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-
1.1. 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 Adaptive batch-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 PL 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-
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 SCSCG (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 backtracking 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 ∧ and ∨ denote the minimum and the maximum. Let [n] := {1, ..., n}. For a set S, let |S| be its cardinality and define ∇fS(·) := 1/|S| ∑s∈S ∇fs(·).

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

\[
\frac{m}{m} \mathbb{E}(f(\hat{x}^{s+1}) - f(\hat{x}^s)) \leq -\phi \frac{m}{m} \mathbb{E}\|v_t\|^2 + \psi I(\mathcal{N}_s < n) \frac{m}{m}
\]

where \( \phi, \psi > 0 \) are constants, the indicator function \( I(A) \) equals 1 if the event \( A \) is true and 0 otherwise, \( \hat{x}^s \) is the snapshot in epoch \( s \), \( v_t \) is a stochastic estimation of \( \nabla f(x_t) \) within epoch \( s \), and \( \mathcal{N}_s \) is the batch size used at the outer-loop iteration. The above bound naturally suggests that \( \mathcal{N}_s \) 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.

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 $\beta_s$ of stochastic gradients in the preceding epoch $s-1$ as given below

$$N_s = \min\{c_\beta \sigma^2 \beta_s^{-1}, c_\epsilon \sigma^2 \epsilon^{-1}, n\},$$

where $c_\beta, c_\epsilon > 0$ are constants and $\sigma^2$ 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 $x, y \in \mathbb{R}^d$,  

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

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 $x \in \mathbb{R}^d$, $\frac{1}{n} \sum_{i=1}^{n} \|\nabla f_i(x) - \nabla f(x)\|^2 \leq \sigma^2$.

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 $x^\epsilon$ is an $\epsilon$-accurate solution for the optimization problem (P) if $\mathbb{E}\|\nabla f(x^\epsilon)\|^2 \leq \epsilon$, where $x^\epsilon$ 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 $\mathbb{E}_i[\nabla f_i(x)] = \nabla f(x)$.

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 $R_t^\epsilon = \mathbb{E}[f(x_t^\epsilon) + c_\epsilon \|x_t^\epsilon - \bar{x}^{s-1}\|^2]$ and proves that $R_t^\epsilon$ decreases by the accumulated gradient norms $\sum_{t=0}^{s-1} \mathbb{E}\|\nabla f_i(x_t^\epsilon)\|^2$ within an epoch $s$, and Li & Li 2018 directly showed that $\mathbb{E}(f(x_t^\epsilon))$ decreases by $\sum_{t=0}^{s-1} \mathbb{E}\|\nabla f(x_t^\epsilon)\|^2$ using tighter bounds. As a comparison, our analysis shows that $\mathbb{E}(f(x_t^\epsilon))$ decreases by the accumulated stochastic gradient norms $\sum_{t=0}^{s-1} \mathbb{E}\|v_t^\epsilon\|^2$. 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 $c_\beta, c_\epsilon \geq \alpha$ for constant $\alpha > 0$. Let $\psi = \frac{2\eta L^2 m^2}{B} + \frac{2}{\alpha} + 2$ and choose $\beta_1, \alpha$ and $\eta$ such that $\beta_1 \leq \epsilon S$ and $\phi = \frac{1}{2} - \frac{1}{2\alpha} - \frac{L}{2T} - \frac{\psi L^2 m^2}{2B^2} > 0$, where $S$ denotes the number of epochs. Then, the output $x_S$ returned by AbaSVRG satisfies

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

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 $\phi$ is positive, i.e. $\frac{L}{2} + \frac{\psi L^2 m^2}{B} \leq \frac{1}{2} - \frac{1}{2\alpha}$, and thus allows very flexible choices of the stepsize $\eta$, 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 $\eta = \frac{1}{4\epsilon}$, the epoch length $m = \sqrt{B}$ (which $B$ denotes the mini-batch size) and $c_\beta, c_\epsilon \geq 16$. Then, to achieve $\mathbb{E}\|\nabla f(x_S)\|^2 \leq \epsilon$, the total SFO complexity of AbaSVRG is given by

$$\sum_{s=1}^{S} \min\left\{\frac{c_\beta \sigma^2}{\sum_{i=1}^{m} \|v_i^\epsilon\|^2/m}, c_\epsilon \sigma^2 \epsilon^{-1}, n\right\} + KB$$

$$< S \min\left\{c_\epsilon \sigma^2 \epsilon^{-1}, n\right\} + KB \leq O\left(\frac{n \wedge \epsilon^{-1}}{\sqrt{B} \epsilon} + \frac{B}{\epsilon}\right).$$

