History-Gradient Aided Batch Size Adaptation for Variance Reduced Algorithms

Kaiyi Ji [1] **Zhe Wang** [1] **Bowen Weng** [1] **Yi Zhou** [2] **Wei Zhang** [3] **Yingbin Liang** [1]

Abstract

Variance-reduced algorithms, although achieve great theoretical performance, can run slowly in practice due to the periodic gradient estimation with a large batch of data. Batch-size adaptation thus arises as a promising approach to accelerate such algorithms. However, existing schemes either apply prescribed batch-size adaption rule or exploit the information along optimization path via additional backtracking and condition verification steps. In this paper, we propose a novel scheme, which eliminates backtracking line search but still exploits the information along optimization path by adapting the batch size via history stochastic gradients. We further theoretically show that such a scheme substantially reduces the overall complexity for popular variance-reduced algorithms SVRG and SARAH/SPIDER for both conventional nonconvex optimization and reinforcement learning problems. To this end, we develop a new convergence analysis framework to handle the dependence of the batch size on history stochastic gradients. Extensive experiments validate the effectiveness of the proposed batch-size adaptation scheme.

1. Introduction

Stochastic gradient descent (SGD) (Ghadimi & Lan, 2013) algorithms have been extensively used to efficiently solve large-scale optimization problems recently. Furthermore, various variance reduced algorithms such as SAGA (Defazio et al., 2014), SVRG (Johnson & Zhang, 2013; Reddi et al., 2016a), SARAH (Nguyen et al., 2017a;b), and SPIDER (Fang et al., 2018)/SpiderBoost (Wang et al., 2019), have been proposed to reduce the variance of SGD. Such variance reduction techniques have also been applied to policy gradient algorithms to develop SVRPG (Papini et al., 2018), SRVR-PG (Xu et al., 2019b) and SARAPO (Yuan et al., 2018) in reinforcement learning (RL). Though variance reduced algorithms have been shown to have order-level lower computational complexity than SGD (and than vanilla policy gradient in RL), they often do not perform as well as SGD in practice, largely due to the periodic large-batch gradient estimation. In fact, variance-reduced gradient estimation plays an important role only towards the later stage of the algorithm execution, and hence a promising way to accelerate variance reduced algorithms is to adaptively increase the batch size.

Two types of batch-size adaptation schemes have been proposed so far to accelerate stochastic algorithms (Smith et al., 2018; Friedlander & Schmidt, 2012; Devarakonda et al., 2017). The first approach follows a *prescribed* rule to adapt the batch size, which can be *exponential* and *polynomial* increase of batch size as in hybrid SGD (HSGD) (Friedlander & Schmidt, 2012; Zhou et al., 2018b) and *linear* increase of batch size (Zhou et al., 2018b). Moreover, Harikandeh et al. 2015 proposed to use exponential increase of batch size at each outer-loop iteration for SVRG.

The second approach adapts the batch size based on the information along the optimization path. For example, De et al. 2016; 2017 proposed Big Batch SGD, which adapts the batch size so that the resulting gradient and variance satisfy certain optimization properties. Since the batch size needs to be chosen *even before* its resulting gradient is calculated, the algorithm adopts the backtracking line search to iteratively check that the chosen batch size ensures the resulting gradient to satisfy a variance bound. Clearly, the backtracking step adds undesired complexity, but seems to be unavoidable, because the convergence analysis exploits the *instantaneous* variance bound.

Our contribution lies in designing an easy-to-implement scheme to adapt the batch size, which incorporates the information along the optimization path, but does not involve backtracking and condition verification. We further show by both theory and experiments that such a scheme achieves much better performance than vanilla variance reduced algo-

[1]Department of Electrical and Computer Engineering, The Ohio State University [2]Department of Electrical and Computer Engineering, University of Utah [3]Department of Mechanical Engineering, Southern University of Science and Technology. Correspondence to: Kaiyi Ji <ji.367@osu.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020.

rithms in both conventional optimization and RL problems.

1.1. Our Contributions

New batch-size adaptation scheme via history gradients: We propose to adapt the batch size of each epoch (i.e., each outer loop) of variance reduced algorithms SVRG and SPIDER inversely proportional to the average of stochastic gradients over each epoch, and call the algorithms as **A**daptive **ba**tch-size SVRG (AbaSVRG) and AbaSPIDER. We further apply the scheme to the variance reduced policy gradient algorithms SVRPG and SPIDER-PG (which refers to SRVR-PG in Xu et al. 2019b) in RL, and call the resulting algorithms as AbaSVRPG and AbaSPIDER-PG. These algorithms initially use small batch size (due to large gradients) and enjoy fast iteration, and then gradually increase the batch size (due to the reduced gradients) and enjoy reduced variance and stable convergence. Further technical justification is provided in Section 2.1.

Since the batch size should be set at the beginning of the epoch at which point the gradients in that epoch has not been calculated yet. It is a similar situation as in De et al. 2016; 2017, which introduced the backtracking line search to guarantee the variance bound. Here, we propose to use the average of stochastic gradients over the *preceding* epoch as an approximation of the present gradient information to avoid the complexity of backtracking line search, which we further show theoretically to still achieve guaranteed improved performance.

New convergence analysis: Since the updates in our algorithms depend on the past gradients, it becomes much more challenging to establish the provable convergence guarantee. The technical novelty of our analysis mainly lies in the following two aspects.

- We develop a new framework to analyze the convergence of variance reduced algorithms with batch size adapted to history gradients for nonconvex optimization. In particular, we bound the function values for each epoch by the average gradient in the preceding epoch due to the batch size dependence, which further facilitates the bounding of the accumulative change of the objective value over the entire execution. Such an analysis is different from the existing analysis of SVRG in Reddi et al. 2016a; Li & Li 2018 and SPIDER/SpiderBoost in Fang et al. 2018; Wang et al. 2019, which are based on guaranteeing the decrease of the objective value iterationwisely or epochwisely.
- We develop a simpler convergence analysis for SVRG-type algorithms than previous studies (Reddi et al., 2016a; Li & Li, 2018) for nonconvex optimization, which allows more flexible choices of parameters. More importantly, such an analysis fits well to the analysis framework we develop to handle the dependence of the adaptive batch size on the stochastic gradients in the previous epoch.

Based on the new analysis framework, we show that both AbaSVRG and AbaSPIDER for conventional nonconvex optimization and AbaSVRPG and AbaSPIDER-PG for policy optimization in RL achieve improved complexity over their corresponding vanilla counterpart (without batch-size adaptation). The worst-case complexity of these algorithms all match the best known complexity. We also provide the convergence analysis of AbaSVRG and AbaSPIDER for nonconvex problems under the PŁ condition in Appendix C.

Experiments: We provide extensive experiments on both supervised learning and RL problems and demonstrate that the proposed adaptive batch-size scheme substantially speeds up the convergence of variance reduced algorithms.

1.2. Related Work

Variance reduced algorithms for conventional optimization. In order to improve the performance of SGD (Robbins & Monro, 1951), various variance reduced algorithms have been proposed such as SAG (Roux et al., 2012), SAGA (Defazio et al., 2014), SVRG (Allen-Zhu & Hazan, 2016; Johnson & Zhang, 2013), SARAH (Nguyen et al., 2017a;b; 2019), SNVRG (Zhou et al., 2018a), SPIDER (Fang et al., 2018), SpiderBoost (Wang et al., 2019). This paper shows that two representative algorithms SVRG and SPIDER can be equipped with the proposed adaptive batch size and attain substantial performance gain.

