn. (1), the generalized gradient $g_{X,t}$ is defined as

$$g_{X,t} = \frac{1}{\alpha_t}(x_t - x_{t+1}^+), \text{ where } x_{t+1}^+ = \text{argmin}_x \{\alpha_t \langle \nabla f(x_t), x\rangle + \alpha_t h(x) + B_{\psi_t}(x, x_t)\}$$

Correspondingly, we change the convergence criterion into the stationarity of the generalized gradient $\mathbb{E}[\|g_{X,t^*}\|^2] \leq \epsilon$. In Definition 1, we replace the stochastic gradient g_t in Eqn. (1) by the full-batch gradient $\nabla f(x_t)$. In other words, x_{t+1}^+ is the next-step parameter if we use the full batch gradient $\nabla f(x_t)$ for mirror descent at time t, and $g_{X,t}$ is the difference between x_t and x_{t+1}^+ scaled by the step size. Therefore, the generalized gradient is small when x_{t+1}^+ and x_t are close enough and thus the algorithm converges. We emphasize that Definition 1 and the convergence criterion are commonly observed in the mirror descent algorithm literature such as Ghadimi et al. (2016); Li and Li (2018). If $\psi_t(x) = \frac{1}{2}\|x\|_2^2$, then the definition is equivalent to the generalized gradient in Li and Li (2018). If we further assume $h(x)$ is a constant, then Definition 1 reduces to the original full batch gradient $\nabla f(x)$. When $h(x)$ is a constant and $\psi_t(x) = \frac{1}{2}\langle x, H_t x\rangle$, the generalized gradient is equivalent to $H_t^{-1}\nabla f(x)$, which is the update rule of self-adaptive algorithms.

3.3 Assumptions and Definitions for P-L condition

Next, we present the additional assumptions and definitions needed for the convergence analysis under the generalized Polyak-Lojasiewicz (P-L) condition (Polyak, 1963), which is also known as the gradient dominant condition. The original P-L condition (in smooth problems $h(x) = 0$) is defined as

$$\exists \mu > 0, \ s.t. \ \|\nabla F(x)\|^2 \geq 2\mu(F(x) - F^*), \ \forall x \in \mathbb{R}^d$$

which is even weaker than restricted strong convexity and it has been studied in many nonconvex convergence analyses such as Reddi et al. (2016a); Zhou et al. (2018b); Lei et al. (2017). Note that P-L condition implies all stationary points are global minimums. Therefore the original convergence criterion ($\mathbb{E}[\|\nabla F(x)\|^2] \leq \epsilon$) is equivalent to

$$2\mu \left[\mathbb{E}[F(x)] - F^*\right] \leq \epsilon$$

However, because of the existence of $h(x)$ in nonsmooth problems, we utilize the definition of the generalized gradient $g_{X,t}$ to define the generalized P-L condition as follows.

Definition 2 The loss function $F(x)$ satisfies the generalized P-L condition if

$$\exists \mu > 0, \ s.t. \ \|g_{X,t}\|^2 \geq 2\mu \left(F(x) - F^*\right), \ \forall x \in \mathbb{R}^d$$

where $g_{X,t}$ is the generalized gradient defined in Definition 1.

Li and Li (2018) used a similar definition of the general P-L condition, and ours is a natural extension since the proximal functions in Eqn. 1 are undetermined. The above definition reduces to the P-L condition by Li and Li (2018) when $\psi_t(x) = \frac{1}{2}\|x\|_2^2$. If we further assume that $h(x)$ is a constant, then Definition 2 is the same as the original P-L condition. For simplicity, we further assume that the constant

Table 1: Comparisons of the SFO complexity of different algorithms to reach ϵ-stationary point of the generalized gradient. n is the total number of samples and b is the mini-batch size. $\widetilde{O}$ notation omits the logarithm term $\log \frac{1}{\epsilon}$

Algorithms	Nonconvex Nonsmooth	P-L condition
ProxGD (Ghadimi et al., 2016)	$O\left(\frac{n}{\epsilon}\right)$	$\widetilde{O}\left(\frac{n}{\mu}\right)$
ProxSVRG (Reddi et al., 2016b)	$O\left(\frac{n}{\epsilon\sqrt{b}}+n\right)$	$\widetilde{O}\left(\frac{n}{\mu\sqrt{b}}+n\right)$
SCSG (Lei et al., 2017)	$O\left(\frac{n^{2/3}b^{1/3}}{\epsilon}\wedge\frac{b^{1/3}}{\epsilon^{5/3}}\right)$	$\widetilde{O}\left(\frac{nb^{1/3}}{\mu}\wedge\frac{b^{1/3}}{\mu^{5/3}\epsilon^{2/3}}+n\wedge\frac{1}{\mu\epsilon}\right)$
ProxSVRG+ (Li and Li, 2018)	$O\left(\frac{n}{\epsilon\sqrt{b}}\wedge\frac{1}{\epsilon^2\sqrt{b}}+\frac{b}{\epsilon}\right)$	$\widetilde{O}\left((n\wedge\frac{1}{\mu\epsilon})\frac{1}{\mu\sqrt{b}}+\frac{b}{\mu}\right)$
Adaptive SMD (Algorithm 1)	$O\left(\frac{n}{\epsilon}\wedge\frac{1}{\epsilon^2}\right)$	$\widetilde{O}\left(\frac{n}{\mu}\wedge\frac{1}{\mu^2\epsilon}\right)$
SVRAMD (Algorithm 2)	$O\left(\frac{n}{\epsilon\sqrt{b}}\wedge\frac{1}{\epsilon^2\sqrt{b}}+\frac{b}{\epsilon}\right)$	$\widetilde{O}\left((n\wedge\frac{1}{\mu\epsilon})\frac{1}{\mu\sqrt{b}}+\frac{b}{\mu}\right)$

μ satisfies $L/(m^2\mu)\geq\sqrt{n}$, which is similar to what Li and Li (2018); Reddi et al. (2016b) assumed in their papers. The condition is similar to the "high condition number regime" for strongly convex optimization problems and it is assumed only because we want to use the same step size $\alpha_t = m/L$ in all the theorems (especially for Theorem 2 and Theorem 4) in this paper. If it is not satisfied, we can simply use a more complicated step size setting as in the Appendix A.2 in Li and Li (2018).

3.4 Comparison Measure

We use the stochastic first-order oracle (SFO) complexity to compare the convergence rates of different algorithms. When given the parameter x, SFO returns one stochastic gradient $\nabla f_i(x)$. In other words, the SFO complexity measures the number of gradient computations in the optimization process. We summarize the SFO complexity of a few mirror descent algorithms in both the nonconvex nonsmooth setting and the generalized P-L condition in Table 1.

4 Algorithm and Convergence

In this section, we present our main theoretical claims. In subsection 4.1, we provide the convergence rate of the Adaptive SMD algorithm as a competitive baseline, which matches the best existing rates of non-adaptive mirror descent algorithms. In subsections 4.2, 4.3, and 4.4, we present our generalized variance reduced mirror descent algorithm framework SVRAMD and provide its convergence analysis to show that variance reduction can accelerate the convergence of most adaptive mirror descent algorithms in both the nonconvex nonsmooth problem and the generalized P-L conditioned problem.

4.1 The Adaptive Mirror Descent Algorithm and Its Convergence

We first establish the the SFO complexity of the general adaptive SMD algorithm (Algorithm 1) as the baseline, which is to our best knowledge, a new result in liter-

ature. We prove in both the nonconvex nonsmooth case and the P-L conditioned case, the convergence rates of Algorithm 1 are similar to those of the non-adaptive SMD algorithm (Ghadimi et al., 2016), (Karimi et al., 2016). The proof of these SFO complexities are provided in Appendix A.

Algorithm 1 General Adaptive SMD Algorithm

1: **Input:** Number of stages T, initial x_1, step sizes $\{\alpha_t\}_{t=1}^T$, proximal functions $\{\psi_t\}_{t=1}^T$
2: **for** $t = 1$ **to** T **do**
3: Randomly sample a batch $\mathcal{I}_t$ with size b
4: $g_t = \nabla f_{\mathcal{I}_t}(x_t)$
5: $x_{t+1} = \text{argmin}_x\{\alpha_t\langle g_t, x\rangle + \alpha_t h(x) + B_{\psi_t}(x, x_t)\}$
6: **end for**
7: **Return** Uniformly sample t^* from $\{t\}_{t=1}^T$ and ouput x_{t^*}

Theorem 1 *Suppose that $\psi_t(x)$ satisfies the m-strong convexity assumption (**1**), and f satisfies the Lipschitz gradients and bounded variance assumptions (**2, 3**). Further assume that the learning rate and the mini batch sizes are set to be $\alpha_t = m/L, b = n \wedge (12\sigma^2/(m^2\epsilon))$. Then the output of Algorithm 1 converges with gradient computations*

$$O\left(\frac{n}{\epsilon} \wedge \frac{\sigma^2}{\epsilon^2} + n \wedge \frac{\sigma^2}{\epsilon}\right)$$

Remark. The above complexity can be treated as $O(n\epsilon^{-1} \wedge \epsilon^{-2})$ and it is the same as the complexity of non-adaptive mirror descent algorithms by Ghadimi et al. (2016). Algorithm 1 needs a relatively large batch size ($O(\epsilon^{-1})$) to obtain a convergence rate close to that of GD ($O(n\epsilon^{-1})$) and SGD ($O(\epsilon^{-2})$) (Reddi et al., 2016a). However, it is still only asymptotically as fast as one of them, depending on the relationship between $O(\epsilon^{-1})$ and the sample size n.

We now provide the convergence rate of Algorithm 1 in the generalized P-L conditioned problem.

Theorem 2 *Suppose that $\psi_{tk}(x)$ satisfies the m-strong convexity assumption (**1**), and f satisfies the Lipschitz gradients and the bounded variance assumptions (**2, 3**). Further assume that the P-L condition (Definition 2) is satisfied. The learning rate and the mini-batch sizes are set to be $\alpha_t = m/L, b_t = n \wedge (2(1+m^2)\sigma^2/(\epsilon m^2\mu))$. Then the output of Algorithm 1 converges with gradient computations*

$$O\left((\frac{n}{\mu} \wedge \frac{\sigma^2}{\mu^2\epsilon})\log\frac{1}{\epsilon}\right)$$

Remarks. The above result is $\widetilde{O}(n\mu^{-1} \wedge \mu^{-2}\epsilon^{-1})$ if we hide the logarithm term. Similar to Theorem 1, the SFO complexity matches the smaller complexity of ProxSGD and ProxGD in the P-L conditioned problem (Karimi et al., 2016). We emphasize that since the general Algorithm 1 covers ProxSGD ($b = 1$) and ProxGD ($b = n$) with $\psi_t(x) = \frac{1}{2}\|x\|_2^2$, we believe that our convergence rates in Theorem 1 and Theorem 2 are the best rates achievable.

4.2 The General Variance Reduced Algorithm-SVRAMD

We now present the Stochastic Variance Reduced Adaptive Mirror Descent (SVRAMD) algorithm, which is a simple and generalized variance reduction framework for any mirror descent algorithm. We provide the pseudo-code in Algorithm 2. The input parameters B_t and b_t are called the batch sizes and the mini-batch sizes. The algorithm consists of two loops: the outer loop has T iterations while the inner loop has K iterations. At each iteration of the outer loop (line 4), we estimate a large batch gradient $g_t = \nabla f_{\mathcal{I}_t}(x_t)$ with batch size B_t at the reference point x_t, both of which will be stored during the inner loop. At each iteration of the inner loop (line 7), the large batch gradient is used to reduce the variance of the small batch gradient $\nabla f_{\widetilde{\mathcal{I}}_t}(y_k^t)$ so that the convergence becomes more stable. Note that in line 8, the proximal function is an undetermined $\psi_{tk}(x)$ which can change over time, therefore Algorithm 2 covers ProxSVRG+ and a lot more algorithms with different proximal functions. For example, if $\psi_{tk}(x) = \frac{1}{2}\|x\|_2^2$, then $B_{tk}(y, y_k^t) = \frac{1}{2}\|y - y_k^t\|_2^2$ and SVRAMD reduces to ProxSVRG+ (Li and Li, 2018). If $\psi_{tk}(x) = \frac{1}{2}\langle x, H_{tk}x\rangle$, where $H_{tk} \succeq mI$, then SVRAMD reduces to variance reduced self-adaptive algorithms for different matrices H_{tk}, e.g., VR-AdaGrad in Section 4.5.

Algorithm 2 Stochastic Variance Reduced Adaptive Mirror Descent (SVRAMD)

1: **Input:** Number of rounds T, initial $x_1 \in \mathbb{R}^d$, step sizes $\{\alpha_t\}_{t=1}^T$, batch, mini-batch sizes $\{B_t, b_t\}_{t=1}^T$, inner-loop iterations K, proximal functions $\{\psi_{tk}\}_{t=1,k=1}^{T,K}$
2: **for** $t = 1$ **to** T **do**
3: Randomly sample a batch $\mathcal{I}_t$ with size B_t
4: $g_t = \nabla f_{\mathcal{I}_t}(x_t)$; $y_1^t = x_t$
5: **for** $k = 1$ **to** K **do**
6: Randomly pick sample $\widetilde{\mathcal{I}}_t$ of size b_t
7: $v_k^t = \nabla f_{\widetilde{\mathcal{I}}_t}(y_k^t) - \nabla f_{\widetilde{\mathcal{I}}_t}(y_1^t) + g_t$
8: $y_{k+1}^t = \mathrm{argmin}_y\{\alpha_t\langle v_k^t, y\rangle + \alpha_t h(x) + B_{\psi_{tk}}(y, y_k^t)\}$
9: **end for**
10: $x_{t+1} = y_{K+1}^t$
11: **end for**
12: **Return** x_{t^*} where t^* is determined by
(Nonsmooth case) Uniformly sample t^* from $[T]$;
(P-L condition case) $t^* = T + 1$.

4.3 Nonconvex Nonsmooth Convergence

Now given the general form of Algorithm 2, we provide its SFO complexity in the nonconvex nonsmooth problem in the following theorem.

