In *Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems* (AAMAS 2019), Montreal, Canada May 2019

Reducing Sampling Error in Policy Gradient Learning

Josiah P. Hanna and Peter Stone
The University of Texas at Austin
Austin, TX
jphanna,pstone@cs.utexas.edu

ABSTRACT

This paper studies a class of reinforcement learning algorithms known as policy gradient methods. Policy gradient methods optimize the performance of a policy by estimating the gradient of the expected return with respect to the policy parameters. One of the core challenges of applying policy gradient methods is obtaining an accurate estimate of this gradient. Most policy gradient methods rely on Monte Carlo sampling to estimate this gradient. When only a limited number of environment steps can be collected, Monte Carlo policy gradient estimates may suffer from sampling error – samples receive more or less weight than they will in expectation. In this paper, we introduce the Sampling Error Corrected policy gradient estimator that corrects the inaccurate Monte Carlo weights. Our approach treats the observed data as if it were generated by a *different* policy than the policy that actually generated the data. It then uses importance sampling between the two – in the process correcting the inaccurate Monte Carlo weights. Under a limiting set of assumptions we can show that this gradient estimator will have lower variance than the Monte Carlo gradient estimator. We show experimentally that our approach improves the learning speed of two policy gradient methods compared to standard Monte Carlo sampling even when the theoretical assumptions fail to hold.

KEYWORDS

Reinforcement learning; Policy gradient; importance sampling; Monte Carlo

ACM Reference Format:
Josiah P. Hanna and Peter Stone. 2019. Reducing Sampling Error in Policy Gradient Learning. In *Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), Montreal, Canada, May 13–17, 2019,* IFAAMAS, 9 pages.

1 INTRODUCTION

The ability to learn is a key capability for autonomous agents and an important capability for the widespread deployment of autonomous agents on real world tasks such as robotics or healthcare. Reinforcement learning (RL) algorithms have the promise to allow autonomous agents to learn without direct human instructions. RL algorithms optimize the expected return – sum of rewards – by learning a *policy* that specifies an action-selection rule for any state in the task environment.

One of the most effective classes of policy learning algorithms is the class of *policy gradient* methods. Policy gradient RL is conceptually simple: the learning agent interacts with the environment, uses the observed states, actions, and rewards to estimate the gradient of the expected return with respect to the policy parameters, updates the policy with this gradient, and then repeats interaction. Despite this simplicity, variants of policy gradient RL have consistently produced state-of-the-art results in RL [14, 21, 26]. In this paper, we analyze a theoretical problem with standard policy gradient implementations, propose a practical solution to this problem, and show that our solution leads to empirically faster learning for sequential decision-making tasks.

Policy gradient implementations typically rely on some form of sampling to approximate the gradient of the policy's expected return. Sampling-based approaches are easy to implement yet can be inefficient – it may take a large number of samples to obtain an accurate estimate of the gradient. This inefficiency means that a learning agent must run its current policy on the task for a long time before it can accurately estimate how it should update its policy. Many advances in policy gradient methodology come from variance reduction techniques that make sampling more efficient[4, 6, 20]. In this paper, we propose a variance reduction technique that is complementary to previously proposed approaches.

We first observe that sampling-based approaches approximate expectations using the frequency that states, actions, and rewards appear in the observed data. In general these frequencies for a finite data set will be different than the frequency we would expect if the agent ran its policy to collect a much larger data set. More importantly, these frequencies will be different than the correct sample weightings that are determined by the environment and policy. While we cannot hope to correct for inaccuracy due to sampling from the environment, the policy is known and so at least some of the randomness is under our control.

We next observe that though the data was collected with the agent's current policy, for a finite data sample, it may appear more likely that the data was generated by a *different* policy. For example, if from a specific state a uniformly-randomly-acting agent is observed to move right twice and move left once, then it appears that the samples were generated from a policy that is more likely to move right than left. Under this view, we draw a connection to *off-policy* RL and use importance sampling to correct the observed action distribution to be closer to the expected distribution of actions under the current policy. We call this approach the Sampling

Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), N. Agmon, M. E. Taylor, E. Elkind, M. Veloso (eds.), May 13–17, 2019, Montreal, Canada.

Error Corrected (SEC) policy gradient estimator. The SEC estimator first estimates the policy that generated the observed samples. This estimated policy will generally be different than the agent's current policy. Once this policy is estimated, SEC applies importance sampling to correct the sample-based weighting of the return following each action. We present a theoretical analysis of the variance of this new policy gradient estimator and show that its variance is less than that of the commonly used Monte Carlo gradient estimator. We also conduct experiments with two policy gradient methods using SEC in place of Monte Carlo sampling. Empirical results show that applying the Sampling Error Corrected policy gradient estimator leads to faster learning across several reinforcement learning tasks.

2 PRELIMINARIES

This section formalizes our problem setting and provides relevant background on policy gradient reinforcement learning.

2.1 Preliminaries

The learning agent acts in an episodic *Markov decision process* with state space $\mathcal{S}$, action space $\mathcal{A}$, transition probabilities, P, reward function R, and discount factor γ [16]. The agent selects actions according to a stochastic policy π where π is a probability distribution over actions conditioned on state. The agent begins in state s_0 and selects action a_0 according to its policy. The environment then responds with state s_1 and reward r_0. The process repeats until a terminal state, s_∞ is reached.[1] We use τ to denote the trajectory: $s_0, a_0, r_0, ...s_\infty$. The probability of a trajectory depends on both the (known) action probabilities under π and the (unknown) state-transition and reward probabilities.

We assume that the policy is parameterized with parameter vector $\boldsymbol{\theta}$ and denote the parameterized policy as $\pi_{\boldsymbol{\theta}}$. Let $Q^\pi(s,a) = \mathbf{E}_\tau[\sum_{t=0}^\infty \gamma^t r_t | s_0 = s, a_0 = a]$ be the expected return (discounted sum of rewards) obtained when taking action a in state s and then following π until the end of the trajectory. The performance of a policy, η, is the expected return obtained when running the policy:

$$\eta(\pi) := \mathbf{E}_{s\sim\rho_\pi, a\sim\pi}[Q^\pi(s,a)]$$

where $\rho_\pi(s) = \sum_{t=0}^\infty \gamma^t \Pr(s_t = s)$ denotes the probability of being in state s while following policy π.

Given the objective η, reinforcement learning algorithms find $\boldsymbol{\theta}$ that maximize $\eta(\pi_{\boldsymbol{\theta}})$.

2.2 Policy Gradient Reinforcement Learning

One of the most common classes of reinforcement learning algorithms is the class of *policy gradient* methods. Policy gradient methods learn a (locally) optimal policy by updating the policy parameters with respect to the gradient of η:

$$\nabla_{\boldsymbol{\theta}}\eta(\pi_{\boldsymbol{\theta}}) := \mathbf{E}_{s\sim\rho_{\pi_{\boldsymbol{\theta}}}, a\sim\pi_{\boldsymbol{\theta}}}[Q^{\pi_{\boldsymbol{\theta}}}(s,a)\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s)] \quad (1)$$

[1] We assume finite trajectory lengths and only use s_∞ to denote the terminal state.