Variance reduced policy gradient for RL: Variance reduction methods have also been applied to policy gradient methods (S. Sutton et al., 2000) in RL. One way is to incorporate a baseline in the gradient estimator, e.g., Williams 1992; Weaver & Tao 2001; Wu et al. 2018. Optimization techniques have also been applied. For example, Papini et al. 2018; Xu et al. 2019a applied SVRG to develop stochastic variance reduced policy gradient (SVRPG) algorithm. Yuan et al. 2018 and Xu et al. 2019b applied SARAH/SPIDER to develop stochastic recursive gradient policy optimization (SARAPO) and stochastic recursive variance reduced policy gradient (SRVR-PG), respectively. Shen et al. 2019 developed Hessian aided policy gradient (HAPG). This paper shows that the batch size adaptation scheme can also be applied to variance reduced policy gradient algorithms to significantly improve their performance.

Stochastic algorithms with adaptive batch size. Adaptively changing the batch size emerges as a powerful approach for accelerating stochastic algorithms (Smith et al., 2018; Friedlander & Schmidt, 2012; Devarakonda et al., 2017). Hybrid SGD (HSGD) applies *exponential* and *polynomial* increase of batch size (Friedlander & Schmidt, 2012; Zhou et al., 2018b) and *linear* increase of batch size (Zhou et al., 2018b). De et al. 2016; 2017 proposed Big Batch SGD with the batch size adaptive to the *instantaneous* gradient and variance information (which needs to be ensured, e.g., by backtracking line search) at each iteration. More-

Algorithm 1 AbaSVRG

Input: x_0, m, B, η, $c_\beta, c_\epsilon, \beta_1 > 0$
$\tilde{x}^0 = x_0$
for $s = 1, 2, ..., S$ **do**
  $x_0^s = \tilde{x}^{s-1}$
  Sample $\mathcal{N}_s$ from $[n]$ without replacement, where $N_s = \min\{c_\beta \sigma^2 \beta_s^{-1}, c_\epsilon \sigma^2 \epsilon^{-1}, n\}$
  $g^s = \nabla f_{\mathcal{N}_s}(\tilde{x}^{s-1})$
  Set $\beta_{s+1} = 0$
  for $t = 1, 2, ..., m$ **do**
    Sample $\mathcal{B}$ from $[n]$ with replacement
    $v_{t-1}^s = \nabla f_{\mathcal{B}}(x_{t-1}^s) - \nabla f_{\mathcal{B}}(\tilde{x}^{s-1}) + g^s$
    $x_t^s = x_{t-1}^s - \eta v_{t-1}^s$
    $\beta_{s+1} \leftarrow \beta_{s+1} + \|v_{t-1}^s\|^2/m$
  end for
  $\tilde{x}^s = x_m^s$
end for
Output: x_ζ from $\{x_{t-1}^s\}_{s\in[S],t\in[m]}$ uniformly at random

Algorithm 2 AbaSPIDER

Input: x_0, m, B, η, $c_\beta, c_\epsilon, \beta_1 > 0$
$\tilde{x}^0 = x_0$
for $s = 1, 2, ..., S$ **do**
  $x_0^s = \tilde{x}^{s-1}$
  Sample $\mathcal{N}_s$ from $[n]$ without replacement, where $N_s = \min\{c_\beta \sigma^2 \beta_s^{-1}, c_\epsilon \sigma^2 \epsilon^{-1}, n\}$
  $v_0^s = \nabla f_{\mathcal{N}_s}(\tilde{x}^{s-1})$
  $x_1^s = x_0^s - \eta v_0^s$
  Set $\beta_{s+1} = \|v_0^s\|^2/m$
  for $t = 1, 2, ..., m-1$ **do**
    Sample $\mathcal{B}$ from $[n]$ with replacement
    $v_t^s = \nabla f_{\mathcal{B}}(x_t^s) - \nabla f_{\mathcal{B}}(x_{t-1}^s) + v_{t-1}^s$
    $x_{t+1}^s = x_t^s - \eta v_t^s$
    $\beta_{s+1} \leftarrow \beta_{s+1} + \|v_t^s\|^2/m$
  end for
  $\tilde{x}^s = x_m^s$
end for
Output: x_ζ from $\{x_{t-1}^s\}_{s\in[S],t\in[m]}$ uniformly at random

over, Harikandeh et al. 2015 and Lei & Jordan 2019 proposed to use exponential increase of batch size at each outer-loop iteration respectively for SVRG and for an adaptively sampled variance reduced algorithm SCSG (Lei et al., 2017). Our algorithms adapt the batch size to history gradients, which differs from the prescribed adaptive schemes, and is easier to implement than Big Batch SGD by eliminating backtraking line search and still guarantees the convergence.

We note that a concurrent work (Sievert & Charles, 2019) also proposed an improved SGD algorithm by adapting the batch size to history gradients, but only as a practice without convergence proof. Our analysis framework is applicable to their algorithm as we show in Appendix D.

Notations. Let $\wedge$ and $\vee$ denote the minimum and the maximum. Let $[n] := \{1, ..., n\}$. For a set $\mathcal{S}$, let S be its cardinality and define $\nabla f_{\mathcal{S}}(\cdot) := \frac{1}{S}\sum_{i\in\mathcal{S}} \nabla f_i(\cdot)$.

2. Batch Size Adaptation for Nonconvex Optimization

In this section, we consider the following finite-sum optimization problem

$$\min_{x\in\mathbb{R}^d} f(x) := \frac{1}{n}\sum_{i=1}^{n} f_i(x). \tag{P}$$

In the context of machine learning problems, each function $f_i(\cdot)$ evaluates the loss on a particular i-th data sample, and is generally nonconvex due to the complex models.

2.1. Proposed Algorithms with Batch-Size Adaptation

Two popular variance reduced algorithms to solve the optimization problem (P) are SVRG (Johnson & Zhang, 2013) and SARAH (Nguyen et al., 2017a)/SPIDER (Fang et al., 2018), which have been shown to outperform SGD. However, SVRG and SARAH/SPIDER often run slowly in practice due to the full/large-batch gradient evaluation at the beginning of each epoch. We propose a batch-size adaptation scheme to mitigate such an issue for these algorithms, and we call the corresponding algorithms as AbaSVRG and AbaSPIDER (see Algorithms 1 and 2). Note that AbaSPIDER adopts the improved version SpiderBoost (Wang et al., 2019) of the original SPIDER (Fang et al., 2018).

