Hessian Aided Policy Gradient

Zebang Shen [1] **Hamed Hassani** [2] **Chao Mi** [1] **Hui Qian** [1] **Alejandro Ribeiro** [2]

Abstract

Reducing the variance of estimators for policy gradient has long been the focus of reinforcement learning research. While classic algorithms like REINFORCE find an ϵ-approximate first-order stationary point in $\mathcal{O}(1/\epsilon^4)$ random trajectory simulations, no provable improvement on the complexity has been made so far. This paper presents a Hessian aided policy gradient method with the first improved sample complexity of $\mathcal{O}(1/\epsilon^3)$. While our method exploits information from the policy Hessian, it can be implemented in linear time with respect to the parameter dimension and is hence applicable to sophisticated DNN parameterization. Simulations on standard tasks validate the efficiency of our method.

1. Introduction

A Markov Decision Process (MDP) is determined by time-varying system states, actions to affect the transition probability between states, and *instantaneous* rewards collected as a function of the state visited and the action taken (Puterman, 2014). Whenever the system visits a particular state, the agent chooses an action that is dictated by a (possibly stochastic) policy. As the agent moves from state to state, it collects an *aggregate* reward given by a discounted sum of the instantaneous rewards. The optimal policy of an MDP is the one that maximizes such aggregate reward. The focus of this paper is to find the optimal policy in *model-free* reinforcement learning (RL) problems where transition probabilities and rewards are unknown and can only be estimated by probing the system through the execution of policy actions (Sutton and Barto, 2018).

Policy gradient methods and its variants constitute the set of tools that are most widely used for finding good policies in MDPs (Williams, 1992; Sutton et al., 2000; Baxter and Bartlett, 2001). Such methods directly find the optimal policy through the use of stochastic first-order differentials of the accumulated reward relative to policy variations. Their popularity notwithstanding, they are known to have low sample efficiency demanding hundreds of thousands of Monte Carlo simulations of full trajectories even when dealing with relatively simple MDPs. Large sample complexity is inherent to MDPs because to evaluate individual stochastic policy gradients we need to run a full trajectory under the current policy. Such trajectories are composed of large number of individual transitions which not only makes them costly to simulate but also results in gradient estimates with large variances as the randomness of individual transitions is compounded over the trajectory's time horizon (Williams, 1992; Sutton, 1984; Arulkumaran et al., 2017; Sutton and Barto, 2018).

Besides the aforementioned MDP-specific challenges, policy gradient inherits the limitations of any stochastic gradient descent method used for optimizing a nonconvex function. This manifests in the need for $\mathcal{O}(1/\epsilon^4)$ stochastic first-order queries to find an ϵ-approximate first order stationary point (ϵ-FOSP) (Nesterov, 2013). In the supervised learning literature this slow convergence is alleviated with variance reduction techniques which exploit correlations between consecutive stochastic gradient estimators (Roux et al., 2012; Johnson and Zhang, 2013; Defazio et al., 2014; Nguyen et al., 2017; Fang et al., 2018). Give the dramatic impact of VR on the SL, we should expect it to be even more effective in reinforcement learning where policy gradients are prone to larger variances.

However, direct transplanting the variance reduction techniques tailored for supervised learning does not reduced the $\mathcal{O}(1/\epsilon^4)$ sample complexity (Papini et al., 2018). This happens because variance reduction techniques make explicit use of the fact that the randomness in the objective function is *oblivious* to the argument, i.e., the randomness that affects the choice of a function does not depend the variables. This is not true for RL problems in which the transition probabilities themselves are controlled by the policy. Consequently, when applying variance reduction techniques to these *non-oblivious* objectives, the improvements in convergence rates that are obtained in supervised learning do not materialize.

Contributions. In this paper, we develop a Hessian-aided variance reduction method that is applicable to non-oblivious objectives. In particular, we give the *first provable* sample-efficiency improvement over stochastic gradient based policy-gradient type methods: We reduce the

[1]Zhejiang University [2]University of Pennsylvania.

trajectory complexity from $\mathcal{O}(1/\epsilon^4)$ to $\mathcal{O}(1/\epsilon^3)$ to obtain an ϵ-FOSP. To do so, we construct a novel, efficiently computable, and unbiased variance reduced gradient estimator for non-oblivious objectives by integrating carefully designed Hessian-vector products along the solution path. Our estimator utilizes the stochastic approximation of the second-order policy differential without compromising the *linear* per-iteration computation complexity and is hence suitable for complex and high dimensional parameterizations. Additionally, existing RL techniques like actor-critic and GAE can be seamlessly integrated to our algorithm. We also provide extensive simulations on various reinforcement learning tasks to validate our analysis and illustrate the efficiency of our method.

Notation. Lowercase boldface $\mathbf{v}$ denotes a vector and uppercase boldface $\mathbf{A}$ denotes a matrix. We use $\|\mathbf{v}\|$ to denote the Euclidean norm of vector $\mathbf{v}$ and use $\|\mathbf{A}\|$ to denote the spectral norm of matrix $\mathbf{A}$. $\mathbb{E}[X]$ and $\mathbb{V}[X]$ denote the expectation and variance of a random variable X.

2. Preliminaries

Consider a discrete time index $h \geq 0$ and a Markov system with time varying states $s_h \in \mathcal{S}$ and actions $a_h \in \mathcal{A}$. The probability distribution of the initial state is $\rho(s_0)$ and the conditional probability distribution of transitioning into s_{h+1} given that we are in state s_h and take action a_h is $\mathcal{P}(s_{h+1}|s_h, a_h)$. Actions are chosen according to a possibly random policy π in which $\pi(a_h|s_h)$ is the distribution for taking action a_h when observing state s_h. We assume policies are parametrized by a vector $\theta \in \mathbb{R}^d$ and use π_θ as a shorthand for the conditional distribution $\pi(a_h|s_h;\theta)$ associated to θ. For a given time horizon H we define the trajectory $\tau := (s_1, a_1, \ldots, s_H, a_H)$ as the collection of state action pairs experienced up until time $h = H$. Given the initial distribution $\rho(s_0)$, the transition kernel $\mathcal{P}(s_{h+1}|s_h, a_h)$, and the Markov property of the system, it follows that the probability distribution over trajectories τ is

$$p(\tau;\pi_\theta) := \rho(s_0) \prod_{h=1}^{H} \mathcal{P}(s_{h+1}|s_h, a_h)\, \pi(a_h|s_h). \quad (1)$$

Associated with a state action pair we have a reward function $r(s_h, a_h)$. When following a trajectory $\tau := (s_1, a_1, \ldots, s_H, a_H)$, we consider the accumulated reward discounted by a geometric factor γ

$$\mathcal{R}(\tau) := \sum_{h=1}^{H} \gamma^h r(s_h, a_h). \quad (2)$$

Our goal is to find the policy parameter θ that maximizes the expected discounted trajectory reward

$$\max_{\theta\in\mathbb{R}^d} J(\theta) := \mathbb{E}_\tau[\mathcal{R}(\tau)] = \int \mathcal{R}(\tau) p(\tau;\pi_\theta) \mathbf{d}\tau. \quad (3)$$

Unlike traditional supervised learning problems, the underlying distribution p depends on the variable θ and hence varies through the whole optimization procedure. We refer to such property as *non-oblivious*.

