An Improved Convergence Analysis of Stochastic Variance-Reduced Policy Gradient

Pan Xu
Department of Computer Science
University of California, Los Angeles
Los Angeles, CA 90095

Felicia Gao
Department of Computer Science
University of California, Los Angeles
Los Angeles, CA 90095

Quanquan Gu
Department of Computer Science
University of California, Los Angeles
Los Angeles, CA 90095

Abstract

We revisit the stochastic variance-reduced policy gradient (SVRPG) method proposed by Papini et al. (2018) for reinforcement learning. We provide an improved convergence analysis of SVRPG and show that it can find an ϵ-approximate stationary point of the performance function within $O(1/\epsilon^{5/3})$ trajectories. This sample complexity improves upon the best known result $O(1/\epsilon^2)$ by a factor of $O(1/\epsilon^{1/3})$. At the core of our analysis is (i) a tighter upper bound for the variance of importance sampling weights, where we prove that the variance can be controlled by the parameter distance between different policies; and (ii) a fine-grained analysis of the epoch length and batch size parameters such that we can significantly reduce the number of trajectories required in each iteration of SVRPG. We also empirically demonstrate the effectiveness of our theoretical claims of batch sizes on reinforcement learning benchmark tasks.

1 INTRODUCTION

Reinforcement learning (RL) is a sequential decision process that learns the best actions to solve a task by repeated, direct interaction with the environment (Sutton & Barto, 2018). In detail, an RL agent starts at one state and sequentially takes an action according to a certain policy, observes the resulting reward signal, and lastly, evaluates and improves its policy before it transits to the next state. A policy tells the agent which action to take at each state. Therefore, a good policy is critically important in a RL problem. Recently, policy gradient methods (Sutton et al., 2000) have achieved impressive successes in many challenging deep reinforcement learning applications (Kakade, 2002; Schulman et al., 2015), which directly optimizes the performance function $J(\boldsymbol{\theta})$ (We will formally define it later) over a class of policies parameterized by some model parameter $\boldsymbol{\theta}$. In particular, policy gradient methods seek to find the best policy $\pi_{\boldsymbol{\theta}}$ that maximizes the expected return of the agent. They are generally more effective in the high-dimensional action space and enjoy the additional flexibility of stochasticity, compared with deterministic value-function based methods such as Q-learning and SARSA (Sutton et al., 2000).

In many RL applications, the performance function $J(\boldsymbol{\theta})$ is non-concave and the goal is to find a stationary point $\boldsymbol{\theta}^*$ such that $\|\nabla J(\boldsymbol{\theta}^*)\|_2 = 0$ using gradient based algorithms. Due to the specialty of reinforcement learning, the objective function $J(\boldsymbol{\theta})$ is calculated based on cumulative rewards arriving in a sequential way, which makes it impossible to calculate the full gradient directly. Therefore, most algorithms such as REINFORCE (Williams, 1992) and GPOMDP (Baxter & Bartlett, 2001) need to actively sample trajectories to approximate the gradient $\nabla J(\boldsymbol{\theta})$. This resembles the stochastic gradient (SG) based algorithms in stochastic optimization (Robbins & Monro, 1951) which require $O(1/\epsilon^2)$ trajectories to obtain $\mathbb{E}[\|\nabla J(\boldsymbol{\theta})\|_2^2] \leq \epsilon$ Due to the large variances caused by stochastic gradient, the convergence of SG based methods can be rather sample inefficient when the required precision ϵ is very small.

To mitigate the negative effect of large variance on the convergence of SG methods, a large class of stochastic variance-reduced gradient (SVRG) algorithms were proposed for both convex (Johnson & Zhang, 2013; Xiao & Zhang, 2014; Harikandeh et al., 2015; Nguyen et al., 2017) and nonconvex (Allen-Zhu & Hazan, 2016; Reddi et al., 2016; Lei et al., 2017; Li & Li, 2018; Fang et al., 2018; Zhou et al., 2018) objective functions. SVRG has proved to achieve faster convergence in terms of the total number of stochastic gradient evaluations. These variance-reduced algorithms have since been applied to reinforcement learning in policy evaluation (Du et al., 2017), trust-region policy optimization (Xu et al., 2017) and policy gradient (Papini et al., 2018). In particular, Papini et al. (2018) recently proposed a stochastic variance-reduced policy gradient (SVRPG) algorithm that marries SVRG to policy gradient for reinforcement learning. The algorithm saves on sample computation and improves the performance of the vanilla policy gradient methods based on SG. However, from a theoretical perspective, the authors only showed that SVRPG converges to a stationary point within $\mathbb{E}[\|\nabla J(\boldsymbol{\theta})\|_2^2] \leq \epsilon$ with $O(1/\epsilon^2)$ stochastic gradient evaluations (trajectory samples), which in fact only matches the sample complexity of SG based policy gradient methods. This leaves open the important question:

Can SVRPG be provably better than SG based policy gradient methods?

We answer this question affirmatively and fill this gap between theory and practice in this paper. Specifically, we provide a sharp convergence analysis of SVRPG and show that it only requires $O(1/\epsilon^{5/3})$ stochastic gradient evaluations in order to converge to a stationary point $\boldsymbol{\theta}$ of the performance function, i.e., $\mathbb{E}[\|\nabla J(\boldsymbol{\theta})\|_2^2] \leq \epsilon$. This sample complexity of SVRPG is strictly lower than that of SG based policy gradient methods by a factor of $O(1/\epsilon^{1/3})$. By the same argument, our result is also better than the sample complexity provided in Papini et al. (2018) by a factor of $O(1/\epsilon^{1/3})$. The key ideas in our theoretical analysis are twofold: (i) we prove a key lemma that controls the variance of importance weights introduced in SVRPG to deal with the non-stationarity of the sample distribution in reinforcement learning. This helps offset the additional variance introduced by importance sampling; and (ii) we provide a refined proof of the convergence of SVRPG and carefully investigate the trade-off between the convergence rate and computational efficiency of SG methods. This enables us to choose a smaller batch size to reduce the sample complexity while maintaining the convergence rate. In addition, we demonstrate the advantage of SVRPG over GPOMDP and validate our theoretical results on Cartpole and Mountain Car problems.

Notation In this paper, scalars, vectors and matrices are denoted by lower case, lower case bold face, and upper case bold face letters respectively. We use $\|\mathbf{v}\|_2$ and $\|\mathbf{A}\|_2$ to denote the vector 2-norm of a vector $\mathbf{v} \in \mathbb{R}^d$ and the spectral norm of a matrix $\mathbf{A} \in \mathbb{R}^{d\times d}$ respectively. We denote $a_n = O(b_n)$ if $a_n \leq Cb_n$ for some constant $0 < C$. For $\alpha > 0$, the Rényi divergence (Rényi et al., 1961) between distributions P, Q is

$$D_\alpha(P||Q) = \frac{1}{\alpha - 1}\log_2 \int_x P(x)\bigg(\frac{P(x)}{Q(x)}\bigg)^{\alpha-1} \mathrm{d}x,$$

which is non-negative for all $\alpha > 0$. The exponentiated Rényi divergence is defined as $d_\alpha(P||Q) = 2^{D_\alpha(P||Q)}$.

2 ADDITIONAL RELATED WORK

In this section, we review additional relevant work that is not discussed in the introduction.

Deep RL models (Mnih et al., 2015) have been popular in solving complex problems such as robot locomotion, playing grandmaster skill-level Go, and safe autonomous driving (Levine et al., 2015; Silver et al., 2016; Shalev-Shwartz et al., 2016). Policy gradient (Sutton et al., 2000) is one of the most effective algorithms, where the policy is usually approximated by linear functions or nonlinear functions such as neural networks, and can be both stochastic and deterministic (Silver et al., 2014). One major drawback of traditional policy gradient methods such as REINFORCE (Williams, 1992), GPOMDP (Baxter & Bartlett, 2001) and TRPO (Schulman et al., 2015) is the large variance caused in the estimation of the gradient (Sehnke et al., 2010), which leads to a poor convergence performance in practice. One way of reducing the variance in gradient estimation is to introduce various baselines as control variates (Weaver & Tao, 2001; Greensmith et al., 2004; Peters & Schaal, 2008; Gu et al., 2017; Tucker et al., 2018). (Pirotta et al., 2013) proposed to use adaptive step size to offset the effect of variance of the policy. Papini et al. (2017) further studied the adaptive batch size used to approximate the gradient and proposed to jointly optimize the adaptive step size and batch size. It has also been extensively studied to reduce the variance of policy gradient by importance sampling (Liu, 2008; Cortes et al., 2010). Metelli et al. (2018) reduced the variance caused by importance sampling by deriving a surrogate objective with a Renyi penalty.

3 PRELIMINARIES

In this section, we introduce the preliminaries on reinforcement learning and policy gradient.

