Sample Efficient Policy Gradient Methods with Recursive Variance Reduction

Pan Xu
Department of Computer Science
University of California, Los Angeles
Los Angeles, CA 90095
`panxu@cs.ucla.edu`

Felicia Gao
Department of Computer Science
University of California, Los Angeles
Los Angeles, CA 90095
`fxgao1160@engineering.ucla.edu`

Quanquan Gu
Department of Computer Science
University of California, Los Angeles
Los Angeles, CA 90095
`qgu@cs.ucla.edu`

Abstract

Improving the sample efficiency in reinforcement learning has been a long-standing research problem. In this work, we aim to reduce the sample complexity of existing policy gradient methods. We propose a novel policy gradient algorithm called SRVR-PG, which only requires $O(1/\epsilon^{3/2})$[1] episodes to find an ϵ-approximate stationary point of the nonconcave performance function $J(\boldsymbol{\theta})$ (i.e., $\boldsymbol{\theta}$ such that $\|\nabla J(\boldsymbol{\theta})\|_2^2 \leq \epsilon$). This sample complexity improves the best known result $O(1/\epsilon^{5/3})$ for policy gradient algorithms by a factor of $O(1/\epsilon^{1/6})$. We conduct numerical experiments on classic control problems in reinforcement learning to validate the performance of our proposed algorithm.

1 Introduction

Reinforcement learning (RL) [37] has received significant success in solving various complex problems such as learning robotic motion skills [14], autonomous driving [33] and Go game [36], where the agent progressively interacts with the environment in order to learn a good policy to solve the task. In RL, the agent makes its decision by choosing the action based on the current state and the historical rewards it has received so far. After performing the chosen action, the agent's state will change according to some transition probability model and a new reward would be revealed to the agent by the environment based on the action and new state. Then the agent continues to choose the next action until it reaches a terminal state. The aim of the agent is to maximize its expected cumulative rewards. Therefore, the pivotal problem in RL is to find a good policy which is a function that maps the state space to the action space and thus informs the agent which action to take at each state. To optimize the agent's policy in the high dimensional continuous action space, the most popular approach is the policy gradient [38] method that parameterizes the policy by an unknown parameter $\boldsymbol{\theta} \in \mathbb{R}^d$ and directly optimizes the policy by finding the optimal $\boldsymbol{\theta}$. The objective function $J(\boldsymbol{\theta})$ is chosen to be the performance function, which is the expected return under a specific policy and is usually non-concave. Our goal is to maximize the value of $J(\boldsymbol{\theta})$ by finding a stationary point $\boldsymbol{\theta}^*$ such that $\|\nabla J(\boldsymbol{\theta}^*)\|_2 = 0$ using gradient based algorithms.

Due to the expectation in the definition of $J(\boldsymbol{\theta})$, it is usually infeasible to compute the gradient exactly. In practice, one often uses stochastic gradient estimators such as REINFORCE [42], PGT [38] and GPOMDP [2] to approximate the gradient of the expected return based on a batch of

[1]$O(\cdot)$ notation hides constant factors.

Optimization Foundations for Reinforcement Learning Workshop at NeurIPS 2019, Vancouver, Canada.

sampled trajectories. However, this approximation will introduce additional variance and slow down the convergence of policy gradient, which thus requires a huge amount of trajectories to find a good policy. Theoretically, these stochastic gradient (SG) based algorithms require $O(1/\epsilon^2)$ trajectories [28] to find an ϵ-approximate stationary point such that $\mathbb{E}[\|\nabla J(\boldsymbol{\theta})\|_2^2] \le \epsilon$. In order to reduce the variance of policy gradient algorithms, Papini et al. [21] proposed a stochastic variance-reduced policy gradient (SVRPG) algorithm by borrowing the idea from the stochastic variance reduced gradient (SVRG) [10, 1, 25] in stochastic optimization. The key idea is to use a so-called semi-stochastic gradient to replace the stochastic gradient used in SG methods. The semi-stochastic gradient combines the stochastic gradient in the current iterate with a snapshot of stochastic gradient stored in an early iterate which is called a reference iterate. In practice, SVRPG saves computation on trajectories and improves the performance of SG based policy gradient methods. Papini et al. [21] also proved that SVRPG converges to an ϵ-approximate stationary point $\boldsymbol{\theta}$ of the nonconcave performance function $J(\boldsymbol{\theta})$ with $\mathbb{E}[\|\nabla J(\boldsymbol{\theta})\|_2^2] \le \epsilon$ after $O(1/\epsilon^2)$ trajectories, which seems to have the same sample complexity as SG based methods. Recently, the sample complexity of SVRPG has been improved to $O(1/\epsilon^{5/3})$ by a refined analysis [45], which theoretically justifies the advantage of SVRPG over SG based methods.

This paper continues on this line of research. We propose a Stochastic Recursive Variance Reduced Policy Gradient algorithm (SRVR-PG), which provably improves the sample complexity of SVRPG [21, 45]. At the core of our proposed algorithm is a recursive semi-stochastic policy gradient inspired from the stochastic path-integrated differential estimator [6], which accumulates all the stochastic gradients from different iterates to reduce the variance. We prove that SRVR-PG only takes $O(1/\epsilon^{3/2})$ trajectories to converge to an ϵ-approximate stationary point $\boldsymbol{\theta}$ of the performance function, i.e., $\mathbb{E}[\|\nabla J(\boldsymbol{\theta})\|_2^2] \le \epsilon$. We summarize the comparison of SRVR-PG with existing policy gradient methods in terms of sample complexity in Table 1. Evidently, the sample complexity of SRVR-PG is lower than that of REINFORCE, PGT and GPOMDP by a factor of $O(1/\epsilon^{1/2})$, and it is also lower than that of SVRPG [45] by a factor of $O(1/\epsilon^{1/6})$. Our experimental results on classical control tasks in reinforcement learning demonstrate the superior performance of the proposed SRVR-PG and verify our theoretical analysis.

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

Algorithms	Sample Complexity
REINFORCE [42] PGT [38] GPOMDP [2]	$O(1/\epsilon^2)$
SVRPG [21]	$O(1/\epsilon^2)$
SVRPG [45]	$O(1/\epsilon^{5/3})$
SRVR-PG (This paper)	$O(1/\epsilon^{3/2})$

1.1 Additional Related Work

In this subsection, we briefly review other relevant work to ours with a focus on policy gradient based methods. For other important RL methods such as value based [41, 18] and actor-critic [12, 22, 35] methods, we refer the reader to [23, 11, 37] for a complete review.