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

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 $\frac{c_\beta \sigma^2}{\sum_{i=1}^{m} \|v_i^\epsilon\|^2/m}$ can be much smaller...
than \( \min\{c, \sigma^2 \epsilon^{-1}, n\} \) 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 \( O\left(\frac{1}{n} (n \wedge \frac{1}{2})^{3/3}\right) \), 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 \( \epsilon^{-1} \). In addition, for a given \( B \), the resulting worst-case complexity \( O\left(\frac{m\epsilon^{-1}}{\sqrt{B}} + \frac{B}{\epsilon}\right) \) 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 \( c_\beta \) and \( c_\epsilon \). More specifically, Theorem 1 only requires \( c_\beta \) and \( c_\epsilon \) to be greater than \( \alpha \) for any positive \( \alpha \). 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 \( c_\beta, c_\epsilon \) 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 \( \psi = \frac{2n^2L^2m}{\sqrt{B}} + \frac{2}{\alpha} + 2 \) and choose \( \beta_1, \alpha \) and \( \eta \) such that \( \beta_1 \leq \psi \) and \( \phi = \frac{1}{2} - \frac{1}{2\psi} - \frac{L}{2n} - \frac{n^2L^2m}{2\psi} > 0 \). Then, the output \( x_\epsilon \) returned by AbaSPIDER satisfies

\[
\mathbb{E}\|\nabla f(x_\epsilon)\|^2 \leq \frac{\psi f(x_0) - f^*)}{\phi_0 K} + \frac{\psi \epsilon}{2\phi_0 \alpha} + \frac{4\epsilon}{\alpha},
\]

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 \( B \geq \Theta(m \tau^2) \) (see Theorem 2) to guarantee \( \phi \) 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 \( B \leq n^{3/2} \wedge \epsilon^{-1/2} \), if we set the epoch length \( m = (n \wedge \frac{1}{2}) B^{-1} \), the stepsize \( \eta = \frac{1}{\sqrt{B}} \sqrt{m} \) and \( c_\beta, c_\epsilon \geq 16 \), then to obtain an \( \epsilon \)-accurate solution \( x_\epsilon \), the total SFO complexity of AbaSPIDER is given by

\[
\sum_{s=1}^{S} \min\left\{ \frac{c_\beta \sigma^2}{\sum_{t=1}^{m} \|\nabla f_{s,t} - f^*\|^2/\sqrt{m}}, c_\epsilon \sigma^2 \epsilon^{-1}, n \right\} + KB
\]

where \( KB \) is a constant term, and the complexity of vanilla SPIDER

\[
< S \min\left\{ c_\epsilon \sigma^2 \epsilon^{-1}, n \right\} + KB = O\left(\epsilon^{-1} (n \wedge \epsilon^{-1})^{1/2}\right).
\]

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 \( O\left(\frac{1}{n} (n \wedge \frac{1}{2})^{1/2}\right) \) under a proper selection of \( m \) and \( \eta \). 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 \( c_\beta, c_\epsilon \) for AbaSPIDER in Corollary 2 can be chosen more flexibly, e.g., \( \beta_1, \epsilon > 0.5 \).

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) \( M = \{S, A, \mathcal{P}, \mathcal{R}, \gamma, \rho\} \), where \( S \) denotes the state space; \( A \) 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 \( \rho \) 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 \( \tau \) consists of a sequence of states and actions \((s_0, a_0, \ldots, s_{H-1}, a_{H-1})\) observed by following a policy
we adapt the batch size $N$ (Baxter & Bartlett, 2001), are introduced in Appendix A. A $g(\theta)$, which iteratively updates the value of the policy $\pi_\theta$. The goal of the problem is to find a policy that achieves the maximum accumulative reward by solving

$$\max_{\theta \in \mathbb{R}^d} \mathbb{E}_{\tau \sim \pi(\cdot|\theta)} [R(\tau)]. \quad (Q)$$