Since the expectation in (1) depends on the unknown environment and reward probabilities (via $\rho_{\pi_{\boldsymbol{\theta}}}$ and $Q^{\pi_{\boldsymbol{\theta}}}$), the gradient is typically approximated with sampling. Given a set, $\mathcal{D}$, of m state-action pairs observed while following $\pi_{\boldsymbol{\theta}}$ in the environment, the *Monte Carlo* policy gradient estimator is:

$$\nabla_{\boldsymbol{\theta}}\eta(\pi_{\boldsymbol{\theta}}) \approx g_{\texttt{mc}}(\mathcal{D}) = \frac{1}{m}\sum_{j=1}^{m} \widehat{Q}^{\pi_{\boldsymbol{\theta}}}(s_j,a_j)\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_j|s_j) \quad (2)$$

where $\widehat{Q}^{\pi_{\boldsymbol{\theta}}}(s_j,a_j)$ is an estimate of the sum of rewards following (s_i,a_i) and the state-action pairs (s_i,a_i) are observed while running the current policy $\pi_{\boldsymbol{\theta}}$. For sufficiently large m, the Monte Carlo estimator approximately weights each $\widehat{Q}^{\pi_{\boldsymbol{\theta}}}(s,a)\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s)$ by the probability $\rho_{\pi_{\boldsymbol{\theta}}}(s)\pi_{\boldsymbol{\theta}}(a|s)$ and $g_{\texttt{mc}}(\mathcal{D}_i)$ closely approximates $\nabla_{\boldsymbol{\theta}}\eta(\pi_{\boldsymbol{\theta}})$. While this estimator is known to have high variance, the policy gradient and its Monte Carlo approximation form the basis for many other methods that give strong performance. In particular, policy gradient methods have been shown to produce state-of-the-art reinforcement learning results (e.g., [6, 21, 26]).

Policy gradient algorithms usually share the general iterative steps:

(1) Collect m state-action pairs from the environment by running the current policy $\pi_{\boldsymbol{\theta}_i}$. We will call the set of these state-action pairs $\mathcal{D}_i$.
(2) Use $\mathcal{D}_i$ to compute $\widehat{Q}^{\pi_{\boldsymbol{\theta}_i}}(s,a)$ for all s,a that occur in $\mathcal{D}_i$.
(3) Approximate $\nabla_{\boldsymbol{\theta}}\eta(\pi_{\boldsymbol{\theta}_i})$ with (2) using $\mathcal{D}_i$ and the $\widehat{Q}^{\pi_{\boldsymbol{\theta}_i}}$ values.
(4) Set $\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i + \alpha_i g_{\texttt{mc}}(\mathcal{D}_i)$ where α_i is a step-size that may vary across iterations.

The exact implementation of any of these steps can vary from method to method. For example, Williams [27] uses $\widehat{Q}^{\pi_{\boldsymbol{\theta}}}(s,a) = \sum_{t=0}^\infty \gamma^t r_t$ to estimate $Q^{\pi_{\boldsymbol{\theta}}}$ while Sutton et al. [23] fit a linear function approximator, $\widehat{Q}_w$, and use it as the estimate of $Q^{\pi_{\boldsymbol{\theta}}}$. It is also common to use the *advantage function*, $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$, in place of $Q^\pi(s,a)$ where $V^\pi(s) = \mathbf{E}_{a\sim\pi}[Q(s,a)]$. Replacing Q^π with A^π leaves the gradient unchanged [4, 27].

3 SAMPLING ERROR IN THE MONTE CARLO GRADIENT

The Monte Carlo estimator, $g_{\texttt{mc}}$ is the standard approach to estimating the gradient in policy gradient learning. In this section, we discuss approximation error in $g_{\texttt{mc}}$ and present the view that – for a fixed $\mathcal{D}_i$ – $g_{\texttt{mc}}$ is the gradient estimated under the wrong distribution of states and actions. The view we present will suggest a simple solution to reduce the approximation error of the Monte Carlo policy gradient estimator.

For a finite number of sampled states and actions, $g_{\texttt{mc}}$, will have error unless $\mathcal{D}$ happens to contain each state and action (s,a) at its long-run expected frequency, $\rho_{\pi_{\boldsymbol{\theta}}}(s)\pi(a|s)$, and $\widehat{Q}^{\pi_{\boldsymbol{\theta}}}(s,a) = Q^{\pi_{\boldsymbol{\theta}}}(s,a)$ for all s,a. For ease of exposition, we will ignore differences in $\widehat{Q}^{\pi_{\boldsymbol{\theta}}}(s,a)$ and $Q^{\pi_{\boldsymbol{\theta}}}(s,a)$ and focus

on error due to sampling in the states and actions. Let $\rho_{\mathcal{D}_i}(s)$ be the proportion of times that s occurs in $\mathcal{D}_i$ and $\pi_{\mathcal{D}_i}(a|s)$ be the proportion of times that action a occurred in state s in $\mathcal{D}_i$. Formally, let $m(s)$ be the number of times that we observe state s in $\mathcal{D}_i$ and let $m(s,a)$ be the number of times that we observe action a in state s. We define $\rho_{\mathcal{D}_i}(s) = \frac{m(s)}{m}$ and $\pi_{\mathcal{D}_i}(a|s) = \frac{m(s,a)}{m(s)}$. Finally, we define the function $\bar{Q}^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ as:

$$\bar{Q}^\pi(s,a) = \begin{cases} \widehat{Q}^\pi(s,a) & m(s,a) > 0 \\ 0 & m(s,a) = 0 \end{cases}$$

Given these definitions, the Monte Carlo policy gradient estimator can be re-written as:

$$\begin{aligned} g_{\texttt{mc}}(\mathcal{D}_i) &= \frac{1}{m}\sum_{j=1}^{m} \widehat{Q}^{\pi_{\boldsymbol{\theta}}}(s_j,a_j)\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_j|s_j) \\ &= \frac{1}{m}\sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} m(s,a)\bar{Q}^{\pi_{\boldsymbol{\theta}}}(s,a)\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s) \\ &= \sum_{s\in\mathcal{S}} \rho_{\mathcal{D}_i}(s)\sum_{a\in\mathcal{A}} \pi_{\mathcal{D}_i}(a|s)\bar{Q}^{\pi_{\boldsymbol{\theta}}}(s,a)\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s) \\ &= \mathbf{E}_{s\sim\rho_{\mathcal{D}_i}, a\sim\pi_{\mathcal{D}_i}}\left[\bar{Q}^{\pi_{\boldsymbol{\theta}}}(s,a)\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s)\right] \end{aligned}$$

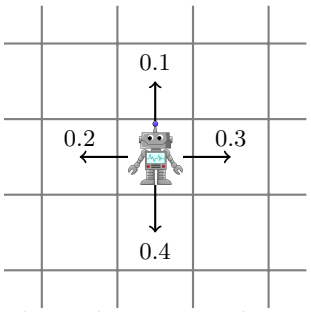
Notably, the sample average in (2) has been replaced with an exact expectation over actions as in (1). However, the expectation is taken over the action distribution $\pi_{\mathcal{D}i}$ and not $\pi_{\boldsymbol{\theta}}$. This expression suggests that sampling error in the Monte Carlo approximation can be viewed as evaluating the gradient under the wrong distribution. Figure 1 expresses how sampling error relates to weighting each $a \in \mathcal{A}$ in a Monte Carlo gradient estimate for a fixed state.