To reduce the variance of policy gradient methods, early works have introduced unbiased baseline functions [2, 8, 23] to reduce the variance, which can be constant, time-dependent or state-dependent. Schulman et al. [30] proposed the generalized advantage estimation (GAE) to explore the trade-off between bias and variance of policy gradient. Recently, action-dependent baselines are also used in [39, 43] which introduces bias but reduces variance at the same time. Sehnke et al. [31, 32] proposed policy gradient with parameter-based exploration (PGPE) that explores in the parameter space. It has been shown that PGPE enjoys a much smaller variance in nature [49]. The Stein variational policy gradient method is proposed in [17]. See [23, 4, 15] for a more detailed survey on policy gradient.

Stochastic variance reduced gradient techniques such as SVRG [10, 44], batching SVRG [9], SAGA [3] and SARAH [19] were first developed in stochastic convex optimization. When the objective function is nonconvex (or nonconcave for maximization problems), nonconvex SVRG [1, 25] and SCSG [13, 16] were proposed and proved to converge to a first-order stationary point faster than vanilla SGD [28] with no variance reduction. The state-of-the-art stochastic variance reduced gradient methods for nonconvex functions are the SNVRG [51, 50] and SPIDER [6] algorithms, which have been proved to achieve near optimal convergence rate for smooth functions.

There are yet not many papers studying variance reduced gradient techniques in RL. Du et al. [5] first applied SVRG in policy evaluation for a fixed policy. Xu et al. [46] introduced SVRG into trust region policy optimization for model-free policy gradient and showed that the resulting algorithm SVRPO is more sample efficient than TRPO. Yuan et al. [48] further applied the techniques in SARAH [19] and SPIDER [6] to TRPO [29]. However, no analysis on sample complexity (i.e., number of trajectories required) was provided in the aforementioned papers [46, 48]. We note that a recent work by Shen et al. [34] proposed a Hessian aided policy gradient (HAPG) algorithm that converges to the stationary point of the performance function within $O(H^2/\epsilon^{3/2})$ trajectories, which is worse than our result by a factor of $O(H^2)$ where H is the horizon length of the environment. Moreover, they need additional samples to approximate the Hessian vector product, and cannot handle the policy in a constrained parameter space. Another related work is Yang and Zhang [47], which extended the stochastic mirror descent algorithm [7] in the optimization field to policy gradient methods and achieved $O(H^2/\epsilon^2)$ sample complexity.

Notation For a vector $\mathbf{v} \in \mathbb{R}^d$ we use $\|\mathbf{v}\|_2$ to denote its Euclidean norm and for a matrix $\mathbf{A} \in \mathbb{R}^{d\times d}$ we use $\|\mathbf{A}\|_2$ to denote its spectral norm. We denote $a_n = O(b_n)$ if $a_n \leq Cb_n$ for some constant $C > 0$. The Dirac delta function $\delta(x)$ is defined as: $\delta(x) = +\infty$ if $x = 0$, and $\delta(x) = 0$ otherwise. Note that $\delta(x)$ satisfies $\int_{-\infty}^{+\infty} \delta(x)\mathrm{d}x = 1$. For any $\alpha > 0$, we define $D_\alpha(P||Q) = 1/(\alpha-1)\log_2 \int_x P(x)(P(x)/Q(x))^{\alpha-1}\mathrm{d}x$ as the Rényi divergence [27] between two distributions P and Q, which is non-negative for all $\alpha > 0$. We also define the exponentiated Rényi divergence as $d_\alpha(P||Q) = 2^{D_\alpha(P||Q)}$.

2 Backgrounds on Policy Gradient

Markov Decision Process: A reinforcement learning task is usually modeled as a discrete-time Markov Decision Process (MDP): $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma, \rho\}$. $\mathcal{S}$ and $\mathcal{A}$ are the state space and action space respectively. $\mathcal{P}(s'|s,a)$ is the transition probability that the agent transits to state s' after taking action a at state s. Function $r(s,a): \mathcal{S}\times\mathcal{A} \to [-R, R]$ emits a bounded reward after the agent takes action a at state s, where $R > 0$ is a constant. $\gamma \in (0,1)$ is the discount factor. ρ is the distribution of the starting state. A policy at state s is a probability function $\pi(a|s)$ over action space $\mathcal{A}$. In episodic tasks, following any stationary policy, the agent can observe and collect a sequence of state-action pairs $\tau = \{s_0, a_0, s_1, a_1, \ldots, s_{H-1}, a_{H-1}, s_H\}$, which is called a trajectory or episode. H is the trajectory horizon or episode length, which can be a random variable in general. In practice, we can set H to be the maximum value among all the actual trajectory horizons we have collected. The sample return over one trajectory τ is defined as the discounted cumulative reward

$$\mathcal{R}(\tau) = \textstyle\sum_{h=0}^{H-1} \gamma^h r(s_h, a_h). \tag{2.1}$$

Policy Gradient: Suppose the policy, denoted by $\pi_{\boldsymbol{\theta}}$, is parameterized by an unknown parameter $\boldsymbol{\theta} \in \mathbb{R}^d$. We denote the trajectory distribution induced by policy $\pi_{\boldsymbol{\theta}}$ as $p(\tau|\boldsymbol{\theta})$. Then

$$p(\tau|\boldsymbol{\theta}) = \rho(s_0)\textstyle\prod_{h=0}^{H-1} \pi_{\boldsymbol{\theta}}(a_h|s_h)P(s_{h+1}|s_h, a_h). \tag{2.2}$$

We define the expected return under policy $\pi_{\boldsymbol{\theta}}$ as $J(\boldsymbol{\theta}) = \mathbb{E}_{\tau\sim p(\cdot|\boldsymbol{\theta})}[\mathcal{R}(\tau)|\mathcal{M}]$, which is also called the performance function. To maximize the performance function, we can update the policy parameter $\boldsymbol{\theta}$ by iteratively running gradient ascent based algorithms, i.e., $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \eta\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_k)$, where $\eta > 0$ is the step size and the gradient $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ is derived as follows:

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) &= \textstyle\int_\tau \mathcal{R}(\tau)\nabla_{\boldsymbol{\theta}} p(\tau|\boldsymbol{\theta})\mathrm{d}\tau = \int_\tau \mathcal{R}(\tau)\frac{\nabla_{\boldsymbol{\theta}} p(\tau|\boldsymbol{\theta})}{p(\tau|\boldsymbol{\theta})} p(\tau|\boldsymbol{\theta})\mathrm{d}\tau \\ &= \mathbb{E}_{\tau\sim p(\cdot|\boldsymbol{\theta})}[\nabla_{\boldsymbol{\theta}} \log p(\tau|\boldsymbol{\theta})\mathcal{R}(\tau)|\mathcal{M}].\end{aligned} \tag{2.3}$$

However, it is intractable to calculate the exact gradient in (2.3) since the trajectory distribution $p(\tau|\boldsymbol{\theta})$ is unknown. In practice, policy gradient algorithm samples a batch of trajectories $\{\tau_i\}_{i=1}^N$ to approximate the exact gradient based on the sample average over all sampled trajectories:

$$\widehat{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = 1/N \textstyle\sum_{i=1}^N \nabla_{\boldsymbol{\theta}} \log p(\tau_i|\boldsymbol{\theta})\mathcal{R}(\tau_i). \tag{2.4}$$

At the k-th iteration, the policy is then updated by $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \eta\widehat{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_k)$. According to (2.2), we know that $\nabla_{\boldsymbol{\theta}} \log p(\tau_i|\boldsymbol{\theta})$ is independent of the transition probability matrix P. Therefore, combining this with (2.1), we can rewrite the approximate gradient as follows

$$\begin{aligned}\widehat{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) &= 1/N \textstyle\sum_{i=1}^N \big(\sum_{h=0}^{H-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_h^i|s_h^i)\big)\big(\sum_{h=0}^{H-1} \gamma^h r(s_h^i, a_h^i)\big) \\ &\overset{\text{def}}{=} 1/N \textstyle\sum_{i=1}^N g(\tau_i|\boldsymbol{\theta}),\end{aligned} \tag{2.5}$$

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 $i = 1, \ldots, N$ and $g(\tau_i|\boldsymbol{\theta})$ is an unbiased gradient estimator computed based on the i-th trajectory τ_i. The gradient estimator in (2.5) is based on the likelihood ratio methods and is often referred to as the REINFORCE gradient estimator [42]. Since $\mathbb{E}[\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s)] = 0$, we can add any constant baseline b_t to the reward that is independent of the current action and the gradient estimator still remains unbiased. With the observation that future actions do not depend on past rewards, another famous policy gradient theorem (PGT) estimator [38] removes the rewards from previous states:

$$g(\tau_i|\boldsymbol{\theta}) = \sum_{h=0}^{H-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_h^i|s_h^i)\big(\sum_{t=h}^{H-1} \gamma^t r(s_t^i, a_t^i) - b_t\big), \tag{2.6}$$

where b_t is a constant baseline. It has been shown [23] that the PGT estimator is equivalent to the commonly used GPOMDP estimator [2] defined as follows:

$$g(\tau_i|\boldsymbol{\theta}) = \sum_{h=0}^{H-1} \big(\sum_{t=0}^{h} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t^i|s_t^i)\big)\big(\gamma^h r(s_h^i, a_h^i) - b_h\big). \tag{2.7}$$

All the three gradient estimators mentioned above are unbiased [23]. It has been proved that the variance of the PGT/GPOMDP estimator is independent of horizon H while the variance of REINFORCE depends on H polynomially [49, 24]. Therefore, we will focus on the PGT/GPOMDP estimator in this paper and refer to them interchangeably due to their equivalence.

3 The Proposed Algorithm

The approximation in (2.4) using a batch of trajectories often causes a high variance in practice. In this section, we propose a novel variance reduced policy gradient algorithm called stochastic recursive variance reduced policy gradient (SRVR-PG), which is displayed in Algorithm 1. Our SRVR-PG algorithm consists of S epochs. At the beginning of the s-th epoch, where $s = 0, \ldots, S$, SRVR-PG stores a reference policy parameterized by $\widetilde{\boldsymbol{\theta}}^s = \boldsymbol{\theta}_0^{s+1}$. Based on policy $\pi_{\widetilde{\boldsymbol{\theta}}^s}$, it samples N episodes $\{\tau_i\}_{i=1}^N$ to compute a gradient estimator $\mathbf{v}_0^s = 1/N \sum_{i=1}^N g(\tau_i|\widetilde{\boldsymbol{\theta}}^s)$, where $g(\tau_i|\widetilde{\boldsymbol{\theta}}^s)$ is the PGT/GPOMDP estimator. Then the policy is immediately update as in Line 6 of Algorithm 1.

Within the epoch, at the t-th iteration, SRVR-PG samples B episodes $\{\tau_j\}_{j=1}^B$ based on the current policy $\pi_{\boldsymbol{\theta}_t^{s+1}}$. We then define the following recursive semi-stochastic gradient estimator:

$$\mathbf{v}_t^{s+1} = 1/B \sum_{j=1}^B g(\tau_j|\boldsymbol{\theta}_t^{s+1}) - \frac{1}{B}\sum_{j=1}^B g_\omega(\tau_j|\boldsymbol{\theta}_{t-1}^{s+1}) + \mathbf{v}_{t-1}^{s+1}, \tag{3.1}$$

where the first term is a stochastic gradient based on B episodes sampled from the current policy, and the difference between the last two terms can be viewed as a control variate to reduce the variance of the stochastic gradient. In many practical applications, the policy parameter space is a subset of $\mathbb{R}^d$, i.e., $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ with $\boldsymbol{\Theta} \subseteq \mathbb{R}^d$ being a convex set. In this case, we need to project the updated policy parameter onto the constraint set. Base on the semi-stochastic gradient (3.1), we can update the policy parameter using projected gradient ascent along the direction of $\mathbf{v}_t^{s+1}$: $\boldsymbol{\theta}_{t+1}^{s+1} = \mathcal{P}_{\boldsymbol{\Theta}}(\boldsymbol{\theta}_t^{s+1} + \eta\mathbf{v}_t^{s+1})$, where $\eta > 0$ is the step size. Here the projection operator associated with $\boldsymbol{\Theta}$ is defined as follows