Markov Decision Process: We will model the reinforcement learning task as a discrete-time Markov Decision Process (MDP): $M = \{\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \rho\}$, where $\mathcal{S}$ is the state space and $\mathcal{A}$ is the action space. $\mathcal{P}(s'|s, a)$ defines the probability that the agent transits to state s' when taking action a in state s. The reward function $\mathcal{R}(s, a) : \mathcal{S}\times\mathcal{A} \mapsto [0, R]$ gives the reward after the agent takes action a at state s for some constant $R > 0$, and $\gamma \in (0, 1)$ is the discount factor. ρ is the initial state distribution. The probability that the agent chooses action a at state s is modeled by its policy $\pi(a|s)$. Following any stationary policy, the agent can observe and collect a trajectory $\tau = \{s_0, a_0, s_1, a_1, \ldots, s_{H-1}, a_{H-1}, s_H\}$ which is a sequence of state-action pairs, where H is the trajectory horizon. Along with the state-action pairs, the agent also observes an cumulative discounted reward

$$\mathcal{R}(\tau)=\textstyle\sum_{h=0}^{H-1}\gamma^h\mathcal{R}(s_h,a_h). \tag{3.1}$$

Policy Gradients: Suppose that the policy π is parameterized by an unknown parameter $\boldsymbol{\theta}\in\mathbb{R}^d$ and denoted by $\pi_{\boldsymbol{\theta}}$. We denote the distribution induced by policy $\pi_{\boldsymbol{\theta}}$ as $p(\tau|\pi_{\boldsymbol{\theta}})$, also referred to as $p(\tau|\boldsymbol{\theta})$ for simplicity. Then

$$p(\tau|\boldsymbol{\theta})=\rho(s_0)\prod_{h=0}^{H-1}\pi_{\boldsymbol{\theta}}(a_h|s_h)P(s_{h+1}|s_h,a_h). \tag{3.2}$$

To measure the performance of a given policy $\pi_{\boldsymbol{\theta}}$, we define the expected total reward under this policy as $J(\boldsymbol{\theta})=\mathbb{E}_{\tau\sim p(\cdot|\boldsymbol{\theta})}[\mathcal{R}(\tau)|M]$. Taking the gradient of $J(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$ gives

$$\begin{aligned}\nabla_{\boldsymbol{\theta}}J(\boldsymbol{\theta})&=\int_\tau\mathcal{R}(\tau)\nabla_{\boldsymbol{\theta}}p(\tau|\boldsymbol{\theta})\mathrm{d}\tau\\&=\int_\tau\mathcal{R}(\tau)\frac{\nabla_{\boldsymbol{\theta}}p(\tau|\boldsymbol{\theta})}{p(\tau|\boldsymbol{\theta})}p(\tau|\boldsymbol{\theta})\mathrm{d}\tau\\&=\mathbb{E}_{\tau\sim p(\cdot|\boldsymbol{\theta})}[\nabla_{\boldsymbol{\theta}}\log p(\tau|\boldsymbol{\theta})\mathcal{R}(\tau)|M]. \end{aligned}\tag{3.3}$$

We can update the policy by running gradient ascent based algorithms on $\boldsymbol{\theta}$. However, it is impossible to calculate the full gradient in reinforcement learning. In particular, policy gradient samples a batch of trajectories $\{\tau_i\}_{i=1}^N$ to approximate the full gradient in (3.3). At the k-th iteration, the policy is then updated by

$$\boldsymbol{\theta}_{k+1}=\boldsymbol{\theta}_k+\eta\widehat{\nabla}_NJ(\boldsymbol{\theta}_k), \tag{3.4}$$

where $\eta>0$ is the step size and the estimated gradient $\widehat{\nabla}_NJ(\boldsymbol{\theta}_k)$ is an approximation of (3.3) based on trajectories $\{\tau_i\}_{i=1}^N$, which is defined as follows

$$\widehat{\nabla}_NJ(\boldsymbol{\theta})=\frac{1}{N}\sum_{i=1}^N\nabla_{\boldsymbol{\theta}}\log p(\tau_i|\boldsymbol{\theta})\mathcal{R}(\tau_i).$$

According to (3.2), we know that $\nabla_{\boldsymbol{\theta}}\log p(\tau_i|\boldsymbol{\theta})$ is independent of the transition matrix P. Therefore, combining this with (3.1) yields

$$\begin{aligned}&\widehat{\nabla}_NJ(\boldsymbol{\theta})\\&=\frac{1}{N}\sum_{i=1}^N\underbrace{\left[\sum_{h=0}^{H-1}\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(a_h^i|s_h^i)\right]\left[\sum_{h=0}^{H-1}\gamma^h\mathcal{R}(s_h^i,a_h^i)\right]}_{g(\tau_i|\boldsymbol{\theta})},\end{aligned}$$

where $\tau_i=\{s_0^i,a_0^i,s_1^i,a_1^i,\ldots,s_{H-1}^i,a_{H-1}^i,s_H^i\}$ for all $i=1,\ldots,N$ are sampled from policy $\pi_{\boldsymbol{\theta}}$, and $g(\tau_i|\boldsymbol{\theta})$ is the unbiased gradient estimator based on sample τ_i. Then we can rewrite the gradient in (3.4) as $\widehat{\nabla}_NJ(\boldsymbol{\theta})=1/N\sum_{i=1}^Ng(\tau_i|\boldsymbol{\theta})$. Based on the above estimator, we can obtain the most well-known gradient estimators for policy gradient such as REINFORCE (Williams, 1992) and GPOMDP (Baxter & Bartlett, 2001). In particular, the REINFORCE estimator introduces an additional term b as the constant baseline:

$$\begin{aligned}&g(\tau_i|\boldsymbol{\theta})\\&=\left[\sum_{h=0}^{H-1}\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(a_h^i|s_h^i)\right]\left[\sum_{h=0}^{H-1}\gamma^h\mathcal{R}(s_h^i,a_h^i)-b\right].\end{aligned}\tag{3.5}$$

GPOMDP is a refined estimator of REINFORCE based on the fact that the current action does not affect previous decisions:

$$\begin{aligned}&g(\tau_i|\boldsymbol{\theta})\\&=\sum_{h=0}^{H-1}\left[\sum_{t=0}^{h}\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(a_t^i|s_t^i)\right]\left(\gamma^hr(s_h^i,a_h^i)-b_h\right).\end{aligned}\tag{3.6}$$

4 ALGORITHM

In each iteration of the gradient ascent update (3.4), policy gradient methods need to sample a batch of trajectories to estimate the expected gradient. This subsampling introduces a high variance and undermines the convergence speed of the algorithm. Inspired by the success of stochastic variance-reduced gradient (SVRG) techniques in stochastic optimization (Johnson & Zhang, 2013; Reddi et al., 2016; Allen-Zhu & Hazan, 2016), Papini et al. (2018) proposed a stochastic variance reduced policy gradient (SVRPG) method, which is displayed in Algorithm 1.

SVRPG consists of multiple epochs. At the beginning of the s-th epoch, it treats the current policy as a reference point denoted by $\widetilde{\boldsymbol{\theta}}^s=\boldsymbol{\theta}_0^{s+1}$. It then computes a gradient estimator $\mu_s=1/N\sum_{i=1}^Ng(\tau_i|\widetilde{\boldsymbol{\theta}}^s)$ based on N trajectories $\{\tau_i\}_{i=1}^N$ sampled from the current policy, where $g(\tau_i|\widetilde{\boldsymbol{\theta}}^s)$ is the REINFORCE or GPOMDP estimator. At the t-th iteration within the s-th epoch, SVRPG samples B trajectories $\{\tau_j\}_{j=1}^B$ based on the current policy $\boldsymbol{\theta}_t^{s+1}$. Then it updates the policy based on the following semi-stochastic gradient

$$\begin{aligned}\mathbf{v}_t^{s+1}&=\frac{1}{B}\textstyle\sum_{j=1}^Bg(\tau_j|\boldsymbol{\theta}_t^{s+1})\\&\quad+\mu_s-\frac{1}{B}\sum_{j=1}^B\omega(\tau_j|\widetilde{\boldsymbol{\theta}}^s,\boldsymbol{\theta}_t^{s+1})g(\tau_j|\widetilde{\boldsymbol{\theta}}^s),\end{aligned}\tag{4.1}$$

where the last two terms serve as a correction to the subsampled gradient estimator which reduces the variance and improves the convergence rate of Algorithm 1. It is worth noting that the semi-stochastic gradient in (4.1) differs from the common one used in SVRG due to the additional term $\omega(\tau|\widetilde{\boldsymbol{\theta}}^s,\boldsymbol{\theta}_t^{s+1})=p(\tau|\widetilde{\boldsymbol{\theta}}^s)/p(\tau|\boldsymbol{\theta}_t^{s+1})$, which is called the importance sampling weight from $p(\tau|\boldsymbol{\theta}_t^{s+1})$ to $p(\tau|\widetilde{\boldsymbol{\theta}}^s)$. This term is important in reinforcement learning due to the non-stationarity of the distribution of τ. Specifically, $\{\tau_i\}_{i=1}^N$ are sampled from $\widetilde{\boldsymbol{\theta}}^s$ while $\{\tau_j\}_{j=1}^B$ are sampled based on $\boldsymbol{\theta}_t^{s+1}$. Nevertheless, we have

$$\mathbb{E}_{\pi_{\boldsymbol{\theta}_t^{s+1}}}\big[\omega(\cdot|\widetilde{\boldsymbol{\theta}}^s, \boldsymbol{\theta}_t^{s+1}) g(\cdot|\widetilde{\boldsymbol{\theta}}^s)\big] = \mathbb{E}_{\pi_{\widetilde{\boldsymbol{\theta}}^s}}\big[g(\cdot|\widetilde{\boldsymbol{\theta}}^s)\big],$$

which ensures the correction term is zero mean and thus $\mathbf{v}_t^{s+1}$ is an unbiased gradient estimator.