Due to the non-convexity of (3), we are satisfied with finding an ϵ-approximate First-Order Stationary Point (ϵ-FOSP), denoted by θ_ϵ, such that

$$\|\nabla J(\theta_\epsilon)\| \leq \epsilon. \quad (4)$$

As a standard tool to achieve such goal, *policy gradient*, a.k.a. the first-order differential of the objective (3), can be expressed as

$$\nabla J(\theta) := \mathbb{E}_{\tau\sim p(\tau;\pi_\theta)}[\sum_{h=1}^{H} \Psi_h(\tau) \nabla \log \pi_\theta(a_h|s_h)], \quad (5)$$

where we denote $\Psi_h(\tau) := \sum_{i=h}^{H} \gamma^i r(s_i, a_i)$ for a fixed trajectory τ. Let $\mathcal{M}$ be a set of random trajectories sampled according to distribution $p(\cdot;\pi_\theta)$. An unbiased stochastic policy-gradient estimator can be constructed by

$$\mathbf{g}(\theta;\mathcal{M}) := \frac{1}{|\mathcal{M}|} \sum_{\tau\in\mathcal{M}} \sum_{h=1}^{H} \Psi_h(\tau) \nabla \log \pi_\theta(a_h|s_h). \quad (6)$$

2.1. Variance Reduced Gradient Estimator

Most of the literature on variance reduction techniques focuses on the *oblivious* setting. For example, supervised learning is usually characterized through the following stochastic optimization framework

$$\max_{\mathbf{x}\in\mathbb{R}^d} F(\mathbf{x}) := \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z})}[f(\mathbf{x};\mathbf{z})], \quad (7)$$

where the samples $\mathbf{z} \in \mathcal{Z}$ have distribution $q(\mathbf{z})$ and the functions $f(;\mathbf{z}) : \mathbb{R}^d \to \mathbb{R}$ are smooth, potentially non-convex loss functions with respect to sample $\mathbf{z}$. Since the underlying distribution $q(\mathbf{z})$ is invariant to the variable $\mathbf{x}$, we refer to (7) as an *oblivious* objective.

For oblivious objectives, a recent method called *Variance Reduced Gradient Estimator* has been proposed to reduce sample complexity of vanilla stochastic gradient methods with provable guarantees (Reddi et al., 2016; Fang et al., 2018; Zhou et al., 2018). Let $\tilde{\mathbf{x}}$ be some reference point and $\tilde{\mathbf{h}}$ be an unbiased gradient estimator at $\tilde{\mathbf{x}}$. Given $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{h}}$, the variance-reduced gradient estimator at a point $\mathbf{x}^t$, which we denote by $\mathbf{h}^t_{vr}$, is of the form:

$$\mathbf{h}^t_{vr} := \tilde{\mathbf{h}} + \mathbf{h}(\mathbf{x}^t;\mathcal{M}) - \mathbf{h}(\tilde{\mathbf{x}};\mathcal{M}), \quad (8)$$

where $\mathcal{M}$ is a set of samples drawn from $q(\mathbf{z})$ and

$$\mathbf{h}(\mathbf{x};\mathcal{M}) := \frac{1}{|\mathcal{M}|} \sum_{\mathbf{z}\in\mathcal{M}} \nabla f(\mathbf{x};\mathbf{z}). \quad (9)$$

Algorithm 1 Hessian Aided Policy Gradient (HAPG)

for $t = 0$ **to** T **do**
  if $\mod (t, p) = 0$ **then**
    Sample $|\mathcal{M}_0|$ trajectories to compute minibatch stochastic policy gradient estimator at θ^t

$$\mathbf{g}^t = \frac{1}{|\mathcal{M}_0|} \sum_{\tau \in \mathcal{M}_0} \sum_{h=1}^{H} \Psi_h(\tau) \cdot \nabla \log \pi_{\theta^t}(a_h; s_h);$$

  else
    Sample $|\mathcal{M}|$ random tuples $(a, \tau(a))$ to construct gradient difference estimator Δ^t using (15)
    $\mathbf{g}^t = \mathbf{g}^{t-1} + \Delta^t$;
  end if
  $\theta^{t+1} = \theta^t + \eta \cdot \mathbf{g}^t / \|\mathbf{g}^t\|$;
end for

Note that $\mathbf{h}(\mathbf{x}^t; \mathcal{M})$ and $\mathbf{h}(\tilde{\mathbf{x}}; \mathcal{M})$ use the same sample batch $\mathcal{M}$, which is critical for reducing variance. The formulation (8) captures several variance-reduction techniques such as the seminal SVRG estimator (Johnson and Zhang, 2013) and the recent SPIDER estimator (Fang et al., 2018).

While it is tempting to generalize the above-mentioned variance reduction technique to reinforcement learning (i.e. the non-oblivious setting), there has been no provable success. For example, the Stochastic Variance Reduced Policy Gradient (SVRPG) from (Papini et al., 2018) incorporates such technique in its gradient estimator design, but still requires the same amount of random trajectories as the vanilla stochastic policy-gradient type method, i.e. $\mathcal{O}(\frac{1}{\epsilon^4})$, to achieve an ϵ-FOSP (4).

3. Methodology

In this section, we derive our Hessian Aided Policy Gradient (HAPG) method for the non-oblivious and non-convex objective (3). HAPG is the first algorithm to achieve provably better sample efficiency than the policy-gradient type method, from $\mathcal{O}(1/\epsilon^4)$ to $\mathcal{O}(1/\epsilon^3)$. Note that while our method conceptually utilizes curvature information, HAPG can be implemented in linear time in terms of the parameter dimension d, without explicitly computing the Hessian matrix.

Suppose we are given a variable sequence $\{\theta^s\}_{s=0}^t$. Then the gradient at θ^t can be written in a path-integral form:

$$\nabla J(\mathbf{x}^t) = \nabla J(\theta^0) + \sum_{s=1}^{t} \nabla J(\theta^s) - \nabla J(\theta^{s-1}). \quad (10)$$

Let Δ^s be an unbiased estimator for the gradient difference $\nabla J(\theta^s) - \nabla J(\theta^{s-1})$ and recall the policy gradient defined in (6). We can recursively construct the following estimator for $\nabla J(\theta^t)$:

$$\mathbf{g}^t = \begin{cases} \mathbf{g}(\theta^t; \mathcal{M}_0) & \mod (t, p) = 0, \\ \mathbf{g}^{t-1} + \Delta^t & \mod (t, p) \neq 0. \end{cases} \quad (11)$$

where $\mathcal{M}_0$ is a mini-batch of trajectories sampled according to $p(\cdot|\pi_\theta)$ and p is a given epoch length. In words, after every p iterations, we directly estimate the gradient using a min-batch $\mathcal{M}_0$ of stochastic trajectories, and in between, we maintain an unbiased estimate by recursively adding the correction term Δ^t to the current estimate $\mathbf{g}^{t-1}$.

Having estimator (11), we will use $\mathbf{g}^t$ to update θ^{t+1} in a *normalized* gradient ascent manner:

$$\theta^{t+1} := \theta^t + \eta^t \cdot \frac{\mathbf{g}^t}{\|\mathbf{g}^t\|}. \quad (12)$$

Our method is presented in Algorithm 1.