Theorem 3 *Suppose that $\psi_{tk}(x)$ satisfies the m-strong convexity assumption (**1**) and f satisfies the Lipschitz gradients and bounded variance assumptions (**2, 3**). Further assume that the learning rate, the batch sizes, the mini-batch sizes, and the number of inner-loop iterations are set to be $\alpha_t = m/L$, $B_t = n \wedge (20\sigma^2/m^2\epsilon)$, $b_t = b$, $K = \left\lfloor \sqrt{b/20} \right\rfloor \vee 1$. Then the output of Algorithm 2 converges with SFO com-*

plexity

$$O\left(\frac{n}{\epsilon\sqrt{b}} \wedge \frac{\sigma^2}{\epsilon^2\sqrt{b}} + \frac{b}{\epsilon}\right)$$

Remarks. The proof is provided in Appendix B. Theorem 3 essentially claims that with Assumptions **1, 2, 3**, we can guarantee the SFO complexity of Algorithm 2 is $O\left(\frac{n}{\epsilon\sqrt{b}} \wedge \frac{\sigma^2}{\epsilon^2\sqrt{b}} + \frac{b}{\epsilon}\right)$, with an undetermined mini-batch size b to be tuned later.

Although α_t is fixed in the theorem, ψ_{tk} can change over time and hence results in the adaptivity in the algorithm. For example, our theory indicates that using different designs of time-varying step sizes is allowed (Example 2 in Section 3). When we take the proximal function to be $\psi_{tk}(x) = \frac{c_{tk}}{2}\|x\|_2^2, c_{tk} \geq m$, Algorithm 2 reduces to ProxSVRG+ with time-varying effective step size α_t/c_{tk} (i.e., η_t in Li and Li (2018)). As long as the effective step sizes α_t/c_{tk} are upper bounded by a constant $(m/L)/m = 1/L$, Algorithm 2 still converges with the same complexity. Such a constraint is easy to satisfy for many step size schedules such as (linearly) decreasing step sizes, cyclic step sizes and warm up. Besides, ψ_{tk} can be even more complicated, such as Example 1 and 3 we have mentioned in Section 3, which will be discussed later. Another interesting result observed in our theorem is that when m is small, we require a relatively larger batch size B_t to guarantee the fast convergence rate. We will later show that this intuition is actually supported by our experiments in Section 5.

Similar to ProxSVRG+, when SCSG (Lei et al., 2017) and ProxSVRG (Reddi et al., 2016b) achieve their best convergence rate at either a too small or a too large mini-batch size, i.e., $b = 1$ or $b = n^{2/3}$, our algorithm achieves its best performance using a moderate mini-batch size $\epsilon^{-2/3}$. We provide the following corollary.

Corollary 1 *Under all the assumptions and parameter settings in Theorem 3, further assume that $b = \epsilon^{-2/3}$, where $\epsilon^{-2/3} \leq n$. Then the output of Algorithm 2 converges with SFO complexity*

$$O\left(\frac{n}{\epsilon^{2/3}} \wedge \frac{1}{\epsilon^{5/3}} + \frac{1}{\epsilon^{5/3}}\right)$$

Remarks. The above SFO complexity is the same as the best SFO complexity of ProxSVRG+, and it is provably better than the SFO complexity of adaptive SMD in Theorem 1. Note that the number of samples is usually very large in modern datasets, such as $n = 10^6$, thus $b = \epsilon^{-2/3} \leq n$ can be easily satisfied with a constant mini-batch size and a small enough ϵ, such as $b = 10^2, \epsilon = 10^{-3}$. If the above best complexity cannot be achieved, meaning that either n or ϵ is too small, some sub-optimal solutions are still available for fast convergence. For example, setting $b = n^{2/3}$ would generate $O(n^{2/3}\epsilon^{-1})$ SFO complexity, which is also smaller than the SFO complexity of Algorithm 1. Therefore, we conclude that if the parameters are carefully chosen, variance reduction can reduce the SFO complexity and accelerate the convergence of any adaptive SMD algorithm that satisfies Assumption **1** in the nonsmooth nonconvex problem. We summarize the SFO complexity generated by a few different choices of b in the nonconvex nonsmooth problem in Table 2.

Table 2: Summary of the SFO complexity of SVRAMD given different mini-batch sizes b. All notations follow Table 1. SVRAMD is much faster than adaptive SMD in the middle three cases, i.e., $b = \epsilon^{-2/3}$, $b = \epsilon^{-1}$ and $b = n^{2/3}$. It is asymptotically at most as fast as adaptive SMD when the mini-batch size is either too small or too big, i.e., $b = 1$ and $b = n$.

Mini-batch	Nonconvex Nonsmooth	P-L condition
$b = 1$	$O\left(\frac{n}{\epsilon} \wedge \frac{1}{\epsilon^2} + \frac{n}{\epsilon}\right)$	$\widetilde{O}\left((\frac{n}{\mu} \wedge \frac{1}{\mu^2\epsilon})\right)$
$b = \epsilon^{-2/3}$	$O\left(\frac{n}{\epsilon^{2/3}} \wedge \frac{1}{\epsilon^{5/3}} + \frac{1}{\epsilon^{5/3}}\right)$	$\widetilde{O}\left((\frac{n\epsilon^{1/3}}{\mu^{2/3}} \wedge \frac{\epsilon^{-2/3}}{\mu^{5/3}} + \frac{\epsilon^{-2/3}}{\mu^{5/3}})\right)$
$b = \epsilon^{-1}$	$O\left(\frac{n}{\epsilon^{1/2}} \wedge \frac{1}{\epsilon^{3/2}} + \frac{1}{\epsilon^2}\right)$	$\widetilde{O}\left(\frac{n\epsilon^{1/2}}{\mu} \wedge \frac{1}{\mu^2\epsilon^{1/2}} + \frac{1}{\mu\epsilon}\right)$
$b = n^{2/3}$	$O\left(\frac{n^{2/3}}{\epsilon} \wedge \frac{1}{\epsilon^2 n^{1/3}} + \frac{n^{2/3}}{\epsilon}\right)$	$\widetilde{O}\left((\frac{n^{2/3}}{\mu} \wedge \frac{1}{\mu^2\epsilon}) + \frac{n^{2/3}}{\mu}\right)$
$b = n$	$O\left(\frac{\sqrt{n}}{\epsilon} \wedge \frac{1}{\epsilon^2\sqrt{n}} + \frac{n}{\epsilon}\right)$	$\widetilde{O}\left((\frac{\sqrt{n}}{\mu} \wedge \frac{1}{\mu^2\epsilon}) + \frac{n}{\mu}\right)$

4.4 Convergence under the Generalized P-L Condition

Now we provide the convergence of Algorithm 2 under the generalized P-L condition (Definition 2). Note that the only difference in Algorithm 2 in such problems is the final output, where we directly use the last x_{T+1} rather than randomly sample from the historical x_ts. Therefore, we preserve the simplicity of our algorithm in the nonconvex nonsmooth case, i.e., there is no additional algorithmic components, such as the restart process in ProxSVRG (Reddi et al., 2016b). Now we present the SFO complexity of our variance reduced algorithm under the generalized P-L condition and show its linear convergence rate.

Theorem 4 *Suppose that $\psi_{tk}(x)$ satisfies the m-strong convexity assumption (**1**) and f satisfies the Lipschitz gradients and bounded variance assumptions (**2, 3**). Further assume that the P-L condition (2) is satisfied. The learning rate, the batch sizes, the mini-batch sizes, and the number of inner loop iterations are set to be $\alpha_t = m/L,\ B_t = n \wedge (10\sigma^2/(\epsilon m^2 \mu)),\ b_t = b,\ K = \lfloor\sqrt{b/32}\rfloor \vee 1$. Then the output of Algorithm 2 converges with SFO complexity*

$$O\left((n \wedge \frac{\sigma^2}{\mu\epsilon})\frac{1}{\mu\sqrt{b}}\log\frac{1}{\epsilon} + \frac{b}{\mu}\log\frac{1}{\epsilon}\right)$$

Remarks. The proof is provided in Appendix C. The above SFO complexity is of size $\widetilde{O}((n \wedge (\mu\epsilon)^{-1})(\mu\sqrt{b})^{-1} + b\mu^{-1})$ if we hide the logarithm terms. Compared with the complexity of the original adaptive SMD algorithm, our complexity can be arguably smaller when we choose an appropriate mini-batch size b, which further proves our conclusion that variance reduction can be applied to most adaptive SMD algorithms to accelerate the convergence. We provide the following corollary for the best choice of b to show its effectiveness.

Corollary 2 *Under all the assumptions and parameter settings in Theorem 4, further assume that $b = (\mu\epsilon)^{-2/3} \leq n$. Then the output of Algorithm 2 converges with SFO complexity*

$$O\left((\frac{n\epsilon^{1/3}}{\mu^{2/3}} \wedge \frac{\epsilon^{-2/3}}{\mu^{5/3}} + \frac{\epsilon^{-2/3}}{\mu^{5/3}})\log\frac{1}{\epsilon}\right)$$

Remarks. The above complexity is the same as the best complexity of ProxSVRG+ and it is smaller than the complexity of adaptive SMD in Theorem 2. Moreover, it generalizes the best results of ProxSVRG/ProxSAGA and SCSG, without the need to perform any restart or use the stochastically controlled iterations trick. Therefore, Algorithm 2 automatically switches to fast convergence in regions satisfying the generalized P-L condition. We also provide the SFO complexity of some other choices of b in Table 2 for a clearer comparison. Similarly, if $b = (\mu\epsilon)^{-2/3}$ is impractical, using $b = n^{2/3}$ also generates faster convergence than adaptive SMD.

4.5 Extensions to Self-Adaptive Algorithms

Algorithm 3 Variance Reduced AdaGrad Algorithm

Input: Number of stages T, initial x_1, step sizes $\{\alpha_t\}_{t=1}^T$, batch sizes $\{B_t\}_{t=1}^T$, mini-batch sizes $\{b_t\}_{t=1}^T$, constant m
for $t = 1$ **to** T **do**
  Randomly sample a batch $\mathcal{I}_t$ with size B_t
  $g_t = \nabla f_{\mathcal{I}_t}(x_t)$
  $y_1^t = x_t$
  for $k = 1$ **to** K **do**
    Randomly pick sample $\tilde{\mathcal{I}}_t$ of size b_t
    $v_k^t = \nabla f_{\tilde{\mathcal{I}}_t}(y_k^t) - \nabla f_{\tilde{\mathcal{I}}_t}(y_1^t) + g_t$
    $H_{tk} = \mathrm{diag}(\sqrt{\sum_{\tau=1}^{t-1}\sum_{i=1}^{K} v_i^{\tau 2} + \sum_{i=1}^{k} v_i^{t2}} + m)$
    $y_{k+1}^t = \mathrm{argmin}_y\{\alpha_t\langle v_k^t, y\rangle + \alpha_t h(x) + \frac{1}{2}\langle y - y_k^t, H_{tk}(y - y_k^t)\rangle\}$
  end for
  $x_{t+1} = y_{K+1}^t$
end for
Return (Smooth case) Uniformly sample x_{t^*} from $\{y_k^t\}_{k=1,t=1}^{K,T}$; (P-L case) $x_{t^*} = x_{T+1}$

As we have mentioned in Section 3, self-adaptive algorithms such as AdaGrad are special cases of the general Algorithm 1 and thus we can use Algorithm 2 to accelerate them. The proximal functions of most adaptive algorithms have the form of $\psi_t(x) = \frac{1}{2}\langle x, H_t x\rangle$, where $H_t \in \mathbb{R}^{d\times d}$ is often designed to be a diagonal matrix.

Assumption **1** is satisfied if we consistently add a constant matrix mI to H_t, with I being the identity matrix. We remark that such an operation is commonly used in self-adaptive algorithms to avoid division by zero (Duchi et al., 2010; Kingma and Ba, 2015), and thus variance reduction can be applied. Assumption **1** can also be satisfied for many special designs of H_t with some mild assumptions. For example, if $H_{t,ii} = \sqrt{g_{1,i}^2 + g_{2,i}^2 + \cdots g_{t,i}^2}, \forall i \in [d]$ (the original AdaGrad algorithm), then $H_t \succeq mI$ if there exists one $\tau_i \in [t]$ on each dimension i such that $g_{\tau_i,i} \geq m$, meaning that there is at least one derivative larger than m on each dimension in the whole optimization process. If such a mild condition holds, then the conclusions in Theorem 3, 4 and Corollary 1, 2 still hold.

However, notice that the strong convexity lower bounds of these algorithms are relatively smaller (m is often set as 1e-3 or even smaller in real experiments),

Theorem 3 implies that the batch size B_t needs to be sufficiently large for these algorithm to converge fast. If variance reduction can work with such algorithms with small m, then we should expect good performances with the other algorithms that have large m's. We provide the implementation for Variance Reduced AdaGrad (VR-AdaGrad) in Algorithm 4 and Variance Reduced RMSProp (VR-RMSProp) is similar.

One interesting question here is whether VR-AdaGrad and VR-RMSProp can converge even faster than ProxSVRG+, since AdaGrad and RMSProp are known to converge faster than SGD in convex problems. Our conjecture is that they cannot do so, at least not in the nonconvex case because their rapid convergence relies on not only the convexity of the problem, but also the sparsity of the gradients (Duchi et al., 2011). Neither of these two assumptions is appropriate in the nonconvex analysis framework.

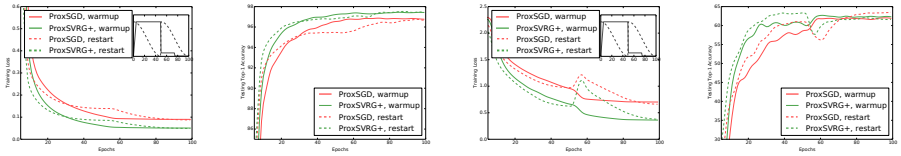
5 Experiments

In this section, we present several experiments on neural networks to show the effectiveness of variance reduction in the adaptive SMD algorithms. We train a two-layer fully connected network on the MNIST dataset, the LeNet (LeCun et al., 1998) on the CIFAR-10 dataset, and the ResNet-20 model on the CIFAR-100 dataset (He et al., 2016).

Implementations. In general, we follow the settings by Zhou et al. (2018b) on SNVRG for all our experiments. Except for normalization, we do not perform any additional data transformation or augmentation techniques such as rotation, flipping, and cropping on the images. For the batch sizes and mini batch sizes B_t and b_t in the algorithms, we follow Zhou et al. (2018b) to use a slightly different but equivalent notation of batch size ratio $r = B_t/b_t$. Note that increasing this ratio when b_t is fixed would be the same as increasing the batch size B_t. Our code is based on the publicly released PyTorch code by yueqiw (2019). All experiments here are run on Nvidia V100 GPUs. We have thoroughly tuned the hyper-parameters for all the algorithms on the different datasets. More details of our experiments can be found in Appendix D.