This section has shown that *for a fixed set of data* the Monte Carlo policy gradient estimator will be equal to an exact expectation taken over the wrong distribution of states and actions. The correct state distribution is unknown, however, we do know the correct action distribution ($\pi_{\boldsymbol{\theta}}$) and thus can potentially correct the inaccurate weighting. In the next section we introduce an algorithm that uses importance sampling to apply this correction.

Before concluding this section, we note that $g_{\texttt{mc}}$ is an unbiased estimator of $\nabla_{\boldsymbol{\theta}}\eta$. That is, if we were to repeatedly sample batches of data and estimate the gradient, the gradient estimates would be correct in expectation. However, once a single batch of data has been collected, we might ask, "can we correct for the sampling inaccuracy observed in this fixed sample?"

4 CORRECTING FOR SAMPLING ERROR

The previous section presented the view that sampling error in Monte Carlo approximations can be viewed as *covariate shift* – we are interested in an expectation under $\rho_{\pi_{\boldsymbol{\theta}}}$ and $\pi_{\boldsymbol{\theta}}$ but instead we have an expectation under $\rho_{\mathcal{D}_i}$ and $\pi_{\mathcal{D}}$. Viewing the sampling error as covariate shift suggests a simple solution: use importance sampling to correct for the distribution shift.



Action	$\pi_{\boldsymbol{\theta}}$	$\pi_{\mathcal{D}}$	$g_{\texttt{mc}}$ weight	$g_{\texttt{sec}}$ weight
Up	0.1	0.15	0.15	0.1
Right	0.3	0.35	0.35	0.3
Down	0.4	0.3	0.3	0.4
Left	0.2	0.2	0.2	0.2

Figure 1: Sampling error in a fixed state s of a gridworld environment. Each action a is sampled with probability $\pi_{\boldsymbol{\theta}}(a|s)$ and is observed in the proportion given by $\pi_{\mathcal{D}}(a|s)$. Monte Carlo weighting gives each return $Q^{\pi_{\boldsymbol{\theta}}}(a|s)$ the weight $\pi_{\mathcal{D}}(a|s)$ while our proposed Sampling Error Corrected weighting gives each return $Q^{\pi_{\boldsymbol{\theta}}}(a|s)$ the weight $\pi_{\mathcal{D}}(a|s)\frac{\pi_{\boldsymbol{\theta}}(a|s)}{\pi_{\mathcal{D}}(a|s)} = \pi_{\boldsymbol{\theta}}(a|s)$. Thus SEC weights each advantage by the correct amount while the Monte Carlo estimator will have error unless the empirical proportion of sampled actions, $\pi_{\mathcal{D}}$, is equal to the expected proportion, $\pi_{\boldsymbol{\theta}}$ for all actions.

Importance Sampling (IS) is a method for reweighting values generated by one policy (commonly called the behavior policy), μ, such that in the limit of infinite samples the values are weighted as if they had come from the policy of interest, $\pi_{\boldsymbol{\theta}}$. In our case, we will treat the empirical distribution of actions, $\pi_{\mathcal{D}}$ as the behavior policy μ and then use IS to correct for the shift between the empirical and desired distribution. We call this approach the *sampling error corrected* (SEC) policy gradient estimator.

In practice, using the true $\pi_{\mathcal{D}}$ may introduce high bias into gradient estimates, particularly in continuous state and action spaces. This bias is because using $\pi_{\mathcal{D}}$ can be shown to be equivalent to assuming that $\widehat{Q}^{\pi_{\boldsymbol{\theta}}}(s,a)$ is zero for all unobserved actions.[2] Instead, let $\pi_{\boldsymbol{\phi}}$ be a parametric estimate of the policy that generated our data.[3] The SEC estimator estimates $\boldsymbol{\phi}_i$ so that $\pi_{\boldsymbol{\phi}}$ is the maximum likelihood policy that generated our data:

$$\boldsymbol{\phi}_i = \underset{\boldsymbol{\phi}}{\text{argmax}} \sum_{i=1}^{m} \log \pi_{\boldsymbol{\phi}}(a_i|s_i) \tag{3}$$

Importantly, SEC estimates $\boldsymbol{\phi}_i$ with the same m samples that will be used to estimate the policy gradient. If $\boldsymbol{\phi}_i$ is estimated with a different set of samples then $\pi_{\boldsymbol{\phi}}$ will contain no information for correcting sampling error – our experiments

[2]This assumption can be seen in the definition of $\bar{Q}$ in Section 3.
[3]We assume in this paper that we use a parametric policy estimate and leave non-parametric estimates to future work.

confirm this observation. For most RL benchmarks, (3) can be formulated as a supervised learning problem.

Given $\pi_{\boldsymbol{\phi}}$, SEC re-weights each $\widehat{Q}^{\pi_{\boldsymbol{\theta}}}(s_i, a_i)\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_i|s_i)$ by the ratio of the true likelihood $\pi_{\boldsymbol{\theta}}$ to the estimated empirical likelihood $\pi_{\boldsymbol{\phi}}$:

$$g_{\mathtt{sec}}(\mathcal{D}_i) = \frac{1}{m}\sum_{j=1}^{m} \frac{\pi_{\boldsymbol{\theta}}(a_j|s_j)}{\pi_{\boldsymbol{\phi}}(a_j|s_j)} \widehat{Q}^{\pi_{\boldsymbol{\theta}}}(s_j, a_j)\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_j|s_j) \quad (4)$$

Intuitively, when an action is sampled more often than its expected proportion, $g_{\mathtt{sec}}$ down-weights the gradient estimate following that action. Similarly, when an action is sampled less often than its expected proportion, $g_{\mathtt{sec}}$ up-weights the gradient estimate following that action. As we will discuss in the next section, if $\pi_{\boldsymbol{\phi}}$ is close to $\pi_{\mathcal{D}}$ then this sampling correction can eliminate variance in the action selection. Full details of this approach are given in Algorithm 1.

Algorithm 1 Sampling Error Corrected Policy Gradient

Input: Initial policy parameters, $\boldsymbol{\theta}_0$, batch size m, a step-size for each iteration, α_i, and number of iterations n.

Output: Optimized policy parameters $\boldsymbol{\theta}_n$.

1: **for all** $i = 0$ to n **do**
2: $\quad\mathcal{D}_i =$ Sample m steps $(s, a) \sim \pi_{\boldsymbol{\theta}_i}$
3: $\quad\boldsymbol{\phi}_i \leftarrow \underset{\boldsymbol{\phi}}{\text{argmax}} \sum_{j=1}^{m} \log \pi_{\boldsymbol{\phi}}(a_j|s_j)$
4: $\quad g_{\mathtt{sec}} \leftarrow \frac{1}{m}\sum_{j=1}^{m} \frac{\pi_{\boldsymbol{\theta}}(a_j|s_j)}{\pi_{\boldsymbol{\phi}}(a_j|s_j)} \widehat{Q}^{\pi_{\boldsymbol{\theta}}}(s_j, a_j)\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}_i}(a_j|s_j)$
5: $\quad\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i + \alpha_i \cdot g_{\mathtt{sec}}$
6: **end for**
7: **Return** $\boldsymbol{\theta}_n$

Importance sampling in reinforcement learning is typically applied for *off-policy* learning, i.e., learning with data that has been generated by a policy that is different from the current policy. Despite this connection to off-policy learning, we remain in the *on-policy* setting: data is collected with the current policy, used to update the current policy, and then discarded.