We take SVRG as an example to briefly explain our idea. Our analysis of SVRG shows that the decrease of the average function value over an epoch s with length m satisfies

$$\frac{\mathbb{E}(f(\tilde{x}^{s+1}) - f(\tilde{x}^s))}{m} \le -\phi\frac{\sum_{t=0}^{m-1}\mathbb{E}\|v_t\|^2}{m} + \frac{\psi I_{(N_s<n)}}{N_s}$$

where $\phi, \psi > 0$ are constants, the indicator function $I_{(A)}$ equals 1 if the event A is true and 0 otherwise, $\tilde{x}^s$ is the snapshot in epoch s, v_t is a stochastic estimation of $\nabla f(x_t)$ within epoch s, and N_s is the batch size used at the outer-loop iteration. The above bound naturally suggests that N_s should be chosen such that the second term is at the same level as the first term, i.e., the batch size should adapt to the average stochastic gradient over the epoch, in which case the convergence guarantee follows easily. However, this is not feasible in practice, because the batch size should be chosen at the beginning of each epoch, at which point the gradients in the same loop have not been calculated yet. Such an issue was previously solved in De et al. 2016; 2017 via backtracking line search, which adds significantly additional complexity. Our main idea here is to use the stochastic gradients calculated in the previous epoch for adapting the batch size of the coming loop, and we show that such a scheme still retains the convergence guarantee and achieves improved computational complexity.

More precisely, AbaSVRG/AbaSPIDER chooses the batch size N_s at epoch s adaptively to the average β_s of stochastic gradients in the *preceding* epoch $s-1$ as given below

$$N_s = \min\{c_\beta \sigma^2 \beta_s^{-1}, c_\epsilon \sigma^2 \epsilon^{-1}, n\}, \beta_s = \frac{\sum_{t=1}^m \|v_{t-1}^{s-1}\|^2}{m},$$

where $c_\beta, c_\epsilon > 0$ are constants and σ^2 is the variance bound. As a comparison, the vanilla SVRG and SPIDER pick a *fixed* batch size to be either n or $\min\{c_\epsilon \sigma^2 \epsilon^{-1}, n\}$.

2.2. Assumptions and Definitions

We adopt the following standard assumptions (Lei et al., 2017; Reddi et al., 2016a) for convergence analysis.

Assumption 1. *The objective function in (P) satisfies:*

(1) $\nabla f_i(\cdot)$ *is L-smooth for* $i \in [n]$*, i.e., for any* $x, y \in \mathbb{R}^d$*,* $\|\nabla f_i(x) - \nabla f_i(y)\| \le L\|x - y\|$.

(2) $f(\cdot)$ *is bounded below, i.e.,* $f^* = \inf_{x\in\mathbb{R}^d} f(x) > -\infty$.

(3) $\nabla f_i(\cdot)$ *(with the index* i *uniformly randomly chosen) has bounded variance, i.e., there exists a constant* $\sigma > 0$ *such that for any* $x \in \mathbb{R}^d$*,* $\frac{1}{n}\sum_{i=1}^n \|\nabla f_i(x) - \nabla f(x)\|^2 \le \sigma^2$.

The item (3) of the bounded variance assumption is commonly adopted for proving the convergence of SGD-type algorithms (e.g., SGD (Ghadimi & Lan, 2013)) and stochastic variance reduced methods (e.g., SCSG (Lei et al., 2017)) that draw a sample batch with size less than n for gradient estimation at each outer-loop iteration.

In this paper, we use the gradient norm as the convergence criterion for nonconvex optimization.

Definition 1. *We say that* x^ζ *is an* ϵ*-accurate solution for the optimization problem (P) if* $\mathbb{E}\|\nabla f(x^\zeta)\|^2 \le \epsilon$*, where* x^ζ *is an output returned by a stochastic algorithm.*

To compare the efficiency of different stochastic algorithms, we adopt the following stochastic first-order oracle (SFO) for the analysis of the computational complexity.

Definition 2. *Given an input* $x \in \mathbb{R}^d$*, SFO randomly takes an index* $i \in [n]$ *and returns a stochastic gradient* $\nabla f_i(x)$ *such that* $\mathbb{E}_i\big[\nabla f_i(x)\big] = \nabla f(x)$.

2.3. Convergence Analysis for AbaSVRG

Since the batch size of AbaSVRG is adaptive to the *history* gradients due to the component $c_\beta \sigma^2 \beta_s^{-1}$, the existing convergence analysis for SVRG type of algorithms in Li & Li 2018; Reddi et al. 2016a does not extend easily. Here, we develop a simpler analysis for SVRG algorithm than that in Li & Li 2018; Reddi et al. 2016a (and can be of independent interest), and enables to handle the dependence of the batch size on the stochastic gradients in the past epoch in the convergence analysis for AbaSVRG. To compare more specifically, Reddi et al. 2016a introduced a Lyapunov function $R_t^s = \mathbb{E}[f(x_t^s) + c_t\|x_t^s - \tilde{x}^{s-1}\|^2]$ and proves that R^s decreases by the accumulated gradient norms $\sum_{t=0}^{m-1}\mathbb{E}\|\nabla f(x_t^s)\|^2$ within an epoch s, and Li & Li 2018 directly showed that $\mathbb{E}f(x^s)$ decreases by $\sum_{t=0}^{m-1}\mathbb{E}\|\nabla f(x_t^s)\|^2$ using tighter bounds. As a comparison, our analysis shows that $\mathbb{E}f(x^s)$ decreases by the accumulated *stochastic gradient* norms $\sum_{t=0}^{m-1}\mathbb{E}\|v_t^s\|^2$. More details about our proof can be found in Appendix E. The following theorem provides a general convergence result for AbaSVRG.

Theorem 1. *Suppose Assumption 1 is satisfied. Let* $\epsilon > 0$ *and* $c_\beta, c_\epsilon \ge \alpha$ *for certain constant* $\alpha > 0$*. Let* $\psi = \frac{2\eta^2L^2m^2}{B} + \frac{2}{\alpha} + 2$ *and choose* β_1*,* α *and* η *such that* $\beta_1 \le \epsilon S$ *and* $\phi = \frac{1}{2} - \frac{1}{2\alpha} - \frac{L\eta}{2} - \frac{\eta^2L^2m^2}{2B} > 0$*, where* S *denotes the number of epochs. Then, the output* x_ζ *returned by AbaSVRG satisfies*

$$\mathbb{E}\|\nabla f(x_\zeta)\|^2 \le \frac{\psi(f(x_0) - f^*)}{\phi\eta K} + \frac{\psi\epsilon}{\phi\alpha} + \frac{4\epsilon}{\alpha},$$

where $f^* = \inf_{x\in\mathbb{R}^d} f(x)$ *and* $K = Sm$ *represents the total number of iterations.*

Theorem 1 guarantees the convergence of AbaSVRG as long as ϕ is positive, i.e. $\frac{L\eta}{2} + \frac{\eta^2L^2m^2}{B} \le \frac{1}{2} - \frac{1}{2\alpha}$, and thus allows very flexible choices of the stepsize η, the epoch length m and the mini-batch size B. Such flexibility and generality are also due to the aforementioned simpler proof that we develop for SVRG-type algorithms.

In the following corollary, we provide the complexity performance of AbaSVRG under certain choices of parameters.