We now focus on the construction of Δ^t. From the Taylor's expansion, the gradient difference can be written as

$$\begin{aligned} \nabla J(\theta^t) - \nabla J(\theta^{t-1}) &= \int_0^1 [\nabla^2 J(\theta(a)) \cdot \mathbf{v}] \mathbf{d}a \\ &= \left[\int_0^1 \nabla^2 J(\theta(a)) \mathbf{d}a \right] \cdot \mathbf{v}, \quad (13) \end{aligned}$$

where we denote $\theta(a) := a \cdot \theta^t + (1-a) \cdot \theta^{t-1}$ and $\mathbf{v} := \theta^t - \theta^{t-1}$. Note that the integral in (13) is just the expectation $\mathbb{E}_a[\nabla^2 J(\theta(a))]$, which admits the unbiased estimator $\nabla^2 J(\theta(\bar{a}))$ with $\bar{a}$ uniformly sampled from $[0, 1]$. Therefore, if we have an unbiased estimator of $\nabla^2 J(\theta)$ for arbitrary θ, we can construct Δ^t. To do so, note that the policy Hessian $\nabla^2 J(\theta)$ can be expressed by[1]

$$\nabla^2 J(\theta) = \mathbb{E}_\tau \left[\nabla \Phi(\theta; \tau) \nabla \log p(\tau; \pi_\theta)^\top + \nabla^2 \Phi(\theta; \tau) \right],$$

where the trajectory τ has the distribution $p(\tau; \pi_\theta)$ and

$$\Phi(\theta; \tau) = \sum_{h=1}^{H} \sum_{i=h}^{H} \gamma^i r(s_i, a_i) \log \pi_\theta(a_h | s_h).$$

Hence, we can construct an unbiased estimator $\tilde{\nabla}^2(\theta; \tau)$ for $\nabla^2 J(\theta)$ by sampling τ according to $p(\tau; \pi_\theta)$ and let

$$\tilde{\nabla}^2(\theta; \tau) := \nabla \Phi(\theta; \tau) \nabla \log p(\tau; \pi_\theta)^\top + \nabla^2 \Phi(\theta; \tau). \quad (14)$$

Putting things together, let $\mathcal{M}$ be a mini-batch of random tuple $(a, \tau(a))$ where $a \in \mathbb{R}$ is uniformly distributed over $[0, 1]$ and $\tau(a)$ is a trajectory sampled according to distribution $p(\tau(a); \pi_{\theta(a)})$ (recall $\theta(a) := a \cdot \theta^t + (1-a) \cdot \theta^{t-1}$). We have the construction of Δ^t by

$$\Delta^t := \tilde{\nabla}^2(\theta; \mathcal{M}) \cdot \mathbf{v}, \quad (15)$$

[1] A detailed derivation is provided in the appendix.

where $\tilde{\nabla}^2(\theta;\mathcal{M})$ is a minibatch version of (14) defined by

$$\tilde{\nabla}^2(\theta;\mathcal{M}) := \frac{1}{|\mathcal{M}|}\sum_{(a,\tau(a))\in\mathcal{M}} \tilde{\nabla}^2(\theta(a);\tau(a)), \quad (16)$$

and $\mathbf{v} = \theta^t - \theta^{t-1}$.

Let us reemphasize that the estimator (15) can be computed in linear time in terms of the parameter dimension d: First note that computing Δ^t is equivalent to computing $|\mathcal{M}|$ matrix-vector product $\tilde{\nabla}^2(\theta;\tau)\cdot\mathbf{v}$. From (14) we can write

$$\tilde{\nabla}^2(\theta;\tau)\cdot\mathbf{v} = (\nabla \log p(\tau;\pi_\theta)^\top \mathbf{v})\nabla\Phi(\theta;\tau) + \nabla^2\Phi(\theta;\tau)\cdot\mathbf{v}.$$

Clearly, the first term can be computed in time $\mathcal{O}(Hd)$. Additionally, the second term is a Hessian-vector product, and can be computed either with Pearlmutter's algorithm (Pearlmutter, 1994) or using finite difference (Wright and Nocedal, 1999) in $\mathcal{O}(Hd)$ time as discussed later in Section 3.1.

Remark 3.1. *The choice of $\Psi_h(\tau) = \sum_{i=h}^H \gamma^i r(s_i,a_i)$ is for deriving a tight bound on the norm of the stochastic policy gradient and policy Hessian. In practice, we can incorporate a state-dependent baseline into $\Psi_h(\tau)$ to further improve the performance, see (31) in the Experiment section. In that case, the $\mathcal{O}(1/\epsilon^3)$ trajectory complexity still holds as the estimator $\mathbf{g}$ in (6) remains unbiased.*

Remark 3.2. *Note that estimator (8) for the oblivious loss (7) shares the same spirit of (11) except that it estimates $\nabla F(\mathbf{x}^i)$ and $\nabla F(\mathbf{x}^{i-1})$ separately by $\mathbf{h}(\mathbf{x}^t;\mathcal{M})$ and $\mathbf{h}(\mathbf{x}^{t-1};\mathcal{M})$ (taking $\tilde{\mathbf{x}} = \mathbf{x}^{t-1}$). However, such separated estimation is biased in reinforcement learning: Let $\mathcal{M}$ be a minibatch of trajectories sampled from $p(\cdot|\pi_{\theta^t})$ and recall the definition of $\mathbf{g}(\theta;\mathcal{M})$ in (6). We have $\mathbb{E}\mathbf{g}(\theta^t;\mathcal{M}) = \nabla J(\theta^t)$ but $\mathbb{E}\mathbf{g}(\tilde{\theta};\mathcal{M}) \neq \nabla J(\tilde{\theta})$.*

3.1. Finite Difference for Hessian-Vector Product

In this section, we briefly describe the finite difference method for computing the Hessian-vector product. Let $\phi : \mathbb{R}^d \to \mathbb{R}$ be a twice differential function. Our goal is to compute, for arbitrary $\theta, \mathbf{v} \in \mathbb{R}$, $\nabla^2\phi(\theta)\cdot\mathbf{v}$. From the mean value theorem, we have for $\epsilon > 0$

$$\xi_\epsilon(\mathbf{v};\phi) := \frac{\nabla\phi(\theta+\epsilon\mathbf{v}) - \nabla\phi(\theta-\epsilon\mathbf{v})}{2\epsilon} = \nabla^2\phi(\theta(\epsilon))\cdot\mathbf{v},$$

where $\theta(\epsilon) = \theta + a(\epsilon)\cdot\mathbf{v}$ with $a \in [-\epsilon,\epsilon]$. By taking ϵ sufficiently small and assuming the second-order smoothness of ϕ, i.e.

$$\|\nabla^2\phi(\mathbf{x}) - \nabla^2\phi(\mathbf{y})\| \le L_2\|\mathbf{x}-\mathbf{y}\|,$$

for arbitrary $\mathbf{x},\mathbf{y}\in\mathbb{R}^d$, we bound

$$\|\xi_\epsilon(\mathbf{v};\phi) - \nabla^2\phi(\theta)\cdot\mathbf{v}\| \le L_2\|\mathbf{v}\|\epsilon, \quad (17)$$

which can be made arbitrarily small by taking a sufficiently small ϵ. Note that the complexity of computing $\xi_\epsilon(\mathbf{v};\phi)$ is twice the complexity of evaluating the gradient of $\nabla\phi(\cdot)$. Therefore, we can approximate $\nabla^2\Phi(\theta;\tau)\cdot\mathbf{v}$ to arbitrary accuracy by employing $\xi_\epsilon(\mathbf{v};\Phi)$ within time $\mathcal{O}(Hd)$.

4. Convergence Analysis

We prove the convergence of Algorithm 1 and analyze the trajectory complexity to find an ϵ-FOSP under the following assumptions.

Assumption 4.1 (bounded reward)**.** *The instantaneous reward function is bounded, i.e., for all $a\in\mathcal{A}$ and $s\in\mathcal{S}$,*

$$|r(a|s)| \le R. \quad (18)$$

Assumption 4.2 (parameterization regularity)**.** *For any choice of parameter θ and state-action pair (s,a), we have*

$$\|\nabla\log\pi(a|s;\theta)\| \le G \quad \text{and} \quad \|\nabla^2\log\pi(a|s;\theta)\| \le L.$$

Remark 4.1. *We note that Assumptions 4.1 and 4.2 are standard in the literature and are used in recent work like (Papini et al., 2018).*

These two assumptions imply the following technical lemma that characterizes the properties of the stochastic approximations $\mathbf{g}(\theta;\{\tau\})$ (6) and $\tilde{\nabla}^2(\theta;\tau)$ (14) for first and second order differential of $J(\theta)$.

