Clipped Action Policy Gradient

Yasuhiro Fujita [1] **Shin-ichi Maeda** [1]

Abstract

Many continuous control tasks have bounded action spaces and clip out-of-bound actions before execution. Policy gradient methods often optimize policies as if actions were not clipped. We propose clipped action policy gradient (CAPG) as an alternative policy gradient estimator that exploits the knowledge of actions being clipped to reduce the variance in estimation. We prove that CAPG is unbiased and achieves lower variance than the original estimator that ignores action bounds. Experimental results demonstrate that CAPG generally outperforms the original estimator, indicating its promise as a better policy gradient estimator for continuous control tasks.

1. Introduction

Reinforcement learning (RL) has achieved remarkable success in recent years in a wide range of challenging tasks, such as games (Mnih et al., 2015; Silver et al., 2016; 2017), robotic manipulation (Levine et al., 2016), and locomotion (Schulman et al., 2015; 2017; Heess et al., 2017), with the help of deep neural networks. Policy gradient methods are among the most successful model-free RL approaches (Mnih et al., 2016; Schulman et al., 2015; 2017; Gu et al., 2017b). They are particularly suitable for continuous control tasks, i.e., environments with continuous action spaces, because they directly improve policies that represent continuous distributions of actions in order to maximize expected returns. For continuous control tasks, policies are typically represented by Gaussian distributions conditioned on current and past observations.

Although Gaussian policies have unbounded support, continuous control tasks often have bounded action sets that they can execute (Duan et al., 2016; Brockman et al., 2016; Tassa et al., 2018). For example, when controlling the torques of motors, effective torque values will be physically constrained. Policies with unbounded support like Gaussian policies are usually applied to such tasks by clipping sampled actions into their bounds (Duan et al., 2016; Dhariwal et al., 2017). Policy gradients for such policies are estimated as if actions were not clipped (Chou et al., 2017).

In this study, we demonstrate that we can improve policy gradient methods by exploiting the knowledge of actions being clipped. We prove that the variance of policy gradient estimates can be strictly reduced under mild assumptions that hold for popular policy classes such as Gaussian policies with diagonal covariance matrices. Our proposed algorithm, termed clipped action policy gradient (CAPG), is an alternative unbiased policy gradient estimator with lower variance than the original estimator. Our experimental results on challenging MuJoCo-simulated continuous control benchmark problems (Todorov et al., 2012; Brockman et al., 2016) show that CAPG generally outperforms the original policy gradient estimator when incorporated into deep RL algorithms.

2. Preliminaries

We consider a Markov decision process (MDP) defined by the tuple $(\mathcal{S}, \mathcal{A}, P, r, \rho_0, \gamma)$, where $\mathcal{S}$ is a set of possible states, $\mathcal{A}$ is a set of possible actions, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the transition probability distribution, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $\rho_0 : \mathcal{S} \to [0, 1]$ is the distribution of the initial state s_0, and $\gamma \in (0, 1]$ is the discount factor.

A probability distribution of action conditioned on state is referred to as a policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$. RL algorithms aim to find a policy that maximizes the expected cumulative discounted reward from initial states.

$$\eta(\pi) = \mathbb{E}_{s_0,a_0,\dots}\Big[\sum_t \gamma^t r(s_t, a_t)\Big],$$

where $\mathbb{E}_{s_0,a_0,\dots}[\cdot]$ denotes an expected value with respect to a state-action-reward sequence $s_0 \sim \rho_0(\cdot)$, $\{a_t \sim \pi(\cdot|s_t), s_{t+1} \sim P(\cdot|s_t, a_t) | t = 1, \cdots\}$.

The state-action value function of a policy π is defined as

$$Q^\pi(s, a) = \mathbb{E}_{s_1,a_1,\dots}\Big[\sum_t \gamma^t r(s_t, a_t) | s_0 = s, a_0 = a\Big].$$

One way to find $\pi^* = \mathrm{argmax}_\pi \eta(\pi)$ is to adjust the parameters θ of a parameterized policy π_θ by following the gradient

[1]Preferred Networks, Inc., Japan. Correspondence to: Yasuhiro Fujita <fujita@preferred.jp>, Shin-ichi Maeda <ichi@preferred.jp>.

$\nabla_\theta \eta(\pi_\theta)$, which is referred to a policy gradient. The policy gradient theorem (Sutton et al., 1999) states that

$$\nabla_\theta \eta(\pi_\theta) = \mathbb{E}_s \Big[\mathbb{E}_a [Q^{\pi_\theta}(s,a) \nabla_\theta \log \pi_\theta(a|s) | s] \Big], \quad (1)$$

where $\mathbb{E}_a[\cdot|s]$ denotes a conditional expected values with respect to $\pi_\theta(\cdot|s)$, and $\mathbb{E}_s[\cdot]$ denotes an (improper) expected value with respect to the (improper) discounted state distribution $\rho^{\pi_\theta}(s)$, which is defined as

$$\rho^{\pi}(s) = \sum_t \gamma^t \int \rho_0(s_0) p(s_t = s | s_0, \pi) ds_0.$$

In practice, the policy gradient is often estimated by a finite number of samples $\{(s^i, a^i) | a^i \sim \pi_\theta(\cdot|s^i), i = 1, \dots, N\}$.

$$\nabla_\theta \eta(\pi_\theta) \approx \frac{1}{N} \sum_i Q^{\pi_\theta}(s^i, a^i) \nabla_\theta \log \pi_\theta(a^i|s^i). \quad (2)$$

RL algorithms that rely on this estimation are referred to as policy gradient methods. While this estimation is unbiased, its variance is typically high, which is considered as a crucial problem of policy gradient methods.