Corollary 1. *Under the setting of Theorem 1, we choose the constant stepsize* $\eta = \frac{1}{4L}$*, the epoch length* $m = \sqrt{B}$ *(which* B *denotes the mini-batch size) and* $c_\beta, c_\epsilon \ge 16$*. Then, to achieve* $\mathbb{E}\|\nabla f(x^\zeta)\|^2 \le \epsilon$*, the total SFO complexity of AbaSVRG is given by*

$$\underbrace{\sum_{s=1}^S \min\left\{\frac{c_\beta\sigma^2}{\sum_{t=1}^m\|v_{t-1}^{s-1}\|^2/m}, c_\epsilon\sigma^2\epsilon^{-1}, n\right\} + KB}_{\text{complexity of AbaSVRG}}$$

$$< \underbrace{S\min\{c_\epsilon\sigma^2\epsilon^{-1}, n\} + KB}_{\text{complexity of vanilla SVRG}} = \mathcal{O}\Big(\frac{n\wedge\epsilon^{-1}}{\sqrt{B}\epsilon} + \frac{B}{\epsilon}\Big).$$

If we choose $B = n^{2/3}\wedge\epsilon^{-2/3}$*, then the worst-case complexity is given by* $\mathcal{O}\left(\epsilon^{-1}(n\wedge\epsilon^{-1})^{2/3}\right)$.

We make the following remarks on Corollary 1.

First, the worst-case SFO complexity under the specific choice of $B = n^{2/3}\wedge\epsilon^{-2/3}$ matches the best known result for SVRG-type algorithms. More importantly, since the adaptive component $\frac{c_\beta\sigma^2}{\sum_{t=1}^m\|v_{t-1}^{s-1}\|^2/m}$ can be much smaller

than $\min\{c_\epsilon\sigma^2\epsilon^{-1}, n\}$ during the optimization process particularly in the initial stage, the actual SFO complexity of AbaSVRG can be much lower than that of SVRG with fixed batch size as well as the worst-case complexity of $\mathcal{O}\left(\frac{1}{\epsilon}(n\wedge\frac{1}{\epsilon})^{2/3}\right)$, as demonstrated in our experiments.

Second, our convergence and complexity results hold for any choice of mini-batch size B, and thus we can safely choose a small mini-batch size rather than the large one $n^{2/3}\wedge\epsilon^{-2/3}$ in the regime with large n and ϵ^{-1}. In addition, for a given B, the resulting worst-case complexity $\mathcal{O}\big(\frac{n\wedge\epsilon^{-1}}{\sqrt{B}\epsilon}+\frac{B}{\epsilon}\big)$ still matches the best known order given by ProxSVRG+ (Li & Li, 2018) for SVRG-type algorithms.

Third, Corollary 1 sets the mini-batch size $B=m^2$ to obtain the best complexity order. However, our experiments suggest that $B=m$ has better performance. Hence, we also provide analysis for this case in Appendix E.3.

Note that although Corollary 1 requires $c_\beta, c_\epsilon \geq 16$, the same complexity result can be achieved by more flexible choices for c_β and c_ϵ. More specifically, Theorem 1 only requires c_β and c_ϵ to be greater than α for any positive α. Hence, $\alpha=0.5, \eta=1/(4L), B=m^2, c_\beta>0.5$, and $c_\epsilon>0.5$ are also valid parameters, which can be checked to yield the same complexity order in Corollary 1. Hence, the choices of c_β, c_ϵ in the experiments are consistent with our theory for AbaSVRG.

2.4. Convergence Analysis for AbaSPIDER

In this subsection, we study AbaSPIDER, and compare our results with that for AbaSVRG. Note that AbaSPIDER in Algorithm 2 adopts the improved version SpiderBoost (Wang et al., 2019) of the original SPIDER (Fang et al., 2018).

The following theorem provides a general convergence result for AbaSPIDER.

Theorem 2. *Suppose Assumption 1 holds. Let $\epsilon>0$ and $c_\beta, c_\epsilon \geq \alpha$ for certain constant $\alpha>0$. Let $\psi=\frac{2\eta^2L^2m}{B}+\frac{2}{\alpha}+2$ and choose β_1, α and η such that $\beta_1\leq S\epsilon$ and $\phi=\frac{1}{2}-\frac{1}{2\alpha}-\frac{L\eta}{2}-\frac{\eta^2L^2m}{2B}>0$. Then, the output x_ζ returned by AbaSPIDER satisfies*

$$\mathbb{E}\|\nabla f(x_\zeta)\|^2 \leq \frac{\psi(f(x_0)-f^*)}{\phi\eta K}+\frac{\psi\epsilon}{2\phi\alpha}+\frac{4\epsilon}{\alpha},$$

where $f^*=\inf_{x\in\mathbb{R}^d} f(x)$ and $K=Sm$.

To guarantee the convergence, AbaSPIDER allows a smaller mini-batch size B than AbaSVRG, because AbaSPIDER requires $B\geq\Theta(m\eta^2)$ (see Theorem 2) to guarantee ϕ to be positive, whereas AbaSVRG requires $B\geq\Theta(m^2\eta^2)$ (see Theorem 1). Thus, to achieve the same-level of target accuracy, AbaSPIDER uses fewer mini-batch samples than AbaSVRG, and thus achieves a lower worst-case SFO complexity, as can be seen in the following corollary.

Corollary 2. *Under the setting of Theorem 2, for any mini-batch size $B\leq n^{1/2}\wedge\epsilon^{-1/2}$, if we set the epoch length $m=(n\wedge\frac{1}{\epsilon})B^{-1}$, the stepsize $\eta=\frac{1}{4L}\sqrt{\frac{B}{m}}$ and $c_\beta, c_\epsilon\geq 16$, then to obtain an ϵ-accurate solution x_ζ, the total SFO complexity of AbaSPIDER is given by*

$$\underbrace{\sum_{s=1}^{S}\min\left\{\frac{c_\beta\sigma^2}{\sum_{t=1}^m\|v_{t-1}^{s-1}\|^2/m}, c_\epsilon\sigma^2\epsilon^{-1}, n\right\}+KB}_{\text{complexity of AbaSPIDER}}$$
$$<\underbrace{S\min\left\{c_\epsilon\sigma^2\epsilon^{-1}, n\right\}+KB}_{\text{complexity of vanilla SPIDER}}=\mathcal{O}\big(\epsilon^{-1}(n\wedge\epsilon^{-1})^{1/2}\big).$$

Corollary 2 shows that for a *wide range* of mini-batch size B (as long as $B\leq n^{1/2}\wedge\epsilon^{-1/2}$), AbaSPIDER achieves the near-optimal worst-case complexity $\mathcal{O}\big(\frac{1}{\epsilon}(n\wedge\frac{1}{\epsilon})^{1/2}\big)$ under a proper selection of m and η. Thus, our choice of mini-batch size is much less restrictive than $B=n^{1/2}\wedge\epsilon^{-1/2}$ used by SpiderBoost (Wang et al., 2019) to achieve the optimal complexity, which can be very large in practice. More importantly, our practical complexity can be much better than those of SPIDER (Fang et al., 2018) and SpiderBoost (Wang et al., 2019) with a fixed batch size due to the batch size adaptation.

Similarly to the argument at the end of Section 2.3, the parameters c_β, c_ϵ for AbaSPIDER in Corollary 2 can be chosen more flexibly, e.g., $c_\beta, c_\epsilon>0.5$. Hence, the choices of c_β, c_ϵ in the experiments are consistent with our theory for AbaSPIDER.

3. Batch Size Adaptation for Policy Gradient

In this section, we demonstrate an important application of our proposed batch-size adaptation scheme to variance reduced policy gradient algorithms in RL.