The SEC estimator is related to the use of importance sampling for off-policy reinforcement learning where the behavior policy μ must be estimated before it can be used to form the importance weights. In practice, behavior policy estimation can be challenging when the distribution class of the true behavior policy is unknown [7, 17]. Fortunately, in the on-policy, policy gradient setting, we have complete access to the behavior policy and can specify the model class of $\pi_{\boldsymbol{\phi}}$ to be the same as $\pi_{\boldsymbol{\theta}}$. We can even simplify the $\pi_{\boldsymbol{\phi}}$ model class by estimating a policy that conditions on intermediate representations of $\pi_{\boldsymbol{\theta}}$. For example if $\pi_{\boldsymbol{\theta}}$ is a convolutional neural network, we can use all but the last layer of $\pi_{\boldsymbol{\theta}}$ as a feature extractor and then model $\pi_{\boldsymbol{\phi}}$ as a linear function of these features. We evaluate this technique in our experiments.

4.1 Variance Analysis

In this section we analyze the variance of $g_{\mathtt{sec}}$ compared to that of $g_{\mathtt{mc}}$. We make a few assumptions that simply the analysis:

(1) The action space is discrete and if a state is observed then all actions have also been observed in that state.
(2) The return estimate $\widehat{Q}^{\pi_{\boldsymbol{\theta}}}$ is computed independently of $\mathcal{D}$. This assumption implies $\widehat{Q}^{\pi_{\boldsymbol{\theta}}}(s, a)$ is a constant with respect to a fixed (s, a) in $\mathcal{D}$.
(3) For all observed states, our estimated policy $\pi_{\boldsymbol{\phi}}$ is equal to $\pi_{\mathcal{D}}$, i.e., if action a occurs k times in state s and s occurs n times in $\mathcal{D}$ then $\pi_{\boldsymbol{\phi}}(a|s) = \frac{k}{n}$.

Let $\mathbb{S}$ be the random variable representing the states in $\mathcal{D}$ and $\mathbb{A}$ be the random variable representing the actions in $\mathcal{D}$. We will use $\mathcal{D} = \{\mathbb{S}, \mathbb{A}, \widehat{Q}^{\pi_{\boldsymbol{\theta}}}\}$ to make explicit that $\mathcal{D}$ depends on both the randomness in the set of sampled states and sampled actions. We can now give the central theoretical claim of this paper.

PROPOSITION 1. *Let* $\text{Var}_{\mathbb{S},\mathbb{A}}(g)$ *denote the variance of estimator* g *with respect to random variables* $\mathbb{S}$ *and* $\mathbb{A}$*. For the Monte Carlo estimator,* $g_{\mathtt{mc}}$*, and the* SEC *estimator,* $g_{\mathtt{sec}}$*:*

$$\text{Var}_{\mathbb{S},\mathbb{A}}(g_{\mathtt{sec}}(\{\mathbb{S}, \mathbb{A}\})) < \text{Var}_{\mathbb{S},\mathbb{A}}(g_{\mathtt{mc}}(\{\mathbb{S}, \mathbb{A}\}))$$

PROOF. We provide a partial proof in this section. Additional details are given in Appendix A.

We first note that both $g_{\mathtt{sec}}$ and $g_{\mathtt{mc}}$ can be written as:

$$g(\{\mathbb{S}, \mathbb{A}\}) = \sum_{s\in\mathcal{S}} \rho_{\mathcal{D}}(s) \sum_{a\in\mathcal{A}} \pi_{\mathcal{D}}(a|s) w(s, a) \widehat{Q}(s, a) \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s) \quad (5)$$

where $w(s, a) = \frac{\pi_{\boldsymbol{\theta}}(a|s)}{\pi_{\mathcal{D}}(a|s)}$ for $g_{\mathtt{sec}}$ and $w(s, a) = 1$ for $g_{\mathtt{mc}}$.

Using the law of total variance, the variance of (5) can be decomposed as:

$$\text{Var}_{\mathbb{S},\mathbb{A}}(g(\{\mathbb{S}, \mathbb{A}\})) = \underbrace{\mathbf{E}_{\mathbb{S}}\left[\text{Var}_{\mathbb{A}}(g(\{\mathbb{S}, \mathbb{A}\})|\mathbb{S})\right]}_{\Sigma_{\mathbb{A}}} + \underbrace{\text{Var}_{\mathbb{S}}\left(\mathbf{E}_{\mathbb{A}}[g(\{\mathbb{S}, \mathbb{A}\})|\mathbb{S}]\right)}_{\Sigma_{\mathbb{S}}}$$

The first term, $\Sigma_{\mathbb{A}}$, is the variance due to stochasticity in the action selection.

CLAIM 1. $\text{Var}_{\mathbb{A}}(g_{\mathtt{sec}}(\{\mathbb{S}, \mathbb{A}\}|\mathbb{S})) = 0.$

PROOF. See Appendix A. Intuitively, this claim follows from the fact that using $w(s, a) = \frac{\pi_{\boldsymbol{\theta}}(a|s)}{\pi_{\mathcal{D}}(a|s)}$ results in all randomness due to $\mathbb{A}$ canceling. □

From Claim 1, $\Sigma_{\mathbb{A}}$ will be zero since $\mathbf{E}_{\mathbb{S}}[0] = 0$. However, this term will be positive for $g_{\mathtt{mc}}$ since in general the Monte Carlo estimator does *not* have zero variance.[4]

The second term, $\Sigma_{\mathbb{S}}$, is the variance due to only visiting a limited number of states before estimating the gradient.

CLAIM 2. $\mathbf{E}_{\mathbb{A}}[g_{\mathtt{sec}}(\{\mathbb{S}, \mathbb{A}\})|\mathbb{S}] = \mathbf{E}_{\mathbb{A}}[g_{\mathtt{mc}}(\{\mathbb{S}, \mathbb{A}\})|\mathbb{S}].$

[4]The Monte Carlo estimator has zero variance with respect to action sampling only when $\widehat{Q}^{\pi_{\boldsymbol{\theta}}}(s, a)$ is equal for all actions in any state.