Lemma 4.1 (properties of stochastic differential estimators)**.** *Under Assumption 4.1 and 4.2, we have for all θ*

$$\begin{aligned}\|\mathbf{g}(\theta;\{\tau\}) - \nabla J(\theta)\|^2 &\le \frac{G^2R^2}{(1-\gamma)^4} := G_g^2,\\ \|\tilde{\nabla}^2(\theta;\tau)\|^2 &\le \frac{H^2G^4R^2 + L^2R^2}{(1-\gamma)^4} := G_H^2,\end{aligned} \quad (19)$$

where τ is a trajectory sampled according to $p(\tau;\theta)$.

Proof. **Bound for stochastic first-order differential:** Recall the unbiasedness of $\mathbf{g}(\theta;\{\tau\})$. Using $\mathbb{E}[(X - \mathbb{E}[X])^2] \le \mathbb{E}[X^2]$ for all random variable X, we have

$$\mathbb{E}_\tau\|\mathbf{g}(\theta;\{\tau\}) - \nabla J(\theta)\|^2 \le \mathbb{E}_\tau\|\mathbf{g}(\theta;\{\tau\})\|^2.$$

Use the definition of $\mathbf{g}(\theta;\{\tau\})$ to obtain

$$\begin{aligned}\|\mathbf{g}(\theta;\tau)\| =& \|\sum_{h=1}^H \Psi_h(\tau)\nabla\log\pi_\theta(a_h|s_h)\|\\ \le& \sum_{h=1}^H |\Psi_h(\tau)|\cdot\|\nabla\log\pi_\theta(a_h|s_h)\|,\\ \le& G\sum_{h=1}^H |\Psi_h(\tau)|,\end{aligned}$$

where we use triangle inequality in the first inequality and Assumption 4.2 in the second one. On the other hand, using the definition of Ψ_h, we derive

$$|\Psi_h| = |\sum_{i=h}^{H} \gamma^i r(s_i, a_i)| \leq \sum_{i=h}^{H} \gamma^i R \leq \frac{R\gamma^h}{1-\gamma},$$

where the first inequality uses Assumption 4.1 and the second uses the summation structure of geometric sequence. Combining these two bounds, we obtain

$$\|\mathbf{g}(\theta; \{\tau\})\| \leq \frac{GR}{(1-\gamma)} \sum_{h=1}^{H} \gamma^h \leq \frac{GR}{(1-\gamma)^2}.$$

Consequently, we have

$$\mathbb{E}_\tau \|\mathbf{g}(\theta; \tau) - \nabla J(\theta)\|^2 \leq \frac{G^2 R^2}{(1-\gamma)^4}.$$

Bound for stochastic second-order differential: Recall the definition of $\tilde{\nabla}^2(\theta; \tau)$ (14). Using $\|\mathbf{A} + \mathbf{B}\|^2 \leq 2\|\mathbf{A}\|^2 + 2\|\mathbf{B}\|^2$ we bound

$$\begin{aligned}\mathbb{E}_\tau \|\tilde{\nabla}^2(\theta; \tau)\|^2 \leq & 2\mathbb{E}_\tau \|\nabla \Phi(\theta; \tau)\|^2 \|\nabla \log p(\tau; \pi_\theta)\|^2 \\ & + 2\mathbb{E}_\tau \|\nabla^2 \Phi(\theta; \tau)\|^2.\end{aligned}$$

To bound the first product, use $\nabla \Phi(\theta; \tau) = \mathbf{g}(\theta; \{\tau\})$ so

$$\|\nabla \Phi(\theta; \tau)\| = \|\mathbf{g}(\theta; \{\tau\})\| \leq \frac{GR}{(1-\gamma)^2}.$$

Additionally, use

$$\|\nabla \log p(\tau; \pi_\theta)\| \leq \sum_{h=1}^{H} \|\nabla \log \pi(a_h | s_h; \theta)\| \leq HG, \tag{20}$$

so that the first term is bounded by

$$\|\nabla \Phi(\theta; \tau)\|^2 \|\nabla \log p(\tau; \pi_\theta)\|^2 \leq \frac{H^2 G^4 R^2}{(1-\gamma)^4}. \tag{21}$$

The second term is bounded by

$$\begin{aligned}\|\nabla^2 \Phi(\theta; \tau)\| &= \|\sum_{h=1}^{H} \Psi_h(\tau) \nabla^2 \log \pi_\theta(a_h | s_h)\| \\ &\leq \sum_{h=0}^{H-1} |\Psi_h(\tau)| \|\nabla^2 \log \pi(a_h | s_h; \theta)\|.\end{aligned} \tag{22}$$

Use the bound on $|\Psi_h(\tau)|$ and Assumption 4.2 to arrive at

$$\|\nabla^2 \Phi(\theta; \tau)\| \leq \frac{LR}{1-\gamma} \sum_{h=0}^{H-1} \gamma^h \leq \frac{LR}{(1-\gamma)^2}. \tag{23}$$

Combining these two bounds, we have

$$\mathbb{E}_\tau \|\tilde{\nabla}^2(\theta; \tau)\|^2 \leq \frac{H^2 G^4 R^2 + L^2 R^2}{(1-\gamma)^4}.$$

□

Having Lemma 4.1, we are now ready to bound the variance of the unbiased gradient estimator $\mathbf{g}^t$.

Lemma 4.2 (variance bound for gradient estimator)**.** *Recall the definition of $\mathbf{g}^t$ from (11). By setting $p = \frac{G_g}{G_H \epsilon}$, $|\mathcal{M}_h| = \frac{G_g}{G_H \epsilon}$, and $|\mathcal{M}_0| = \frac{G_g^2}{G_H^2 \epsilon^2}$, we have*

$$\mathbb{E}\|\mathbf{g}^t - \nabla J(\theta^t)\|^2 \leq 2G_H^2 \epsilon^2. \tag{24}$$

Proof. For ease of presentation, we consider $t < p$. The general case is a direct extension. Recall the definition of $\mathbf{g}^t$ and use its unbiasedness to obtain

$$\begin{aligned}&\mathbb{E}\|\mathbf{g}^t - \nabla J(\theta^t)\|^2 \\ =&\mathbb{E}\|\tilde{\nabla}^2(\theta; \mathcal{M})[\theta^t - \theta^{t-1}] + \mathbf{g}^{t-1} - \nabla J(\theta^t)\|^2 \\ =&\mathbb{V}\|\tilde{\nabla}^2(\theta; \mathcal{M})[\theta^t - \theta^{t-1}]\| + \mathbb{E}\|\mathbf{g}^{t-1} - \nabla J(\theta^{t-1})\|^2\end{aligned}$$

To bound the first term, use $\mathbb{V}[\frac{1}{n}\sum_{i=1}^n X_i] \leq \frac{1}{n}\mathbb{E}[X_1]$ for i.i.d. random variables $\{X_i\}_{i=1}^n$ to obtain

$$\begin{aligned}&\mathbb{V}\|\tilde{\nabla}^2(\theta; \mathcal{M})[\theta^t - \theta^{t-1}]\| \\ \leq&\frac{1}{|\mathcal{M}|}\mathbb{E}\|\tilde{\nabla}^2(\theta(a); \tau(a))[\theta^t - \theta^{t-1}]\|^2 \\ \leq&\frac{1}{|\mathcal{M}|}\mathbb{E}\|\tilde{\nabla}^2(\theta(a); \tau(a))\|^2 \cdot \|\theta^t - \theta^{t-1}\|^2 = \frac{G_H^2 \epsilon^2}{|\mathcal{M}_h|},\end{aligned}$$

where we use in the last equality $\|\theta^{t+1} - \theta^t\| = \epsilon$, due to the normalized update (12) and the step-size choice $\eta^t = \epsilon$. Consequently we have the recursion

$$\mathbb{E}\|\mathbf{g}^t - \nabla J(\theta^t)\|^2 \leq \frac{G_H^2 \epsilon^2}{|\mathcal{M}_h|} + \mathbb{E}\|\mathbf{g}^{t-1} - \nabla J(\theta^{t-1})\|^2.$$