Algorithm 1 SVRPG

```
1:  Input: number of epochs $S$, epoch size $m$, step size $\eta$, batch size $N$, mini-batch size $B$, gradient estimator $g$, initial parameter $\boldsymbol{\theta}_m^0 := \widetilde{\boldsymbol{\theta}}^0 := \boldsymbol{\theta}_0$
2:  for $s = 0, \ldots, S-1$ do
3:    $\boldsymbol{\theta}_0^{s+1} = \widetilde{\boldsymbol{\theta}}^s = \boldsymbol{\theta}_m^s$
4:    Sample $N$ trajectories $\{\tau_i\}$ from $p(\cdot|\widetilde{\boldsymbol{\theta}}^s)$
5:    $\mu_s = \widehat{\nabla}_N J(\widetilde{\boldsymbol{\theta}}^s) := \frac{1}{N}\sum_{i=1}^N g(\tau_i|\widetilde{\boldsymbol{\theta}}^s)$
6:    for $t = 0, \ldots, m-1$ do
7:      Sample $B$ trajectories $\{\tau_j\}$ from $p(\cdot|\boldsymbol{\theta}_t^{s+1})$
8:      $\mathbf{v}_t^{s+1} = \mu_s + \frac{1}{B}\sum_{j=1}^B \big(g(\tau_j|\boldsymbol{\theta}_t^{s+1}) - \omega(\tau_j|\widetilde{\boldsymbol{\theta}}^s, \boldsymbol{\theta}_t^{s+1}) g(\tau_j|\widetilde{\boldsymbol{\theta}}^s)\big)$
9:      $\boldsymbol{\theta}_{t+1}^{s+1} = \boldsymbol{\theta}_t^{s+1} + \eta \mathbf{v}_t^{s+1}$
10:   end for
11: end for
12: return $\boldsymbol{\theta}_{\text{out}}$: uniformly picked from $\{\boldsymbol{\theta}_t^s\}$ for $t = 0, \ldots, m; s = 0, \ldots, S$
```

5 THEORY

In this section, we are going to provide a sharp analysis of Algorithm 1. We first lay down the following common assumption on the log-density of the policy function.

Assumption 5.1. Let $\pi_{\boldsymbol{\theta}}(a|s)$ be the policy of an agent at state s. There exist constants $G, M > 0$ such that the log-density of the policy function satisfies

$$\|\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s)\| \leq G, \quad \big\|\nabla_{\boldsymbol{\theta}}^2 \log \pi_{\boldsymbol{\theta}}(a|s)\big\|_2 \leq M,$$

for all $a \in \mathcal{A}$ and $s \in \mathcal{S}$.

In many real-world problems, we require that policy parameterization to change smoothly over time instead of drastically. Assumption 5.1 is an important condition in nonconvex optimization (Reddi et al., 2016; Allen-Zhu & Hazan, 2016), which guarantees the smoothness of the objective function $J(\boldsymbol{\theta})$. Our assumption is slightly different from that in Papini et al. (2018), which assumes that $\frac{\partial}{\partial \theta_i} \log \pi_{\boldsymbol{\theta}}(a|s)$ and $\frac{\partial^2}{\partial \theta_i \partial \theta_j} \log \pi_{\boldsymbol{\theta}}(a|s)$ are upper bounded elementwisely. It can be easily verified that our Assumption 5.1 is milder than theirs. It should also be noted that although in reinforcement learning we make the assumptions on the parameterized policy, there is no difference in imposing the smoothness assumption on the performance function $J(\boldsymbol{\theta})$ directly. In fact, Assumption 5.1 implies the following proposition on $J(\boldsymbol{\theta})$.

Proposition 5.2. Under Assumption 5.1, $J(\boldsymbol{\theta})$ is L-smooth with $L = HR(M + HG^2)/(1-\gamma)$. In addition, let $g(\tau|\boldsymbol{\theta})$ be the REINFORCE or GPOMDP gradient estimators. Then for all $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \in \mathbb{R}^d$, it holds that

$$\|g(\tau|\boldsymbol{\theta}_1) - g(\tau|\boldsymbol{\theta}_2)\|_2 \leq L_g \|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|_2$$

and $\|g(\tau|\boldsymbol{\theta})\|_2 \leq C_g$ for all $\boldsymbol{\theta} \in \mathbb{R}^d$, where $L_g = HM(R + |b|)/(1-\gamma), C_g = HG(R + |b|)/(1-\gamma)$ and b is the baseline reward.

The next assumption requires that the variance of the gradient estimator is bounded.

Assumption 5.3. There exists a constant σ such that

$$\text{Var}\big(g(\tau|\boldsymbol{\theta})\big) \leq \sigma^2, \quad \text{for all policy } \pi_{\boldsymbol{\theta}}.$$

The above assumption is widely made in stochastic optimization. It can be easily verified for Gaussian policies with REINFORCE estimator (Zhao et al., 2011; Pirotta et al., 2013; Papini et al., 2018).

The following assumption is needed due to the non-stationarity of the sample distribution, which is also made in Papini et al. (2018).

Assumption 5.4. There is a constant $W < \infty$ such that for each policy pairs encountered in Algorithm 1, it holds

$$\text{Var}(\omega(\tau|\boldsymbol{\theta}_1, \boldsymbol{\theta}_2)) \leq W, \quad \forall \boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \in \mathbb{R}^d, \tau \sim p(\cdot|\boldsymbol{\theta}_2).$$

We now present our convergence result for SVRPG.

Theorem 5.5. Under Assumptions 5.1, 5.3 and 5.4. In Algorithm 1, suppose the step size $\eta \leq 1/(4L)$ and epoch length m and mini-batch size B satisfy

$$\frac{B}{m^2} \geq \frac{3(C_\omega C_g^2 + L_g^2)}{2L^2},$$

where $C_\omega = H(2HG^2 + M)(W+1)$, and L_g, C_g and L are defined in Proposition 5.2. Then the output of Algorithm 1 satisfies

$$\mathbb{E}\big[\big\|\nabla J(\boldsymbol{\theta}_{\text{out}})\big\|_2^2\big] \leq \frac{8(J(\boldsymbol{\theta}^*) - J(\boldsymbol{\theta}_0))}{\eta S m} + \frac{6\sigma^2}{N},$$

where $\boldsymbol{\theta}^*$ is the maximizer of $J(\boldsymbol{\theta})$.

Remark 5.6. Let $T = Sm$ be the total number of iterations Algorithm 1 needs to achieve $\mathbb{E}\big[\big\|\nabla J(\boldsymbol{\theta}_{\text{out}})\big\|_2^2\big] \leq \epsilon$. The first term on the right hand side in Theorem 5.5 gives an $O(1/T)$ convergence rate which matches that of Papini et al. (2018) and the results in nonconvex optimization (Allen-Zhu & Hazan, 2016; Reddi et al., 2016).

Table 1: Comparison on sample complexity required to achieve $\|\nabla J(\boldsymbol{\theta})\|_2^2 \le \epsilon$.

METHODS	COMPLEXITY
SG	$O(1/\epsilon^2)$
SVRPG (Papini et al., 2018)	$O(1/\epsilon^2)$
SVRPG (This paper)	$O(1/\epsilon^{5/3})$

The second term $O(1/N)$ comes from the full gradient approximation at the beginning of each epoch in Algorithm 1. Compared with the result in Papini et al. (2018), Theorem 5.5 does not have the additional term $O(1/B)$, which is offset by our elaborate and careful analysis of the variance of importance weights. This also enables us to choose a much smaller batch size B in the inner loops of Algorithm 1 and leads to a lower sample complexity.

Based on Theorem 5.5, we can calculate the total trajectory samples Algorithm 1 requires to achieve ϵ-precision.

Corollary 5.7. Under the same conditions as in Theorem 5.5, let $\epsilon > 0$, if we set $\eta = 1/(4L)$, $N = O(1/\epsilon)$, $B = O(1/\epsilon^{2/3})$ and $m = \sqrt{B}$, then Algorithm 1 needs $O(1/\epsilon^{5/3})$ trajectories in order to achieve $\mathbb{E}[\|\nabla J(\boldsymbol{\theta}_{\text{out}})\|_2^2] \le \epsilon$.

Remark 5.8. In Theorem 4.4 of Papini et al. (2018), the authors showed that the sample complexity of SVRPG is $O((B + N/m)/\epsilon)$. In order to make the gradient small enough, they essentially require that $B, N = O(1/\epsilon)$, which leads to $O(1/\epsilon^2)$ sample complexity. In sharp contrast, our Corollary 5.7 shows that the SVRPG algorithm only needs $O(1/\epsilon^{5/3})$ number of trajectories to achieve $\|\nabla J(\boldsymbol{\theta})\|_2^2 \le \epsilon$, which is obviously lower than the sample complexity proved in Papini et al. (2018). We present a straightforward comparison in Table 1 to show the sample complexities of different methods. SG represents vanilla stochastic gradient based methods such as REINFORCE and GPOMDP. It can be seen from Table 1 that our analysis yields the lowest complexity.