3.1. Problem Formulation

Consider a discrete-time Markov decision process (MDP) $\mathcal{M}=\{\mathcal{S},\mathcal{A},\mathcal{P},\mathcal{R},\gamma,\rho\}$, where $\mathcal{S}$ denotes the state space; $\mathcal{A}$ denotes the action space; $\mathcal{P}$ denotes the Markovian transition model, $\mathcal{P}(s'|s,a)$ denotes the transition probability from state-action pair (s,a) to state s'; $\mathcal{R}\in[-R,R]$ denotes the reward function, $\mathcal{R}(s,a)$ denotes the reward at state-action pair (s,a); $\gamma\in[0,1)$ denotes the discount factor; and ρ denotes the initial state distribution. The agent's decision strategy is captured by the policy $\pi:=\pi(\cdot|s)$, which represents the density function over space $\mathcal{A}$ at state s. Assume that the policy is parameterized by $\theta\in\mathbb{R}^d$. Then, the policy class can be represented as $\Pi=\{\pi_\theta|\theta\in\mathbb{R}^d\}$.

We consider a MDP problem with a finite horizon H. Then, a trajectory τ consists of a sequence of states and actions $(s_0,a_0,\cdots,s_{H-1},a_{H-1})$ observed by following a policy

π_θ and $s_0 \sim \rho$. The total reward of such a trajectory τ is given by $\mathcal{R}(\tau) = \sum_{t=0}^{H} \gamma^t \mathcal{R}(s_t, a_t)$. For a give policy π_θ, the corresponding expected reward is given by $J(\theta) = \mathbb{E}_{\tau\sim p(\cdot|\theta)} \mathcal{R}(\tau)$, where $p(\cdot|\theta)$ represents the probability distribution of the trajectory τ by following the policy π_θ. The goal of the problem is to find a policy that achieves the maximum accumulative reward by solving

$$\max_{\theta\in\mathbb{R}^d} J(\theta), \quad \text{where } J(\theta) = \mathop{\mathbb{E}}_{\tau\sim p(\cdot|\theta)} [\mathcal{R}(\tau)]. \qquad \text{(Q)}$$

3.2. Preliminaries of Policy Gradient and Variance Reduction

Policy gradient is a popular approach to solve the problem (Q), which iteratively updates the value of θ based on the trajectory gradient of the above objective function. Since the distribution $p(\cdot|\theta)$ of τ is unknown because the MDP is unknown, policy gradient adopts the trajectory gradient (denoted by $g(\tau|\theta)$) based on the *sampled* trajectory τ for its update. Two types of commonly used trajectory gradients, namely REINFORCE (Williams, 1992) and G(PO)MDP (Baxter & Bartlett, 2001), are introduced in Appendix A. A key difference of such policy gradient algorithms from SGD in conventional optimization is that the sampling distribution $p(\cdot|\theta)$ changes as the policy parameter θ is iteratively updated, and hence trajectories here are sampled by a varying distributions during the policy gradient iteration.

This paper focuses on the following two variance reduced policy gradient algorithms, which were developed recently to improve the computational efficiency of policy gradient algorithms. First, Papini et al. 2018 proposed a stochastic variance reduced policy gradient (SVRPG) algorithm by adopting the SVRG structure in conventional optimization. In particular, SVRPG continuously adjusts the gradient estimator by *importance sampling* due to the iteratively changing trajectory distribution. Furthermore, Xu et al. 2019b applied the SARAH/SPIDER estimator in conventional optimization to develop a stochastic recursive variance reduced policy gradient (SRVR-PG) algorithm, which we refer to as SPIDER-PG in this paper.

3.3. Proposed Algorithms with Batch-size Adaptation

Both variance reduced policy gradient algorithms SVRPG and SPIDER-PG choose a large batch size N for estimating the policy gradient at the beginning of each epoch. As the result, they often run slowly in practice, and do not show significant advantage over the vanilla policy gradient algorithms. This motivates us to apply our developed batch-size adaptation scheme to reduce their computational complexity.

Thus, we propose AbaSVRPG and AbaSPIDER-PG algorithms, as outlined in Algorithms 3 and 4. More specifically, we adapt the batch size N based on the average of the trajectory gradients in the preceding epoch as

$$N = \frac{\alpha\sigma^2}{\frac{\beta}{m}\sum_{i=n_k-m}^{n_k-1} \|v_i\|^2 + \epsilon}, \qquad (1)$$

where k denotes the iteration number, $n_k = \lfloor k/m \rfloor \times m$, and $\|v_{-1}\| = \cdots = \|v_{-m}\| = 0$.

3.4. Assumptions and Definitions

We take the following standard assumptions, as also adopted by Xu et al. 2019a;b; Papini et al. 2018.

Assumption 2. *The trajectory gradient $g(\tau|\theta)$ is an unbiased gradient estimator, i.e., $\mathbb{E}_{\tau\sim p(\cdot|\theta)}[g(\tau|\theta)] = \nabla J(\theta)$.*

Note that the commonly used trajectory gradients $g(\tau|\theta)$ such as REINFORCE and G(PO)MDP as given in Appendix A satisfy Assumption 2.

Assumption 3. *For any state-action pair (s, a), at any value of θ, and for $1 \le i, j \le d$, there exist constants $0 \le G, H, R < \infty$ such that $|\nabla_{\theta_i} \log \pi_\theta(a|s)| \le G$ and*

$$|\mathcal{R}(s, a)| \le R, \ \left|\frac{\partial^2}{\partial\theta_i\partial\theta_j} \log \pi_\theta(a|s)\right| \le H.$$

Assumption 3 assumes that the reward function $\mathcal{R}$, and the gradient and Hessian of $\log \pi_\theta(a|s)$ are bounded.

Assumption 4. *The estimation variance of the trajectory gradient is bounded, i.e., there exists a constant $\sigma^2 < \infty$ such that, for any $\theta \in \mathbb{R}^d$:*

$$\mathbb{V}\mathrm{ar}[g(\tau|\theta)] = \mathop{\mathbb{E}}_{\tau\sim p(\cdot|\theta)} \|g(\tau|\theta) - \nabla J(\theta)\|^2 \le \sigma^2.$$

Since the problem (Q) in general is nonconvex, we take the following standard convergence criterion.

Definition 3. *We say that $\bar{\theta}$ is an ϵ-accurate stationary point for the problem (Q) if $\mathbb{E}\left\|\nabla J(\bar{\theta})\right\|^2 \le \epsilon$.*

To measure the computational complexity of various policy gradient methods, we take the stochastic trajectory-gradient oracle (STO) complexity as the metric, which meansures the number of trajectory-gradient computations to attain an ϵ-accurate stationary point.