$$\mathcal{P}_{\boldsymbol{\Theta}}(\boldsymbol{\theta}) = \underset{\mathbf{u}\in\boldsymbol{\Theta}}{\operatorname{argmin}} \|\boldsymbol{\theta} - \mathbf{u}\|_2^2 = \underset{\mathbf{u}\in\mathbb{R}^d}{\operatorname{argmin}} \big\{\mathbb{1}_{\boldsymbol{\Theta}}(\mathbf{u}) + 1/(2\eta)\|\boldsymbol{\theta} - \mathbf{u}\|_2^2\big\}, \tag{3.2}$$

where $\mathbb{1}_{\boldsymbol{\Theta}}(\mathbf{u})$ is the set indicator function on $\boldsymbol{\Theta}$, i.e., $\mathbb{1}_{\boldsymbol{\Theta}}(\mathbf{u}) = 0$ if $\mathbf{u} \in \boldsymbol{\Theta}$ and $\mathbb{1}_{\boldsymbol{\Theta}}(\mathbf{u}) = +\infty$ otherwise. $\eta > 0$ is any finite real value and is chosen as the step size in our paper. It is easy to see that $\mathbb{1}_{\boldsymbol{\Theta}}(\cdot)$ is nonsmooth. The goal of our algorithm is to find a point $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ that maximizes the performance function $J(\boldsymbol{\theta})$ subject to the constraint, namely, $\max_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} J(\boldsymbol{\theta}) = \max_{\boldsymbol{\theta}\in\mathbb{R}^d}\{J(\boldsymbol{\theta}) - \mathbb{1}_{\boldsymbol{\Theta}}(\boldsymbol{\theta})\}$. The gradient norm $\|\nabla J(\boldsymbol{\theta})\|_2$ is not sufficient to characterize the convergence of the algorithm due to additional the constraint. Following the literature on nonsmooth optimization [26, 7, 19, 16, 40], we use the generalized first-order stationary condition: $\mathcal{G}_\eta(\boldsymbol{\theta}) = \mathbf{0}$, where the *gradient mapping* $\mathcal{G}_\eta$ is defined as follows

$$\mathcal{G}_\eta(\boldsymbol{\theta}) = 1/\eta(\mathcal{P}_{\boldsymbol{\Theta}}(\boldsymbol{\theta} + \eta\nabla J(\boldsymbol{\theta})) - \boldsymbol{\theta}). \tag{3.3}$$

We can view $\mathcal{G}_\eta$ as a generalized projected gradient at $\boldsymbol{\theta}$. By definition if $\boldsymbol{\Theta} = \mathbb{R}^d$, we have $\mathcal{G}_\eta(\boldsymbol{\theta}) \equiv \nabla J(\boldsymbol{\theta})$. Therefore, the policy is update is displayed in Line 10 in Algorithm 1, where prox is the proximal operator defined in (3.2). Similar recursive semi-stochastic gradients to (3.1) were first proposed in stochastic optimization for finite-sum problems, leading to the stochastic recursive gradient algorithm (SARAH) [19, 20] and the stochastic path-integrated differential estimator (SPIDER) [6, 40]. However, our gradient estimator in (3.1) is noticeably different from that in

Algorithm 1 Stochastic Recursive Variance Reduced Policy Gradient (SRVR-PG)

```
1: Input: number of epochs $S$, epoch size $m$, step size $\eta$, batch size $N$, mini-batch size $B$, gradient
   estimator $g$, initial parameter $\boldsymbol{\theta}_m^0 := \widetilde{\boldsymbol{\theta}}^0 := \boldsymbol{\theta}_0 \in \boldsymbol{\Theta}$
2: for $s = 0, \ldots, S-1$ do
3:   $\boldsymbol{\theta}_0^{s+1} = \widetilde{\boldsymbol{\theta}}^s$
4:   Sample $N$ trajectories $\{\tau_i\}$ from $p(\cdot|\widetilde{\boldsymbol{\theta}}^s)$
5:   $\mathbf{v}_0^{s+1} = \widehat{\nabla}_{\boldsymbol{\theta}} J(\widetilde{\boldsymbol{\theta}}^s) := \frac{1}{N}\sum_{i=1}^N g(\tau_i|\widetilde{\boldsymbol{\theta}}^s)$
6:   $\boldsymbol{\theta}_1^{s+1} = \boldsymbol{\theta}_0^{s+1} + \eta \mathbf{v}_0^{s+1}$
7:   for $t = 1, \ldots, m-1$ do
8:     Sample $B$ trajectories $\{\tau_j\}$ from $p(\cdot|\boldsymbol{\theta}_t^{s+1})$
9:     $\mathbf{v}_t^{s+1} = \mathbf{v}_{t-1}^{s+1} + \frac{1}{B}\sum_{j=1}^B \big(g(\tau_j|\boldsymbol{\theta}_t^{s+1}) - g_\omega(\tau_j|\boldsymbol{\theta}_{t-1}^{s+1})\big)$
10:    $\boldsymbol{\theta}_{t+1}^{s+1} = \mathcal{P}_{\boldsymbol{\Theta}}(\boldsymbol{\theta}_t^{s+1} + \eta \mathbf{v}_t^{s+1})$
11:  end for
12: end for
13: return $\boldsymbol{\theta}_{\text{out}}$, which is uniformly picked from $\{\boldsymbol{\theta}_t^s\}_{t=0,\ldots,m-1;s=0,\ldots,S}$
```