PROOF. See Appendix A. Under Assumption (1) and (3) the expectation over $\pi_{\mathcal{D}}$ ($\sum_{a \in \mathcal{A}} \pi_{\mathcal{D}}(a|s)$) in (5) is converted to an exact expectation over $\pi_{\boldsymbol{\theta}}$ ($\sum_{a \in \mathcal{A}} \pi_{\boldsymbol{\theta}}(a|s)$) and $g_{\texttt{mc}}$ is an unbiased estimator of this exact expectation. □

From Claim 2, it follows that $\Sigma_{\mathbb{S}}$ will be the same for both $g_{\texttt{mc}}$ and $g_{\texttt{sec}}$. Since $\Sigma_{\mathbb{S}}$ is identical for both terms and $\Sigma_{\mathbb{A}}$ is zero for $g_{\texttt{sec}}$, the variance of $g_{\texttt{sec}}$ can be no more than that of $g_{\texttt{mc}}$. □

Our claim that the variance of $g_{\texttt{sec}}$ is less than that of $g_{\texttt{mc}}$ has been shown under a limiting set of assumptions. The assumption that all actions have been observed in all sampled states and that we can estimate $\pi_{\mathcal{D}}$ exactly limits the analysis to discrete state and action domains. Analyzing the estimators' variances under relaxed assumptions is an interesting direction for future work.

Finally, we note that in typical policy gradient implementations the assumption that $\widehat{Q}^{\pi_{\boldsymbol{\theta}}}$ is computed independently of $\mathcal{D}$ is typically violated. In this case, the variance decomposition will have a third term that is due to variance in the return estimates:

$$\Sigma_{\tau} = \mathbf{E}_{\mathbb{S},\mathbb{A}} \left[\text{Var} \left(g(\{\mathbb{S}, \mathbb{A}\} | \mathbb{S}, \mathbb{A}) \right) \right]$$

This term may not necessarily be less for either estimator and we leave its analysis to future work. We also discuss in our future work section how the SEC estimator could be modified to lower the variance of the return estimates.

5 EMPIRICAL RESULTS

In this section we present an empirical evaluation of the sampling error corrected policy gradient estimator. While the analysis in the previous section was based on limiting assumptions, we now evaluate whether $g_{\texttt{sec}}$ can lead to faster learning in practice, even when these assumptions are violated. Specifically, we study $g_{\texttt{sec}}$ in both continuous and discrete state spaces, in discrete and continuous action spaces, and when the return estimates are *not* independent of the gradient estimate. Our main empirical question is, "Does replacing $\widehat{Q}^{\pi}(s,a)$ with $\frac{\pi(a|s)}{\pi_{\boldsymbol{\phi}}(a|s)} \widehat{Q}^{\pi}(s,a)$ lead to faster learning within a policy gradient method?"

5.1 Empirical Set-up

We first describe four reinforcement learning tasks and the motivation for evaluating SEC in these domains.

Gridworld. Our first domain is a 4×4 gridworld and we use REINFORCE [27] as the underlying policy gradient algorithm. The agent begins in grid cell $(0,0)$ and trajectories terminate when it reaches $(3,3)$. The agent receives a reward of 100 at termination, -10 at $(1,1)$ and -1 otherwise. The agent's policy is a state-dependent softmax distribution over actions. The SEC estimator estimates the policy by counting how many times each action is taken in each state. This domain closely matches the assumptions made in our theoretical analysis. Specifically, the state and action spaces are discrete and $\pi_{\boldsymbol{\phi}}$ is exactly equal to $\pi_{\mathcal{D}}$. While we do not explicitly enforce the assumption that all actions are observed in all states, the small size of the state and action space ($|\mathcal{S}| = 16$ and $|\mathcal{A}| = 4$) makes it likely that this assumption holds.

Tabular Mountain Car. Our second domain is a discretized version of the classic mountain car domain. We use REINFORCE as the policy gradient algorithm. The agent attempts to move an under-powered car up a steep hill by accelerating to the left or right. The agent's policy is a state-dependent softmax distribution over the two discrete actions. The SEC estimator estimates the policy by counting how many times each action is taken in each state. This domain has a large number of discrete states and it is unlikely that all actions are observed in all states. In this setting, $g_{\texttt{sec}}$ will have higher bias. This domain matches our theoretical setting in that states and actions are discrete and $\pi_{\boldsymbol{\phi}}$ is exactly equal to $\pi_{\mathcal{D}}$.

Linear Dynamical System. Our third domain is a two-dimensional linear dynamical system with additive Gaussian noise. The reward is the agent's distance to the origin and trajectories last for 20 time-steps. In this domain the learning agent uses a linear Gaussian policy to select continuous valued accelerations in the x and y direction. We use the OpenAI Baselines [3] implementation of *trust-region policy optimization* (TRPO) as the underlying policy gradient algorithm [20]. We set the generalized advantage estimation parameters (γ, λ) both to 1. Unless noted otherwise, we use the default OpenAI Baselines default values for all other hyper-parameters. We estimate $\pi_{\boldsymbol{\phi}}$ with ordinary least squares and estimate a state-independent variance parameter. In this domain, none of our theoretical assumptions hold. We include it to evaluate $g_{\texttt{sec}}$ with simple function approximation. We estimate the TRPO surrogate objective and constraint with 1000 steps per batch and set the KL-Divergence constraint, $\epsilon = 0.01$.

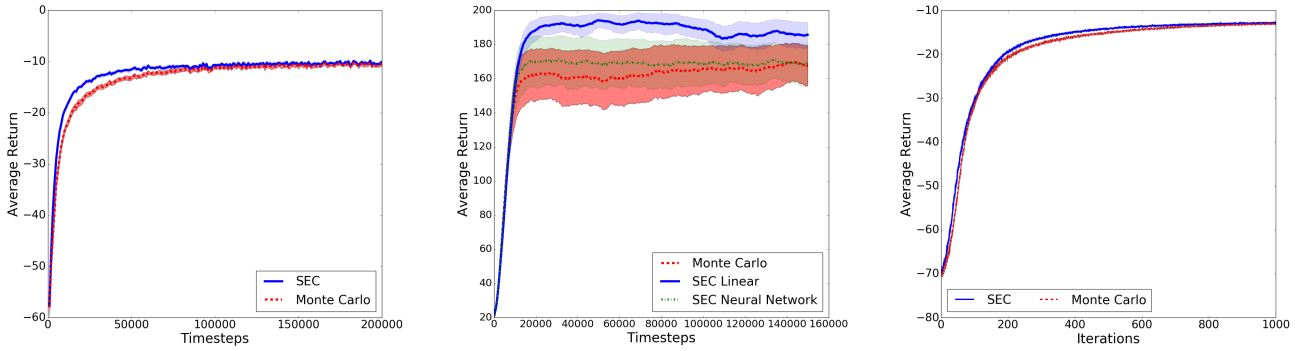
CartPole. Our final domain is the CartPole-v0 domain from OpenAI Gym and we again use TRPO. We estimate the TRPO surrogate objective and constraint with 200 steps per batch and set the KL-Divergence constraint, $\epsilon = 0.001$. The policy representation is a two layer neural network with 32 hidden units in each layer. The output of the network is the parameters of a softmax distribution over the two actions. We consider two parameterizations of $\pi_{\boldsymbol{\phi}}$:

(1) $\pi_{\boldsymbol{\phi}}$ is a neural network with the same architecture as $\pi_{\boldsymbol{\theta}}$. We estimate $\pi_{\boldsymbol{\phi}}$ with batch gradient descent. This method is labeled **SEC Neural Network**.
(2) $\pi_{\boldsymbol{\phi}}$ is a linear policy that receives the activations of the last hidden layer of $\pi_{\boldsymbol{\theta}}$ as input. The dual $\pi_{\boldsymbol{\phi}}$ and $\pi_{\boldsymbol{\theta}}$ architecture is shown in Figure 2. We estimate the weights of $\pi_{\boldsymbol{\phi}}$ with gradient descent. This method is labeled **SEC Linear**.