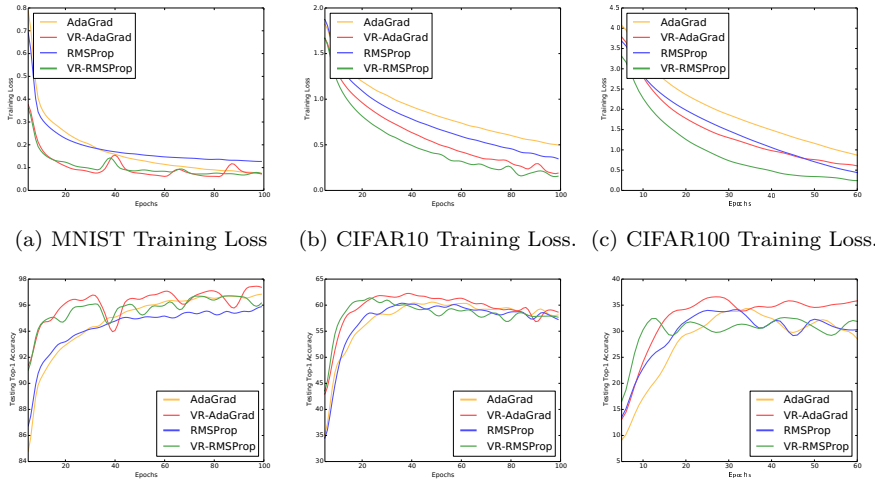
We first validate our claims in Section 4 by showing that variance reduction can work well with different step size schedules. We choose two special schedules that are both complicated enough and popular nowadays, warm up and warm restart. We choose to compare ProxSGD and ProxSVRG+ because they are special cases of Algorithm 1 and Algorithm 2 with proximal function $\frac{1}{2\alpha_t}\|x\|_2^2$, where α_t is the step size. For warm up, we increase the step size from zero to the best step size of ProxSGD/ProxSVRG+ linearly for the initial 5 epochs and then multiply the step size by 0.1 at the 50-th and the 75-th epoch. For warm restart, we decrease from the best step size to zero with cosine annealing in 50 epochs and then restart the decay. The training loss, the testing accuracy and the step size schedules on MNIST and CIFAR-10 are plotted in Figure 1. As can be seen in the figures, ProxSVRG+ converges faster than ProxSGD with both of the two special step size schedules, proving our claim that as long as the step sizes are bounded, variance reduction can still improve the convergence rate of ProxSGD with time-varying step sizes.

We then choose VR-AdaGrad and VR-RMSProp as the other two special examples of our general algorithm because they have relatively smaller lower bound of



(a) MNIST Training Loss (b) MNIST Testing Acc. (c) CIFAR10 Training Loss. (d) CIFAR10 Testing Acc.

Fig. 1: Performance of ProxSGD and ProxSVRG+ with different step size schedules. The solid lines are trained with the warmup schedule (warmup for 5 epochs then step decay) and the dashed lines are trained with the warm restart schedule. We plot the two schedules at the top-right corners of the training loss subfigures for the readers' reference. **(a) and (b)**: training loss and testing accuracy using fully connected network on MNIST. **(c) and (d)**: training loss and testing accuracy using LeNet on CIFAR-10. The results were averaged over 5 runs.

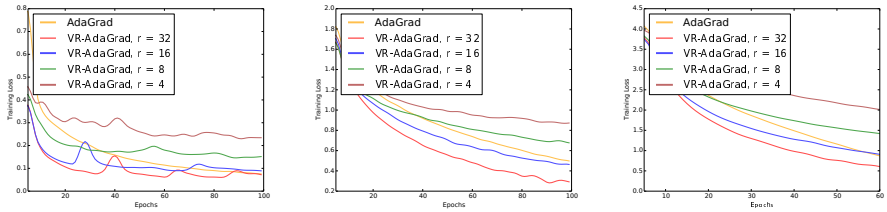


(a) MNIST Training Loss (b) CIFAR10 Training Loss. (c) CIFAR100 Training Loss.

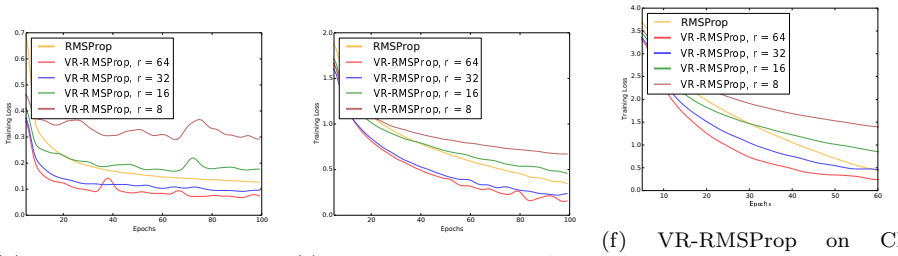
(d) MNIST Testing Acc. (e) CIFAR10 Testing Acc. (f) CIFAR100 Testing Acc.

Fig. 2: (a) and (d): training loss and testing accuracy on MNIST. (b) and (e): training loss and testing accuracy on CIFAR10. (c) and (f): training loss and testing accuracy on CIFAR100. The results are averaged over 5 runs.

the strong convexity and their proximal functions are even more complicated. We use the constant step size schedule for these algorithms as they are self-adaptive. The training loss and the testing Top-1 accuracy of VR-AdaGrad, VR-RMSProp, and their original algorithms on all the three datasets are plotted in Figure 2. As can be observed, the variance reduced algorithms converge faster than their original algorithms and their best testing Top-1 accuracy is also comparable or even higher, proving the effectiveness of variance reduction. We emphasize that the experiments are not designed to pursue the state-of-the-art performances, but to show that variance reduction can work well with any adaptive proximal functions



(a) VR-AdaGrad on MNIST (b) VR-AdaGrad on CIFAR10 (c) VR-AdaGrad on CIFAR100



(d) VR-RMSProp on MNIST (e) VR-RMSProp on CIFAR10 (f) VR-RMSProp on CIFAR100

Fig. 3: (a) and (d): training loss with different r on MNIST. (b) and (e): training loss with different r on CIFAR10 . (c) and (f): training loss with different ratio r on CIFAR100.

and contribute to faster training, even if the proximal functions have very small strong convexity lower bound.

Finally, we also show that algorithms with weaker convexity need a larger batch size B_t to converge fast, which corresponds to our claims in Section 4.3 and 4.4. We fix the mini batch sizes b_t in VR-AdaGrad and VR-RMSProp to be the same as in the previous experiments and gradually increase the batch size ratio r. The baseline ratios of ProxSVRG+, which has a large strong convexity lower bound, are provided in Appendix D for the readers' reference and the training performances of VR-AdaGrad and VR-RMSProp with different r are shown in Figure 3. Note that ProxSVRG+ only needs a small ratio ($r = 4$ or $r = 8$) to be faster than SGD (Li and Li, 2018), but for VR-RMSProp, even when $r = 16$, the algorithms still do not converge faster than RMSProp, proving that algorithms with weaker convexity need larger batch sizes B_t to show the effectiveness of variance reduction.

6 Conclusions

In this work, we generalize the convergence results of variance reduced algorithms to almost all adaptive stochastic mirror descent algorithms by proposing a general adaptive extension of ProxSVRG+. We prove that the variance reduction technique contributes to the same level of improvement in the convergence rates of adaptive mirror descent algorithms under a very mild assumption, both in the general nonsmooth nonconvex problems and the PL-conditioned problems. In particular, our theory can be applied to proving the soundness of using time-varying step sizes in the training process of ProxSVRG+, and the effectiveness of apply-

ing variance reduction to adaptive algorithms such as AdaGrad and RMSProp. A potential future work direction is whether our Assumption 1 can still be replaced by weaker conditions, such as the CHEF conditions Lei and Jordan (2019) have mentioned in the convex case. Another interesting direction is whether the SFO complexity can be further improved with the help of additional algorithmic components, for example, the nested variance reduction technique by Zhou et al. (2018b) or momentum accelerations.

7 Declarations

The authors have no conflicts of interest to declare that are relevant to the content of this article.

References

Allen-Zhu Z (2017a) Natasha: Faster non-convex stochastic optimization via strongly non-convex parameter. 1702.00763

Allen-Zhu Z (2017b) Natasha 2: Faster non-convex optimization than sgd. 1708.08694

Beck A, Teboulle M (2003) Mirror descent and nonlinear projected subgradient methods for convex optimization. Operations Research Letters

Chen X, Liu S, Sun R, Hong M (2019) On the convergence of a class of adam-type algorithm for non-convex optimization. Proceedings of 7th International Conference on Learning Representations(ICLR)

Defazio A, Bach F, Lacoste-Julien S (2014) Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. Advances in Neural Information Processing Systems 27

Dubois-Taine B, Vaswani S, Babanezhad R, Schmidt M, Lacoste-Julien S (2021) Svrg meets adagrad: Painless variance reduction. 2102.09645

Duchi J, Shalev-Shwartz S, Singer Y, Tewari A (2010) Composite objective mirror descent. In Proceedings of the Twenty Third Annual Conference on Computational Learning Theory

Duchi J, Hazan E, Singer Y (2011) Adaptive subgradient methods for online learning and stochastic optimization. Journal of Machine Learning Research (JMLR) pp 12:2121–2159

Fang C, Li CJ, Lin Z, Zhang T (2018) Spider: Near-optimal non-convex optimization via stochastic path-integrated differential estimator. Advances in Neural Information Processing Systems 31

Ge R, Li Z, Wang W, Wang X (2019) Stabilized svrg: Simple variance reduction for nonconvex optimization. 1905.00529

Ghadimi S, Lan G, Zhang H (2016) Mini-batch stochastic approximation methods for nonconvex stochastic composite optimization. arXiv preprint arXiv:13086594

Goyal P, Dollár P, Girshick R, Noordhuis P, Wesolowski L, Kyrola A, Tulloch A, Jia Y, He K (2017) Accurate, large minibatch sgd: Training imagenet in 1 hour. 1706.02677

He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 770–778

Huang H, Wang C, Dong B (2019) Nostalgic adam: Weighting more of the past gradients when designing the adaptive learning rate. arXiv preprint arXiv: 180507557

Johnson R, Zhang T (2013) Accelerating stochastic gradient descent using predictive variance reduction. Advances in Neural Information Processing Systems 27

Karimi H, Nutini J, Schmidt M (2016) Linear convergence of gradient and proximal-gradient methods under the polyak-Łojasiewicz condition. 1608.04636

Kingma DP, Ba JL (2015) Adam: A method for stochastic optimization. Proceedings of the 3rd International Conference on Learning Representations (ICLR)

Krizhevsky A, Nair V, Hinton G (2009) Cifar-10 (canadian institute for advanced research). 1

LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proceedings of the IEEE 86(11):2278–2324

Lei L, Jordan MI (2019) On the adaptivity of stochastic gradient-based optimization. 1904.04480

Lei L, Ju C, Chen J, Jordan MI (2017) Non-convex finite-sum optimization via scsg methods. Advances in Neural Information Processing Systems 30 pp 2348–2358

Li W, Zhang Z, Wang X, Luo P (2020) Adax: Adaptive gradient descent with exponential long term memory. arXiv preprint arXiv:200409740

Li Z, Li J (2018) A simple proximal stochastic gradient method for nonsmooth nonconvex optimization. Advances in Neural Information Processing Systems 31 pp 5564–5574

Liu L, Jiang H, He P, Chen W, Liu X, Gao J, Han J (2019) On the variance of the adaptive learning rate and beyond. arXiv preprint arXiv:190803265

Liu M, Zhang W, Orabona F, Yang T (2020) Adam$^+$: A stochastic method with adaptive variance reduction. 2011.11985

Loshchilov I, Hutter F (2016) Sgdr: Stochastic gradient descent with warm restarts. International Conference on Learning Representations

Luo L, Xiong Y, Liu Y, Sun X (2019) Adaptive gradient methods with dynamic bound of learning rate. Proceedings of 7th International Conference on Learning Representations

Nair V, Hinton G (2010) Rectified linear units improve restricted boltzmann machines. Proceedings of 27th International Conference on Machine Learning(ICML)

Polyak BT (1963) Gradient methods for minimizing functionals. Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki

Reddi S, Sra S, Poczos B, Smola AJ (2016b) Proximal stochastic methods for nonsmooth nonconvex finite-sum optimization. Advances in Neural Information Processing Systems 29

Reddi SJ, Hefny A, Sra S, Poczos B, Smola A (2016a) Stochastic variance reduction for nonconvex optimization. 1603.06160

Reddi SJ, Kale S, Kumar S (2018) On the convergence of adam and beyond. Proceedings of the 6th International Conference on Learning Representations (ICLR)

Schölkopf B, Smola AJ (2002) Learning with kernels: support vector machines, regularization, optimization, and beyond. MIT press

Tieleman T, Hinton G (2012) Rmsprop: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural networks for machine learning pp 4(2):26–31

Wilson AC, Roelofs R, Stern M, Srebro N, Recht B (2017) The marginal value of adaptive gradient methods in machine learning. In: Advances in Neural Information Processing Systems, Curran Associates, Inc.

yueqiw (2019) Svrg for neural networks (pytorch). URL https://github.com/yueqiw/OptML-SVRG-PyTorch

Zhou D, Tang Y, Yang Z, Cao Y, Gu Q (2018a) On the convergence of adaptive gradient methods for nonconvex optimization. 1808.05671

Zhou D, Xu P, Gu Q (2018b) Stochastic nested variance reduction for nonconvex optimization. Advances in Neural Information Processing Systems 32

Zhuang J, Tang T, Ding Y, Tatikonda S, Dvornek N, Papademetris X, Duncan JS (2020) Adabelief optimizer: Adapting stepsizes by the belief in observed gradients. In: Advances in Neural Information Processing Systems, Curran Associates, Inc.

Appendix

A Convergence of the Adaptive SMD Algorithm

Definition 3 We define the generalized stochastic gradient at t as

$$\tilde{g}_{X,t} = \frac{1}{\alpha_t}(x_t - x_{t+1})$$

A.1 Auxiliary Lemmas

Lemma 1 *[**Lemma 1 in Ghadimi et al. (2016)**]. Let g_t be the stochastic gradient in Algorithm 1 obtained at t and $\tilde{g}_{X,t}$ be defined as in Definition 3, then*

$$\langle g_t, \tilde{g}_{X,t}\rangle \geq m\|\tilde{g}_{X,t}\|^2 + \frac{1}{\alpha_t}[h(x_{t+1}) - h(x_t)] \tag{2}$$

Proof. By the optimality of the mirror descent update rule, it implies for any $x \in \mathcal{X}$ and $\nabla h(x_{t+1}) \in \partial h(x_{t+1})$

$$\langle g_t + \frac{1}{\alpha_t}(\nabla\psi_t(x_{t+1}) - \nabla\psi_t(x_t)) + \nabla h(x_{t+1}), x_t - x_{t+1}\rangle \geq 0 \tag{3}$$

Let $x = x_t$ in the above in equality, we get

$$\begin{aligned}\langle g_t, x_t - x_{t+1}\rangle &\geq \frac{1}{\alpha_t}\langle\nabla\psi_t(x_{t+1}) - \nabla\psi_t(x_t), x_{t+1} - x_t\rangle + \langle\nabla h(x_{t+1}), x_{t+1} - x_t\rangle \\ &\geq \frac{m}{\alpha_t}\|x_{t+1} - x_t\|_2^2 + h(x_{t+1}) - h(x)\end{aligned} \tag{4}$$

where the second inequality is due to the strong convexity of the function $\psi_t(x)$ and the convexity of $h(x)$, by noting that $x_t - x_{t+1} = \alpha_t\tilde{g}_{X,t}$, the inequality follows.