6 PROOF OF THEORETICAL RESULTS

In this section, we prove our main theoretical results.

6.1 PROOF OF THE MAIN THEORY

Before we provide the proof of Theorem 5.5, we first lay down the following key lemma that controls the variance of the importance sampling weights $\omega(\tau|\widetilde{\boldsymbol{\theta}}^s, \boldsymbol{\theta}_t^{s+1})$.

Lemma 6.1. Let $\omega(\tau|\widetilde{\boldsymbol{\theta}}^s, \boldsymbol{\theta}_t^{s+1}) = p(\tau|\widetilde{\boldsymbol{\theta}}^s)/p(\tau|\boldsymbol{\theta}_t^{s+1})$. Under Assumptions 5.1 and 5.4, it holds that

$$\mathrm{Var}\big(\omega\big(\tau|\widetilde{\boldsymbol{\theta}}^s, \boldsymbol{\theta}_t^{s+1}\big)\big) \le C_\omega \|\widetilde{\boldsymbol{\theta}}^s - \boldsymbol{\theta}_t^{s+1}\|_2^2,$$

where $C_\omega = H(2HG^2 + M)(W + 1)$.

Lemma 6.1 shows that the variance of the importance weight is proportional to the distance between the behavioral and the target policies. Note that this upper bound could be trivial based on Assumption 5.4 when the distance is large. However, Lemma 6.1 also provides a fine-grained control of the variance when the behavioral and target polices are sufficiently close.

Now we are ready to present the proof of our main theorem, which is also inspired from that in Li & Li (2018).

Proof of Theorem 5.5. By Proposition 5.2, $J(\boldsymbol{\theta})$ is L-smooth, which leads to

$$\begin{aligned}
J\big(\boldsymbol{\theta}_{t+1}^{s+1}\big) &\ge J\big(\boldsymbol{\theta}_t^{s+1}\big) + \big\langle \nabla J\big(\boldsymbol{\theta}_t^{s+1}\big), \boldsymbol{\theta}_{t+1}^{s+1} - \boldsymbol{\theta}_t^{s+1}\big\rangle \\
&\quad - L/2\big\|\boldsymbol{\theta}_{t+1}^{s+1} - \boldsymbol{\theta}_t^{s+1}\big\|_2^2 \\
&= J\big(\boldsymbol{\theta}_t^{s+1}\big) + \big\langle \nabla J\big(\boldsymbol{\theta}_t^{s+1}\big) - \mathbf{v}_t^{s+1}, \eta\mathbf{v}_t^{s+1}\big\rangle \\
&\quad + \eta\big\|\mathbf{v}_t^{s+1}\big\|_2^2 - L/2\big\|\boldsymbol{\theta}_{t+1}^{s+1} - \boldsymbol{\theta}_t^{s+1}\big\|_2^2 \\
&\ge J\big(\boldsymbol{\theta}_t^{s+1}\big) - \eta/2\big\|\nabla J\big(\boldsymbol{\theta}_t^{s+1}\big) - \mathbf{v}_t^{s+1}\big\|_2^2 \\
&\quad + \eta/2\big\|\mathbf{v}_t^{s+1}\big\|_2^2 - L/2\big\|\boldsymbol{\theta}_{t+1}^{s+1} - \boldsymbol{\theta}_t^{s+1}\big\|_2^2 \\
&\ge J\big(\boldsymbol{\theta}_t^{s+1}\big) - 3\eta/4\big\|\nabla J\big(\boldsymbol{\theta}_t^{s+1}\big) - \mathbf{v}_t^{s+1}\big\|_2^2 \\
&\quad + (1/(4\eta) - L/2)\big\|\boldsymbol{\theta}_{t+1}^{s+1} - \boldsymbol{\theta}_t^{s+1}\big\|_2^2 \\
&\quad + \eta/8\big\|\nabla J\big(\boldsymbol{\theta}_t^{s+1}\big)\big\|_2^2, \qquad (6.1)
\end{aligned}$$

where the second inequality holds due to Young's inequality and the last inequality comes from the fact that $\|\nabla J(\boldsymbol{\theta}_t^{s+1})\|_2^2 \le 2\|\mathbf{v}_t^{s+1}\|_2^2 + 2\|\nabla J(\boldsymbol{\theta}_t^{s+1}) - \mathbf{v}_t^{s+1}\|_2^2$. Let $\mathbb{E}_{N,B}$ denote the expectation only over the randomness of the sampling trajectories $\{\tau_i\}_{i=1}^N$ and $\{\tau_j\}_{j=1}^B$

$$\begin{aligned}
&\mathbb{E}_{N,B}\big\|\nabla J\big(\boldsymbol{\theta}_t^{s+1}\big) - \mathbf{v}_t^{s+1}\big\|_2^2 \\
&= \mathbb{E}_{N,B}\bigg\|\nabla J\big(\boldsymbol{\theta}_t^{s+1}\big) - \mu_s \\
&\quad + \frac{1}{B}\sum_{j=1}^B \big(\omega(\tau_j|\widetilde{\boldsymbol{\theta}}^s, \boldsymbol{\theta}_t^{s+1}) g\big(\tau_j|\widetilde{\boldsymbol{\theta}}^s\big) - g\big(\tau_j|\boldsymbol{\theta}_t^{s+1}\big)\big)\bigg\|_2^2 \\
&= \mathbb{E}_{N,B}\bigg\|\nabla J\big(\boldsymbol{\theta}_t^{s+1}\big) - \nabla J(\widetilde{\boldsymbol{\theta}}^s) + \nabla J(\widetilde{\boldsymbol{\theta}}^s) - \mu_s \\
&\quad + \frac{1}{B}\sum_{j=1}^B \big(\omega(\tau_j|\widetilde{\boldsymbol{\theta}}^s, \boldsymbol{\theta}_t^{s+1}) g\big(\tau_j|\widetilde{\boldsymbol{\theta}}^s\big) - g\big(\tau_j|\boldsymbol{\theta}_t^{s+1}\big)\big)\bigg\|_2^2 \\
&= \mathbb{E}_{N,B}\bigg\|\nabla J\big(\boldsymbol{\theta}_t^{s+1}\big) - \nabla J(\widetilde{\boldsymbol{\theta}}^s) \\
&\quad + \frac{1}{B}\sum_{j=1}^B \big(\omega(\tau_j|\widetilde{\boldsymbol{\theta}}^s, \boldsymbol{\theta}_t^{s+1}) g\big(\tau_j|\widetilde{\boldsymbol{\theta}}^s\big) - g\big(\tau_j|\boldsymbol{\theta}_t^{s+1}\big)\big)\bigg\|_2^2 \\
&\quad + \mathbb{E}_{N,B}\bigg\|\nabla J(\widetilde{\boldsymbol{\theta}}^s) - \frac{1}{N}\sum_{i=1}^N g\big(\tau_i|\widetilde{\boldsymbol{\theta}}^s\big)\bigg\|_2^2 \qquad (6.2)
\end{aligned}$$

$$= \frac{1}{B^2}\sum_{j=1}^{B}\mathbb{E}_{N,B}\big\|\nabla J\big(\boldsymbol{\theta}_t^{s+1}\big) - \nabla J(\widetilde{\boldsymbol{\theta}}^s) + \omega(\tau_j|\widetilde{\boldsymbol{\theta}}^s, \boldsymbol{\theta}_t^{s+1})g\big(\tau_j|\widetilde{\boldsymbol{\theta}}^s\big) - g\big(\tau_j|\boldsymbol{\theta}_t^{s+1}\big)\big\|_2^2 + \frac{1}{N^2}\sum_{i=1}^{N}\mathbb{E}_{N,B}\big\|\nabla J(\widetilde{\boldsymbol{\theta}}^s) - g\big(\tau_i|\widetilde{\boldsymbol{\theta}}^s\big)\big\|_2^2 \qquad (6.3)$$

$$\le \frac{1}{B^2}\sum_{j=1}^{B}\mathbb{E}_{N,B}\big\|\omega(\tau_j|\widetilde{\boldsymbol{\theta}}^s, \boldsymbol{\theta}_t^{s+1})g\big(\tau_j|\widetilde{\boldsymbol{\theta}}^s\big) - g\big(\tau_j|\boldsymbol{\theta}_t^{s+1}\big)\big\|_2^2 + \sigma^2/N, \qquad (6.4)$$

where (6.2) holds due to the independence between trajectories $\{\tau_i\}_{i=1}^N$ and $\{\tau_j\}_{j=1}^B$, (6.3) is due to $\mathbb{E}\|\mathbf{x}_1 + \ldots + \mathbf{x}_n\|_2^2 = \mathbb{E}\|\mathbf{x}_1\|_2^2 + \ldots + \mathbb{E}\|\mathbf{x}_n\|_2^2$ for independent and zero mean variables $\mathbf{x}_1, \ldots, \mathbf{x}_n$, and (6.4) follows Assumption 5.3 and the fact that $\mathbb{E}\|\mathbf{x} - \mathbb{E}\mathbf{x}\|_2^2 \le \mathbb{E}\|\mathbf{x}\|_2^2$. Note that we have