[19, 6, 40, 20] due to the gradient estimator $g_\omega(\tau_j|\boldsymbol{\theta}_{t-1}^{s+1})$ that is equipped with step-wise importance weights. Take the GPOMDP estimator for example, we define the step-wise importance weighted estimator as follows

$$g_\omega(\tau_j|\boldsymbol{\theta}) = \sum_{h=0}^{H-1} \omega_{0:h}(\tau|\boldsymbol{\theta}_1, \boldsymbol{\theta}_2)\big(\sum_{t=0}^h \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t^j|s_t^j)\big)\gamma^h r(s_h^j, a_h^j), \tag{3.4}$$

where $\omega_{0:h}(\tau|\boldsymbol{\theta}_1, \boldsymbol{\theta}_2) = \prod_{h'=0}^h \pi_{\boldsymbol{\theta}_1}(a_h|s_h)/\pi_{\boldsymbol{\theta}_2}(a_h|s_h)$ is the importance weight from $p(\tau_h|\boldsymbol{\theta}_t^{s+1})$ to $p(\tau_h|\boldsymbol{\theta}_{t-1}^{s+1})$ and τ_h is a truncated trajectory $\{(a_t, s_t)\}_{t=0}^h$ from the full trajectory τ. This term is essential to deal with the non-stationarity of the distribution of the trajectory τ in (3.1). Specifically, $\{\tau_j\}_{j=1}^B$ are sampled from policy $\pi_{\boldsymbol{\theta}_t^{s+1}}$ while the PGT/GPOMDP estimator $g(\cdot|\boldsymbol{\theta}_{t-1}^{s+1})$ is defined based on policy $\pi_{\boldsymbol{\theta}_{t-1}^{s+1}}$ according to (2.7). This inconsistency introduces extra challenges in the convergence analysis of SRVR-PG. Using importance weighting, we can obtain

$$\mathbb{E}_{\tau\sim p(\tau|\boldsymbol{\theta}_t^{s+1})}[g_\omega(\tau|\boldsymbol{\theta}_{t-1}^{s+1})] = \mathbb{E}_{\tau\sim p(\tau|\boldsymbol{\theta}_{t-1}^{s+1})}[g(\tau|\boldsymbol{\theta}_{t-1}^{s+1})].$$

which eliminates the inconsistency caused by the varying trajectory distribution.

It is worth noting that our semi-stochastic gradient in (3.1) also differs from the one used in SVRPG [21] because we recursively update $\mathbf{v}_t^{s+1}$ using $\mathbf{v}_{t-1}^{s+1}$ from the previous iteration, while SVRPG uses a reference gradient that is only updated at the beginning of each epoch. Moreover, SVRPG wastes N trajectories without updating the policy at the beginning of each epoch, while Algorithm 1 updates the policy immediately after this sampling process (Line 6), which saves computation in practice.

We notice that very recently another algorithm called SARAPO [48] is proposed which also uses a recursive gradient update in trust region policy optimization [29]. Our Algorithm 1 differs from their algorithm at least in the following ways: (1) our recursive gradient $\mathbf{v}_t^s$ defined in (3.1) has an importance weight from the snapshot gradient while SARAPO does not; (2) we are optimizing the expected return while [48] optimizes the total advantage over state visitation distribution and actions under Kullback–Leibler divergence constraint; and most importantly (3) there is no convergence or sample complexity analysis for SARAPO.

4 Main Theory

In this section, we present the theoretical analysis of Algorithm 1. We first introduce some common assumptions used in the convergence analysis of policy gradient methods.

Assumption 4.1. Let $\pi_{\boldsymbol{\theta}}(a|s)$ be the policy parametrized by $\boldsymbol{\theta}$. There exist constants $G, M > 0$ such that the gradient and Hessian matrix of $\log \pi_{\boldsymbol{\theta}}(a|s)$ with respect to $\boldsymbol{\theta}$ satisfy $\|\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s)\| \leq G$, $\big\|\nabla_{\boldsymbol{\theta}}^2 \log \pi_{\boldsymbol{\theta}}(a|s)\big\|_2 \leq M$, for all $a \in \mathcal{A}$ and $s \in \mathcal{S}$.

The above boundedness assumption is reasonable since we usually require the policy function to be twice differentiable and easy to optimize in practice. Similarly, in [21], the authors assume 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, which is actually stronger than our Assumption 4.1.

In the following proposition, we show that Assumption4.1 directly implies that the Hessian matrix of the performance function $\nabla^2 J(\boldsymbol{\theta})$ is bounded, which is often referred to as the smoothness assumption and is crucial in analyzing the convergence of nonconvex optimization [25, 1].

Proposition 4.2. Let $g(\tau|\boldsymbol{\theta})$ be the PGT estimator defined in (2.6). Assumption 4.1 implies

(1). $\|g(\tau|\boldsymbol{\theta}_1) - g(\tau|\boldsymbol{\theta}_2)\|_2 \le L\|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|_2, \forall \boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \in \mathbb{R}^d$, with $L = MR/(1-\gamma)^2$.
(2). $J(\boldsymbol{\theta})$ is L-smooth, namely $\|\nabla^2_{\boldsymbol{\theta}} J(\boldsymbol{\theta})\|_2 \le L$.
(3). $\|g(\tau|\boldsymbol{\theta})\|_2 \le C_g$ for all $\boldsymbol{\theta} \in \mathbb{R}^d$, with $C_g = GR/(1-\gamma)^2$.

Similar properties are also proved in [45]. However, in contrast to their results, the smoothness parameter L and the bound on the policy gradient in Proposition 4.2 do not rely on horizon H and are therefore tighter. The next assumption requires that the variance of the gradient estimator is bounded.

Assumption 4.3. There exists a constant $\xi > 0$ such that $\mathrm{Var}\big(g(\tau|\boldsymbol{\theta})\big) \le \xi^2$, for all policy $\pi_{\boldsymbol{\theta}}$.

In Algorithm 1, we have used importance sampling to connect the trajectories between two different iterations. The following assumption ensures that the variance of the importance weight is bounded, which is also made in [21, 45].

Assumption 4.4. Let $\omega(\cdot|\boldsymbol{\theta}_1, \boldsymbol{\theta}_2) = p(\cdot|\boldsymbol{\theta}_1)/p(\cdot|\boldsymbol{\theta}_2)$. There is a constant $W < \infty$ such that for each policy pairs encountered in Algorithm 1, $\mathrm{Var}(\omega(\tau|\boldsymbol{\theta}_1, \boldsymbol{\theta}_2)) \le W$ for all $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \in \mathbb{R}^d, \tau \sim p(\cdot|\boldsymbol{\theta}_2)$.