Lemma 2 *Let $g_{X,t}, \tilde{g}_{X,t}$ be defined as in Definition 3 and Definition 1 respectively, then*

$$\|g_{X,t} - \tilde{g}_{X,t}\|_2 \leq \frac{1}{m}\|\nabla f(x_t) - g_t\|_2 \tag{5}$$

Proof. By the definition of g_{X_t} and $\tilde{g}_{X,t}$,

$$\|g_{X,t} - \tilde{g}_{X,t}\|_2 = \frac{1}{\alpha_t}\|(x_t - x_{t+1}^+) - (x_t - x_{t+1})\|_2 = \frac{1}{\alpha_t}\|x_{t+1} - x_{t+1}^+\|_2 \tag{6}$$

Similar to Lemma 1, by the optimality of the mirror descent update rule, we have the following two inequalities

$$\begin{aligned}&\langle g_t + \frac{1}{\alpha_t}(\nabla\psi_t(x_{t+1}) - \nabla\psi_t(x_t)) + \nabla h(x_{t+1}), x - x_{t+1}\rangle \geq 0, \forall x \in \mathcal{X}, \nabla h(x_{t+1}) \in \partial h(x_{t+1}) \\ &\langle \nabla f(x_t) + \frac{1}{\alpha_t}(\nabla\psi_t(x_{t+1}^+) - \nabla\psi_t(x_t)) + \nabla h(x_{t+1}^+), x - x_{t+1}^+\rangle \geq 0, \forall x \in \mathcal{X}, \nabla h(x_{t+1}^+) \in \partial h(x_{t+1}^+)\end{aligned} \tag{7}$$

Take $x = x_{t+1}^+$ in the first inequality and $x = x_{t+1}$ in the second one, we can get

$$\begin{aligned}&\langle -g_t, x_{t+1} - x_{t+1}^+\rangle \geq \frac{1}{\alpha_t}\langle\nabla\psi_t(x_{t+1}) - \nabla\psi_t(x_t), x_{t+1} - x_{t+1}^+\rangle + h(x_{t+1}) - h(x_{t+1}^+) \\ &\langle \nabla f(x_t), x_{t+1} - x_{t+1}^+\rangle \geq \frac{1}{\alpha_t}\langle\nabla\psi_t(x_{t+1}^+) - \nabla\psi_t(x_t), x_{t+1}^+ - x_{t+1}\rangle + h(x_{t+1}^+) - h(x_{t+1})\end{aligned} \tag{8}$$

Summing up the above inequalities, we can get

$$\begin{aligned}
&\langle \nabla f(x_t) - g_t, x_{t+1} - x_{t+1}^+ \rangle \\
&\geq \frac{1}{\alpha_t}(\langle \nabla \psi_t(x_{t+1}) - \nabla \psi_t(x_t), x_{t+1} - x_{t+1}^+ \rangle + \langle \nabla \psi_t(x_{t+1}^+) - \nabla \psi_t(x_t), x_{t+1}^+ - x_{t+1} \rangle) \\
&= \frac{1}{\alpha_t}(\langle \nabla \psi_t(x_{t+1}) - \nabla \psi_t(x_{t+1}^+), x_{t+1} - x_{t+1}^+ \rangle) \\
&\geq \frac{m}{\alpha_t}\|x_{t+1} - x_{t+1}^+\|_2^2
\end{aligned} \tag{9}$$

Therefore by Cauchy Schwarz inequality,

$$\|g_t - \nabla f(x_t)\|_2 \geq \frac{m}{\alpha_t}\|x_{t+1} - x_{t+1}^+\|_2 \tag{10}$$

Hence the inequality in the lemma follows.

Lemma 3 *[Lemma A.1 in Lei et al. (2017)]. Let $x_1, \cdots, x_M \in \mathbb{R}^d$ be an arbitrary population of M vectors with the condition that*

$$\sum_{i=1}^{M} x_j = 0 \tag{11}$$

Further let J be a uniform random subset of $\{1, \cdots, M\}$ with size m, then

$$\mathbb{E}[\|\frac{1}{m}\sum_{j \in J} x_j\|^2] \leq \frac{I(m < M)}{mM}\sum_{j=1}^{M}\|x_j\|^2 \tag{12}$$

Proof of the above general lemma can be found in Lei et al. (2017).

A.2 Convergence of the Adaptive SMD Algorithm in the Nonconvex Nonsmooth Problem

Proof of Theorem 1. From the L-Lipshitz gradients and Lemma 1, we know that

$$\begin{aligned}
f(x_{t+1}) &\leq f(x_t) + \langle \nabla f(x_t), x_{t+1} - x_t \rangle + \frac{L}{2}\|x_{t+1} - x_t\|^2 \\
&= f(x_t) - \alpha_t \langle \nabla f(x_t), \tilde{g}_{X,t} \rangle + \frac{L}{2}\alpha_t^2 \|\tilde{g}_{X,t}\|_2^2 \\
&= f(x_t) - \alpha_t \langle g_t, \tilde{g}_{X,t} \rangle + \frac{L}{2}\alpha_t^2 \|\tilde{g}_{X,t}\|_2^2 + \alpha_t \langle g_t - \nabla f(x_t), \tilde{g}_{X,t} \rangle \\
&\leq f(x_t) + \frac{L}{2}\alpha_t^2 \|\tilde{g}_{X,t}\|_2^2 - \alpha_t m \|\tilde{g}_{X,t}\|_2^2 - [h(x_{t+1}) - h(x_t)] + \alpha_t \langle g_t - \nabla f(x_t), \tilde{g}_{X,t} \rangle
\end{aligned} \tag{13}$$

Therefore since $F(x) = f(x) + h(x)$, we get

$$\begin{aligned}
F(x_{t+1}) &\leq F(x_t) - (\alpha_t m - \frac{L}{2}\alpha_t^2)\|\tilde{g}_{X,t}\|_2^2 + \alpha_t \langle g_t - \nabla f(x_t), \tilde{g}_{X,t} \rangle + \alpha_t \langle g_t - \nabla f(x_t), \tilde{g}_{X,t} - g_{X,t} \rangle \\
&\leq F(x_t) - (\alpha_t m - \frac{L}{2}\alpha_t^2)\|\tilde{g}_{X,t}\|_2^2 + \alpha_t \langle g_t - \nabla f(x_t), \tilde{g}_{X,t} \rangle + \alpha_t \|\nabla f(x_t) - g_t\|_2 \|\tilde{g}_{X,t} - g_{X,t}\|_2 \\
&\leq F(x_t) - (\alpha_t m - \frac{L}{2}\alpha_t^2)\|\tilde{g}_{X,t}\|_2^2 + \alpha_t \langle g_t - \nabla f(x_t), \tilde{g}_{X,t} \rangle + \frac{\alpha_t}{m}\|\nabla f(x_t) - g_t\|_2^2
\end{aligned} \tag{14}$$

where the second last one is a direct result from Cauchy-Schwarz inequality and the last inequality is from Lemma 2. Rearrange the above inequalities and sum up from 1 to T, we get

$$\begin{aligned}\sum_{t=1}^{T}(\alpha_t m-\frac{L}{2}\alpha_t^2)\|\tilde{g}_{X,t}\|_2^2 &\le \sum_{t=1}^{T}[F(x_t)-F(x_{t+1})]+\sum_{t=1}^{T}[\alpha_t\langle g_t-\nabla f(x_t),\tilde{g}_{X,t}\rangle+\frac{\alpha_t}{m}\|\nabla f(x_t)-g_t\|_2^2]\\ &= F(x_1)-F(x_{T+1})+\sum_{t=1}^{T}[\alpha_t\langle g_t-\nabla f(x_t),\tilde{g}_{X,t}\rangle+\frac{\alpha_t}{m}\|\nabla f(x_t)-g_t\|^2]\\ &\le F(x_1)-F^*+\sum_{t=1}^{T}[\alpha_t\langle g_t-\nabla f(x_t),\tilde{g}_{X,t}\rangle+\frac{\alpha_t}{m}\|\nabla f(x_t)-g_t\|^2]\end{aligned} \tag{15}$$

where the last inequality is due to $F^*\le F(x),\forall x$. Define the filtration $\mathcal{F}_t=\sigma(x_1,\cdots,x_t)$. Note that we suppose g_t is an unbiased estimate of $\nabla f(x_t)$, hence $\mathbb{E}[\langle\nabla f(x_t)-g_t,g_{X,t}\rangle|\mathcal{F}_t]=0$. Moreover, since the sampled gradients has bounded variance σ^2, hence by applying Lemma 3 with $x_i=\nabla_{i\in\mathcal{I}_j}f_i(x_t)-\nabla f(x_t)$

$$\mathbb{E}[\|\nabla f(x_t)-g_t\|^2]\le\frac{\sigma^2}{b_t}I(b_t<n) \tag{16}$$

where I is the indicator function. Since the final x_{t^*} is uniformly sampled from all $\{x_t\}_{t=1}^T$, therefore

$$\mathbb{E}[\|\tilde{g}_{X,t^*}\|_2^2]=\mathbb{E}[\mathbb{E}[\|\tilde{g}_{X,t^*}\|_2^2|t^*]]=\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\|\tilde{g}_{X,t}\|_2^2] \tag{17}$$

Therefore when α_t,b_t are constants, the average can be found as

$$\begin{aligned}T(\alpha_t m-\frac{L}{2}\alpha_t^2)\mathbb{E}(\|\tilde{g}_{X,t^*}\|_2^2)&\le F(x_1)-F^*+\sum_{t=1}^{T}\frac{\alpha_t}{m}\mathbb{E}[\|\nabla f(x_t)-g_t\|_2^2]\\ &= \Delta_F+T\frac{\alpha_t\sigma^2}{mb_t}I(b_t<n)\end{aligned} \tag{18}$$

where we define $\Delta_F=F(x_1)-F^*$. Take $\alpha_t=\frac{m}{L}$, then $\alpha_t m-\frac{L}{2}\alpha_t^2=\frac{m^2}{2L}$ and

$$\mathbb{E}(\|\tilde{g}_{X,t^*}\|_2^2)\le\frac{2\Delta_F L}{m^2T}+\frac{2\sigma^2}{b_t m^2}I(b_t<n) \tag{19}$$

Also by Lemma 2, the difference between g_{X,t^*} and $\tilde{g}_{X,t^*}$ are bounded, hence

$$\begin{aligned}\mathbb{E}[\|g_{X,t^*}\|_2^2]&\le 2\mathbb{E}[\|\tilde{g}_{X,t^*}\|_2^2]+2\mathbb{E}[\|g_{X,t^*}-\tilde{g}_{X,t^*}\|_2^2]\\ &\le\frac{4\Delta_F L}{m^2T}+\frac{4\sigma^2}{b_t m^2}I(b_t<n)+\frac{2\sigma^2}{b_t m^2}I(b_t<n)\\ &=\frac{4\Delta_F L}{m^2T}+\frac{6\sigma^2}{b_t m^2}I(b_t<n)\end{aligned} \tag{20}$$

Take $b_t=n\wedge(12\sigma^2/m^2\epsilon),T=1\vee(8\Delta_F L/m^2\epsilon)$ as in the theorem, the expectation is

$$\begin{aligned}\mathbb{E}[\|g_{X,t^*}\|_2^2]&\le\frac{4\Delta_F L}{m^2T}+\frac{6\sigma^2}{b_t m^2}I(b_t<n)\\ &\le\frac{\epsilon}{2}+\frac{\epsilon}{2}=\epsilon\end{aligned} \tag{21}$$

Therefore since one iteration takes b_t stochastic gradient computations, the total number of stochastic gradient computations is

$$Tb_t\le\frac{8\Delta_F L}{m^2\epsilon}b_t+b_t=O\left(\frac{n}{\epsilon}\wedge\frac{\sigma^2}{\epsilon^2}+n\wedge\frac{\sigma^2}{\epsilon}\right) \tag{22}$$

A.3 Convergence of the Adaptive SMD Algorithm under the P-L condition

Proof of Theorem 2 By the proof in A.2, we have

$$F(x_{t+1}) \leq F(x_t) - (\alpha_t m - \frac{L}{2}\alpha_t^2)\|\tilde{g}_{X,t}\|_2^2 + \alpha_t\langle g_t - \nabla f(x_t), \tilde{g}_{X,t}\rangle + \frac{\alpha_t}{m}\|\nabla f(x_t) - g_t\|_2^2 \quad (23)$$

Take expectation on both sides, we know that

$$\mathbb{E}[F(x_{t+1})] \leq \mathbb{E}[F(x_t)] - (\alpha_t m - \frac{L}{2}\alpha_t^2)\mathbb{E}[\|\tilde{g}_{X,t}\|_2^2] + \frac{\alpha_t}{m}\mathbb{E}[\|\nabla f(x_t) - g_t\|_2^2] \quad (24)$$

Since

$$\mathbb{E}[\|g_{X,t^*}\|_2^2] \leq 2\mathbb{E}[\|\tilde{g}_{X,t^*}\|_2^2] + 2\mathbb{E}[\|g_{X,t^*} - \tilde{g}_{X,t^*}\|_2^2] \quad (25)$$