3.5. Convergence Analysis for AbaSVRPG

In this subsection, we provide the convergence and complexity analysis for AbaSVRPG algorithm. Since the batch size of AbaSVRPG is adaptive to the trajectory gradients calculated in the previous epoch, we will adopt our new analysis framework to bound the change of the function value for each epoch by the trajectory gradients in the preceding epoch due to the batch size dependence. The challenge here arises due to the fact that the sampling distribution is time varying as the policy parameter is updated due to the iteration. Hence, the bound should be tightly developed in order

Algorithm 3 AbaSVRPG

Input: $\eta, \theta_0, \epsilon, m, \alpha, \beta > 0$
for $k = 0, 1, \ldots, K$ **do**
  if $\mathrm{mod}(k, m) = 0$ **then**
    Sample $\{\tau_i\}_{i=1}^N$ from $p(\cdot|\theta_k)$, where N is given by (1)
    $v_k = \frac{1}{N}\sum_{i=1}^N g(\tau_i|\theta_k)$
    $\tilde{\theta} = \theta_k$ and $\tilde{v} = v_k$
  else
    Draw $\{\tau_i\}_{i=1}^B$ samples from $p(\cdot|\theta_k)$
    $v_k = \frac{1}{B}\sum_{i=1}^B \left(g(\tau_i|\theta_k) - \omega(\tau_i|\theta_k, \tilde{\theta})g(\tau_i|\tilde{\theta})\right) + \tilde{v}$
  end if
  $\theta_{k+1} = \theta_k + \eta v_k$
end for
Output: θ_ξ from $\{\theta_0, \cdots, \theta_K\}$ uniformly at random.

Algorithm 4 AbaSPIDER-PG

Input: $\eta, \theta_0, \epsilon, m, \alpha, \beta > 0$
for $k = 0, 1, \ldots, K$ **do**
  if $\mathrm{mod}(k, m) = 0$ **then**
    Sample $\{\tau_i\}_{i=1}^N$ from $p(\cdot|\theta_k)$, where N is given by (1)
    $v_k = \frac{1}{N}\sum_{i=1}^N g(\tau_i|\theta_k)$
  else
    Draw $\{\tau_i\}_{i=1}^B$ samples from $p(\cdot|\theta_k)$.
    $v_k = \frac{1}{B}\sum_{i=1}^B \big(g(\tau_i|\theta_k) - \omega(\tau_i|\theta_k, \theta_{k-1})g(\tau_i|\theta_{k-1})\big) + v_{k-1}$
  end if
  $\theta_{k+1} = \theta_k + \eta v_k$
end for
Output: θ_ξ from $\{\theta_0, \cdots, \theta_K\}$ uniformly at random.

to ensure the decrease of the accumulative change of the objective value over the entire execution of the algorithm. Such bounding procedure is very different from the convergence proofs for vanilla SVRPG in Papini et al. 2018; Xu et al. 2019a.

The following theorem characterizes the convergence of AbaSVRPG. Let $\theta^* := \arg\max_{\theta\in\mathbb{R}^d} J(\theta)$.

Theorem 3. *Suppose Assumption 2, 3, and 4 hold. Choose* $\eta = \frac{1}{2L}, m = \left(\frac{L^2\sigma^2}{Q\epsilon}\right)^{\frac{1}{3}}, B = \left(\frac{Q\sigma^4}{L^2\epsilon^2}\right)^{\frac{1}{3}}, \alpha = 48$ *and* $\beta = 6$, *where* $L > 0$ *is a Lipschitz constant given in Lemma 3 in Appendix F, and* Q *is the constant given in Lemma 5 in Appendix F. Then, the output* θ_ξ *of AbaSVRPG satisfies*

$$\mathbb{E}\left\|\nabla J(\theta_\xi)\right\|^2 \leq \frac{88L}{K+1}\left(J(\theta^*) - J(\theta_0)\right) + \frac{\epsilon}{2}, \quad (2)$$

where K *denotes the total number of iterations.*

Theorem 3 shows that the output of AbaSVRPG converges at a rate of $\mathcal{O}(1/K)$. Furthermore, the following corollary captures the overall STO complexity of AbaSVRPG and its comparison with the vanilla SVRPG.

Corollary 3. *Under the setting of Theorem 3, the overall STO complexity of AbaSVRPG to achieve* $\mathbb{E}\left\|\nabla J(\theta_\xi)\right\|^2 \leq \epsilon$ *is*

$$\underbrace{2KB + \sum_{k=0}^{n_K} \frac{\alpha\sigma^2}{\frac{\beta}{m}\sum_{i=km-m}^{km-1}\|v_i\|^2 + \epsilon}}_{\text{compelxity of AbaSVRPG}} < \underbrace{2KB + \sum_{k=0}^{n_K}\frac{\alpha\sigma^2}{\epsilon}}_{\text{compelxity of vanilla SVRPG}} = \mathcal{O}\left(\epsilon^{-5/3} + \epsilon^{-1}\right).$$

The STO complexity improves the state-of-the-art complexity of vanilla SVRPG characterized in Xu et al. 2019a, especially due to the saving samples at the initial stage. The worst-case STO complexity of AbaSVRPG is $\mathcal{O}(\epsilon^{-5/3} + \epsilon^{-1})$, which matches that of Xu et al. 2019a.

3.6. Convergence Analysis for AbaSPIDER-PG

In this section, we provide the convergence and complexity analysis for AbaSPIDER-PG algorithm.

Theorem 4. *Suppose Assumptions 2, 3, and 4 hold. Choose* $\eta = \frac{1}{2L}, m = \frac{L\sigma}{\sqrt{Q}\epsilon}, B = \frac{\sigma\sqrt{Q}}{L\sqrt{\epsilon}}, \alpha = 48$ *and* $\beta = 6$. *where* $L > 0$ *is a Lipschitz constant given in Lemma 3 in Appendix F, and* Q *is the constant given in Lemma 5 in Appendix F. Then, the output* θ_ξ *of AbaSPIDER-PG satisfies*

$$\mathbb{E}\left\|\nabla J(\theta_\xi)\right\|^2 \leq \frac{40L}{K+1}\left(J(\theta^*) - J(\theta_0)\right) + \frac{\epsilon}{2}.$$

Corollary 4. *Under the setting of Theorem 4, the total STO complexity of AbaSPIDER-PG to achieve* $\mathbb{E}\|\nabla J(\theta_\xi)\|^2 \leq \epsilon$ *is*

$$\underbrace{2KB + \sum_{k=0}^{n_K} \frac{\alpha\sigma^2}{\frac{\beta}{m}\sum_{i=km-m}^{km-1}\|v_i\|^2 + \epsilon}}_{\text{compelxity of AbaSPIDER-PG}} < \underbrace{2KB + \sum_{k=0}^{n_K}\frac{\alpha\sigma^2}{\epsilon}}_{\text{compelxity of vanilla SPIDER-PG}} = \mathcal{O}(\epsilon^{-3/2} + \epsilon^{-1}).$$

Corollary 4 shows that the worst-case STO complexity of AbaSPIDER-PG is $\mathcal{O}(\epsilon^{-3/2} + \epsilon^{-1})$, which orderwisely outperforms that of AbaSVRPG in Corollary 3, by a factor of $\mathcal{O}(\epsilon^{-1/6})$. This is due to the fact that AbaSPIDER-PG avoids the variance accumulation problem of AbaSVRPG by continuously using the gradient information from the immediate preceding step (see Appendix F.4 for more details).

4. Experiments

In this section, we compare our proposed batch-size adaptation algorithms with their corresponding vanilla algorithms in both conventional nonconvex optimization and reinforcement learning problems.

4.1. Nonconvex Optimization

We compare our proposed AbaSVRG and AbaSPIDER with the state-of-the-art algorithms including mini-batch SGD (Ghadimi & Lan, 2013), HSGD (Zhou et al., 2018b), AbaSGD[1] (Sievert & Charles, 2019), SVRG+ (Li & Li,

[1] An improved SGD algorithm with batch size adapting to the history gradients.