$$\begin{aligned}
&\mathbb{E}_{N,B}\big\|\omega(\tau_j|\widetilde{\boldsymbol{\theta}}^s, \boldsymbol{\theta}_t^{s+1})g\big(\tau_j|\widetilde{\boldsymbol{\theta}}^s\big) - g\big(\tau_j|\boldsymbol{\theta}_t^{s+1}\big)\big\|_2^2 \\
&\le \mathbb{E}_{N,B}\big\|\big(\omega(\tau_j|\widetilde{\boldsymbol{\theta}}^s, \boldsymbol{\theta}_t^{s+1}) - 1\big)g\big(\tau_j|\widetilde{\boldsymbol{\theta}}^s\big)\big\|_2^2 + \mathbb{E}_{N,B}\big\|g\big(\tau_j|\widetilde{\boldsymbol{\theta}}^s\big) - g\big(\tau_j|\boldsymbol{\theta}_t^{s+1}\big)\big\|_2^2 \\
&\le C_g^2\mathbb{E}_{N,B}\big\|\omega(\tau_j|\widetilde{\boldsymbol{\theta}}^s, \boldsymbol{\theta}_t^{s+1}) - 1\big\|_2^2 + L_g^2\big\|\widetilde{\boldsymbol{\theta}}^s - \boldsymbol{\theta}_t^{s+1}\big\|_2^2, \qquad (6.5)
\end{aligned}$$

where the second inequality comes from Proposition 5.2. By Lemma 6.1, we have

$$\begin{aligned}
&\mathbb{E}_{N,B}\big\|\omega(\tau_j|\widetilde{\boldsymbol{\theta}}^s, \boldsymbol{\theta}_t^{s+1}) - 1\big\|_2^2 \\
&= \mathrm{Var}_{\widetilde{\boldsymbol{\theta}}^s, \boldsymbol{\theta}_t^{s+1}}\big(\omega(\tau_j|\widetilde{\boldsymbol{\theta}}^s, \boldsymbol{\theta}_t^{s+1})\big) \\
&\le C_\omega\big\|\boldsymbol{\theta}_t^{s+1} - \widetilde{\boldsymbol{\theta}}^s\big\|_2^2. \qquad (6.6)
\end{aligned}$$

where $C_\omega = (2G^2 + M)(W + 1)$. Substituting the results in (6.4), (6.5) and (6.6) into (6.1) yields

$$\begin{aligned}
&\mathbb{E}_{N,B}\big[J\big(\boldsymbol{\theta}_{t+1}^{s+1}\big)\big] \\
&\ge \mathbb{E}_{N,B}\big[J\big(\boldsymbol{\theta}_t^{s+1}\big)\big] + \frac{\eta}{8}\mathbb{E}_{N,B}\big[\big\|\nabla J\big(\boldsymbol{\theta}_t^{s+1}\big)\big\|_2^2\big] + \bigg[\frac{1}{4\eta} - \frac{L}{2}\bigg]\mathbb{E}_{N,B}\big[\big\|\boldsymbol{\theta}_{t+1}^{s+1} - \boldsymbol{\theta}_t^{s+1}\big\|_2^2\big] - \frac{3\eta\sigma^2}{4N} \\
&\quad - \frac{3\eta(C_\omega C_g^2 + L^2)}{4B}\mathbb{E}_{N,B}\big[\big\|\boldsymbol{\theta}_t^{s+1} - \widetilde{\boldsymbol{\theta}}^s\big\|_2^2\big]. \qquad (6.7)
\end{aligned}$$

For the ease of notation, we denote

$$\Psi = 3(C_\omega C_g^2 + L_g^2)/4B. \qquad (6.8)$$

By Young's inequality (Peter-Paul inequality), we have

$$\big\|\boldsymbol{\theta}_{t+1}^{s+1} - \widetilde{\boldsymbol{\theta}}^s\big\|_2^2 \le (1 + \alpha)\big\|\boldsymbol{\theta}_{t+1}^{s+1} - \boldsymbol{\theta}_t^{s+1}\big\|_2^2 + (1 + 1/\alpha)\big\|\boldsymbol{\theta}_t^{s+1} - \widetilde{\boldsymbol{\theta}}^s\big\|_2^2$$

holds for any $\alpha > 0$. For $\eta \le 1/(2L)$, combining the above inequality with (6.7) and (6.8) yields

$$\begin{aligned}
&\mathbb{E}_{N,B}\big[J\big(\boldsymbol{\theta}_{t+1}^{s+1}\big)\big] \\
&\ge \mathbb{E}_{N,B}\big[J\big(\boldsymbol{\theta}_t^{s+1}\big)\big] + \frac{\eta}{8}\mathbb{E}_{N,B}\big[\big\|\nabla J\big(\boldsymbol{\theta}_t^{s+1}\big)\big\|_2^2\big] - \frac{3\eta\sigma^2}{4N} \\
&\quad + \frac{1}{1+\alpha}\bigg[\frac{1}{4\eta} - \frac{L}{2}\bigg]\mathbb{E}_{N,B}\big[\big\|\boldsymbol{\theta}_{t+1}^{s+1} - \widetilde{\boldsymbol{\theta}}^s\big\|_2^2\big] \\
&\quad - \bigg[\eta\Psi + \frac{1}{\alpha}\bigg[\frac{1}{4\eta} - \frac{L}{2}\bigg]\bigg]\mathbb{E}_{N,B}\big[\big\|\boldsymbol{\theta}_t^{s+1} - \widetilde{\boldsymbol{\theta}}^s\big\|_2^2\big]
\end{aligned}$$

Now we set $\alpha = 2t + 1$ and sum up the above inequality over $t = 0, \ldots, m - 1$. Note that $\boldsymbol{\theta}_0^{s+1} = \widetilde{\boldsymbol{\theta}}^s$, $\boldsymbol{\theta}_m^{s+1} = \widetilde{\boldsymbol{\theta}}^{s+1}$. We are able to obtain

$$\begin{aligned}
&\mathbb{E}_N\big[J\big(\widetilde{\boldsymbol{\theta}}^{s+1}\big)\big] \\
&\ge \mathbb{E}_N\big[J\big(\widetilde{\boldsymbol{\theta}}^s\big)\big] + \frac{\eta}{8}\sum_{t=0}^{m-1}\mathbb{E}_N\big[\big\|\nabla J\big(\boldsymbol{\theta}_t^{s+1}\big)\big\|_2^2\big] - \frac{3m\eta\sigma^2}{4N} \\
&\quad + \sum_{t=0}^{m-1}\frac{1/(2\eta) - L}{4(t+1)}\mathbb{E}_N\big[\big\|\boldsymbol{\theta}_{t+1}^{s+1} - \widetilde{\boldsymbol{\theta}}^s\big\|_2^2\big] \\
&\quad - \sum_{t=0}^{m-1}\bigg[\eta\Psi + \frac{1/(2\eta) - L}{2(2t+1)}\bigg]\mathbb{E}_N\big[\big\|\boldsymbol{\theta}_t^{s+1} - \widetilde{\boldsymbol{\theta}}^s\big\|_2^2\big] \\
&= \mathbb{E}_N\big[J\big(\widetilde{\boldsymbol{\theta}}^s\big)\big] + \frac{\eta}{8}\sum_{t=0}^{m-1}\mathbb{E}_N\big[\big\|\nabla J\big(\boldsymbol{\theta}_t^{s+1}\big)\big\|_2^2\big] - \frac{3m\eta\sigma^2}{4N} \\
&\quad + \sum_{t=0}^{m-2}\frac{1/(2\eta) - L}{4(t+1)}\mathbb{E}_N\big[\big\|\boldsymbol{\theta}_{t+1}^{s+1} - \widetilde{\boldsymbol{\theta}}^s\big\|_2^2\big] \\
&\quad - \sum_{t=1}^{m-1}\bigg[\eta\Psi + \frac{1/(2\eta) - L}{2(2t+1)}\bigg]\mathbb{E}_N\big[\big\|\boldsymbol{\theta}_t^{s+1} - \widetilde{\boldsymbol{\theta}}^s\big\|_2^2\big] \\
&\quad + \frac{1/(2\eta) - L}{4m}\mathbb{E}_N\big[\big\|\boldsymbol{\theta}_m^{s+1} - \widetilde{\boldsymbol{\theta}}^s\big\|_2^2\big] \\
&\quad - \bigg[\eta\Psi + \frac{1}{4\eta} - \frac{L}{2}\bigg]\mathbb{E}_N\big[\big\|\boldsymbol{\theta}_0^{s+1} - \widetilde{\boldsymbol{\theta}}^s\big\|_2^2\big] \\
&= \mathbb{E}_N\big[J\big(\widetilde{\boldsymbol{\theta}}^s\big)\big] + \frac{\eta}{8}\sum_{t=0}^{m-1}\mathbb{E}_N\big[\big\|\nabla J\big(\boldsymbol{\theta}_t^{s+1}\big)\big\|_2^2\big] - \frac{3m\eta\sigma^2}{4N} \\
&\quad + \sum_{t=1}^{m-1}\bigg[\frac{1/(4\eta) - L/2}{2t(2t+1)} - \eta\Psi\bigg]\mathbb{E}_N\big[\big\|\boldsymbol{\theta}_t^{s+1} - \widetilde{\boldsymbol{\theta}}^s\big\|_2^2\big] \\
&\quad + \frac{1/(2\eta) - L}{4m}\mathbb{E}_N\big[\big\|\boldsymbol{\theta}_m^{s+1} - \widetilde{\boldsymbol{\theta}}^s\big\|_2^2\big]. \qquad (6.9)
\end{aligned}$$