Hence the inequality becomes

$$\begin{aligned}
\mathbb{E}[F(x_{t+1})] &\leq \mathbb{E}[F(x_t)] - (\alpha_t m - \frac{L}{2}\alpha_t^2)(\frac{1}{2}\mathbb{E}[\|g_{X,t}\|_2^2] - \mathbb{E}[\|\nabla f(x_t) - g_t\|_2^2]) + \frac{\alpha_t}{m}\mathbb{E}[\|\nabla f(x_t) - g_t\|_2^2] \\
&\leq \mathbb{E}[F(x_t)] - (\frac{\alpha_t m}{2} - \frac{L}{4}\alpha_t^2)\mathbb{E}[\|g_{X,t}\|_2^2] + (\frac{\alpha_t}{m} + \alpha_t m - \frac{L}{2}\alpha_t^2)\mathbb{E}[\|\nabla f(x_t) - g_t\|_2^2] \\
&\leq \mathbb{E}[F(x_t)] - \mu(\alpha_t m - \frac{L}{2}\alpha_t^2)(\mathbb{E}[F(x_t)] - F^*) + (\frac{\alpha_t}{m} + \alpha_t m - \frac{L}{2}\alpha_t^2)\mathbb{E}[\|\nabla f(x_t) - g_t\|_2^2]
\end{aligned} \quad (26)$$

Take $\alpha_t = m/L$ and minus $F(x)^*$ on both sides, we get

$$\begin{aligned}
\mathbb{E}[F(x_{t+1})] - F^* &\leq (1 - \mu(\alpha_t m - \frac{L}{2}\alpha_t^2)(\mathbb{E}[F(x_t)] - F^*) + (\frac{\alpha_t}{m} + \alpha_t m - \frac{L}{2}\alpha_t^2)\mathbb{E}[\|\nabla f(x_t) - g_t\|_2^2] \\
&= (1 - \mu\frac{m^2}{2L})(\mathbb{E}[F(x_t)] - F^*) + (\frac{1}{L} + \frac{m^2}{2L})\mathbb{E}[\|\nabla f(x_t) - g_t\|_2^2] \\
&= (1 - \mu\frac{m^2}{2L})(\mathbb{E}[F(x_t)] - F^*) + (\frac{1}{L} + \frac{m^2}{2L})\frac{\sigma^2}{b_t}I(b_t < n)
\end{aligned} \quad (27)$$

Let $\gamma = 1 - \frac{\mu m^2}{2L}$, since $m^2\mu/L \leq \frac{1}{\sqrt{n}}$, $\gamma \in (0,1)$, divide by γ^{t+1} on both sides, we get

$$\frac{\mathbb{E}[F(x_{t+1})] - F^*}{\gamma^{t+1}} \leq \frac{\mathbb{E}[F(x_t)] - F^*}{\gamma^t} + \frac{(\frac{1}{L} + \frac{m^2}{2L})}{\gamma^{t+1}}\frac{\sigma^2}{b_t}I(b_t < n) \quad (28)$$

Take summation with respect to the loop parameter t from $t = 1$ to $t = T$, assume that b_t is a constant, the inequality becomes

$$\begin{aligned}
\mathbb{E}[F(x_{T+1})] - F^* &\leq \gamma^T\Delta_F + \gamma^T\sum_{t=1}^{T}\frac{(\frac{1}{L} + \frac{m^2}{2L})}{\gamma^t}\frac{\sigma^2}{b_t}I(b_t < n) \\
&\leq \gamma^T\Delta_F + (\frac{1}{L} + \frac{m^2}{2L})\frac{1 - \gamma^T}{1 - \gamma}\frac{\sigma^2}{b_t}I(b_t < n) \\
&\leq \gamma^T\Delta_F + (\frac{1}{L} + \frac{m^2}{2L})\frac{2L}{\mu m^2}\frac{\sigma^2}{b_t}I(b_t < n) \\
&= \gamma^T\Delta_F + (\frac{1}{m^2} + 1)\frac{1}{\mu}\frac{\sigma^2}{b_t}I(b_t < n)
\end{aligned} \quad (29)$$

Therefore when taking $T = 1 \vee (\log\frac{2\Delta_F}{\epsilon})/(\log\frac{1}{\gamma}) = O(\log\frac{2\Delta_F}{\epsilon}/\mu)$, $b_t = n \wedge \frac{2(1+m^2)\sigma^2}{\epsilon m^2\mu}$. Then the total number of stochastic gradient computations is

$$\begin{aligned}
Tb &= O((n \wedge \frac{\sigma^2}{\mu\epsilon})(\frac{1}{\mu}\log\frac{1}{\epsilon})) \\
&= O\left((\frac{n}{\mu} \wedge \frac{\sigma^2}{\mu^2\epsilon})\log\frac{1}{\epsilon}\right)
\end{aligned} \quad (30)$$

B Convergence of SVRAMD in the nonconvex nonsmooth problem

Recall the Algorithm 2 in the algorithm section, similarly define

Definition 4 We define the variance reduced generalized gradient mapping as

$$\tilde{g}_{Y,k}^t = \frac{1}{\alpha_t}(y_k^t - y_{k+1}^t)$$

Definition 5 We also define its corresponding term when the algorithm uses non-stochastic full batch gradient

$$g_{Y,k}^t = \frac{1}{\alpha_t}(y_k^t - y_{k+1}^{t+}), \text{ when } y_{k+1}^{t+} = \text{argmin}_y\{\alpha_t\langle\nabla f(y_k^t), y\rangle + \alpha_t h(x) + B_{\psi_{tk}}(y, y_k^t)\} \quad (31)$$

B.1 Auxiliary Lemmas

Lemma 4 *Let v_k^t be defined as in Algorithm 2 and $\tilde{g}_{Y,k}^t$ be defined as in Definition 4, then we have*

$$\langle v_k^t, \tilde{g}_{Y,k}^t\rangle \geq m\|\tilde{g}_{Y,k}^t\|^2 + \frac{1}{\alpha_t}[h(y_{k+1}^t) - h(y_k^t)] \quad (32)$$

Proof. The proof of this inequality is similar to that of Lemma 1. By the optimality of the mirror descent update rule, it implies for any $y \in \mathcal{X}, \nabla h(y_{k+1}^t) \in \partial h(y_{k+1}^t)$,

$$\langle v_k^t + \frac{1}{\alpha_t}(\nabla\psi_{tk}(y_{k+1}^t) - \nabla\psi_{tk}(y_k^t)) + \nabla h(y_{k+1}^t), y - y_{k+1}^t\rangle \geq 0 \quad (33)$$

Let $x = y_k^t$ in the above in equality, we get

$$\begin{aligned}\langle v_k^t, y_k^t - y_{k+1}^t\rangle &\geq \frac{1}{\alpha_t}\langle\nabla\psi_{tk}(y_{k+1}^t) - \nabla\psi_{tk}(y_k^t), y_{k+1}^t - y_k^t\rangle + \langle\nabla h(y_{k+1}^t), y_{k+1}^t - y_k^t\rangle \\ &\geq \frac{m}{\alpha_t}\|y_{k+1}^t - y_k^t\|_2^2 + [h(y_{k+1}^t) - h(y_k^t)]\end{aligned} \quad (34)$$

where the second inequality is due to the m-strong convexity of the function $\psi_{tk}(x)$ and the convexity of h. Note from the definition that $y_k^t - y_{k+1}^t = \alpha_t\tilde{g}_{Y,k}^t$, the inequality follows.

Lemma 5 *Let $g_{Y,k}^t, \tilde{g}_{Y,k}^t$ be defined as in Definition 4 and Definition 5 respectively, then we have*

$$\|\tilde{g}_{Y,k}^t - g_{Y,k}^t\|_2 \leq \frac{1}{m}\|\nabla f(y_k^t) - v_k^t\|_2 \quad (35)$$

Proof. The proof is similar to Lemma 2. By definition of $\tilde{g}_{Y,k}^t$ and $g_{Y,k}^t$,

$$\|\tilde{g}_{Y,k}^t - g_{Y,k}^t\|_2 = \frac{1}{\alpha_t}\|(y_k^t - y_{k+1}^t) - (y_k^t - y_{k+1}^{t+})\|_2 = \frac{1}{\alpha_t}\|y_k^t - y_{k+1}^{t+}\|_2 \quad (36)$$

As in Lemma 4, by the optimality of the mirror descent update rule, we have the following two inequalities

$$\begin{aligned}&\langle v_k^t + \frac{1}{\alpha_t}(\nabla\psi_{tk}(y_{k+1}^t) - \nabla\psi_{tk}(y_k^t)) + \nabla h(y_{k+1}^t), y - y_{k+1}^t\rangle \geq 0, \forall y \in \mathcal{X}, \nabla h(y_{k+1}^t) \in \partial h(y_{k+1}^t) \\ &\langle\nabla f(y_k^t) + \frac{1}{\alpha_t}(\nabla\psi_{tk}(y_{k+1}^{t+}) - \nabla\psi_{tk}(y_k^t)) + \nabla h(y_{k+1}^{t+}), y - y_{k+1}^{t+}\rangle \geq 0, \forall y \in \mathcal{X}, \nabla h(y_{k+1}^{t+1}) \in \partial h(y_{k+1}^{t+})\end{aligned} \quad (37)$$

Take $y = y_{k+1}^{t+}$ in the first inequality and $y = y_{k+1}^t$ in the second one, we can get

$$\begin{aligned}
&\langle v_k^t, y_{k+1}^{t+} - y_{k+1}^t\rangle \geq \frac{1}{\alpha_t}\langle \nabla\psi_{tk}(y_{k+1}^t) - \nabla\psi_{tk}(y_k^t), y_{k+1}^t - y_{k+1}^{t+}\rangle + h(y_{k+1}^t) - h(y_{k+1}^{t+}) \\
&\langle \nabla f(y_k^t), y_{k+1}^t - y_{k+1}^{t+}\rangle \geq \frac{1}{\alpha_t}\langle \nabla\psi_{tk}(y_k^t) - \nabla\psi_{tk}(y_{k+1}^{t+}), y_{k+1}^{t+} - y_{k+1}^t\rangle + h(y_{k+1}^{t+}) - h(y_{k+1}^t)
\end{aligned} \tag{38}$$

Summing up the above inequalities, we can get

$$\begin{aligned}
&\langle v_k^t - \nabla f(y_k^t), y_{k+1}^{t+} - y_{k+1}^t\rangle \\
&\geq \frac{1}{\alpha_t}\langle \nabla\psi_{tk}(y_{k+1}^t) - \nabla\psi_{tk}(y_k^t), y_{k+1}^t - y_{k+1}^{t+}\rangle + \frac{1}{\alpha_t}\langle \nabla\psi_{tk}(y_k^t) - \nabla\psi_{tk}(y_{k+1}^{t+}), y_{k+1}^{t+} - y_{k+1}^t\rangle \\
&= \frac{1}{\alpha_t}(\langle \nabla\psi_{tk}(y_{k+1}^t) - \nabla\psi_{tk}(y_{k+1}^{t+}), y_{k+1}^t - y_{k+1}^{t+}\rangle) \\
&\geq \frac{m}{\alpha_t}\|y_{k+1}^t - y_{k+1}^{t+}\|_2^2
\end{aligned} \tag{39}$$

where the last inequality is due to the strong convexity of $\psi_{tk}(x)$. Therefore by Cauchy Schwarz inequality,

$$\frac{1}{m}\|\nabla f(y_k^t) - v_k^t\|_2 \geq \frac{1}{\alpha_t}\|y_{k+1}^t - y_{k+1}^{t+}\|_2 \geq \|\tilde{g}_{Y,k}^t - g_{Y,k}^t\|_2 \tag{40}$$

Hence the inequality in the lemma follows.

Lemma 6 *Let $\nabla f(y_k^t), v_k^t$ be the full batch gradient and the , then*

$$\mathbb{E}[\|\nabla f(y_k^t) - v_k^t\|_2^2] \leq \frac{L^2}{b_t}\mathbb{E}[\|y_k^t - x_t\|^2] + \frac{I(B_t < n)\sigma^2}{B_t} \tag{41}$$

Proof. Note that the large batch $\mathcal{I}_j$ and the mini-batch $\tilde{\mathcal{I}}_j$ are independent, hence

$$\begin{aligned}
&\mathbb{E}[\|\nabla f(y_k^t) - v_k^t\|_2^2] \\
&= \mathbb{E}[\|\frac{1}{b_t}\sum_{i\in\tilde{\mathcal{I}}_k}(\nabla f_i(y_k^t) - \nabla f_i(x_t)) - (\nabla f(y_k^t) - g_t)\|_2^2] \\
&= \mathbb{E}[\|\frac{1}{b_t}\sum_{i\in\tilde{\mathcal{I}}_k}(\nabla f_i(y_k^t) - \nabla f_i(x_t)) - (\nabla f(y_k^t) - \frac{1}{B_t}\sum_{i\in\mathcal{I}_t}\nabla f_i(x_t))\|_2^2] \\
&= \mathbb{E}[\|\frac{1}{b_t}\sum_{i\in\tilde{\mathcal{I}}_k}(\nabla f_i(y_k^t) - \nabla f_i(x_t)) - \nabla f(y_k^t) + \nabla f(x_t) + \frac{1}{B_t}\sum_{i\in\mathcal{I}_t}(\nabla f_i(x_t) - \nabla f(x_t))\|_2^2] \\
&= \mathbb{E}[\|\frac{1}{b_t}\sum_{i\in\tilde{\mathcal{I}}_k}(\nabla f_i(y_k^t) - \nabla f_i(x_t)) - \nabla f(y_k^t) + \nabla f(x_t)\|_2^2 + \mathbb{E}\|\frac{1}{B_t}\sum_{i\in\mathcal{I}_t}(\nabla f_i(x_t) - \nabla f(x_t))\|_2^2] \\
&= \mathbb{E}[\|\frac{1}{b_t}\sum_{i\in\tilde{\mathcal{I}}_k}(\nabla f_i(y_k^t) - \nabla f(y_k^t)) - (\nabla f_i(x_t) - \nabla f(x_t))\|_2^2 + \mathbb{E}\|\frac{1}{B_t}\sum_{i\in\mathcal{I}_t}(\nabla f_i(x_t) - \nabla f(x_t))\|_2^2] \\
&\leq \mathbb{E}[\|\frac{1}{b_t}\sum_{i\in\tilde{\mathcal{I}}_k}(\nabla f_i(y_k^t) - \nabla f(y_k^t)) - (\nabla f_i(x_t) - \nabla f(x_t))\|_2^2 + \frac{I(B_t < n)\sigma^2}{B_t} \\
&= \frac{1}{b_t^2}\mathbb{E}[\sum_{i\in\tilde{\mathcal{I}}_k}\|\nabla f_i(y_k^t) - \nabla f_i(x_t)) - \nabla f(y_k^t) + \nabla f(x_t)\|^2] + \frac{I(B_t < n)\sigma^2}{B_t} \\
&\leq \frac{1}{b_t^2}\mathbb{E}[\sum_{i\in\tilde{\mathcal{I}}_k}\|\nabla f_i(y_k^t) - \nabla f_i(x_t))\|^2] + \frac{I(B_t < n)\sigma^2}{B_t} \\
&\leq \frac{L^2}{b_t}\mathbb{E}[\|y_k^t - x_t\|^2] + \frac{I(B_t < n)\sigma^2}{B_t}
\end{aligned} \tag{42}$$

where the fourth equality is because of the independence between $\mathcal{I}_j$ and $\tilde{\mathcal{I}}_j$. The first and the second inequalities are by Lemma 3. The third inequality follows from $\mathbb{E}[\|x - \mathbb{E}(x)\|^2] = \mathbb{E}[\|x\|^2]$ and the last inequality follows from the L-smoothness of $f(x)$