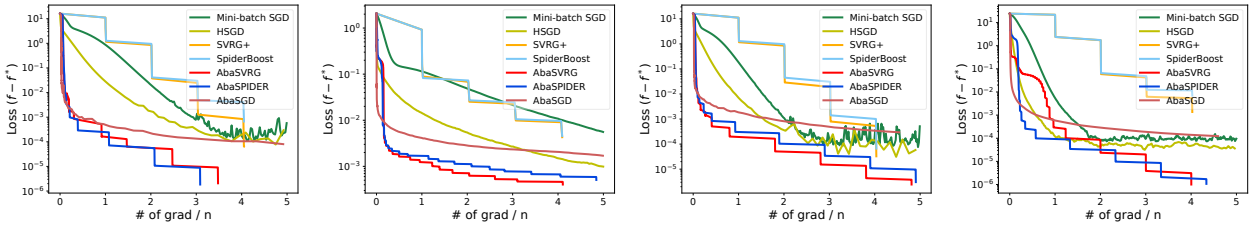
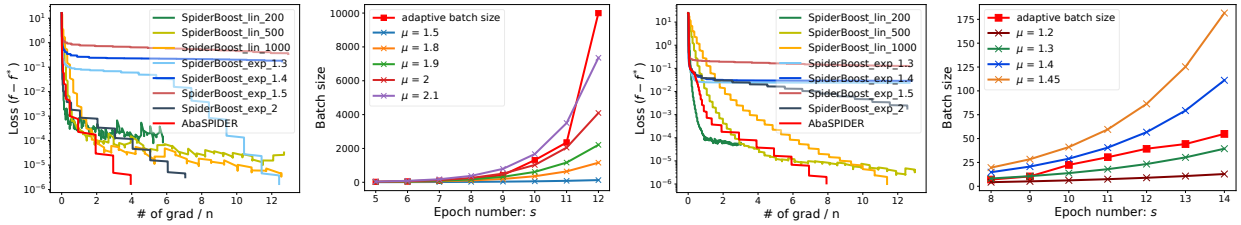


Figure 1. Comparison of various algorithms for logistic regression problem over four datasets. From left to right: a8a, ijcnn1, a9a, w8a.



(a) dataset: a9a (b) dataset: w8a

Figure 2. Comparison of our gradient-based adaptive batch size and exponentially and linearly increased batch sizes. For each dataset, the left figure plots loss v.s. # of gradient evaluations and the right figure plots adaptive batch size and exponentially increasing batch sizes v.s. epoch number s.

2018), and SpiderBoost (Wang et al., 2019) for two nonconvex optimization problems, i.e., nonconvex logistic regression and training multi-layer neural networks. Due to the space limitations, the detailed hyper-parameter settings for all algorithms and the results on training neural networks are relegated to Appendix B.

We consider the following nonconvex logistic regression problem with two classes $\min_{w\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^n \ell(w^T x_i, y_i) + \alpha\sum_{i=1}^d \frac{w_i^2}{1+w_i^2}$, where $x_i \in \mathbb{R}^d$ denote the features, $y_i \in \{\pm 1\}$ are labels, ℓ is the cross-entropy loss, and we set $\alpha = 0.1$. For this problem, we use four datasets from LIBSVM (Chang & Lin, 2011): a8a, w8a, a9a, ijcnn1.

As can be seen from Fig. 1 and Fig. 4 (in Appendix B.2), AbaSVRG and AbaSPIDER converge much faster than all other algorithms in terms of the total number of gradient evaluations (i.e., SFO complexity) on all four datasets. It can be seen that both of them take the advantage of sample-efficient SGD-like updates (due to the small batch size) at the initial stage and attain high accuracy provided by variance-reduced methods at the final stage. This is consistent with the choice of our batch-size adaptation scheme.

We then evaluate the performance of our history-gradient based batch-size adaptation scheme with the other two commonly used prescribed adaptation schemes, i.e., exponential increase of batch size $N_s = \mu^s$ and linear increase of batch size $N_s = \nu(s+1)$. Let SpiderBoost_exp_μ and SpiderBoost_lin_ν denote SpiderBoost algorithms with exponentially and linearly increasing batch sizes under parameters μ and ν, respectively. As shown in Fig. 2, our adaptive batch size scheme achieves the best performance for a9a dataset, and performs better than all other algorithms for w8a dataset except SpiderBoost_lin_200, which, however, does not converge in the high-accuracy regime. Furthermore, the performance of prescribed batch-size adaptation can be problem specific. For example, exponential increase of batch size (with $\mu = 2$ and $\mu = 2.1$) performs better than linear increase of batch size for a9a dataset, but worse for w8a dataset, whereas our scheme adapts to the optimization path, and hence performs the best in both cases.

4.2. Reinforcement Learning

We compare our proposed AbaSVRPG and AbaSPIDER-PG with vanilla SVRPG (Papini et al., 2018) and SPIDER-PG (Xu et al., 2019b) on four benchmark tasks in reinforcement learning, i.e., InvertedPendulum, InvertedDoublePendulum, Swimmer and Hopper. We apply the Gaussian policy which is constructed using a two-layer neural network (NN) with the number of hidden weights being task-dependent. We also include the setup of adaptive standard deviation. The experimental results are averaged over 20 trails with different random seeds, and selections of random seeds are consistent for different algorithms within each task for a fair comparison. Further details about the hyper-parameter setting and task environments are provided in Appendix B.4.

It can be seen from Figure 3 that the proposed AbaSVRPG

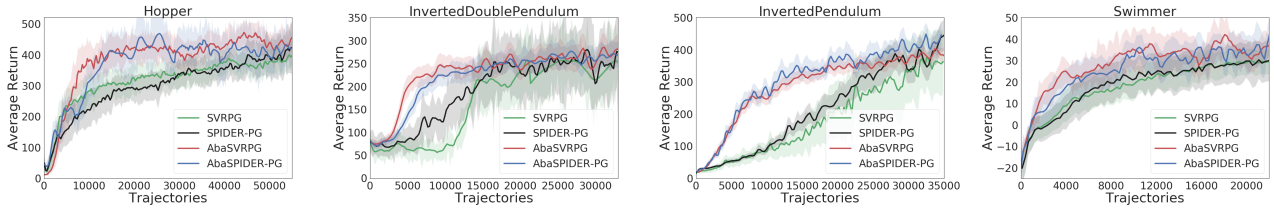


Figure 3. Comparison of various algorithms for reinforcement learning on four tasks.

and AbaSPIDER-PG converge much faster than the vanilla SVRPG and SPIDER-PG (without batch size adaptation) on all four tasks. Such an acceleration is more significant at the initial stage of optimization procedure due to the large trajectory gradient that suggests small batch size.

5. Conclusion

In this paper, we propose a novel scheme for adapting the batch size via history gradients, based on which we develop AbaSVRG and AbaSPIDER for conventional optimization and AbaSVRPG and AbaSPIDER-PG for reinforcement learning. We show by theory and experiments that the proposed algorithms achieve improved computational complexity than their vanilla counterparts (without batch size adaptation). Extensive experiments demonstrate the promising performances of proposed algorithms. We anticipate that such a scheme can be applied to a wide range of other stochastic algorithms to accelerate their theoretical and practical performances.