Now we are ready to present the convergence result of SRVR-PG to a stationary point:

Theorem 4.5. Suppose that Assumptions 4.1, 4.3 and 4.4 hold. In Algorithm 1, we choose the step size $\eta \le 1/(4L)$ and epoch size m and mini-batch size B such that $B \ge 72m\eta G^2(2G^2/M + 1)(W+1)\gamma/(1-\gamma)^3$. Then the generalized projected gradient of the output of Algorithm 1 satisfies

$$\mathbb{E}\big[\big\|\mathcal{G}_\eta\big(\boldsymbol{\theta}_{\text{out}}\big)\big\|_2^2\big] \le 8(J(\boldsymbol{\theta}^*) - J(\boldsymbol{\theta}_0) - \mathbb{1}_{\boldsymbol{\Theta}}(\boldsymbol{\theta}^*) + \mathbb{1}_{\boldsymbol{\Theta}}(\boldsymbol{\theta}_0))/(\eta S m) + 6\xi^2/N,$$

where $\boldsymbol{\theta}^* = \mathrm{argmax}_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} J(\boldsymbol{\theta})$.

Remark 4.6. Recall that S is the number of epochs and m is the epoch length of Algorithm 1. Let $T = Sm$ be the total number of iterations. Theorem 4.5 states that under a proper choice of step size, batch size and epoch length, the squared gradient norm of the output of SRVR-PG is upper bounded by two terms. The first term is in the order of $O(1/T)$, which matches the convergence rate of stochastic variance reduced gradient descent in nonconvex optimization [25]. The second term is in the order of $O(1/N)$ and N is the batch size in the snapshot gradient in Line 6 of SRVR-PG.

Compared with the result in [21] where the squared gradient norm is in the order of $O(1/T + 1/N + 1/B)$, our analysis avoids the additional term that depends on the mini-batch size B within each epoch. Compared with [45], our requirement on the mini-batch size B is also relaxed. This enables us to choose a smaller mini-batch size B while maintaining the same convergence rate. As we will show in the next corollary, this improvement leads to a lower sample complexity.

Corollary 4.7. Under the same conditions as in Theorem 4.5. We set step size as $\eta = 1/(4L)$, the batch sizes in the outer loop and in the inner loop as $N = O(1/\epsilon)$ and $B = O(1/\epsilon^{1/2})$ respectively. Let the epoch length m and the number of epochs S to be in the same order as B. Then Algorithm 1 outputs a point $\boldsymbol{\theta}_{\text{out}}$ that satisfies $\mathbb{E}[\|\mathcal{G}_\eta(\boldsymbol{\theta}_{\text{out}})\|_2^2] \le \epsilon$ within $O(1/\epsilon^{3/2})$ trajectories in total.

Note that the results in [21, 45] are only for $\|\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})\|_2^2 \le \epsilon$, while our result in Corollary 4.7 is more general. In particular, when the policy parameter $\boldsymbol{\theta}$ is defined on the whole space $\mathbb{R}^d$ instead of $\boldsymbol{\Theta}$, our result reduces to the case for $\|\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})\|_2^2 \le \epsilon$ since it holds $\boldsymbol{\Theta} = \mathbb{R}^d$ and $\mathcal{G}_\eta(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$. In [45], the authors improved the sample complexity of SVRPG from $O(1/\epsilon^2)$ to $O(1/\epsilon^{5/3})$ by a more careful analysis of the SVRPG algorithm [21]. According to Corollary 4.7, our new algorithm SRVR-PG only needs $O(1/\epsilon^{3/2})$ number of trajectories to achieve $\|\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})\|_2^2 \le \epsilon$, which is lower than the sample complexity of SVRPG by a factor of $O(1/\epsilon^{1/6})$. This improvement is significant when the required precision ϵ is very small.

5 Conclusions

We propose a novel policy gradient method called SRVR-PG, which is built on a recursively updated stochastic policy gradient estimator. We prove that the sample complexity of SRVR-PG is lower than the sample complexity of the state-of-the-art SVRPG [21, 45] algorithm. Experiments on the classic reinforcement learning benchmarks validate the advantage of our proposed algorithms.

References

[1] Zeyuan Allen-Zhu and Elad Hazan. Variance reduction for faster non-convex optimization. In *International Conference on Machine Learning*, pages 699–707, 2016.

[2] Jonathan Baxter and Peter L Bartlett. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15:319–350, 2001.

[3] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems*, pages 1646–1654, 2014.

[4] Marc Peter Deisenroth, Gerhard Neumann, Jan Peters, et al. A survey on policy search for robotics. *Foundations and Trends® in Robotics*, 2(1–2):1–142, 2013.

[5] Simon S Du, Jianshu Chen, Lihong Li, Lin Xiao, and Dengyong Zhou. Stochastic variance reduction methods for policy evaluation. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1049–1058. JMLR. org, 2017.

[6] Cong Fang, Chris Junchi Li, Zhouchen Lin, and Tong Zhang. Spider: Near-optimal non-convex optimization via stochastic path-integrated differential estimator. In *Advances in Neural Information Processing Systems*, pages 686–696, 2018.

[7] Saeed Ghadimi, Guanghui Lan, and Hongchao Zhang. Mini-batch stochastic approximation methods for nonconvex stochastic composite optimization. *Mathematical Programming*, 155 (1-2):267–305, 2016.

[8] Evan Greensmith, Peter L Bartlett, and Jonathan Baxter. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, 5(Nov): 1471–1530, 2004.

[9] Reza Harikandeh, Mohamed Osama Ahmed, Alim Virani, Mark Schmidt, Jakub Konečnỳ, and Scott Sallinen. Stopwasting my gradients: Practical svrg. In *Advances in Neural Information Processing Systems*, pages 2251–2259, 2015.

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

[11] Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

[12] Vijay R Konda and John N Tsitsiklis. Actor-critic algorithms. In *Advances in Neural Information Processing Systems*, pages 1008–1014, 2000.

[13] Lihua Lei, Cheng Ju, Jianbo Chen, and Michael I Jordan. Non-convex finite-sum optimization via scsg methods. In *Advances in Neural Information Processing Systems*, pages 2348–2358, 2017.

[14] Sergey Levine, Nolan Wagener, and Pieter Abbeel. Learning contact-rich manipulation skills with guided policy search. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 156–163. IEEE, 2015.

[15] Yuxi Li. Deep reinforcement learning: An overview. *CoRR*, abs/1701.07274, 2017. URL http://arxiv.org/abs/1701.07274.