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 $\theta$ based on the trajectory gradient of the above objective function. Since the distribution $p(\cdot|\theta)$ of $\tau$ is unknown because the MDP is unknown, policy gradient adopts the trajectory gradient (denoted by $g(\tau|\theta)$) based on the sampled trajectory $\tau$ 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 $\theta$ 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 (SVRG) algorithm by adopting the SVRG structure in conventional optimization. In particular, SVRG 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 SVRG 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 AbaSVRG 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

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

(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., $E_{\tau \sim p(\cdot|\theta)}[g(\tau|\theta)] = \nabla J(\theta)$.

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 $\theta$, and for $1 \leq i, j \leq d$, there exist constants $0 \leq G, H, R < \infty$ such that $|\nabla \theta_i \log \pi_\theta(a|s)| \leq G$ and

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

Assumption 3 assumes that the reward function $R$, 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 $\theta \in \mathbb{R}^d$:

$$\text{Var}[g(\tau|\theta)] = \mathbb{E}_{\tau \sim p(\cdot|\theta)}[\|g(\tau|\theta) - \nabla J(\theta)\|^2 \leq \sigma^2].$$

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

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

To measure the computational complexity of various policy gradient methods, we take the stochastic trajectory-gradient oracle (STO) complexity as the metric, which measures the number of trajectory-gradient computations to attain an $\epsilon$-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
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 SVRG in Papini et al. 2018; Xu et al. 2019a.

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

**Theorem 3.** Suppose Assumption 2, 3, and 4 hold. Choose $\eta = \frac{1}{\sqrt{m}}, m = \frac{\sigma \sqrt{m}}{B}, B = \frac{\sqrt{m}}{\alpha \sqrt{\sigma}}, \alpha = 48$ 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 $\theta_{\xi}$ of AbaSVRG satisfies

$$\mathbb{E} \| \nabla J(\theta_{\xi}) \|^2 \leq \frac{88L}{K + 1} (J(\theta^{*}) - J(\theta_{0})) + \frac{\epsilon}{2},$$

where $K$ denotes the total number of iterations.

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

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

$$2KB + \sum_{k=0}^{nK} \frac{\alpha \sigma^2}{m} \left( \frac{\sum_{i=k-m}^{k-1} \| v_i \|^2}{\sigma^2} + \epsilon \right) < 2KB + \sum_{k=0}^{nK} \frac{\alpha \sigma^2}{m} \left( \frac{\sum_{i=k-m}^{k-1} \| v_i \|^2}{\sigma^2} + \epsilon \right),$$

where $K$ denotes the total number of iterations.

The STO complexity improves the state-of-the-art complexity of vanilla SVRG characterized in Xu et al. 2019a, especially due to the saving samples at the initial stage. The worst-case STO complexity of AbaSVRG is $O(\epsilon^{-5/3} + \epsilon^{-1})$, 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 $\eta = \frac{1}{\sqrt{m}}, m = \frac{\sigma \sqrt{m}}{B}, B = \frac{\sqrt{m}}{\alpha \sqrt{\sigma}}, \alpha = 48$ 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 $\theta_{\xi}$ of AbaSPIDER-PG satisfies

$$\mathbb{E} \| \nabla J(\theta_{\xi}) \|^2 \leq \frac{40L}{K + 1} (J(\theta^{*}) - J(\theta_{0})) + \frac{\epsilon}{2}.$$
We consider the following nonconvex logistic regression optimization problem, 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 formulate the following nonconvex logistic regression problem with two classes \( \min_{\mathbf{w} \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \ell(w^T x_i; y_i) + \alpha \sum_{i=1}^{d} \frac{w_i^2}{1 + w_i^2} \), where \( x_i \in \mathbb{R}^d \) denote the features, \( y_i \in \{\pm1\} \) are labels, \( \ell \) is the cross-entropy loss, and we set \( \alpha = 0.1 \). For this problem, we use 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_\( \mu \) and SpiderBoost\_lin_\( \nu \) denote SpiderBoost algorithms with exponentially and linearly increasing batch sizes under parameters \( \mu \) and \( \nu \), 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 \( \nu = 2.1 \)) 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...
and AbaSVRPG 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