B.2 Main Proof

Proof of Theorem 3. From the L-Lipshitz gradients and Lemma 4, we know that

$$\begin{aligned}
f(y_{k+1}^t) &\le f(y_k^t) + \langle \nabla f(y_k^t), y_{k+1}^t - y_k^t \rangle + \frac{L}{2}\|y_{k+1}^t - y_k^t\|^2 \\
&= f(y_k^t) - \alpha_t \langle \nabla f(y_k^t), \tilde{g}_{Y,k}^t \rangle + \frac{L}{2}\alpha_t^2 \|\tilde{g}_{Y,k}^t\|_2^2 \\
&= f(y_k^t) - \alpha_t \langle v_k^t, \tilde{g}_{Y,k}^t \rangle + \frac{L}{2}\alpha_t^2 \|\tilde{g}_{Y,k}^t\|_2^2 + \alpha_t \langle v_k^t - \nabla f(y_k^t), \tilde{g}_{Y,k}^t \rangle \\
&\le f(y_k^t) + \frac{L}{2}\alpha_t^2 \|\tilde{g}_{Y,k}^t\|_2^2 - \alpha_t m \|\tilde{g}_{Y,k}^t\|_2^2 + \alpha_t \langle v_k^t - \nabla f(y_k^t), \tilde{g}_{Y,k}^t \rangle - [h(y_{k+1}^t) - h(y_k^t)]
\end{aligned} \tag{43}$$

Since $F(x) = f(x) + h(x)$, we can get

$$\begin{aligned}
F(y_{k+1}^t) &= F(y_k^t) - (\alpha_t m - \frac{L}{2}\alpha_t^2)\|\tilde{g}_{Y,k}^t\|_2^2 + \alpha_t \langle v_k^t - \nabla f(y_k^t), g_{Y,k}^t \rangle + \alpha_t \langle v_k^t - \nabla f(y_k^t), \tilde{g}_{Y,k}^t - g_{Y,k}^t \rangle \\
&\le F(y_k^t) - (\alpha_t m - \frac{L}{2}\alpha_t^2)\|\tilde{g}_{Y,k}^t\|_2^2 + \alpha_t \langle v_k^t - \nabla f(y_k^t), g_{Y,k}^t \rangle + \alpha_t \|\nabla f(y_k^t) - v_k^t\|_2 \|\tilde{g}_{Y,k}^t - g_{Y,k}^t\|_2 \\
&\le F(y_k^t) - (\alpha_t m - \frac{L}{2}\alpha_t^2)\|\tilde{g}_{Y,k}^t\|_2^2 + \alpha_t \langle v_k^t - \nabla f(y_k^t), g_{Y,k}^t \rangle + \frac{\alpha_t}{m}\|\nabla f(y_k^t) - v_k^t\|_2^2
\end{aligned} \tag{44}$$

where the second last inequality is from Cauchy Schwartz inequality and the last inequality is from Lemma 5. Define the filtration $\mathcal{F}_k^t = \sigma(y_1^1, \cdots y_{K+1}^1, y_1^2, \cdots, y_{K+1}^2, \cdots, y_1^t, \cdots, y_k^t)$. Note that $\mathbb{E}[\langle \nabla f(y_k^t) - v_k^t, g_{Y,k}^t \rangle | F_k^t] = 0$. Take expectation on both sides and use Lemma 6, we get

$$\begin{aligned}
\mathbb{E}[F(y_{k+1}^t)] &\le \mathbb{E}[F(y_k^t)] - (\frac{m}{\alpha_t} - \frac{L}{2})\mathbb{E}[\|y_{k+1}^t - y_k^t\|_2^2] + \frac{L^2\alpha_t}{b_t m}\mathbb{E}[\|y_k^t - x_t\|^2] + \frac{\alpha_t I(B_t < n)\sigma^2}{mB_t} \\
&\le \mathbb{E}[F(y_k^t)] - (\frac{m}{2\alpha_t} - \frac{L}{4})\mathbb{E}[\|y_{k+1}^t - y_k^t\|_2^2] + \frac{L^2\alpha_t}{b_t m}\mathbb{E}[\|y_k^t - x_t\|^2] + \frac{\alpha_t I(B_t < n)\sigma^2}{mB_t} \\
&\quad + (\frac{m}{2\alpha_t} - \frac{L}{4})\frac{\alpha_t^2 L^2}{m^2 b_t}\mathbb{E}[\|y_k^t - x_t\|^2] + (\frac{m}{2\alpha_t} - \frac{L}{4})\frac{\alpha_t^2 I(B_t < n)\sigma^2}{m^2 B_t} - (\frac{m}{4\alpha_t} - \frac{L}{8})\mathbb{E}[\|y_{k+1}^{t+} - y_k^t\|_2^2] \\
&= \mathbb{E}[F(y_k^t)] - (\frac{m}{2\alpha_t} - \frac{L}{4})\mathbb{E}[\|y_{k+1}^t - y_k^t\|_2^2] - (\frac{m}{4\alpha_t} - \frac{L}{8})\mathbb{E}[\|y_{k+1}^{t+} - y_k^t\|_2^2] \\
&\quad + (\frac{3L^2\alpha_t}{2b_t m} - \frac{\alpha_t^2 L^3}{4m^2 b_t})\mathbb{E}[\|y_k^t - x_t\|^2] + (\frac{3\alpha_t}{2m} - \frac{\alpha_t^2 L}{4m^2})\frac{I(B_t < n)\sigma^2}{B_t} \\
&= \mathbb{E}[F(y_k^t)] - (\frac{m}{2\alpha_t} - \frac{L}{4})\mathbb{E}[\|y_{k+1}^t - y_k^t\|_2^2] - (\frac{m\alpha_t}{4} - \frac{L\alpha_t^2}{8})\mathbb{E}[\|g_{Y,k}^t\|_2^2] \\
&\quad + (\frac{3L^2\alpha_t}{2b_t m} - \frac{\alpha_t^2 L^3}{4m^2 b_t})\mathbb{E}[\|y_k^t - x_t\|^2] + (\frac{3\alpha_t}{2m} - \frac{\alpha_t^2 L}{4m^2})\frac{I(B_t < n)\sigma^2}{B_t}
\end{aligned} \tag{45}$$

where the second inequality uses the fact that by Lemma 6

$$\begin{aligned}\mathbb{E}[\|y_{k+1}^{t+}-y_k^t\|_2^2] &= \alpha_t^2\mathbb{E}[\|g_{Y,k}^t\|_2^2] \le 2\alpha_t^2\mathbb{E}[\|\tilde{g}_{Y,k}^t\|_2^2] + 2\alpha_t^2\mathbb{E}[\|g_{Y,k}^t-\tilde{g}_{Y,k}^t\|_2^2]\\ &\le 2\alpha_t^2\mathbb{E}[\|\tilde{g}_{Y,k}^t\|_2^2] + \frac{2\alpha_t^2}{m^2}\mathbb{E}[\|\nabla f(y_k^t)-v_k^t\|_2^2]\\ &\le 2\mathbb{E}[\|y_{k+1}^t-y_k^t\|_2^2] + \frac{2\alpha_t^2}{m^2}(\frac{L^2}{b_t}\mathbb{E}[\|y_k^t-x_t\|^2] + \frac{I(B_t<n)\sigma^2}{B_t})\\ &= 2\mathbb{E}[\|y_{k+1}^t-y_k^t\|_2^2] + \frac{2\alpha_t^2L^2}{m^2b_t}\mathbb{E}[\|y_k^t-x_t\|^2] + \frac{2I(B_t<n)\sigma^2\alpha_t^2}{B_tm^2}\end{aligned} \tag{46}$$

Since by Young's inequality, we know that

$$\|y_{k+1}^t-x_t\| \le (1+\frac{1}{p})\|y_k^t-x_t\|_2^2 + (1+p)\|y_{k+1}^t-y_k^t\|_2^2, \forall p\in\mathbb{R} \tag{47}$$

Hence substitute into equation 45, we can get

$$\begin{aligned}\mathbb{E}[F(y_{k+1}^t)] &\le \mathbb{E}[F(y_k^t)] - (\frac{m}{2\alpha_t}-\frac{L}{4})\mathbb{E}(\frac{\|y_{k+1}^t-x_t\|_2^2}{1+p} - \frac{\|y_k^t-x_t\|_2^2}{p}) - (\frac{m\alpha_t}{4}-\frac{L\alpha_t^2}{8})\mathbb{E}[\|g_{Y,k}^t\|_2^2]\\ &\quad + (\frac{3L^2\alpha_t}{2b_tm} - \frac{\alpha_t^2L^3}{4m^2b_t})\mathbb{E}[\|y_k^t-x_t\|^2] + (\frac{3\alpha_t}{2m}-\frac{\alpha_t^2L}{4m^2})\frac{I(B_t<n)\sigma^2}{B_t}\\ &\le \mathbb{E}[F(y_k^t)] - (\frac{m}{2\alpha_t}-\frac{L}{4})\mathbb{E}(\frac{\|y_{k+1}^t-x_t\|_2^2}{1+p}) - (\frac{m\alpha_t}{4}-\frac{L\alpha_t^2}{8})\mathbb{E}[\|g_{Y,k}^t\|_2^2]\\ &\quad + (\frac{3L^2\alpha_t}{2b_tm} - \frac{\alpha_t^2L^3}{4m^2b_t} + \frac{m}{2\alpha_tp} - \frac{L}{4p})\mathbb{E}[\|y_k^t-x_t\|^2] + (\frac{3\alpha_t}{2m}-\frac{\alpha_t^2L}{4m^2})\frac{I(B_t<n)\sigma^2}{B_t}\end{aligned} \tag{48}$$

Let $p=2k-1$ and take summation with respect to the inner loop parameter k, we can get

$$\begin{aligned}&\mathbb{E}[F(x^{t+1})]\\ &\le \mathbb{E}[F(x^t)] - \sum_{k=1}^{K}(\frac{m}{2\alpha_t(2k)}-\frac{L}{4(2k)})\mathbb{E}(\|y_{k+1}^t-x_t\|_2^2) - \sum_{k=1}^{K}(\frac{m\alpha_t}{4}-\frac{L\alpha_t^2}{8})\mathbb{E}[\|g_{Y,k}^t\|_2^2]\\ &\quad + \sum_{k=1}^{K}(\frac{3L^2\alpha_t}{2b_tm} - \frac{\alpha_t^2L^3}{4m^2b_t} + \frac{m}{2\alpha_t(2k-1)} - \frac{L}{4(2k-1)})\mathbb{E}[\|y_k^t-x_t\|^2] + \sum_{k=1}^{K}(\frac{3\alpha_t}{2m}-\frac{\alpha_t^2L}{4m^2})\frac{I(B_t<n)\sigma^2}{B_t}\\ &\le \mathbb{E}[F(x^t)] - \sum_{k=1}^{K-1}(\frac{m}{2\alpha_t(2k)}-\frac{L}{4(2k)})\mathbb{E}(\|y_{k+1}^t-x_t\|_2^2) - \sum_{k=1}^{K}(\frac{m\alpha_t}{4}-\frac{L\alpha_t^2}{8})\mathbb{E}[\|g_{Y,k}^t\|_2^2]\\ &\quad + \sum_{k=2}^{K}(\frac{3L^2\alpha_t}{2b_tm} - \frac{\alpha_t^2L^3}{4m^2b_t} + \frac{m}{2\alpha_t(2k-1)} - \frac{L}{4(2k-1)})\mathbb{E}[\|y_k^t-x_t\|^2] + \sum_{k=1}^{K}(\frac{3\alpha_t}{2m}-\frac{\alpha_t^2L}{4m^2})\frac{I(B_t<n)\sigma^2}{B_t}\\ &\le \mathbb{E}[F(x^t)] - \sum_{k=1}^{K}(\frac{m\alpha_t}{4}-\frac{L\alpha_t^2}{8})\mathbb{E}[\|g_{Y,k}^t\|_2^2] + \sum_{k=1}^{K}(\frac{3\alpha_t}{2m}-\frac{\alpha_t^2L}{4m^2})\frac{I(B_t<n)\sigma^2}{B_t}\\ &\quad + \sum_{k=1}^{K-1}(\frac{3L^2\alpha_t}{2b_tm} - \frac{\alpha_t^2L^3}{4m^2b_t} + \frac{m}{2\alpha_t(2k+1)} - \frac{L}{4(2k+1)} - (\frac{m}{2\alpha_t(2k)}-\frac{L}{4(2k)}))\mathbb{E}[\|y_k^t-x_t\|^2]\\ &= \mathbb{E}[F(x^t)] - \sum_{k=1}^{K}(\frac{m\alpha_t}{4}-\frac{L\alpha_t^2}{8})\mathbb{E}[\|g_{Y,k}^t\|_2^2] + \sum_{k=1}^{K}(\frac{3\alpha_t}{2m}-\frac{\alpha_t^2L}{4m^2})\frac{I(B_t<n)\sigma^2}{B_t}\\ &\quad + \sum_{k=1}^{K-1}(\frac{3L^2\alpha_t}{2b_tm} - \frac{\alpha_t^2L^3}{4m^2b_t} + (\frac{L}{4}-\frac{m}{2\alpha_t})(\frac{1}{2k(2k+1)}))\mathbb{E}[\|y_k^t-x_t\|^2]\end{aligned} \tag{49}$$

where the second inequality is due to the fact that $x_t = y_1^t$ and $\|x_{t+1} - x_t\| > 0$. Take $\alpha_t = m/L$

$$\begin{aligned}\mathbb{E}[F(x^{t+1})] &\leq \mathbb{E}[F(x^t)] - \sum_{k=1}^{K} \frac{m^2}{8L}\mathbb{E}[\|g_{Y,k}^t\|_2^2] + \sum_{k=1}^{K}(\frac{5}{4L})\frac{I(B_t < n)\sigma^2}{B_t} + \sum_{k=1}^{K-1}(\frac{5L}{4b_t} - \frac{L}{8k(2k+1)})\mathbb{E}[\|y_k^t - x_t\|^2] \\ &\leq \mathbb{E}[F(x^t)] - \sum_{k=1}^{K} \frac{m^2}{8L}\mathbb{E}[\|g_{Y,k}^t\|_2^2] + \sum_{k=1}^{K}(\frac{5}{4L})\frac{I(B_t < n)\sigma^2}{B_t}\end{aligned} \tag{50}$$

where the last inequality follows from the setting $K \leq \left\lfloor \sqrt{b_t/20} \right\rfloor$ and therefore

$$\frac{5L}{4b_t} - \frac{L}{8(K-1)(2(K-1)+1)} \leq \frac{5L}{4b_t} - \frac{L}{16K^2} \leq 0 \tag{51}$$