By repeating the above recursion t times, we obtain

$$\begin{aligned}\mathbb{E}\|\mathbf{g}^t - \nabla J(\theta^t)\|^2 &\leq \frac{t \cdot G_H^2 \epsilon^2}{|\mathcal{M}_h|} + \mathbb{E}\|\mathbf{g}^0 - \nabla J(\theta^0)\|^2 \\ &\leq \frac{p \cdot G_H^2 \epsilon^2}{|\mathcal{M}_h|} + \frac{G_g^2}{|\mathcal{M}_0|}.\end{aligned}$$

By setting $p = \frac{G_g}{G_H \epsilon}$, $|\mathcal{M}_h| = \frac{G_g}{G_H \epsilon}$, and $|\mathcal{M}_0| = \frac{G_g^2}{G_H^2 \epsilon^2}$, the result of the theorem folows. □

Theorem 4.1. *Recall the definition of smoothness parameter G_g and G_H in Lemma 4.1. Under Assumptions 4.1,4.2, and by setting $p = \frac{G_g}{G_H \epsilon}$, $|\mathcal{M}_h| = \frac{G_g}{G_H \epsilon}$, $|\mathcal{M}_0| = \frac{G_g^2}{G_H^2 \epsilon^2}$, $T = \frac{2(J(\theta^0) - J^*)}{G_H \cdot \epsilon^2}$ in Algorithm 1, we have*

$$\mathbb{E}\|\nabla J(\theta^{\bar{t}})\| \leq 4G_H \epsilon, \tag{25}$$

where $\bar{t}$ is uniformly sampled from $\{0, \ldots, T-1\}$.

Proof. Lemma 4.1 implies that $J(\theta)$ is G_H-smooth, i.e.

$$\|\nabla^2 J(\theta)\| = \|\mathbb{E}_\tau \mathbf{H}(\theta;\tau)\| \le \mathbb{E}_\tau \|\mathbf{H}(\theta;\tau)\| \le G_H. \quad (26)$$

We can thus write

$$\begin{aligned} J(\theta^{t+1}) \ge& J(\theta^t) + \langle \nabla J(\theta^t), \theta^{t+1} - \theta^t \rangle - \frac{G_H}{2}\|\theta^{t+1} - \theta^t\|^2 \\ =& J(\theta^t) + \langle \mathbf{g}^t, \theta^{t+1} - \theta^t \rangle - \frac{G_H}{2}\|\theta^{t+1} - \theta^t\|^2 \\ &+ \langle \nabla J(\theta^t) - \mathbf{g}^t, \theta^{t+1} - \theta^t \rangle. \end{aligned}$$

Plug in $\|\theta^{t+1} - \theta^t\| = \epsilon$ and $\theta^{t+1} - \theta^t = \frac{\epsilon \mathbf{g}^t}{\|\mathbf{g}^t\|}$ to obtain

$$\begin{aligned} J(\theta^{t+1}) \ge& J(\theta^t) + \epsilon\|\mathbf{g}^t\| - \frac{G_H \epsilon^2}{2} + \langle \nabla J(\theta^t) - \mathbf{g}^t, \frac{\epsilon \mathbf{g}^t}{\|\mathbf{g}^t\|} \rangle \\ \ge& J(\theta^t) + \epsilon\|\mathbf{g}^t\| - G_H\epsilon^2 - \frac{1}{2G_H}\|\nabla J(\theta^t) - \mathbf{g}^t\|^2, \end{aligned}$$

where we use Young's inequality in the second inequality. Using the triangle inequality, we have

$$\begin{aligned} J(\theta^{t+1}) \ge& J(\theta^t) + \epsilon(\|\nabla J(\theta^t)\| - \|\nabla J(\theta^t) - \mathbf{g}^t\|) - G_H\epsilon^2 \\ &- \frac{1}{2G_H}\|\nabla J(\theta^t) - \mathbf{g}^t\|^2. \end{aligned} \quad (27)$$

Take expectation on both sides of (27) and rearrange terms to obtain:

$$\begin{aligned} &\epsilon \cdot \mathbb{E}[\|\nabla J(\theta^t)\|] \\ \le& \mathbb{E}[J(\theta^{t+1})] - \mathbb{E}[J(\theta^t)] + G_H\epsilon^2 \\ &+ \frac{1}{2G_H}\mathbb{E}[\|\nabla J(\theta^t) - \mathbf{g}^t\|^2] + \epsilon \cdot \mathbb{E}[\|\nabla J(\theta^t) - \mathbf{g}^t\|] \\ \le& \mathbb{E}[J(\theta^{t+1})] - \mathbb{E}[J(\theta^t)] + G_H\epsilon^2 \\ &+ \frac{1}{2G_H}\mathbb{E}[\|\nabla J(\theta^t) - \mathbf{g}^t\|^2] + \epsilon \cdot \sqrt{\mathbb{E}[\|\nabla J(\theta^t) - \mathbf{g}^t\|^2]} \\ \le& \mathbb{E}[J(\theta^{t+1})] - \mathbb{E}[J(\theta^t)] + 4G_H\epsilon^2, \end{aligned} \quad (28)$$

where we use the Jensen's inequality in the second inequality and Lemma (4.2) in the third one. Sum (28) from $t = 0$ to $T-1$ and divide by $T\epsilon$ on both sides to arrive at

$$\begin{aligned} \frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla J(\theta^t)\| \le& \frac{\mathbb{E}[J(\theta^T)] - J(\theta^0)}{T\epsilon} + 4G_H\epsilon \\ \le& \frac{J^* - J(\theta^0)}{T\epsilon} + 4G_H\epsilon. \end{aligned} \quad (29)$$

We have the result by taking $T = \frac{2(J^* - J(\theta^0))}{G_H \cdot \epsilon^2}$ and $\bar{t}$ to be uniformly sampled from $\{0, \dots, T-1\}$. □

Having Theorem 4.1, the following corollary translates the convergence results to the overall trajectory complexity in order to achieve ϵ-FOSP.

Corollary 4.1. *Under Assumption 4.1 and 4.2, Algorithm 1 finds an $\bar{\epsilon}$-FOSP with no more than $\frac{256 G_g G_H (J^* - J(\theta^0))}{\bar{\epsilon}^3}$ random trajectories.*

Proof. Using Theorem 4.1 with $\epsilon = \frac{\bar{\epsilon}}{4G_H}$, we have $|\mathcal{M}_h| = \frac{4G_g}{\bar{\epsilon}}$ and the amortized per-iteration trajectory complexity is $2|\mathcal{M}_h|$. Besides, $T = \frac{32 G_H (J^* - J(\theta^0))}{\bar{\epsilon}^2}$, and hence, the overall trajectory complexity is $2|\mathcal{M}_h| \cdot T = \frac{256 G_g G_H (J^* - J(\theta^0))}{\bar{\epsilon}^3}$. □

5. Related Work

5.1. MDP-Specific Variance

In the policy-based and model-free reinforcement learning, the large variance in gradient estimation and the resulting high trajectory complexity have for long been identified as key challenges. This is mainly due to the term $\Psi_h(\tau)$ having large variance in the policy gradient (5), due to which the randomness grows exponentially with respect to the horizon. Ideally, by setting $\Psi_h(\tau)$ to be the advantage function of the MDP under policy π_θ, i.e. the difference of the state-action value function and the state-value function, the estimator given in (6) achieves the minimum possible variance. However, we generally do not have direct access to the advantage functions and can only use estimations like the discounted cumulative reward $\sum_{i=h}^{H} \gamma^i r(s_i, a_i)$ for approximation.