[16] Zhize Li and Jian Li. A simple proximal stochastic gradient method for nonsmooth nonconvex optimization. In *Advances in Neural Information Processing Systems*, pages 5569–5579, 2018.

[17] Yang Liu, Prajit Ramachandran, Qiang Liu, and Jian Peng. Stein variational policy gradient. *CoRR*, abs/1704.02399, 2017. URL http://arxiv.org/abs/1704.02399.

[18] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[19] Lam M Nguyen, Jie Liu, Katya Scheinberg, and Martin Takáč. Sarah: A novel method for machine learning problems using stochastic recursive gradient. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2613–2621. JMLR. org, 2017.

[20] Lam M Nguyen, Marten van Dijk, Dzung T Phan, Phuong Ha Nguyen, Tsui-Wei Weng, and Jayant R Kalagnanam. Optimal finite-sum smooth non-convex optimization with sarah. *CoRR, abs/1901.07648*, 2019. URL http://arxiv.org/abs/1901.07648.

[21] Matteo Papini, Damiano Binaghi, Giuseppe Canonaco, Matteo Pirotta, and Marcello Restelli. Stochastic variance-reduced policy gradient. In *International Conference on Machine Learning*, pages 4023–4032, 2018.

[22] Jan Peters and Stefan Schaal. Natural actor-critic. *Neurocomputing*, 71(7-9):1180–1190, 2008.

[23] Jan Peters and Stefan Schaal. Reinforcement learning of motor skills with policy gradients. *Neural Networks*, 21(4):682–697, 2008.

[24] Matteo Pirotta, Marcello Restelli, and Luca Bascetta. Adaptive step-size for policy gradient methods. In *Advances in Neural Information Processing Systems*, pages 1394–1402, 2013.

[25] Sashank J Reddi, Ahmed Hefny, Suvrit Sra, Barnabas Poczos, and Alex Smola. Stochastic variance reduction for nonconvex optimization. In *International Conference on Machine Learning*, pages 314–323, 2016.

[26] Sashank J Reddi, Suvrit Sra, Barnabas Poczos, and Alexander J Smola. Proximal stochastic methods for nonsmooth nonconvex finite-sum optimization. In *Advances in Neural Information Processing Systems*, pages 1145–1153, 2016.

[27] Alfréd Rényi et al. On measures of entropy and information. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*. The Regents of the University of California, 1961.

[28] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, pages 400–407, 1951.

[29] John Schulman, Sergey Levine, Pieter Abbeel, Michael I Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, volume 37, pages 1889–1897, 2015.

[30] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *CoRR, abs/1506.02438*, 2015. URL https://arxiv.org/abs/1506.02438.

[31] Frank Sehnke, Christian Osendorfer, Thomas Rückstieß, Alex Graves, Jan Peters, and Jürgen Schmidhuber. Policy gradients with parameter-based exploration for control. In *International Conference on Artificial Neural Networks*, pages 387–396. Springer, 2008.

[32] Frank Sehnke, Christian Osendorfer, Thomas Rückstieß, Alex Graves, Jan Peters, and Jürgen Schmidhuber. Parameter-exploring policy gradients. *Neural Networks*, 23(4):551–559, 2010.

[33] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. Safe, multi-agent, reinforcement learning for autonomous driving. *CoRR*, abs/1610.03295, 2016. URL http://arxiv.org/abs/1610.03295.

[34] Zebang Shen, Alejandro Ribeiro, Hamed Hassani, Hui Qian, and Chao Mi. Hessian aided policy gradient. In *International Conference on Machine Learning*, pages 5729–5738, 2019.

[35] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International Conference on Machine Learning*, 2014.

[36] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

[37] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[38] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, pages 1057–1063, 2000.

[39] George Tucker, Surya Bhupatiraju, Shixiang Gu, Richard Turner, Zoubin Ghahramani, and Sergey Levine. The mirage of action-dependent baselines in reinforcement learning. In *International Conference on Machine Learning*, pages 5022–5031, 2018.

[40] Zhe Wang, Kaiyi Ji, Yi Zhou, Yingbin Liang, and Vahid Tarokh. Spiderboost: A class of faster variance-reduced algorithms for nonconvex optimization. *CoRR*, abs/1810.10690, 2018. URL http://arxiv.org/abs/1810.10690.

[41] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

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

[43] Cathy Wu, Aravind Rajeswaran, Yan Duan, Vikash Kumar, Alexandre M Bayen, Sham Kakade, Igor Mordatch, and Pieter Abbeel. Variance reduction for policy gradient with action-dependent factorized baselines. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=H1tSsb-AW.

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

[45] Pan Xu, Felicia Gao, and Quanquan Gu. An improved convergence analysis of stochastic variance-reduced policy gradient. In *International Conference on Uncertainty in Artificial Intelligence*, 2019.

[46] Tianbing Xu, Qiang Liu, and Jian Peng. Stochastic variance reduction for policy gradient estimation. *CoRR*, abs/1710.06034, 2017. URL http://arxiv.org/abs/1710.06034.

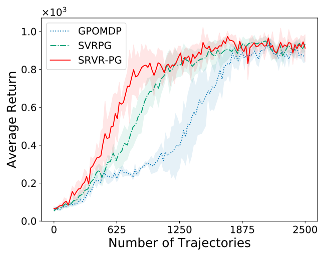
[47] Long Yang and Yu Zhang. Policy optimization with stochastic mirror descent. *arXiv preprint arXiv:1906.10462*, 2019.

[48] Huizhuo Yuan, Chris Junchi Li, Yuhao Tang, and Yuren Zhou. Policy optimization via stochastic recursive gradient algorithm, 2019. URL https://openreview.net/forum?id=rJl3S2A9t7.

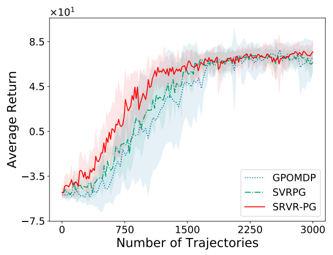
[49] Tingting Zhao, Hirotaka Hachiya, Gang Niu, and Masashi Sugiyama. Analysis and improvement of policy gradient estimation. In *Advances in Neural Information Processing Systems*, pages 262–270, 2011.