Again, this domain violates all assumptions made in our theoretical analysis. We include this domain to study $g_{\texttt{sec}}$ with more complex function approximation. This setting allows us to study $g_{\texttt{sec}}$ with neural network policies but is simple enough to avoid extensive tuning of hyper-parameters.

5.2 Empirical Results

We now present our empirical results.



(a) Linear Dynamical System (b) Cart Pole (c) Mountain Car

Figure 3: Learning results for the Linear Dynamical System, Cart Pole, and Mountain Car domains. The x-axis is the number of timesteps and the y-axis is the average return of a policy. We run 25 trials of each method using different random seeds. Error bars show a 95% confidence interval. In both domains we see that all variants of Sampling Error Corrected policy gradient outperforms standard Monte Carlo policy gradient in either time to optimal convergence or final performance.

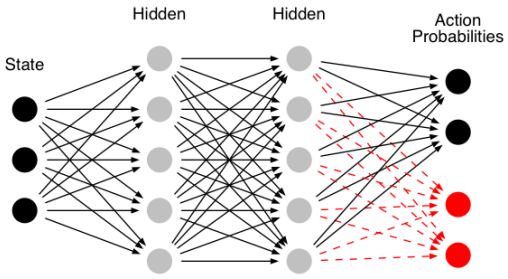


Figure 2: A simplified version of the neural network architecture used in CartPole. The true architecture has 32 hidden units in each layer. The current policy $\pi_{\boldsymbol{\theta}}$ is given by a neural network that outputs the action probabilities as a function of state (black nodes). The estimated policy, $\pi_{\boldsymbol{\phi}}$, is a linear policy that takes as input the activations of the final hidden layer of $\pi_{\boldsymbol{\theta}}$. Only the weights on the red, dashed connections are changed when estimating $\pi_{\boldsymbol{\phi}}$.

5.2.1 Main Results. Results for Mountain Car, Linear Dynamical System (LDS), and Cart Pole environment are given in Figure 3. In all three domains, we see that the SEC methods lead to learning speed-up compared to the Monte Carlo based approaches. In the LDS and Mountain Car environments, SEC outperforms Monte Carlo in time to convergence to optimal. In Cart Pole, both variants of SEC learn faster initially, however, Monte Carlo catches up to the neural network version of SEC. This result demonstrates that we can leverage intermediate representations of $\pi_{\boldsymbol{\theta}}$ (in this case, the activations of the final hidden layer) to learn $\pi_{\boldsymbol{\phi}}$ with a simpler model class. In fact, results suggest that fitting a simpler model improves performance. We hypothesize that simpler models require less hyper-parameter re-tuning throughout learning and so we get a more accurate estimate of $\pi_{\mathcal{D}}$ which leads to a more accurate sampling error correction.

5.2.2 Gridworld Ablations. Figure 4 shows several results in the Gridworld domain. First, Figure 4(a) shows SEC leads to faster convergence compared to Monte Carlo. This domain most closely matches our theoretical assumptions where we showed SEC has lower variance than Monte Carlo gradient estimates. The lower variance translates into faster learning.

We also use the Gridworld domain to perform a quantitative evaluation of sampling error. As a measure of sampling error we use the Earth Mover's distance between the current policy $\pi_{\boldsymbol{\theta}}$ and the empirical frequency of actions, $\pi_{\mathcal{D}}$. Intuitively, for any state, s, the Earth Mover's distance measures how much probability mass must be moved to transform $\pi_{\mathcal{D}}(\cdot|s)$ into $\pi_{\boldsymbol{\theta}}(\cdot|s)$.[5] Figure 4(b) shows that sampling error increases and then decreases during learning. Peak sampling error is aligned with where the learning curve gap between the two methods is greatest. Note that sampling error naturally decreases as learning converges because the policy becomes more deterministic. Figure 4(c) shows that the entropy of the current policy goes to zero, i.e., becomes more deterministic. A more deterministic policy will have less sampling error and so we expect to see less advantage from SEC as learning progresses.

Finally, we also verify the importance of using the same data to both estimate $\pi_{\boldsymbol{\phi}}$ and estimate the policy gradient. Figure 4(d) introduces two alternatives to SEC:

- INDEPENDENT: Estimates $\pi_{\boldsymbol{\phi}}$ with a separate set of m samples and then uses this estimate to estimate $\mathcal{D}_i$

[5] We choose the Earth Mover's distance (also known as the Wasserstein distance) as opposed to the more commonly used KL-divergence since $\pi_{\mathcal{D}}$ and $\pi_{\boldsymbol{\theta}}$ may not share support. That is, there may be an action, a, where $\pi_{\mathcal{D}}(a|s)$ is 0 and $\pi_{\boldsymbol{\theta}}(a|s) > 0$.

- RANDOM: Instead of computing importance weights, we randomly sample weights from a normal distribution and use these in place of the learned SEC weights.

Figure 4(a) shows that INDEPENDENT hurts performance compared to Monte Carlo. RANDOM performs marginally worse than Monte Carlo. This result demonstrates the need to use the same set of data to estimate $\pi_{\boldsymbol{\phi}}$ and the gradient.

6 RELATED LITERATURE

There is a wide body of literature on variance reduction for policy gradient methods. One of the most common techniques is to use a state-dependent baseline which preserves unbiasedness while lowering variance [4]. Recently multiple works have studied action-dependent baselines [5, 13] though the benefit of this approach has been questioned [25]. Schulman et al. use *generalized advantage estimation* which allows the user to better control the bias-variance trade-off of advantage estimation with a parameter, λ [20]. Thomas [24] and Schulman et al. [20] also consider the use of a discount factor in undiscounted problems (i.e., $\gamma = 1$) as a method for variance reduction at the cost of some bias. Stochastic policy actor-critic methods reduce variance with a learned value-function replacing part or all of a Monte Carlo return estimate. All of these variance reduction techniques use Monte Carlo sampling in some form and could be improved with sampling error corrections.

Deterministic policy gradient (DPG) methods avoid sampling in the action space by learning a deterministic policy [12, 22]. Since there is no action-space sampling, the variance reduction presented here is inapplicable to DPG based methods. However, DPG methods may have high bias that makes them unstable and thus stochastic policy gradient methods may be preferred in practice. An alternative to DPG and action-space sampling stochastic policy gradient methods is the expected policy gradient (EPG) approach of Ciosek and Whiteson [1]. This method learns a stochastic policy but avoids sampling by analytically integrating over the action-space. This approach requires learning a critic (an approximation of $Q^{\pi_{\boldsymbol{\theta}}}$) and requires the policy and critic to have a form that can be analytically integrated.