Recall the definition of Ψ in (6.8). If we set step size η and the epoch length B to satisfy

$$\eta \le \frac{1}{4L}, \quad \frac{B}{m^2} \ge \frac{3(C_\omega C_g^2 + L_g^2)}{2L^2}, \qquad (6.10)$$

then (6.9) leads to

$$\mathbb{E}_N\big[J\big(\widetilde{\boldsymbol{\theta}}^{s+1}\big)\big] \ge \mathbb{E}_N\big[J\big(\widetilde{\boldsymbol{\theta}}^s\big)\big] - \frac{3m\eta\sigma^2}{4N}$$

$$+\frac{\eta}{8}\sum_{t=0}^{m-1}\mathbb{E}_N\big[\big\|\nabla J\big(\boldsymbol{\theta}_t^{s+1}\big)\big\|_2^2\big].$$

Telescoping the above inequality yields

$$\frac{\eta}{8}\sum_{s=0}^{S-1}\sum_{t=0}^{m-1}\mathbb{E}\big[\big\|\nabla J\big(\boldsymbol{\theta}_t^{s+1}\big)\big\|_2^2\big]$$
$$\le \mathbb{E}\big[J\big(\widetilde{\boldsymbol{\theta}}^S\big)\big]-\mathbb{E}\big[J\big(\widetilde{\boldsymbol{\theta}}^0\big)\big]+\frac{3Sm\eta\sigma^2}{4N},$$

which immediately implies

$$\mathbb{E}\big[\big\|\nabla J\big(\boldsymbol{\theta}_{\text{out}}\big)\big\|_2^2\big]\le\frac{8\big(\mathbb{E}\big[J\big(\widetilde{\boldsymbol{\theta}}^S\big)\big]-\mathbb{E}\big[J\big(\widetilde{\boldsymbol{\theta}}^0\big)\big]\big)}{\eta Sm}+\frac{6\sigma^2}{N}$$
$$\le\frac{8(J(\boldsymbol{\theta}^*)-J(\boldsymbol{\theta}_0))}{\eta Sm}+\frac{6\sigma^2}{N}.$$

This completes the proof. □

Proof of Corollary 5.7. By Theorem 5.5, in order to ensure $\mathbb{E}\big[\big\|\nabla J\big(\boldsymbol{\theta}_{\text{out}}\big)\big\|_2^2\big]\le\epsilon$, it suffices to ensure

$$\frac{8(J(\boldsymbol{\theta}^*)-J(\boldsymbol{\theta}_0))}{\eta Sm}=\frac{\epsilon}{2},\quad\frac{6\sigma^2}{N}=\frac{\epsilon}{2},$$

which implies $Sm=O(1/\epsilon)$ and $N=O(1/\epsilon)$. Note that we have set $m=O(\sqrt{B})$. The total number of stochastic gradient evaluations $\mathcal{T}_g$ we need is

$$\mathcal{T}_g=SN+SmB=O\bigg(\frac{N}{\sqrt{B}\epsilon}+\frac{B}{\epsilon}\bigg)=O\bigg(\frac{1}{\epsilon^{5/3}}\bigg),$$

where we set $B=N^{2/3}=1/\epsilon^{2/3}$. □

6.2 PROOF OF TECHNICAL LEMMAS

In this subsection, we provide the proofs of the technical lemmas used in the proof of main theory. We first prove the smoothness of $J(\boldsymbol{\theta})$.

Proof of Proposition 5.2. Recall the notion in (3.3) as

$$\nabla J(\boldsymbol{\theta})=\int_\tau\mathcal{R}(\tau)\nabla_{\boldsymbol{\theta}}p(\tau|\boldsymbol{\theta})\mathrm{d}\tau,$$

which directly implies the Hessian matrix

$$\nabla^2 J(\boldsymbol{\theta})=\int_\tau\mathcal{R}(\tau)\nabla^2_{\boldsymbol{\theta}}p(\tau|\boldsymbol{\theta})\mathrm{d}\tau. \qquad (6.11)$$

Note that the Hessian of the log-density function is

$$\nabla^2_{\boldsymbol{\theta}}\log p(\tau|\boldsymbol{\theta})=-p(\tau|\boldsymbol{\theta})^{-2}\nabla_{\boldsymbol{\theta}}p(\tau|\boldsymbol{\theta})\nabla_{\boldsymbol{\theta}}p(\tau|\boldsymbol{\theta})^\top$$
$$+p(\tau|\boldsymbol{\theta})^{-1}\nabla^2_{\boldsymbol{\theta}}p(\tau|\boldsymbol{\theta}). \qquad (6.12)$$

Substituting (6.12) into (6.11) yields

$$\nabla^2 J(\boldsymbol{\theta})=\int_\tau p(\tau|\boldsymbol{\theta})\mathcal{R}(\tau)\big[\nabla^2_{\boldsymbol{\theta}}\log p(\tau|\boldsymbol{\theta})$$
$$+\nabla_{\boldsymbol{\theta}}\log p(\tau|\boldsymbol{\theta})\nabla_{\boldsymbol{\theta}}\log p(\tau|\boldsymbol{\theta})^\top\big]\mathrm{d}\tau.$$

Therefore, we have

$$\|\nabla^2 J(\boldsymbol{\theta})\|_2\le\int_\tau p(\tau|\boldsymbol{\theta})\mathcal{R}(\tau)\big[\|\nabla^2_{\boldsymbol{\theta}}\log p(\tau|\boldsymbol{\theta})\|_2$$
$$+\|\nabla_{\boldsymbol{\theta}}\log p(\tau|\boldsymbol{\theta})\|_2^2\big]\mathrm{d}\tau$$
$$\le\int_\tau p(\tau|\boldsymbol{\theta})\mathcal{R}(\tau)H(M+HG^2)\mathrm{d}\tau$$
$$\le\frac{R(1-\gamma^H)H(M+HG^2)}{1-\gamma}$$
$$\le RH(M+HG^2)/(1-\gamma), \qquad (6.13)$$

where the second inequality comes from Assumption 5.1 and the third inequality holds due to the definition of $\mathcal{R}$ in (3.1). Thus $J(\boldsymbol{\theta})$ is L-smooth with $L=RH(M+HG^2)/(1-\gamma)$. Recall the REINFORCE estimator defined in (3.5). We immediately have

$$\|\nabla g(\tau|\boldsymbol{\theta})\|_2\le\Bigg[\sum_{t=0}^{H-1}\big\|\nabla^2\log\pi_{\boldsymbol{\theta}}(a_t|s_t)\big\|_2\Bigg]\frac{R+|b|}{1-\gamma}$$
$$\le HM(R+|b|)/(1-\gamma).$$

Similarly, we have

$$\|g(\tau|\boldsymbol{\theta})\|_2\le HG\bigg[\frac{R(1-\gamma^H)}{1-\gamma}+|b|\bigg]\le\frac{HG(R+|b|)}{1-\gamma}.$$

The proof of the GPOMDP estimator is similar and we omit it for simplicity. This completes the proof. □

The analysis of Lemma 6.1 relies on the following important properties of importance sampling weights.

Lemma 6.2 (Lemma 1 in Cortes et al. (2010))**.** Let $\omega(x)=P(x)/Q(x)$ be the importance weight for distributions P and Q. Then the following identities hold:

$$\mathbb{E}[\omega]=1,\quad\mathbb{E}[\omega^2]=d_2(P||Q),$$

where $d_2(P||Q)=2^{D_2(P||Q)}$ and $D_2(P||Q)$ is the Rényi divergence between distributions P and Q. Note that this immediately implies $\text{Var}(\omega)=d_2(P||Q)-1$.