A notable portion of the literature has focused on deriving better choices of $\Psi_h(\tau)$ in order to reduce the variance in estimating the advantage function and can be classified into two categories depending on whether or not bootstrapping is used to update the state-value function (Sutton and Barto, 2018). By estimating $\Psi_h(\tau)$ with a critic which uses bootstrapping, the actor-critic type methods effectively drive down the variance at the cost of introducing bias to the gradient estimator (Konda and Tsitsiklis, 2000; Mnih et al., 2016). Alternatively, if we directly incorporate a baseline into $\Psi_h(\tau)$, the estimator (6) remains unbiased but potentially suffers from larger variance (Greensmith et al., 2004; Wu et al., 2018; Duan et al., 2016). The Generalized Advantage Estimation (GAE) proposed by (Schulman et al., 2016) incorporates the temporal-difference structure into the advantage function approximation and allows to control the tradeoff between bias and variance. We emphasize that all these refined advantage estimators can be directly incorporated to our method by placing $\Psi_h(\tau)$ correspondingly.

5.2. Variance Reduced Gradient in RL

While variance reduced gradient estimators have been successful in the oblivious supervised learning (see Section 2.1), its development in the non-oblivious reinforcement learning setting has been limited. Most of the existing work

only apply such techniques to solve *oblivious* subproblems in RL rather than using it to estimate the gradients: (Du et al., 2017) considers estimating the state-value function under the current policy via minimizing the empirical mean squared projected Bellman error. This is equivalent to a finite-sum convex-concave saddle point problem where the underlying distribution is the variable-independent uniform distribution, and can be solved by existing variance reduced-variants (Palaniappan and Bach, 2016). In another work Xu et al. (2017) use the SVRG estimator (Johnson and Zhang, 2013) to solve the trust region subproblem of TRPO (Schulman et al., 2015), which again is a finite sum minimization problem with an oblivious uniform underlying distribution.

Recently, Papini et al. (2018) propose SVRPG which is the first method that employs the variance-reduced type gradient estimator to directly optimize the non-oblivious loss (3). Concretely, denoting the importance weight

$$w(\theta^t, \tilde{\theta}; \tau) := \frac{p(\tau|\pi_{\tilde{\theta}})}{p(\tau|\pi_{\theta^t})} = \prod_{h=1}^{H} \frac{\pi_{\tilde{\theta}}(a_h|s_h)}{\pi_{\theta^t}(a_h|s_h)}, \tag{30}$$

SVRPG constructs the gradient estimator by

$$\mathbf{g}_{vr}^t := \tilde{\mathbf{g}} + \mathbf{g}(\theta^t; \mathcal{M}) - \frac{1}{|\mathcal{M}|} \sum_{\tau \in \mathcal{M}} w(\theta^t, \tilde{\theta}; \tau) \mathbf{g}(\tilde{\theta}; \{\tau\}),$$

where $\tilde{\theta}$ and $\tilde{\mathbf{g}}$ are the reference point and its corresponding unbiased estimator respectively, and $\mathcal{M}$ is a mini-batch of trajectories sampled from $p(\cdot|\pi_{\theta^t})$. Note that $\mathbf{g}_{vr}^t$ shares the same structure as $\mathbf{h}_{vr}^t$ (see Eqn. (8)) except the correction term $w(\theta^t, \tilde{\theta}; \tau)$, which ensures the unbiasedness of $\mathbf{g}_{vr}^t$ under the non-oblivious setting. However, by scrutinizing the convergence result, $\mathcal{O}(\frac{1}{\epsilon^4})$ random trajectories are still required to achieve an ϵ-FOSP (4), which is the same as the original policy-gradient type method. In the appendix, we briefly discuss why the trajectory complexity is not reduced.

As we presented in the previous section, HAPG directly estimate their difference by sampling from the Hessian integral instead of estimating the policy gradient at two point (θ^t and $\tilde{\theta}$) separately, which is the key to our success.

6. Experiments

In this section, we evaluate the performance of the proposed HAPG method on several standard reinforcement learning tasks. The REINFORCE method from (Sutton et al., 2000) is used as baseline for comparison.

The performance of HAPG and REINFORCE are tested on six continuous reinforcement learning tasks, namely CartPole, Swimmer, 2dWalker, Reacher, Humanoid, and HumanoidStandup, where the latter five are commonly used Mujoco environments (Todorov et al., 2012). We use the environment implementations in the garage library (Duan et al., 2016) on which our implementation is based as well. For all the tasks, we use deep Gaussian policy with the mean and variance parameterized by a fully-connected neural network. The number of network layers and hidden units, and the nonlinear activation functions follows (Papini et al., 2018) with the details given in the appendix[2]. For fair comparison, in each run we generate a common random policy for HAPG and REINFORCE as initialization. To limit the influence of randomness, each task is repeated 10 times and the performance is evaluated by averaging the minibatch episode return with 90% bootstrap confidence interval.

In terms of the sample complexity measurement, we use the number of system probes, i.e. the number of state transitions after taking an action according to the policy, instead of the number of trajectories. Such quantity serves a better criterion because different trajectories might have varying number of system probes: In many tasks, the environment returns FAILURE flag (usually at the beginning of the training procedure) and the current episode terminates before reaching the maximum horizon.

In our experiments, we find the hyper-parameters such as the minibatch size ($|\mathcal{M}_0|$ and $|\mathcal{M}|$), the epoch length p, and step-size have a large impact on the efficiency of the algorithms. The details of the parameter choices are also given in the appendix.

While we set the function $\Psi_h(\tau)$ to be the discounted cumulative reward $\sum_{i=h}^{H} \gamma^i r(s_i, a_i)$, we can incorporate more sophisticated choices from the literature to obtain better performance. For simplicity, in our experiment, we adopt the standard linear baseline from the garage library which predicts a state-dependent baseline function $b : \mathcal{S} \to \mathbb{R}$ and then we use the GAE($\gamma, 1$) type advantage estimation:

$$\Psi_h(\tau) = \sum_{i=h}^{H} \gamma^i r(s_i, a_i) - b(s_h), \tag{31}$$

where b is updated with the previous empirical trajectories. Note that under such choice of $\Psi_h(\tau)$ the gradient estimator (6) remains unbiased and all of our theoretical guarantee carries over (with different parameters G_g and G_H). In practice, replacing the baseline with a critic from the actor-critic type method may further improve the performance of HAPG and will be our future work. We emphasize that both methods use the same baseline in our implementation.

We present the results of the comparison are plotted in Figure 1. In the CartPole experiment, we see that HAPG has only a little advantage over REINFORCE. This is because the CartPole task is relatively easy. From hyper-parameter

[2]We observe that the SVRPG method from (Papini et al., 2018) is extremely sensitive to the initialization policy. When initialized by a random policy, SVRPG usually diverges and hence is excluded in our comparison.