Our SEC method is related to applications of importance sampling that first estimate the data sampling distribution (i.e., the behavior policy, μ) and then apply importance sampling with the estimated behavior policy instead of the true behavior policy. Though it may be natural to assume that such an estimator will perform worse than using the true behavior policy, much work in the causal inference [9, 18], Monte Carlo integration [2, 8], and multi-armed bandit literature [11] has shown that this is *not* in fact the case. More recently, Hanna et al. have shown that this approach can lower the variance of importance sampling policy evaluation in off-policy evaluation in Markov decision processes [7]. Our work contrasts with this earlier work in that we are concerned with policy gradient reinforcement learning and are in the on-policy setting. Using off-policy data with an estimated behavior policy to improve data-efficiency could be an interesting step for future work.

Our proposed approach combines importance sampling with policy gradient RL. Importance sampling has been used before in policy gradient learning to incorporate off-policy data [10]. Both TRPO and PPO use importance sampling to correct for the policy shift during a line search on a policy performance objective [19, 21]. The ACER algorithm uses importance sampling and the Retrace method [15] to incorporate off-policy trajectory segments [26]. In contrast, we use importance sampling to correct the empirical distribution of actions to better match the true distribution over actions when sampling in an on-policy fashion.

7 DISCUSSION AND FUTURE WORK

We have proposed the sampling error corrected policy gradient estimator and showed that empirically it can increase the data-efficiency of REINFORCE and Trust Region Policy Optimization. In this section, we discuss our proposed approach and highlight next steps for extending our results.

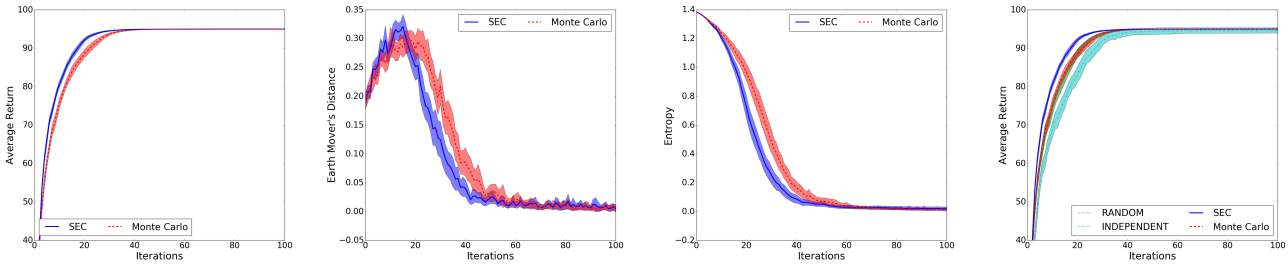
Policy gradient methods typically have lower bias updates than other RL algorithms. The SEC gradient estimator trades-off some of this bias to achieve lower variance estimates. Though there is a benefit to doing this when the base algorithm is a low-bias, high-variance method such as TRPO or REINFORCE, it is less clear that the benefit would remain when incorporating sampling error corrections into higher bias methods such as actor-critic methods. Exploring SEC with actor-critic methods is one direction for future work.

Replacing $\widehat{Q}^{\pi_{\boldsymbol{\theta}}}(a|s)$ with $\frac{\pi_{\boldsymbol{\theta}}(a|s)}{\pi_{\boldsymbol{\phi}}(a|s)}\widehat{Q}^{\pi_{\boldsymbol{\theta}}}(a|s)$ was shown to reduce variance and speed-up policy improvement. However, this approach only reduces variance in the initial action selection and fails to account for variance in the $\widehat{Q}^{\pi_{\boldsymbol{\theta}}}(a|s)$ estimates. Since $\widehat{Q}^{\pi_{\boldsymbol{\theta}}}(a|s)$ is dependent on future action selection it may be possible to further lower variance by using a *full trajectory* SEC estimator:

$$g_{\mathsf{sec}}(\mathcal{D}) = \frac{1}{m}\sum_{j=0}^{m}\widehat{Q}^{\pi_{\boldsymbol{\theta}}}(a_0^j|s_0^j)\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(a_t^j|s_t^j)\prod_{t=0}^{\infty}\frac{\pi_{\boldsymbol{\theta}}(a_t^j|s_t^j)}{\pi_{\boldsymbol{\phi}}(a_t^j|s_t^j)}.$$

This advantage estimator would reduce sampling error in action sampling after the first step, however, a poorly fit $\pi_{\boldsymbol{\phi}}$ could have its error amplified over the long horizon.

The biggest limitation of our current approach is that estimating $\pi_{\boldsymbol{\phi}}$ requires solving a supervised learning problem at every iteration. Since RL changes the policy and thus the data distribution at every iteration, the best hyper-parameters of the selected supervised learning algorithm may change at every iteration as well. On our experimental results to date we have not found the need to re-tune these parameters across iterations, however, re-tuning could be more necessary on more complex domains. For RL domains where the policy is typically represented by a convolutional neural network this supervised learning problem also bears a prohibitive computational cost. We have shown that the computational cost of training a more complex $\pi_{\boldsymbol{\phi}}$ can be avoided by training a simpler model on the learned representation of $\pi_{\boldsymbol{\theta}}$. However, more work is needed to fully understand whether such simpler



(a) Avg. Return (b) Sampling Error (c) Entropy (d) Alternatives

Figure 4: Sampling Error Corrections in the Gridworld Domain. Figure 4(a) shows the average return for SEC and MC. Figure 4(b) shows earth mover's distance between the current policy and estimated policy at each iteration. Figure 4(c) shows policy entropy at each iteration. Figure 4(d) shows two alternative weight corrections. Results are averaged over 25 trials and confidence bars are for a 95% confidence interval.

$\pi_{\boldsymbol{\phi}}$ representations can always suffice when a more complex model is otherwise necessary.

8 CONCLUSION

In this paper we have proposed the sampling error corrected policy gradient estimator (SEC). SEC attempts to weight the observed samples by their true probability of occurring when executing the current policy $\pi_{\boldsymbol{\theta}_i}$. This contrasts to the commonly used Monte Carlo policy gradient estimator that weights each sample by its empirical frequency. Theoretical results show that under a limiting set of conditions SEC has lower variance than the Monte Carlo estimator. We also presented an empirical study of SEC and found that it can increase the learning speed of REINFORCE and trust-region policy optimization even when these theoretical conditions fail to hold.

ACKNOWLEDGMENTS

This work has taken place in the Learning Agents Research Group (LARG) at the Artificial Intelligence Laboratory, The University of Texas at Austin. LARG research is supported in part by grants from the National Science Foundation (IIS-1637736, IIS-1651089, IIS-1724157), the Office of Naval Research (N00014-18-2243), Future of Life Institute (RFP2-000), DARPA, Intel, Raytheon, and Lockheed Martin. Josiah Hanna is supported by an IBM PhD Fellowship. Peter Stone serves on the Board of Directors of Cogitai, Inc. The terms of this arrangement have been reviewed and approved by the University of Texas at Austin in accordance with its policy on objectivity in research.

A PROOF OF THEORETICAL RESULTS

In this appendix we prove several properties of the SEC policy gradient estimator that are used in the variance analysis presented in the main paper. Before we present the proofs, we recall the assumption made in Section 4.1 of the main text:

(1) The action space is discrete and if a state is observed then all actions have also been observed in that state.
(2) The return estimates $\widehat{Q}^{\pi_{\boldsymbol{\theta}}}$ for any (s, a) is a fixed constant that is independent of $\mathcal{D}$.
(3) For all observed states, our estimated policy $\pi_{\boldsymbol{\phi}}$ is equal to $\pi_{\mathcal{D}}$, i.e., if action a occurs k times in state s and s occurs n times in $\mathcal{D}$ then $\pi_{\boldsymbol{\phi}}(a|s) = \frac{k}{n}$.