Acknowledgements

The work was supported in part by the U.S. National Science Foundation under the grants CCF-1761506, ECCS-1818904, CCF-1900145, and CCF-1909291.

References

Allen-Zhu, Z. and Hazan, E. Variance reduction for faster non-convex optimization. In *Proc. International Conference on Machine Learning (ICML)*, pp. 699–707, 2016.

Baxter, J. and Bartlett, P. L. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15(1), November 2001.

Chang, C.-C. and Lin, C.-J. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):27, 2011.

De, S., Yadav, A., Jacobs, D., and Goldstein, T. Big batch SGD: Automated inference using adaptive batch sizes. *arXiv preprint arXiv:1610.05792*, 2016.

De, S., Yadav, A., Jacobs, D., and Goldstein, T. Automated inference with adaptive batches. In *Proc. Artificial Intelligence and Statistics (AISTATS)*, pp. 1504–1513, 2017.

Defazio, A., Bach, F., and Lacoste-Julien, S. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, pp. 1646–1654. 2014.

Devarakonda, A., Naumov, M., and Garland, M. Adabatch: adaptive batch sizes for training deep neural networks. *arXiv preprint arXiv:1712.02029*, 2017.

Duan, Y., Chen, X., Houthooft, R., Schulman, J., and Abbeel, P. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning (ICML)*, pp. 1329–1338, 2016.

Fang, C., Li, C. J., Lin, Z., and Zhang, T. SPIDER: Near-optimal non-convex optimization via stochastic path-integrated differential estimator. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, pp. 689–699, 2018.

Friedlander, M. P. and Schmidt, M. Hybrid deterministic-stochastic methods for data fitting. *SIAM Journal on Scientific Computing*, 34(3):A1380–A1405, 2012.

Ghadimi, S. and Lan, G. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013.

Harikandeh, R., Ahmed, M. O., Virani, A., Schmidt, M., Konečnỳ, J., and Sallinen, S. Stop wasting my gradients: practical SVRG. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, pp. 2251–2259, 2015.

Johnson, R. and Zhang, T. Accelerating stochastic gradient descent using predictive variance reduction. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, pp. 315–323, 2013.

Lei, L. and Jordan, M. I. On the adaptivity of stochastic gradient-based optimization. *arXiv preprint arXiv:1904.04480*, 2019.

Lei, L., Ju, C., Chen, J., and Jordan, M. I. Non-convex finite-sum optimization via SCSG methods. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, pp. 2348–2358, 2017.

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

Nesterov, Y. and Polyak, B. T. Cubic regularization of newton method and its global performance. *Mathematical Programming*, 108(1):177–205, 2006.

Nguyen, L. M., Liu, J., Scheinberg, K., and Takáč, M. SARAH: A novel method for machine learning problems using stochastic recursive gradient. In *Proc. International Conference on Machine Learning (ICML)*, pp. 2613–2621, 2017a.

Nguyen, L. M., Liu, J., Scheinberg, K., and Takáč, M. Stochastic recursive gradient algorithm for nonconvex optimization. *arXiv preprint arXiv:1705.07261*, 2017b.

Nguyen, L. M., van Dijk, M., Phan, D. T., Nguyen, P. H., Weng, T.-W., and Kalagnanam, J. R. Finite-sum smooth optimization with SARAH. *arXiv preprint arXiv:1901.07648*, 2019.

Papini, M., Binaghi, D., Canonaco, G., Pirotta, M., and Restelli, M. Stochastic variance-reduced policy gradient. In *Proc. International Conference on Machine Learning (ICML)*, pp. 4026–4035, 2018.

Polyak, B. T. Gradient methods for the minimisation of functionals. *USSR Computational Mathematics and Mathematical Physics*, 3(4):864–878, 1963.

Reddi, S. J., Hefny, A., Sra, S., Poczos, B., and Smola, A. Stochastic variance reduction for nonconvex optimization. In *Proc. International Conference on Machine Learning (ICML)*, pp. 314–323, 2016a.

Reddi, S. J., Sra, S., Poczos, B., and Smola, A. Proximal stochastic methods for nonsmooth nonconvex finite-sum optimization. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, pp. 1145–1153. 2016b.

Robbins, H. and Monro, S. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3): 400–407, 1951.

Roux, N. L., Schmidt, M., and Bach, F. R. A stochastic gradient method with an exponential convergence rate for finite training sets. In *Proc. Advances in Neural Information Processing Systems (NIPS)*, pp. 2663–2671. 2012.

S. Sutton, R., McAllester, D., Singh, S., and Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. In *Proc. Neural Information Processing Systems (NIPS)*, 2000.

Shen, Z., Ribeiro, A., Hassani, H., Qian, H., and Mi, C. Hessian aided policy gradient. In *Proc. International Conference on Machine Learning (ICML)*, 2019.

Sievert, S. and Charles, Z. Improving the convergence of SGD through adaptive batch sizes. *arXiv preprint arXiv:1910.08222*, 2019.

Smith, S. L., Kindermans, P.-J., Ying, C., and Le, Q. V. Don't decay the learning rate, increase the batch size. *Proc. International Conference on Learning Representations (ICLR)*, 2018.

Wang, Z., Ji, K., Zhou, Y., Liang, Y., and Tarokh, V. SpiderBoost and momentum: Faster variance reduction algorithms. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, pp. 2403–2413, 2019.

Weaver, L. and Tao, N. The optimal reward baseline for gradient-based reinforcement learning. In *Proc. Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 538–545, 2001.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, May 1992.

Wu, C., Rajeswaran, A., Duan, Y., Kumar, V., Bayen, A. M., Kakade, S., Mordatch, I., and Abbeel, P. Variance reduction for policy gradient with action-dependent factorized baselines. In *Proc. International Conference on Learning Representations (ICLR)*, 2018.

Xu, P., Gao, F., and Gu, Q. An improved convergence analysis of stochastic variance-reduced policy gradient. *arXiv preprint arXiv:1905.12615*, 2019a.

Xu, P., Gao, F., and Gu, Q. Sample efficient policy gradient methods with recursive variance reduction. *arXiv preprint arXiv:1909.08610*, 2019b.

Yuan, H., Li, C. J., Tang, Y., and Zhou, Y. Policy optimization via stochastic recursive gradient algorithm. 2018.

Zhong, K., Song, Z., Jain, P., Bartlett, P. L., and Dhillon, I. S. Recovery guarantees for one-hidden-layer neural networks. In *Proc. International Conference on Machine Learning (ICML)*, pp. 4140–4149, 2017.

Zhou, D., Xu, P., and Gu, Q. Stochastic nested variance reduced gradient descent for nonconvex optimization. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, pp. 3921–3932, 2018a.

Zhou, P., Yuan, X., and Feng, J. New insight into hybrid stochastic gradient descent: Beyond with-replacement sampling and convexity. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, pp. 1234–1243, 2018b.

Zhou, Y. and Liang, Y. Characterization of gradient dominance and regularity conditions for neural networks. *arXiv preprint arXiv:1710.06910*, 2017.

Zhou, Y., Zhang, H., and Liang, Y. Geometrical properties and accelerated gradient solvers of non-convex phase retrieval. In *54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 331–335, 2016.