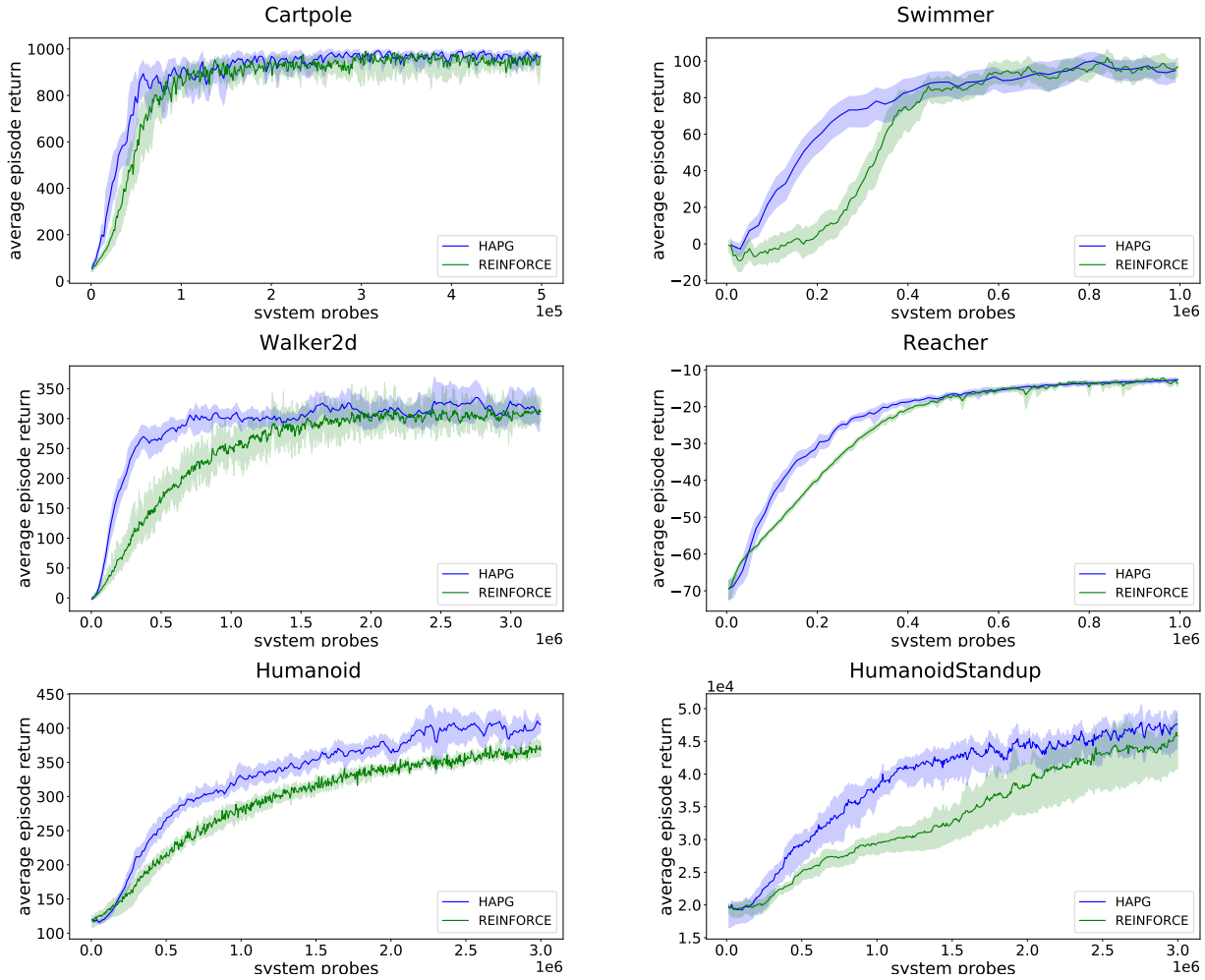

Figure 1. Comparison of the proposed HAPG method with the REINFORCE method on six tasks

setting of REINFORCE after grid searching, we see that a very small mini-batch of samples suffice to ensure the convergence of REINFORCE even when a relatively large step size is used. Under such circumstances, our Hessian-aided technique does not have the biggest advantage.

The advantage of HAPG over REINFORCE gets more clear when the task gets harder. Specifically, in the Mujoco tasks, HAPG converges to the parameter region with a high average reward using significantly less system probes than REINFORCE. Such observation can be explained by grid searching over hyper-parameter space: In these three tasks, REINFORCE requires a larger mini-batch size and its step-size has to be properly controlled to obtain the best performance. In contrast, with the exterior mini-batch size $|\mathcal{M}_0|$ set similar to REINFORCE's choice, the interior mini-batch size $|\mathcal{M}|$ can be chosen to be much smaller without compromising the convergence of HAPG. This is because when the step-size is small, our Hessian-aided scheme effectively reduce the variance with small number of extra trajectories.

Conclusion and Future Work

In this paper, we considered the problem of policy-based, model-free reinforcement learning. By introducing novel variance-reduced gradient estimators, we proposed a Hessian Aided Policy Gradient (HAPG) method which provides the first provable sample complexity improvement over the REINFORCE algorithm. HAPG incorporates the curvature information from the policy Hessian without compromising the $\mathcal{O}(d)$ per-iteration computation cost. Moreover, it can readily employ the state-of-the-art techniques for more sophisticated advantage function estimation which would result in superior performance. While we directly use the estimated gradient as the descent direction, methods like nature gradient (Kakade, 2002) or TRPO (Schulman et al., 2015) usually have better performance as they correct the gradients using an inverse-Fisher information matrix. A possible future direction is to combine HAPG with the nature gradient updates.

References

Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. A brief survey of deep reinforcement learning. *arXiv preprint arXiv:1708.05866*, 2017.

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

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*, 2014.

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*, 2017.

Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning*, pages 1329–1338, 2016.

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 687–697, 2018.

Evan Greensmith, Peter L Bartlett, and Jonathan Baxter. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, 2004.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in neural information processing systems*, 2013.

Sham M Kakade. A natural policy gradient. In *Advances in neural information processing systems*, pages 1531–1538, 2002.

Vijay R Konda and John N Tsitsiklis. Actor-critic algorithms. In *Advances in neural information processing systems*, pages 1008–1014, 2000.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.

Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*. Springer Science & Business Media, 2013.

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*, 2017.

Balamurugan Palaniappan and Francis Bach. Stochastic variance reduction methods for saddle-point problems. In *Advances in Neural Information Processing Systems*, pages 1416–1424, 2016.

Matteo Papini, Damiano Binaghi, Giuseppe Canonaco, Matteo Pirotta, and Marcello Restelli. Stochastic variance-reduced policy gradient. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

Barak A. Pearlmutter. Fast exact multiplication by the hessian. *Neural Computation*, 6:147–160, 1994.

Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

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.

Nicolas L Roux, Mark Schmidt, and Francis R Bach. A stochastic gradient method with an exponential convergence _rate for finite training sets. In *Advances in neural information processing systems*, 2012.

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

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

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

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*, 2000.

Richard Stuart Sutton. Temporal credit assignment in reinforcement learning. 1984.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 5026–5033. IEEE, 2012.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 1992.

Stephen Wright and Jorge Nocedal. Numerical optimization. *Springer Science*, 1999.

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.

Tianbing Xu, Qiang Liu, and Jian Peng. Stochastic variance reduction for policy gradient estimation. *arXiv preprint arXiv:1710.06034*, 2017.

Dongruo Zhou, Pan Xu, and Quanquan Gu. Stochastic nested variance reduced gradient descent for nonconvex optimization. In *Advances in Neural Information Processing Systems*, pages 3925–3936, 2018.