Proof of Lemma 6.1. According to Lemma 6.2, we have

$$\text{Var}\big(\omega\big(\tau|\widetilde{\boldsymbol{\theta}}^s,\boldsymbol{\theta}_t^{s+1}\big)\big)=d_2\big(p(\tau|\widetilde{\boldsymbol{\theta}}^s)||p(\tau|\boldsymbol{\theta}_t^{s+1})\big)-1.$$

In the rest of this proof, we denote $\boldsymbol{\theta}_1=\widetilde{\boldsymbol{\theta}}^s$ and $\boldsymbol{\theta}_2=\boldsymbol{\theta}_t^{s+1}$ to simplify the notation. By definition, we have

$$d_2(p(\tau|\boldsymbol{\theta}_1)||p(\tau|\boldsymbol{\theta}_2))=\int_\tau p(\tau|\boldsymbol{\theta}_1)\frac{p(\tau|\boldsymbol{\theta}_1)}{p(\tau|\boldsymbol{\theta}_2)}\mathrm{d}\tau$$

$$= \int_\tau p(\tau|\boldsymbol{\theta}_1)^2 p(\tau|\boldsymbol{\theta}_2)^{-1}\mathrm{d}\tau.$$

For any fixed $\boldsymbol{\theta}_2 \in \mathbb{R}^d$, computing the gradient of $d_2(p(\tau|\boldsymbol{\theta}_1)||p(\tau|\boldsymbol{\theta}_2))$ with respect to $\boldsymbol{\theta}_1$ yields

$$\begin{aligned}&\nabla_{\boldsymbol{\theta}_1} d_2(p(\tau|\boldsymbol{\theta}_1)||p(\tau|\boldsymbol{\theta}_2))\\ &= 2\int_\tau p(\tau|\boldsymbol{\theta}_1)\nabla_{\boldsymbol{\theta}_1} p(\tau|\boldsymbol{\theta}_1)p(\tau|\boldsymbol{\theta}_2)^{-1}\mathrm{d}\tau,\end{aligned}$$

which implies that if we set $\boldsymbol{\theta}_1 = \boldsymbol{\theta}_2$, we will obtain

$$\begin{aligned}&\nabla_{\boldsymbol{\theta}_1} d_2(p(\tau|\boldsymbol{\theta}_1)||p(\tau|\boldsymbol{\theta}_2))\big|_{\boldsymbol{\theta}_1=\boldsymbol{\theta}_2}\\ &= 2\int_\tau \nabla_{\boldsymbol{\theta}_1} p(\tau|\boldsymbol{\theta}_1)\mathrm{d}\tau\big|_{\boldsymbol{\theta}_1=\boldsymbol{\theta}_2}\\ &= 0.\end{aligned}$$

Hence, applying mean value theorem, we have

$$\begin{aligned}&d_2(p(\tau|\boldsymbol{\theta}_1)||p(\tau|\boldsymbol{\theta}_2)) \qquad (6.14)\\ &= 1 + \frac{1}{2}(\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2)^\top \nabla^2_{\boldsymbol{\theta}} d_2(p(\tau|\boldsymbol{\theta})||p(\tau|\boldsymbol{\theta}_2))(\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2),\end{aligned}$$

where $\boldsymbol{\theta} = t\boldsymbol{\theta}_1 + (1-t)\boldsymbol{\theta}_2$ for some $t \in [0,1]$. Next, we compute the Hessian matrix. For any fixed $\boldsymbol{\theta}_2$, we have

$$\begin{aligned}&\nabla^2_{\boldsymbol{\theta}} d_2(p(\tau|\boldsymbol{\theta})||p(\tau|\boldsymbol{\theta}_2))\\ &= 2\int_\tau \nabla_{\boldsymbol{\theta}} p(\tau|\boldsymbol{\theta})\nabla_{\boldsymbol{\theta}} p(\tau|\boldsymbol{\theta})^\top p(\tau|\boldsymbol{\theta}_2)^{-1}\mathrm{d}\tau\\ &\quad + 2\int_\tau \nabla^2_{\boldsymbol{\theta}} p(\tau|\boldsymbol{\theta})p(\tau|\boldsymbol{\theta})p(\tau|\boldsymbol{\theta}_2)^{-1}\mathrm{d}\tau\\ &= 2\int_\tau \nabla_{\boldsymbol{\theta}} \log p(\tau|\boldsymbol{\theta})\nabla_{\boldsymbol{\theta}} \log p(\tau|\boldsymbol{\theta})^\top \frac{p(\tau|\boldsymbol{\theta})^2}{p(\tau|\boldsymbol{\theta}_2)}\mathrm{d}\tau\\ &\quad + 2\int_\tau \nabla^2_{\boldsymbol{\theta}} p(\tau|\boldsymbol{\theta})p(\tau|\boldsymbol{\theta})p(\tau|\boldsymbol{\theta}_2)^{-1}\mathrm{d}\tau. \qquad (6.15)\end{aligned}$$

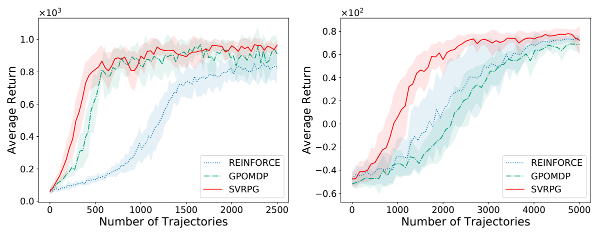
Recall the Hessian of the log-density function in (6.12). Substituting (6.12) into (6.15) yields

$$\begin{aligned}&\|\nabla^2_{\boldsymbol{\theta}} d_2(p(\tau|\boldsymbol{\theta})||p(\tau|\boldsymbol{\theta}_2))\|_2\\ &= \bigg\|4\int_\tau \nabla_{\boldsymbol{\theta}} \log p(\tau|\boldsymbol{\theta})\nabla_{\boldsymbol{\theta}} \log p(\tau|\boldsymbol{\theta})^\top \frac{p(\tau|\boldsymbol{\theta})^2}{p(\tau|\boldsymbol{\theta}_2)}\mathrm{d}\tau\\ &\quad + 2\int_\tau \nabla^2_{\boldsymbol{\theta}} \log p(\tau|\boldsymbol{\theta})\frac{p(\tau|\boldsymbol{\theta})^2}{p(\tau|\boldsymbol{\theta}_2)}\mathrm{d}\tau\bigg\|_2\\ &\le \int_\tau \frac{p(\tau|\boldsymbol{\theta})^2}{p(\tau|\boldsymbol{\theta}_2)}\big(4\|\nabla_{\boldsymbol{\theta}} \log p(\tau|\boldsymbol{\theta})\|_2^2\\ &\quad + 2\|\nabla^2_{\boldsymbol{\theta}} \log p(\tau|\boldsymbol{\theta})\|_2\big)\mathrm{d}\tau\\ &\le (4H^2G^2 + 2HM)\mathbb{E}[\omega(\tau|\boldsymbol{\theta},\boldsymbol{\theta}_2)^2]\\ &\le 2H(2HG^2 + M)(W+1),\end{aligned}$$

where the second inequality comes from Assumption 5.1 and the last inequality is due to Assumption 5.4 and Lemma 6.2. Therefore, by (6.14) we have

$$\begin{aligned}\mathrm{Var}\big(\omega\big(\tau|\widetilde{\boldsymbol{\theta}}^s, \boldsymbol{\theta}_t^{s+1}\big)\big) &= d_2\big(p(\tau|\widetilde{\boldsymbol{\theta}}^s)||p(\tau|\boldsymbol{\theta}_t^{s+1})\big) - 1\\ &\le C_\omega\|\widetilde{\boldsymbol{\theta}}^s - \boldsymbol{\theta}_t^{s+1}\|_2^2,\end{aligned}$$

where $C_\omega = H(2HG^2 + M)(W+1)$. □



(a) Cartpole (b) Mountain Car

Figure 1: The average reward of different algorithms in Cartpole and Mountain Car environments.

7 EXPERIMENTS

In this section, we conduct experiments on reinforcement learning benchmark tasks, i.e., the Cartpole and Mountain Car (continuous) environments (Brockman et al., 2016), to evaluate the performance of Algorithm 1. We measure the performance of an algorithm in terms of the total sample trajectories it needs to achieve a certain reward. We compare SVRPG with vanilla stochastic gradient based algorithms: the REINFORCE (Williams, 1992) and GPOMDP[1] (Baxter & Bartlett, 2001) algorithms. Recall that at each iteration of Algorithm 1, we also need to choose certain stochastic gradient estimator to approximate the full gradient based on sampled trajectories. Since the performance of GPOMDP is always comparable or better than REINFORCE, we only report the results of SVRPG with the GPOMDP estimator.

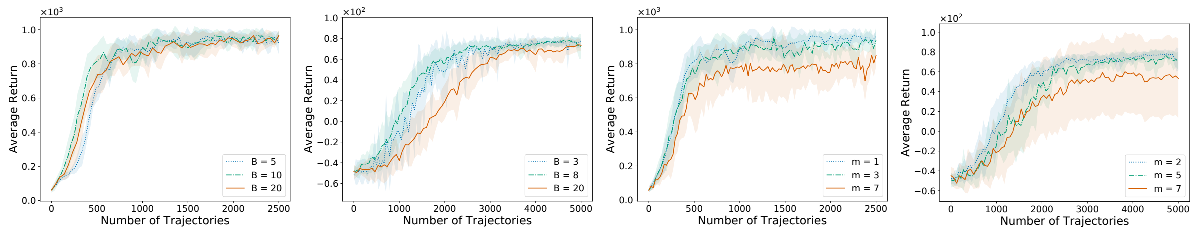
We follow the practical suggestions provided in Papini et al. (2018) to improve the performance including (1) performing one initial gradient update immediately after sampling the N trajectories in the outer loop; (2) using adaptive step sizes; and (3) using adaptive epoch length (terminate the inner loop update early if the step size used in the inner loop is smaller than that used in the outer loop). Following Papini et al. (2018), we use the following Gaussian policy with a fixed standard deviation $\widetilde{\sigma}^2$:

$$\pi_{\boldsymbol{\theta}}(a|s) = 1/\sqrt{2\pi}\sigma \exp\big(-(\boldsymbol{\theta}^\top\phi(s) - a)^2/2\widetilde{\sigma}^2\big),$$

where $\phi : \mathcal{S} \mapsto \mathbb{R}^d$ is a bounded feature map. Under the Gaussian policy, it is easy to verify that Assumptions 5.1 and 5.3 is satisfied with parameters depending on $\phi, \widetilde{\sigma}^2$ and the upper bound of the action a for all $a \in \mathcal{A}$.