Take sum with respect to the outer loop parameter t and re-arrange the inequality

$$\begin{aligned}\sum_{t=1}^{T}\sum_{k=1}^{K} \frac{m^2}{8L}\mathbb{E}[\|g_{Y,k}^t\|_2^2] &\leq \mathbb{E}[F(x^1) - F(x^{T+1})] + \sum_{t=1}^{T}\sum_{k=1}^{K}(\frac{5}{4L})\frac{I(B_t < n)\sigma^2}{B_t} \\ &\leq \Delta_F + TK(\frac{5}{4L})\frac{I(B_t < n)\sigma^2}{B_t}\end{aligned} \tag{52}$$

Therefore when taking $B_t = n \wedge 20\sigma^2/(m^2\epsilon)$, $T = 1 \vee 16\Delta_F L/(m^2\epsilon K)$

$$\mathbb{E}[\|g_{X,t^*}\|_2^2] \leq \frac{8\Delta_F L}{m^2 TK} + \frac{10I(B_t < n)\sigma^2}{B_t m^2} \leq \frac{\epsilon}{2} + \frac{\epsilon}{2} \leq \epsilon \tag{53}$$

The total number of stochastic gradient computations is

$$\begin{aligned}TB + TKb &= O\left((n \wedge \frac{\sigma^2}{\epsilon} + b\sqrt{b})(1 + \frac{1}{\epsilon\sqrt{b}})\right) \\ &= O\left(n \wedge \frac{\sigma^2}{\epsilon} + b\sqrt{b} + \frac{n}{\epsilon\sqrt{b}} \wedge \frac{\sigma^2}{\epsilon^2\sqrt{b}} + \frac{b}{\epsilon}\right) \\ &= O\left(\frac{n}{\epsilon\sqrt{b}} \wedge \frac{\sigma^2}{\epsilon^2\sqrt{b}} + \frac{b}{\epsilon}\right)\end{aligned} \tag{54}$$

where the last inequality is because $b^2 \leq \epsilon^{-2}$ when $b \leq \epsilon^{-1}$ and $\sqrt{b} \leq \epsilon^{-1}$ when $b \leq \epsilon^{-2}$. However,we will never let b to be as large as ϵ^{-2} as it is even larger than the batch size B_t and doing so will make the number of gradient computations $O(\epsilon^{-3})$, which is undesirable.

C Convergence of SVRAMD under the P-L Condition

Proof of Theorem 4. Recall the definition of the PL condition and modify the notations a little bit, we get

$$\exists \mu > 0, s.t. \|g_{Y,k}^t\|^2 \geq 2\mu(F(y_k^t) - F^*) \tag{55}$$

By the proof in appendix B, we know that

$$\begin{aligned}\mathbb{E}[F(y_{k+1}^t)] &\leq \mathbb{E}[F(y_k^t)] - (\frac{m}{2\alpha_t} - \frac{L}{4})\mathbb{E}(\frac{\|y_{k+1}^t - x_t\|_2^2}{1+p} - \frac{\|y_k^t - x_t\|_2^2}{p}) - (\frac{m\alpha_t}{4} - \frac{L\alpha_t^2}{8})\mathbb{E}[\|g_{Y,k}^t\|_2^2] \\ &\quad + (\frac{3L^2\alpha_t}{2b_t m} - \frac{\alpha_t^2 L^3}{4m^2 b_t})\mathbb{E}[\|y_k^t - x_t\|^2] + (\frac{3\alpha_t}{2m} - \frac{\alpha_t^2 L}{4m^2})\frac{I(B_t < n)\sigma^2}{B_t} \\ &\leq \mathbb{E}[F(y_k^t)] - (\frac{m}{2\alpha_t} - \frac{L}{4})\mathbb{E}(\frac{\|y_{k+1}^t - x_t\|_2^2}{1+p}) - (\frac{m\alpha_t}{2} - \frac{L\alpha_t^2}{4})\mu(\mathbb{E}[F(y_k^t)] - F^*) \\ &\quad + (\frac{3L^2\alpha_t}{2b_t m} - \frac{\alpha_t^2 L^3}{4m^2 b_t} + \frac{m}{2\alpha_t p} - \frac{L}{4p})\mathbb{E}[\|y_k^t - x_t\|^2] + (\frac{3\alpha_t}{2m} - \frac{\alpha_t^2 L}{4m^2})\frac{I(B_t < n)\sigma^2}{B_t}\end{aligned} \tag{56}$$

Therefore when $p = 2k-1$, define $\gamma := (1-(\frac{m\alpha_t\mu}{2} - \frac{L\alpha_t^2\mu}{4}))$, we obtain

$$\begin{aligned}\frac{\mathbb{E}[F(y_{k+1}^t)] - F^*}{\gamma^{k+1}} \leq & \frac{(\mathbb{E}[F(y_k^t)] - F^*)}{\gamma^k} - (\frac{m}{2\alpha_t\gamma^{k+1}} - \frac{L}{4\gamma^{k+1}})\mathbb{E}(\frac{\|y_{k+1}^t - x_t\|_2^2}{2k}) \\ & + \frac{1}{\gamma^{k+1}}(\frac{3L^2\alpha_t}{2b_t m} - \frac{\alpha_t^2 L^3}{4m^2 b_t} + \frac{m}{2\alpha_t(2k-1)} - \frac{L}{4(2k-1)})\mathbb{E}[\|y_k^t - x_t\|^2] \\ & + \frac{1}{\gamma^{k+1}}(\frac{3\alpha_t}{2m} - \frac{\alpha_t^2 L}{4m^2})\frac{I(B_t < n)\sigma^2}{B_t}\end{aligned} \tag{57}$$

Summing up with respect to the inner loop parameter k, we get that

$$\begin{aligned}\mathbb{E}[F(x_{t+1})] - F^* \leq & \gamma^K(\mathbb{E}[F(x_t)] - F^*) - \gamma^{K+1}\sum_{k=1}^{K}(\frac{m}{2\alpha_t\gamma^{k+1}} - \frac{L}{4\gamma^{k+1}})\mathbb{E}(\frac{\|y_{k+1}^t - x_t\|_2^2}{2k}) \\ & + \gamma^{K+1}\sum_{k=1}^{K}\frac{1}{\gamma^{k+1}}(\frac{3L^2\alpha_t}{2b_t m} - \frac{\alpha_t^2 L^3}{4m^2 b_t} + \frac{m}{2\alpha_t(2k-1)} - \frac{L}{4(2k-1)})\mathbb{E}[\|y_k^t - x_t\|^2] \\ & + \gamma^{K+1}\sum_{k=1}^{K}\frac{1}{\gamma^{k+1}}(\frac{3\alpha_t}{2m} - \frac{\alpha_t^2 L}{4m^2})\frac{I(B_t < n)\sigma^2}{B_t} \\ = & \gamma^K(\mathbb{E}[F(x_t)] - F^*) - \gamma^{K+1}\sum_{k=1}^{K}(\frac{m}{2\alpha_t\gamma^{k+1}} - \frac{L}{4\gamma^{k+1}})\mathbb{E}(\frac{\|y_{k+1}^t - x_t\|_2^2}{2k}) \\ & + \gamma^{K+1}\sum_{k=1}^{K}\frac{1}{\gamma^{k+1}}(\frac{3L^2\alpha_t}{2b_t m} - \frac{\alpha_t^2 L^3}{4m^2 b_t} + \frac{m}{2\alpha_t(2k-1)} - \frac{L}{4(2k-1)})\mathbb{E}[\|y_k^t - x_t\|^2] \\ & + \frac{1-\gamma^K}{1-\gamma}(\frac{3\alpha_t}{2m} - \frac{\alpha_t^2 L}{4m^2})\frac{I(B_t < n)\sigma^2}{B_t}\end{aligned} \tag{58}$$

By the fact that $x_t = x_1^t$, and $\|x_{t+1} - x_t\| > 0$, we know that

$$
\begin{aligned}
&\mathbb{E}[F(x_{t+1})] - F^* \\
&\leq \gamma^K(\mathbb{E}[F(x_t)] - F^*) - \gamma^{K+1}\sum_{k=1}^{K-1}(\frac{m}{2\alpha_t\gamma^{k+1}} - \frac{L}{4\gamma^{k+1}})\mathbb{E}(\frac{\|y_{k+1}^t - x_t\|_2^2}{2k}) \\
&\quad + \gamma^{K+1}\sum_{k=2}^{K}\frac{1}{\gamma^{k+1}}(\frac{3L^2\alpha_t}{2b_tm} - \frac{\alpha_t^2L^3}{4m^2b_t} + \frac{m}{2\alpha_t(2k-1)} - \frac{L}{4(2k-1)})\mathbb{E}[\|y_k^t - x_t\|^2] \\
&\quad + \frac{1-\gamma^K}{1-\gamma}(\frac{3\alpha_t}{2m} - \frac{\alpha_t^2L}{4m^2})\frac{I(B_t < n)\sigma^2}{B_t} \\
&= \gamma^K(\mathbb{E}[F(x_t)] - F^*) + \frac{1-\gamma^K}{1-\gamma}(\frac{3\alpha_t}{2m} - \frac{\alpha_t^2L}{4m^2})\frac{I(B_t < n)\sigma^2}{B_t} \\
&\quad - \gamma^{K+1}\sum_{k=1}^{K-1}(\frac{m}{2\alpha_t\gamma^{k+1}} - \frac{L}{4\gamma^{k+1}})\mathbb{E}(\frac{\|y_{k+1}^t - x_t\|_2^2}{2k}) \\
&\quad + \gamma^{K+1}\sum_{k=1}^{K-1}\frac{1}{\gamma^{k+1}}(\frac{3L^2\alpha_t}{2b_tm\gamma} - \frac{\alpha_t^2L^3}{4m^2b_t\gamma} + \frac{m}{2\alpha_t(2k+1)\gamma} - \frac{L}{4(2k+1)\gamma})\mathbb{E}[\|y_k^t - x_t\|^2] \\
&= \gamma^K(\mathbb{E}[F(x_t)] - F^*) + \frac{1-\gamma^K}{1-\gamma}(\frac{3\alpha_t}{2m} - \frac{\alpha_t^2L}{4m^2})\frac{I(B_t < n)\sigma^2}{B_t} \\
&\quad + \gamma^{K+1}\sum_{k=1}^{K-1}\frac{1}{\gamma^{k+2}}(\frac{3L^2\alpha_t}{2b_tm} - \frac{\alpha_t^2L^3}{4m^2b_t} + \frac{m}{2\alpha_t(2k+1)} - \frac{L}{4(2k+1)} + \frac{L\gamma}{8k} - \frac{m\gamma}{4k\alpha_t})\mathbb{E}[\|y_k^t - x_t\|^2] \\
&= \gamma^K(\mathbb{E}[F(x_t)] - F^*) + \frac{1-\gamma^K}{1-\gamma}(\frac{3\alpha_t}{2m} - \frac{\alpha_t^2L}{4m^2})\frac{I(B_t < n)\sigma^2}{B_t} \\
&\quad + \gamma^{K+1}\sum_{k=1}^{K-1}\frac{1}{\gamma^{k+2}}(\frac{3L^2\alpha_t}{2b_tm} - \frac{\alpha_t^2L^3}{4m^2b_t} - (\frac{m}{2\alpha_t} - \frac{L}{4})(\frac{\gamma}{2k} - \frac{1}{2k+1}))\mathbb{E}[\|y_k^t - x_t\|^2]
\end{aligned}
\tag{59}
$$

By the definition $\gamma = 1 - \frac{m\alpha_t\mu}{2} + \frac{L\alpha_t^2\mu}{4}$, we know that

$$
\begin{aligned}
\frac{\gamma}{2k} - \frac{1}{2k+1} &= \frac{1}{2k(2k+1)} - \frac{m\alpha_t\mu}{4k} + \frac{L\alpha_t^2\mu}{8k} \\
&= \frac{1}{2k(2k+1)} - \frac{\alpha_t^2\mu}{2k}(\frac{m}{2\alpha_t} - \frac{L}{4})
\end{aligned}
\tag{60}
$$

Therefore when taking $\alpha_t = \frac{m}{L}$ and with the assumption $L/(\mu m^2) > \sqrt{n}$, the last term in the inequality (59) is

$$
\begin{aligned}
&\gamma^{K+1}\sum_{k=1}^{K-1}\frac{1}{\gamma^{k+2}}(\frac{3L^2\alpha_t}{2b_tm} - \frac{\alpha_t^2L^3}{4m^2b_t} - (\frac{m}{2\alpha_t} - \frac{L}{4})(\frac{\gamma}{2k} - \frac{1}{2k+1})\mathbb{E}[\|y_k^t - x_t\|^2] \\
&= \gamma^{K+1}\sum_{k=1}^{K-1}\frac{1}{\gamma^{k+2}}(\frac{3L^2\alpha_t}{2b_tm} - \frac{\alpha_t^2L^3}{4m^2b_t} - (\frac{m}{2\alpha_t} - \frac{L}{4})(\frac{1}{2k(2k+1)} - \frac{\alpha_t^2\mu}{2k}(\frac{m}{2\alpha_t} - \frac{L}{4})))\mathbb{E}[\|y_k^t - x_t\|^2] \\
&= \gamma^{K+1}\sum_{k=1}^{K-1}\frac{1}{\gamma^{k+2}}(\frac{3L^2\alpha_t}{2b_tm} - \frac{\alpha_t^2L^3}{4m^2b_t} - (\frac{m}{2\alpha_t} - \frac{L}{4})(\frac{1}{2k(2k+1)}) + \frac{\alpha_t^2\mu}{2k}(\frac{m}{2\alpha_t} - \frac{L}{4})^2))\mathbb{E}[\|y_k^t - x_t\|^2] \\
&\leq \gamma^{K+1}\sum_{k=1}^{K-1}\frac{1}{\gamma^{k+2}}(\frac{5L}{4b_t} - \frac{L}{4}(\frac{1}{2k(2k+1)}) + \frac{L}{32k\sqrt{n}})\mathbb{E}[\|y_k^t - x_t\|^2]
\end{aligned}
\tag{61}
$$