Under these assumptions, we make the following two claims about $g_{\mathtt{sec}}$:

CLAIM 1. $\mathrm{Var}_{\mathbb{A}}\left(g_{\mathtt{sec}}(\{\mathbb{S}, \mathbb{A}\}|\mathbb{S})\right) = 0.$

CLAIM 2. $\mathbf{E}_{\mathbb{A}}\left[g_{\mathtt{sec}}(\{\mathbb{S}, \mathbb{A}\})|\mathbb{S}\right] = \mathbf{E}_{\mathbb{A}}\left[g_{\mathtt{mc}}(\mathbb{S}, \mathbb{A})|\mathbb{S}\right].$

Recall that we can write either $g_{\mathtt{mc}}$ or $g_{\mathtt{sec}}$ as:

$$g_{\mathtt{sec}}(\{\mathbb{S}, \mathbb{A}\}) = \sum_{s \in \mathcal{S}} \rho_{\mathcal{D}}(s) \sum_{a \in \mathcal{A}} \pi_{\mathcal{D}}(a|s) w(s, a) f(s, a) \quad (6)$$

where $f(s, a) = \widehat{Q}(s, a) \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a|s)$.

In both Claim 1 and Claim 2, the sampled states are fixed and variance only arises from $\pi_{\mathcal{D}}$ and $w(s, a)$ which vary for different realizations of $\mathbb{A}$. When we choose $w(s, a) = \frac{\pi_{\boldsymbol{\theta}}(a|s)}{\pi_{\mathcal{D}}(a|s)}$ (as SEC does) the $\pi_{\mathcal{D}}(a|s)$ factors cancel in 6. Since $\pi_{\mathcal{D}}$ is the only part of $g_{\mathtt{sec}}$ that depends on the random variable $\mathbb{A}$, using $w(s, a)$ eliminates variance due to action selection in the estimator. This proves Claim 1.

Claim 2 also follows from the above discussion. The cancellation of the $\pi_{\mathcal{D}}(a|s)$ factors converts the inner summation over actions into an exact expectation under $\pi_{\boldsymbol{\theta}}$. Since $g_{\mathtt{mc}}$ is an unbiased estimator, the inner summation over actions must be equal to the exact expectation under $\pi_{\boldsymbol{\theta}}$ *in expectation*. Thus the expectation of both estimators *conditioned on* $\mathbb{S}$ is:

$$\mathbf{E}_{\mathbb{A}}\left[g(\{\mathbb{S}, \mathbb{A}\})|\mathbb{S}\right] = \sum_{s \in \mathcal{S}} \rho_{\mathcal{D}}(s) \sum_{a \in \mathcal{A}} \pi_{\boldsymbol{\theta}}(a|s) w(s, a) f(s, a) \quad (7)$$

This proves Claim 2.

REFERENCES

[1] Kamil Ciosek and Shimon Whiteson. 2018. Expected Policy Gradients for Reinforcement Learning. AAAI Conference on Artificial Intelligence (2018).

[2] Bernard Delyon and François Portier. 2016. Integral approximation by kernel smoothing. *Bernoulli* 22, 4 (2016), 2177–2208.

[3] Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu. 2017. OpenAI Baselines. https://github.com/openai/baselines. (2017).

[4] Evan Greensmith, Peter L Bartlett, Jonathan Baxter, et al. 2001. Variance Reduction Techniques for Gradient Estimates in Reinforcement Learning.. In *NIPS*. 1507–1514.

[5] Shixiang Gu, Timothy Lillicrap, Zoubin Ghahramani, Richard E Turner, and Sergey Levine. 2017. Q-prop: Sample-efficient policy gradient with an off-policy critic. In *International Conference on Learning Representations*.

[6] Shixiang Gu, Tim Lillicrap, Richard E Turner, Zoubin Ghahramani, Bernhard Schölkopf, and Sergey Levine. 2017. Interpolated policy gradient: Merging on-policy and off-policy gradient estimation for deep reinforcement learning. In *Advances in Neural Information Processing Systems*. 3846–3855.

[7] Josiah P. Hanna, Scott Niekum, and Peter Stone. 2018. Importance Sampling Policy Evaluation with an Estimated Behavior Policy. *arXiv preprint arXiv:1806.01347* (2018).

[8] Masayuki Henmi, Ryo Yoshida, and Shinto Eguchi. 2007. Importance sampling via the estimated sampler. *Biometrika* 94, 4 (2007), 985–991.

[9] Keisuke Hirano, Guido W Imbens, and Geert Ridder. 2003. Efficient estimation of average treatment effects using the estimated propensity score. *Econometrica* 71, 4 (2003), 1161–1189.

[10] Sergey Levine and Vladlen Koltun. 2013. Guided Policy Search. In *International Conference on Machine Learning, ICML*.

[11] Lihong Li, Rémi Munos, and Csaba Szepesvári. 2015. Toward minimax off-policy value estimation. In *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics (AISTATS)*.

[12] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *CoRR* abs/1509.02971 (2015).

[13] Hao Liu, Yihao Feng, Yi Mao, Dengyong Zhou, Jian Peng, and Qiang Liu. 2018. Action-dependent control variates for policy optimization via stein identity. (2018).

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

[15] Rémi Munos, Tom Stepleton, Anna Harutyunyan, and Marc G Bellemare. 2016. Safe and Efficient Off-Policy Reinforcement Learning. In *In Proceedings of Neural and Information Processing Systems*.

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

[17] Aniruddh Raghu, Omer Gottesman, Yao Liu, Matthieu Komorowski, Aldo Faisal, Finale Doshi-Velez, and Emma Brunskill. 2018. Behaviour Policy Estimation in Off-Policy Policy Evaluation: Calibration Matters. *arXiv preprint arXiv:1807.01066* (2018).

[18] Paul R Rosenbaum. 1987. Model-based direct adjustment. *J. Amer. Statist. Assoc.* 82, 398 (1987), 387–394.

[19] John Schulman, Sergey Levine, Philipp Moritz, Michael Jordan, and Pieter Abbeel. 2015. Trust Region Policy Optimization. In *International Conference on Machine Learning, ICML*.

[20] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. 2015. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438* (2015).

[21] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[22] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Weirstra, and Martin Riedmiller. 2014. Deterministic Policy Gradient Algorithms. In *ICML*.

[23] Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 2000. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *NIPS*.

[24] Phillip S. Thomas. 2014. Bias in Natural Actor-Critic Algorithms. In *31st International Conference on Machine Learning, ICML*.

[25] George Tucker, Surya Bhupatiraju, Shixiang Gu, Richard E Turner, Zoubin Ghahramani, and Sergey Levine. 2018. The mirage of action-dependent baselines in reinforcement learning. *arXiv preprint arXiv:1802.10031* (2018).

[26] Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. 2016. Sample efficient actor-critic with experience replay. *arXiv preprint arXiv:1611.01224* (2016).

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