Cartpole Setup: The neural network of the Cartpole environment has one hidden layer of 64 nodes with the

[1]We thank Papini et al. (2018) for their implementations of GPOMDP and SVRPG as well as Duan et al. (2016) for their implementations from the *rllab* library.



(a) Cartpole (b) Mountain Car (c) Cartpole (d) Mountain Car

Figure 2: (a) - (b): The average reward of SVRPG with different mini-batch sizes B. (c) - (d): The average reward of SVRPG with different epoch lengths m.

$tanh$ activation function. In the comparison between REINFORCE, GPOMDP, and SVRPG, we use learning rate $\eta = [0.005, 0.005, 0.0125]$ for them respectively. Based on our theoretical analysis, we chose $N = 25$ and $B = 10$ for SVRPG. We also the set the batch size of vanilla gradient methods to be $N = 25$ for REINFORCE and $N = 10$ for GPOMDP.

Mountain Car Setup: The neural network for the Mountain Car environment contains one hidden layer with 16 nodes with the $tanh$ activation. In the comparison among REINFORCE, GPOMDP and SVRPG, we set $N = 25$ and $B = 8$ for SVRPG and set batch size $N = 20$ for the vanilla gradient methods. REINFORCE, GPOMDP, and SVRPG have respective learning rates of $\eta = [0.005, 0.005, 0.0075]$.

Experimental Results: Figures 1(a) and 1(b) respectively show the performance of different algorithms on the Cartpole and Mountain Car environments. All the results are averaged over 10 repetitions and the shaded area is a confidence interval corresponding to the standard deviation over different runs. It can be seen that all the methods solved the Cartpole environment (with averaged reward close to 1000). The SVRPG algorithm outperforms the other two by gaining higher rewards with fewer sample trajectories. SVRPG also beats the other methods in solving the Mountain Car environment (with averaged reward close to 90).

Sensitivity Study: We now study the impact of the mini-batch size B within each epoch of SVRPG to validate our theoretical claims in Corollary 5.7. Specifically, in Cartpole environment, we fix $N = 25$ and vary the mini-batch size B in the range of $\{5, 10, 20\}$. In Mountain Car environment, we set $N = 25$ and vary the mini-batch size B in the range of $\{3, 8, 20\}$. The learning rates are tuned accordingly. Figures 2(a) and 2(b) show the effect of different mini-batch sizes B on SVRPG. Note that the outer loop batch size is $N = 25$ in both environments. It can be seen that when $B = 10$ and $B = 8$ for Cartpole and Mountain Car respectively, SVRPG achieve the best performance, which is well aligned with our theory. In particular, with a small mini-batch size, SVRPG acts similarly to the vanilla stochastic gradient based algorithms which needs fewer trajectories in each iteration but converges slowly and requires more trajectories in total. Conversely, using a large mini-batch pushes SVRPG to converge in fewer iterations, but requires more trajectories in total.

We also conduct experiments to show the impact of different epoch lengths m on the performance of SVRPG. Similar to the previous sensitivity study on B, here we fix N and B for all experiments and vary the epoch length m in different experiments. The learning rates are tuned accordingly. Figures 2(c) and 2(d) show the sensitivity of SVRPG with respect to the epoch length m. It can be seen that smaller m tends to give better results, where the algorithm runs more outer iterations with a larger batch size N (compared with the mini-batch size B in the inner loop). This implies that the current choice of epoch length in Corollary 5.7 may not be optimal yet.

8 CONCLUSIONS

We revisited the SVRPG algorithm (Papini et al., 2018) and derived a sharp convergence analysis of SVRPG which achieves the state-of-the-art sample complexity. We provided a detailed discussion and guidance on the choice of batch sizes and epoch length based on our improved analysis so that the total number of samples can be significantly reduced. We also empirically validated the theoretical results on common reinforcement learning tasks. As a future direction, it would be interesting to see whether any better sample complexity can be obtained for policy gradient algorithms.

Acknowledgements

We would like to thank the anonymous reviewers for their helpful comments. This research was sponsored in part by the National Science Foundation IIS-1904183 and IIS-1906169. The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing any funding agencies.

References

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

Baxter, J. and Bartlett, P. L. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15:319–350, 2001.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym, 2016.

Cortes, C., Mansour, Y., and Mohri, M. Learning bounds for importance weighting. In *Advances in Neural Information Processing Systems*, pp. 442–450, 2010.

Du, S. S., Chen, J., Li, L., Xiao, L., and Zhou, D. Stochastic variance reduction methods for policy evaluation. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1049–1058. JMLR. org, 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*, 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 *Advances in Neural Information Processing Systems*, pp. 686–696, 2018.

Greensmith, E., Bartlett, P. L., and Baxter, J. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, 5(Nov):1471–1530, 2004.

Gu, S., Lillicrap, T., Ghahramani, Z., Turner, R. E., and Levine, S. Q-prop: Sample-efficient policy gradient with an off-policy critic. In *International Conference on Learning Representations 2017*. OpenReviews. net, 2017.

Harikandeh, R., Ahmed, M. O., Virani, A., Schmidt, M., Konečnỳ, J., and Sallinen, S. Stopwasting my gradients: Practical svrg. In *Advances in Neural Information Processing Systems*, pp. 2251–2259, 2015.

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

Kakade, S. M. A natural policy gradient. In *Advances in Neural Information Processing Systems*, pp. 1531–1538, 2002.

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

Levine, S., Wagener, N., and Abbeel, P. Learning contact-rich manipulation skills with guided policy search. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 156–163. IEEE, 2015.

Li, Z. and Li, J. A simple proximal stochastic gradient method for nonsmooth nonconvex optimization. In *Advances in Neural Information Processing Systems*, pp. 5569–5579, 2018.

Liu, J. S. *Monte Carlo strategies in scientific computing*. Springer Science & Business Media, 2008.

Metelli, A. M., Papini, M., Faccio, F., and Restelli, M. Policy optimization via importance sampling. In *Advances in Neural Information Processing Systems*, pp. 5447–5459, 2018.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

Nguyen, L. M., Liu, J., Scheinberg, K., and Takáč, M. Sarah: A novel method for machine learning problems using stochastic recursive gradient. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2613–2621. JMLR. org, 2017.

Papini, M., Pirotta, M., and Restelli, M. Adaptive batch size for safe policy gradients. In *Advances in Neural Information Processing Systems*, pp. 3591–3600, 2017.

Papini, M., Binaghi, D., Canonaco, G., Pirotta, M., and Restelli, M. Stochastic variance-reduced policy gradient. In *International Conference on Machine Learning*, pp. 4023–4032, 2018.

Peters, J. and Schaal, S. Reinforcement learning of motor skills with policy gradients. *Neural Networks*, 21(4): 682–697, 2008.

Pirotta, M., Restelli, M., and Bascetta, L. Adaptive step-size for policy gradient methods. In *Advances in Neural Information Processing Systems*, pp. 1394–1402, 2013.

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

Rényi, A. et al. On measures of entropy and information. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Vol-*

ume 1: Contributions to the Theory of Statistics. The Regents of the University of California, 1961.

Robbins, H. and Monro, S. A stochastic approximation method. *The Annals of Mathematical Statistics*, pp. 400–407, 1951.

Schulman, J., Levine, S., Abbeel, P., Jordan, M. I., and Moritz, P. Trust region policy optimization. In *International Conference on Machine Learning*, volume 37, pp. 1889–1897, 2015.

Sehnke, F., Osendorfer, C., Rückstieß, T., Graves, A., Peters, J., and Schmidhuber, J. Parameter-exploring policy gradients. *Neural Networks*, 23(4):551–559, 2010.

Shalev-Shwartz, S., Shammah, S., and Shashua, A. Safe, multi-agent, reinforcement learning for autonomous driving. *CoRR*, abs/1610.03295, 2016. URL `http://arxiv.org/abs/1610.03295`.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. Deterministic policy gradient algorithms. In *International Conference on Machine Learning*, 2014.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T. P., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–489, 2016.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, pp. 1057–1063, 2000.

Tucker, G., Bhupatiraju, S., Gu, S., Turner, R., Ghahramani, Z., and Levine, S. The mirage of action-dependent baselines in reinforcement learning. In *International Conference on Machine Learning*, pp. 5022–5031, 2018.

Weaver, L. and Tao, N. The optimal reward baseline for gradient-based reinforcement learning. In *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, pp. 538–545. Morgan Kaufmann Publishers Inc., 2001.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, 1992.

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

Xu, T., Liu, Q., and Peng, J. Stochastic variance reduction for policy gradient estimation. *CoRR*, abs/1710.06034, 2017. URL `http://arxiv.org/abs/1710.06034`.

Zhao, T., Hachiya, H., Niu, G., and Sugiyama, M. Analysis and improvement of policy gradient estimation. In *Advances in Neural Information Processing Systems*, pp. 262–270, 2011.

Zhou, D., Xu, P., and Gu, Q. Stochastic nested variance reduced gradient descent for nonconvex optimization. In *Advances in Neural Information Processing Systems*, pp. 3922–3933, 2018.