Define $H(x) := -\frac{1}{2x(2x+1)} + \frac{1}{8x\sqrt{n}} + \frac{5}{b_t}$, $H'(x) = \frac{8x+2}{4x^2(2x+1)^2} - \frac{1}{8x^2\sqrt{n}} = \frac{1}{4x^2}(\frac{8x+2}{4x^2+4x+1} - \frac{1}{2\sqrt{n}}) = \frac{1}{4x^2}(\frac{2(8x+2)\sqrt{n}-(4x^2+4x+1)}{2(4x^2+4x+1)\sqrt{n}})$. When $x \leq K-1 < K < \sqrt{\frac{b_t}{16}} \leq \sqrt{\frac{n}{16}}$, $\frac{8x+2}{4x^2+4x+1} -$

$\frac{1}{2\sqrt{n}} \geq \frac{8K+2}{4K^2+4K+1} - \frac{1}{2\sqrt{n}} \geq 0$. Therefore $H(x) \leq H(K-1) \leq \frac{5}{b_t} - \frac{14K+1}{32K(K-1)(2K-1)} \leq 0$ when $K = \lfloor\sqrt{\frac{b_t}{32}}\rfloor$. which means the inequality above is smaller than zero. Hence

$$\mathbb{E}[F(x_{t+1})] - F^* \leq \gamma^K(\mathbb{E}[F(x_t)] - F^*) + \frac{1-\gamma^K}{1-\gamma}\frac{5L}{4}\frac{I(B_t < n)\sigma^2}{B_t} \tag{62}$$

Therefore

$$\frac{\mathbb{E}[F(x_{t+1})] - F(x_t)}{\gamma^{K(t+1)}} \leq \frac{(\mathbb{E}[F(x_t)] - F^*)}{\gamma^{Kt}} + \frac{1-\gamma^K}{(1-\gamma)\gamma^{K(t+1)}}(\frac{5L}{4}\frac{I(B_t < n)\sigma^2}{B_t}) \tag{63}$$

Now take sum with respect to the outer loop parameter t and take B_t as a constant, we can get

$$\begin{aligned}
\mathbb{E}[F(x_{T+1})] - F^* &\leq \gamma^{KT}(F(x_1) - F^*) + \gamma^{K(T+1)}\sum_{t=1}^{T}\frac{1-\gamma^K}{(1-\gamma)\gamma^{K(t+1)}}\frac{5L}{4}\frac{I(B_t < n)\sigma^2}{B_t} \\
&\leq \gamma^{KT}\Delta_F + \gamma^{K(T+1)}\frac{1-\gamma^K}{1-\gamma}\sum_{t=1}^{T}\frac{1}{\gamma^{K(t+1)}}\frac{5L}{4}\frac{I(B_t < n)\sigma^2}{B_t} \\
&= \gamma^{KT}\Delta_F + \gamma^{K(T+1)}\frac{1-\gamma^K}{1-\gamma}\sum_{t=1}^{T}\frac{1}{\gamma^{K(t+1)}}\frac{5L}{4}\frac{I(B_t < n)\sigma^2}{B_t} \\
&= \gamma^{KT}\Delta_F + \frac{5LI(B_t < n)\sigma^2}{4B_t}\frac{1-\gamma^K}{1-\gamma}\frac{1-\gamma^{KT}}{1-\gamma^K} \\
&= \gamma^{KT}\Delta_F + \frac{5LI(B_t < n)\sigma^2}{4B_t}\frac{1-\gamma^{KT}}{1-\gamma}
\end{aligned} \tag{64}$$

Since $1-\gamma^{KT} < 1$, $\gamma = 1 - \frac{m\alpha_t\mu}{2} + \frac{L\alpha_t^2\mu}{4} = 1 - \frac{m^2\mu}{2L} + \frac{m^2\mu}{4L} = 1 - \frac{m^2\mu}{4L}$, hence

$$\begin{aligned}
\mathbb{E}[F(x_{T+1})] - F^* &\leq \gamma^{KT}\Delta_F + \frac{5LI(B_t < n)\sigma^2}{4B_t(1-\gamma)} \\
&= \gamma^{KT}\Delta_F + \frac{5I(B_t < n)\sigma^2}{B_t m^2\mu}
\end{aligned} \tag{65}$$

Therefore when taking $T = 1 \vee (\log\frac{2\Delta_F}{\epsilon})/(K\log\frac{1}{\gamma}) = O((\log\frac{2\Delta_F}{\epsilon})/(K\mu))$, $B_t = n \wedge \frac{10\sigma^2}{\epsilon m^2\mu}$. Then the total number of stochastic gradient computations is

$$\begin{aligned}
TB + TKb &= O\left((n \wedge \frac{\sigma^2}{\mu\epsilon} + b\sqrt{b})(\frac{1}{\mu\sqrt{b}}\log\frac{1}{\epsilon})\right) \\
&= O\left((n \wedge \frac{\sigma^2}{\mu\epsilon})\frac{1}{\mu\sqrt{b}}\log\frac{1}{\epsilon} + \frac{b}{\mu}\log\frac{1}{\epsilon}\right)
\end{aligned} \tag{66}$$

D Algorithm Implementation and More Experimental Details

D.1 Algorithm

We provide the implementation of Variance Reduced AdaGrad (VR-AdaGrad) in Algorithm 4 when $h(x) = 0$. Note that this implementation is actually a simple combination of the AdaGrad algorithm and the SVRAMD algorithm The implementation can be further extended to the case when $h(x) \neq 0$, but the form would depend on the regularization function $h(x)$. For example, the AdaGrad algorithm with $h(x) = \|x\|_1$, a nonsmooth regularization, can be found in Duchi et al. (2011). The VR-AdaGrad algorithm with $h(x) = \|x\|_1$ will therefore have a similar form. To change the algorithm into VR-RMSProp, one can simply replace the global sum design of the denominator with the exponential moving average in line 9.

Algorithm 4 Variance Reduced AdaGrad Algorithm

Input: Number of stages T, initial x_1, step sizes $\{\alpha_t\}_{t=1}^T$, batch sizes $\{B_t\}_{t=1}^T$, mini-batch sizes $\{b_t\}_{t=1}^T$, constant m
for $t = 1$ **to** T **do**
  Randomly sample a batch $\mathcal{I}_t$ with size B_t
  $g_t = \nabla f_{\mathcal{I}_t}(x_t)$
  $y_1^t = x_t$
  for $k = 1$ **to** K **do**
    Randomly pick sample $\tilde{\mathcal{I}}_t$ of size b_t
    $v_k^t = \nabla f_{\tilde{\mathcal{I}}_t}(y_k^t) - \nabla f_{\tilde{\mathcal{I}}_t}(y_1^t) + g_t$
    $y_{k+1}^t = y_k^t - \alpha_t v_k^t / (\sqrt{\sum_{\tau=1}^{t-1}\sum_{i=1}^{K} v_i^{\tau 2} + \sum_{i=1}^{k} v_i^{t2}} + m)$
  end for
  $x_{t+1} = y_{K+1}^t$
end for
Return (Smooth case) Uniformly sample x_{t^*} from $\{y_k^t\}_{k=1,t=1}^{K,T}$; (P-L case) $x_{t^*} = x_{T+1}$

D.2 Experiment Details

Datasets. We used three datasets in our experiments. The MNIST (Schölkopf and Smola, 2002) dataset has 50k training images and 10k testing images of handwritten digits. The images were normalized before fitting into the neural networks. The CIFAR10 dataset (Krizhevsky et al., 2009) also has 50k training images and 10k testing images of different objects in 10 classes. The images were normalized with respect to each channel (3 channels in total) before fitting into the network. The CIFAR-100 dataset splits the original CIFAR-10 dataset further into 100 classes, each with 500 training images and 100 testing images.

Network Architecture. For the MNIST dataset, we used a one-hidden layer fully connected neural network as the architecture. The hidden layer size was 64 and we used the Relu activation function (Nair and Hinton, 2010). The logsoftmax activation function was applied to the final output. For CIFAR-10, we used the standard LeNet (LeCun et al., 1998) with two layers of convolutions of size 5. The two layers have 6 and 16 channels respectively. Relu activation and max pooling are applied to the output of each convolutional layer. The output is then applied sequentially to three fully connected layers of size 120, 84 and 10 with Relu activation functions. For CIFAR-100, the ResNet-20 model follows the official implementation in He et al. (2016) by Li et al. (2020).

Parameter Tuning. For the initial step size α_0, we have tuned over $\{1, 0.5, 0.1, 0.01, 0.005, 0.002, 0.001\}$ for all the algorithms. For the batch sizes of the original algorithms, we have followed the choices by Zhou et al. (2018b) for ProxSGD and adaptive algorithms (Zhou et al. (2018b) used Adam but here we use AdaGrad and RMSProp). For the batch sizes and mini batch sizes B_t and b_t of the variance reduced algorithms, we have tuned over $b_t = \{64, 128, 256, 512, 1024\}$ and $r = \{4, 8, 16, 32, 64\}$. For the constant m added to the denominator matrix H_t in AdaGrad and RMSProp, we choose a reasonable value m=1e-3, which is slightly larger than the values set in the original implementations to guarantee strong convexity (Kingma and Ba, 2015). The other parameters are set to be the default values. For example, the exponential moving average parameter β in RMSProp (and VR-RMSProp) is set to be 0.999. No weight decay is applied to any algorithm in our experiments. The parameters that generate the reported results are provided in Table 3, 4, and 5.

D.3 Additional Experiments

We provide the performances of AdaGrad, RMSProp and their variance reduced variants with different step sizes in figure 4 and 5. A too-large step size actually makes the convergence of AdaGrad and RMSProp slower. Note that variance reduction always works well in these figures, and it results in faster convergence and better testing accuracy under both step size settings. Therefore for different step sizes, we can apply variance reduction to get faster training and better performances.

Table 3: Parameter settings on the MNIST dataset. The batch size B_t is equal to $b_t * r$.

Algorithms	Step size	Mini batch size b_t	Batch size ratio r
ProxSGD	0.1	1024	N.A.
AdaGrad	0.001	2048	N.A.
RMSProp	0.001	1024	N.A.
ProxSVRG+	0.1	256	32
VR-AdaGrad	0.001	256	32
VR-RMSProp	0.001	256	64

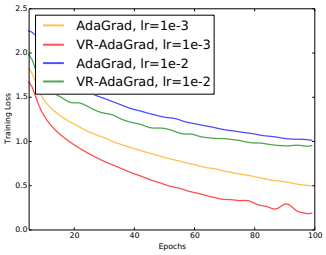
Table 4: Parameter settings on the CIFAR-10 dataset. The batch size B_t is equal to $b_t * r$.

Algorithms	Step size	Mini batch size b_t	Batch size ratio r
ProxSGD	0.1	1024	N.A.
AdaGrad	0.001	1024	N.A.
RMSProp	0.001	1024	N.A.
ProxSVRG+	0.1	512	64
VR-AdaGrad	0.001	512	64
VR-RMSProp	0.001	512	64

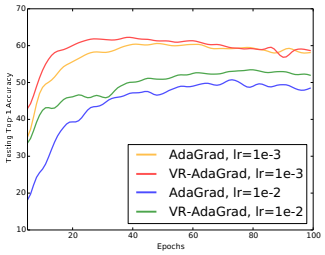
Table 5: Parameter settings on the CIFAR-100 dataset. The batch size B_t is equal to $b_t * r$.

Algorithms	Step size	Mini batch size b_t	Batch size ratio r
ProxSGD	0.1	1024	N.A.
AdaGrad	0.001	1024	N.A.
RMSProp	0.001	1024	N.A.
ProxSVRG+	0.1	512	64
VR-AdaGrad	0.001	512	64
VR-RMSProp	0.001	512	64

The baseline ratios of ProxSVRG+ and ProxSGD in different datasets are provided in Figure 6. Note that ProxSVRG+ only needs a small batch size ratio ($r = 4$) to be faster than ProxSGD.

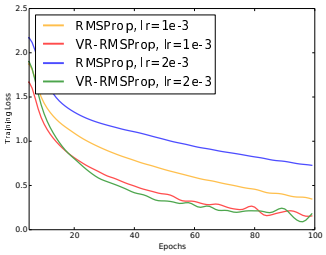


(a) CIFAR10 Training Loss.

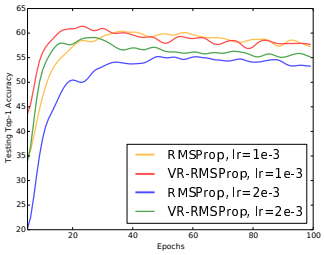


(b) CIFAR10 Testing Acc.

Fig. 4: Comparison of AdaGrad and VR-AdaGrad on CIFAR-10 using different learning rates. The other parameters are the same as in Table 4. "lr" stands for learning rate, which is a different name for step size. The results are averaged over 5 independent runs

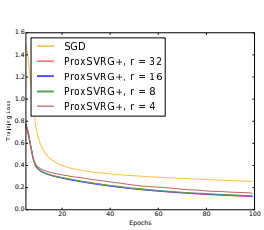


(a) CIFAR10 Training Loss.

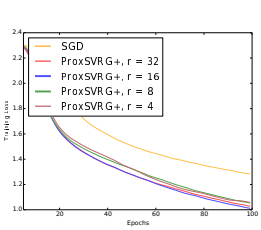


(b) CIFAR10 Testing Acc.

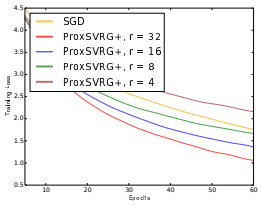
Fig. 5: Comparison of RMSProp and VR-RMSProp on CIFAR-10 using different learning rates. The other parameters are the same as in Table 4. "lr" stands for learning rate, which is a different name for step size. lr=1e-2 is too large for RMSProp and the algorithm diverges. The results are averaged over 5 independent runs



(a) ProxSVRG+ on MNIST



(b) ProxSVRG+ on CIFAR10



(c) ProxSVRG+ on CIFAR100

Fig. 6: 6a. training loss of ProxSGD and ProxSVRG+ with different r on MNIST. 6b. training loss of ProxSGD and ProxSVRG+ with different r on CIFAR-10. 6c. training loss of ProxSGD and ProxSVRG+ with different r on CIFAR-100. The other parameters are the same as in Table 3, 4. The mini batch size is set to be the same as AdaGrad and RMSProp to ensure fair comparisons. The results were averaged over three independent runs.