7. Appendix

7.1. Why SVRPG does not work

While this *importance sampling* technique removes the bias, the variance of estimator $\mathbf{g}_{vr}^t$ cannot be properly bounded since

$$\begin{aligned}&\mathbb{E}_{\mathcal{M}}\|\mathbf{g}^t - \nabla J(\theta^t)\|^2\\ \leq&\frac{1}{|\mathcal{M}|}\mathbb{E}_{\tau}\|\mathbf{g}(\theta^t;\{\tau\}) - w(\theta^t,\tilde{\theta};\tau)\mathbf{g}(\theta^t;\{\tau\})\|^2\\ =&\frac{1}{|\mathcal{M}|}\int_{\tau}\frac{\mathcal{R}(\tau)}{p_\tau^t}\|p_\tau^t\log\nabla p_\tau^t - \tilde{p}_\tau\log\nabla\tilde{p}_\tau\|^2\mathbf{d}\tau,\end{aligned}$$

where $p_\tau^t := p(\tau;\pi_{\theta^t})$, $\tilde{p}_\tau := p(\tau;\pi_{\tilde{\theta}})$, and the term $\frac{1}{p_\tau^t}$ in the integral can be infinity large. The lack of proper variance control deprives SVRPG of its high sample-efficiency: by scrutinizing the convergence result, $\mathcal{O}(\frac{1}{\epsilon^4})$ random trajectories are still required to achieve an ϵ-FOSP (4), which is the same as the original policy-gradient type method.

7.2. Derivation of Policy Gradient and Policy Hessian

Let $\tau = \{s_1, a_1, \ldots, s_H, a_H\}$ be a trajectory sampled according to $p(\tau;\pi_\theta)$ and define $\tau_h := \{s_1, a_1, \ldots, s_h, a_h\}$ for any $h \in [H]$. For simplicity of notation we will denote

$$\ell_\theta^{\tau_h} := \log p(\tau_h;\pi_\theta), \quad \bar{\mathcal{R}}_\gamma^{\tau_h} := \gamma^h\bar{\mathcal{R}}(a_h|s_h)$$

in the following discussion. From (3) and (2), we have

$$J(\theta) = \sum_{h=1}^H \mathbb{E}_{\tau\sim p(\tau;\pi_\theta)}[\bar{\mathcal{R}}_\gamma^{\tau_h}] = \sum_{h=1}^H \mathbb{E}_{\tau_h\sim p(\tau_h;\pi_\theta)}[\bar{\mathcal{R}}_\gamma^{\tau_h}],$$

where we replace τ by τ_h since $\bar{\mathcal{R}}_\gamma^{\tau_h}$ is independent of the randomness after a_h. To compute the policy gradient

$$\begin{aligned}\nabla J(\theta) =&\sum_{h=1}^H\int_{\tau_h}\bar{\mathcal{R}}_\gamma^{\tau_h}\nabla p(\tau_h;\pi_\theta)\mathbf{d}\tau_h\\ =&\sum_{h=1}^H\int_{\tau_h}\bar{\mathcal{R}}_\gamma^{\tau_h}p(\tau_h;\pi_\theta)\nabla\ell_\theta^{\tau_h}\mathbf{d}\tau_h,\end{aligned}$$

where we use the log-trick in the second equation

$$\nabla p(\tau_h;\pi_\theta) = p(\tau_h;\pi_\theta)\nabla\log p(\tau_h;\pi_\theta) = p(\tau_h;\pi_\theta)\nabla\ell_\theta^{\tau_h}.$$

The policy gradient can be further simplified:

$$\begin{aligned}\nabla J(\theta) =&\sum_{h=1}^H\int_{\tau_h}\bar{\mathcal{R}}_\gamma^{\tau_h}p(\tau_h;\pi_\theta)\nabla\ell_\theta^{\tau_h}\mathbf{d}\tau_h\\ =&\sum_{h=1}^H\mathbb{E}_{\tau_h\sim p(\tau_h;\pi_\theta)}[\bar{\mathcal{R}}_\gamma^{\tau_h}\sum_{i=1}^h\nabla\log\pi_\theta(a_i|s_i)]\\ =&\sum_{h=1}^H\sum_{i=1}^h\mathbb{E}_{\tau_h\sim p(\tau_h;\pi_\theta)}[\bar{\mathcal{R}}_\gamma^{\tau_h}\nabla\log\pi_\theta(a_i|s_i)]\\ =&\sum_{h=1}^H\sum_{i=1}^h\mathbb{E}_{\tau\sim p(\tau;\pi_\theta)}[\bar{\mathcal{R}}_\gamma^{\tau_h}\nabla\log\pi_\theta(a_i|s_i)],\end{aligned}$$

where in the last equality we use that $\bar{\mathcal{R}}_\gamma^{\tau_h}\nabla\log\pi_\theta(a_i|s_i)$ with $i \leq h$ is independent of the randomness after a_h. Exchange the summation over i and h to obtain

$$\begin{aligned}\nabla J(\theta) =&\sum_{i=1}^H\sum_{h=i}^H\mathbb{E}_{\tau\sim p(\tau;\pi_\theta)}[\bar{\mathcal{R}}_\gamma^{\tau_h}\nabla\log\pi_\theta(a_i|s_i)]\\ =&\sum_{i=1}^H\mathbb{E}_{\tau\sim p(\tau;\pi_\theta)}[\left(\sum_{h=i}^H\bar{\mathcal{R}}_\gamma^{\tau_h}\right)\nabla\log\pi_\theta(a_i|s_i)]\\ =&\sum_{i=1}^H\mathbb{E}_{\tau\sim p(\tau;\pi_\theta)}[\Psi_i(\tau)\nabla\log\pi_\theta(a_i|s_i)],\end{aligned}$$

where $\Psi_i := \sum_{h=i}^H\gamma^h\bar{\mathcal{R}}(a_h|s_h)$ is the discounted reward after action a_i given state s_i. Let

$$\Phi(\theta;\tau) = \sum_{i=1}^H\Psi_i(\tau)\log p(a_i|s_i;\pi_\theta).$$

Using such notation, we have

$$\begin{aligned}\nabla J(\theta) =&\mathbb{E}_{\tau\sim p(\tau;\pi_\theta)}\nabla\Phi(\theta;\tau)\\ =&\int_\tau p(\tau;\pi_\theta)\nabla\Phi(\theta;\tau)\mathbf{d}\tau.\end{aligned}$$

The second order derivative can be computed by

$$\begin{aligned}&\nabla^2 J(\theta)\\ =&\int_\tau\nabla\Phi(\theta;\tau)\nabla p(\tau;\pi_\theta)^\top + p(\tau;\pi_\theta)\nabla^2\Phi(\theta;\tau)\mathbf{d}\tau\\ =&\int_\tau p(\tau;\pi_\theta)\left[\nabla\Phi(\theta;\tau)\nabla\log p(\tau;\pi_\theta)^\top + \nabla^2\Phi(\theta;\tau)\right]\mathbf{d}\tau\\ =&\mathbb{E}_{\tau\sim p(\tau;\pi_\theta)}\left[\nabla\Phi(\theta;\tau)\nabla\log p(\tau;\pi_\theta)^\top + \nabla^2\Phi(\theta;\tau)\right].\end{aligned}$$

7.3. Detail Hyper-parameter Settings

We present the Hyper-parameter settings in Table 1.

Table 1. Hyper-parameter Settings

	CartPole	Swimmer	Reacher	Walker2d	Humanoid	HumanoidStandup
Horizon	100	500	500	500	500	500
Baseline	No	Linear	Linear	Linear	Linear	Linear
Number of timesteps	$5 \cdot 10^5$	10^6	10^6	$3 \cdot 10^6$	$3 \cdot 10^6$	$3 \cdot 10^6$
NN sizes	8	32x32	32x32	64x64	100x50x25	100x50x25
REINFORCE learning rate	0.05	0.001	0.01	0.001	0.005	0.005
REINFORCE batchsize	5000	5000	5000	5000	5000	5000
HAPG learning rate	0.05	0.01	0.05	0.01	0.01	0.01
HAPG $\lvert\mathcal{M}_0\rvert$	1000	10000	5000	5000	5000	5000
HAPG $\lvert\mathcal{M}\rvert$	100	500	500	500	500	500
HAPG p	10	20	10	10	10	10