[50] Dongruo Zhou, Pan Xu, and Quanquan Gu. Finding local minima via stochastic nested variance reduction. *arXiv preprint arXiv:1806.08782*, 2018.

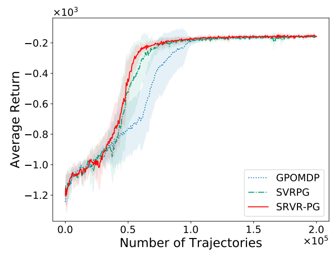
[51] Dongruo Zhou, Pan Xu, and Quanquan Gu. Stochastic nested variance reduced gradient descent for nonconvex optimization. In *Advances in Neural Information Processing Systems*, pages 3922–3933, 2018.



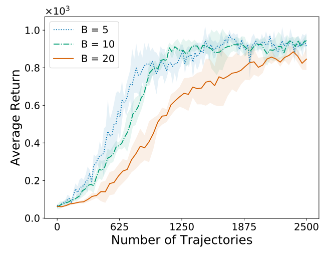
(a) Cartpole



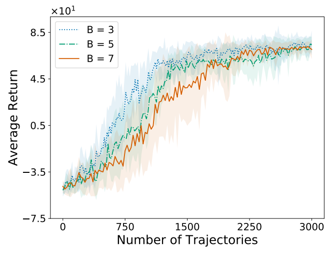
(b) Mountain Car



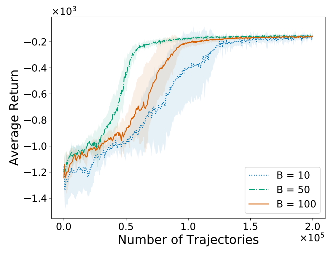
(c) Pendulum



(d) Cartpole



(e) Mountain Car



(f) Pendulum

Figure 1: (a)-(c): Comparison of different algorithms. Experimental results are averaged over 10 repetitions. (d)-(f): Comparison of different batch size B on the performance of SRVR-PG.

A Experiments

In this section, we provide experiment results of the proposed algorithm on benchmark reinforcement learning environments including the Cartpole, Mountain Car and Pendulum problems. In all the experiments, we use the Gaussian policy defined as follows. For bounded action space $\mathcal{A} \subset \mathbb{R}$, a Gaussian policy parametrized by $\boldsymbol{\theta}$ is defined as

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

where σ^2 is a fixed standard deviation parameter and $\phi : \mathcal{S} \mapsto \mathbb{R}^d$ is a mapping from the state space to the feature space.

For baselines, we compare the proposed SRVR-PG algorithm with the most relevant methods: GPOMDP [2] and SVRPG [21]. For the learning rates η in all of our experiments, we use grid search to directly tune η. For instance, we searched η for the Cartpole problem by evenly dividing the interval $[10^{-5}, 10^{-1}]$ into 20 points in the log-space. For the batch size parameters N and B and the epoch length m, according to Corollary 4.7, we choose $N = O(1/\epsilon)$, $B = O(1/\epsilon^{1/2})$ and thus $m = O(1/\epsilon^{1/2})$, where $\epsilon > 0$ is a user-defined precision parameter. In our experiments, we set $N = C_0/\epsilon$, $B = C_1/\epsilon^{1/2}$ and $m = C_2/\epsilon^{1/2}$ and tune the constant parameters C_0, C_1, C_2 using grid search. The details on parameters used in the experiments are presented in Table 2.

We evaluate the performance of different algorithms in terms of the total number of trajectories they require to achieve a certain threshold of cumulative rewards. We run each experiment repeatedly for 10 times and plot the averaged returns with standard deviation. For a given environment, all experiments are initialized from the same random initialization. Figures 1(a), 1(b) and 1(c) show the results on the comparison of GPOMDP, SVRPG, and our proposed SRVR-PG algorithm across three different RL environments. It is evident that, for all environments, GPOMDP is overshadowed by the variance reduced algorithms SVRPG and SRVR-PG significantly. Furthermore, SRVR-PG outperforms SVRPG in all experiments, which is consistent with the comparison on the sample complexity of GPOMDP, SRVRPG and SRVR-PG in Table 1.

Corollary 4.7 suggests that when the mini-batch size B is in the order of $O(\sqrt{N})$, SRVR-PG achieves the best performance. Here N is the number of episodes sampled in the outer loop of Algorithm 1 and B is the number of episodes sampled at each inner loop iteration. To validate our theoretical result, we conduct a sensitivity study to demonstrate the effectiveness of different batch sizes within each epoch of SRVR-PG on its performance. The results on different environments are displayed in Figures 1(d), 1(e) and 1(f) respectively. To interpret these results, we take the Pendulum problem

Table 2: Parameters used in the SRVR-PG experiments.

Parameters	Algorithm	Cartpole	Mountain Car	Pendulum
NN size	-	64	64	8×8
NN activation function	-	Tanh	Tanh	Tanh
Task horizon	-	100	1000	200
Total trajectories	-	2500	3000	2×10^5
Discount factor γ	GPOMDP	0.99	0.999	0.99
	SVRPG	0.999	0.999	0.995
	SRVR-PG	0.995	0.999	0.995
Learning rate η	GPOMDP	0.005	0.005	0.01
	SVRPG	0.0075	0.0025	0.01
	SRVR-PG	0.005	0.0025	0.01
Batch size N	GPOMDP	10	10	250
	SVRPG	25	10	250
	SRVR-PG	25	10	250
Batch size B	GPOMDP	-	-	-
	SVRPG	10	5	50
	SRVR-PG	5	3	50
Epoch size m	GPOMDP	-	-	-
	SVRPG	3	2	1
	SRVR-PG	3	2	1

as an example. In this setting, we choose outer loop batch size N of Algorithm 1 to be $N = 250$. By Corollary 4.7, the optimal choice of batch size in the inner loop of Algorithm 1 is $B = C\sqrt{N}$, where $C > 1$ is a constant depending on horizon H and discount factor γ. Figure 1(f) shows that $B = 50 \approx 3\sqrt{N}$ yields the best convergence results for SRVR-PG on Pendulum, which validates our theoretical analysis and implies that a larger batch size B does not necessarily result in an improvement in sample complexity, as each update requires more trajectories, but a smaller batch size B pushes SRVR-PG to behave more similar to GPOMDP.