The variance of the estimation above is proportional to $\mathbb{V}_{s,a}[Q^{\pi_\theta}(s,a) \nabla_\theta \log \pi_\theta(a|s)]$, which is decomposed by the law of total variance:

$$\begin{aligned} &\mathbb{V}_{s,a}[Q^{\pi_\theta}(s,a) \nabla_\theta \log \pi_\theta(a|s)] \\ &\quad = \mathbb{V}_s[\mathbb{E}_a[Q^{\pi_\theta}(s,a) \nabla_\theta \log \pi_\theta(a|s) | s]] \\ &\quad\quad + \mathbb{E}_s[\mathbb{V}_a[Q^{\pi_\theta}(s,a) \nabla_\theta \log \pi_\theta(a|s) | s]]. \end{aligned}$$

Therefore, reduction of $\mathbb{V}_a[Q^{\pi_\theta}(s,a) \nabla_\theta \log \pi_\theta(a|s) | s]$ for each s will reduce the total variance.

For notational simplicity, $\mathbb{E}_a[\cdot|s]$ and $\mathbb{V}_a[\cdot|s]$ are written as $\mathbb{E}_a[\cdot]$ and $\mathbb{V}_a[\cdot]$ below, respectively.

3. Clipped Action Policy Gradient

We consider the case where any action $a \in \mathbb{R}^d$ chosen by the agent is clipped by the environment into a range $[\alpha, \beta] \subset \mathbb{R}^d$. That is, the transition probability and the reward function are written as

$$P(s, a, s') = P(s, \text{clip}(a, \alpha, \beta), s'), \quad (3)$$

$$r(s, a) = r(s, \text{clip}(a, \alpha, \beta)), \quad (4)$$

respectively. The clip function is defined as $\text{clip}(a, \alpha, \beta) = \max(\min(a, \beta), \alpha)$, where $\max, \min$ are computed elementwise when a is a vector, i.e., $d \geq 2$. Each of α and β can be a constant or a function of s. The case where the reward function depends on actions before clipping is discussed in Section 3.4.

Suppose we aim to estimate the value of $\mathbb{E}[X]$ in an unbiased and lower-variance manner than using the sample average of X. To this end, we should use the sample average of Y instead of X when its variance is lower, i.e., $\mathbb{V}[Y] < \mathbb{V}[X]$, as long as it is unbiased, i.e., $\mathbb{E}[Y] = \mathbb{E}[X]$. Below, we derive such Y to estimate the policy gradient for given s, where $X = Q^{\pi_\theta}(s,a) \nabla_\theta \log \pi_\theta(a|s)$.

Before explaining our algorithm, let us characterize the class of policies we consider in this study.

Definition 3.1 (compatible distribution)**.** *Let $p_\theta(a)$ be a distribution of $a \in \mathbb{R}$ that has a parameter θ. If $p_\theta(a)$ is differentiable with respect to θ and allows the exchange of derivative and integral as $\int_{-\infty}^{\alpha} \nabla_\theta p_\theta(a) da = \nabla_\theta \int_{-\infty}^{\alpha} p_\theta(a) da$ and $\int_{\beta}^{\infty} \nabla_\theta p_\theta(a) da = \nabla_\theta \int_{\beta}^{\infty} p_\theta(a) da$, we call $p_\theta(a)$ a compatible distribution. If $p_\theta(a|s)$ is a compatible distribution conditioned on the variable s, we call it a compatible conditional distribution.*

3.1. The case of scalar actions

First, we derive a lower-variance estimator of the policy gradient for scalar actions, i.e., $d = 1$. The case of vector actions will be covered later in Section 3.2.

From (3) and (4), the state-action value function is written as

$$\begin{aligned} Q^{\pi_\theta}(s,a) &= Q^{\pi_\theta}(s, \text{clip}(a, \alpha, \beta)) \\ &= \begin{cases} Q^{\pi_\theta}(s, \alpha) & \text{if } a \leq \alpha \\ Q^{\pi_\theta}(s, a) & \text{if } \alpha < a < \beta \\ Q^{\pi_\theta}(s, \beta) & \text{if } \beta \leq a \end{cases}. \end{aligned} \quad (5)$$

To derive our unbiased estimator, we decompose the expected value. Let X be a function of a random variable a and $\mathbb{1}_{f(x)}$ be an indicator function that takes 1 when x satisfies the condition $f(x)$ otherwise 0. Since $X = \mathbb{1}_{a \leq \alpha} X + \mathbb{1}_{\alpha < a < \beta} X + \mathbb{1}_{\beta \leq a} X$, $\mathbb{E}_a[X]$ can be decomposed as

$$\mathbb{E}_a[X] = \mathbb{E}_a[\mathbb{1}_{a \leq \alpha} X] + \mathbb{E}_a[\mathbb{1}_{\alpha < a < \beta} X] + \mathbb{E}_a[\mathbb{1}_{\beta \leq a} X]. \quad (6)$$

From (5) and (6), we immediately have

$$\begin{aligned} &\mathbb{E}_a[Q^{\pi_\theta}(s,a) \nabla_\theta \log \pi_\theta(a|s)] \\ &\quad = Q^{\pi_\theta}(s, \alpha) \mathbb{E}_a[\mathbb{1}_{a \leq \alpha} \nabla_\theta \log \pi_\theta(a|s)] \\ &\quad\quad + \mathbb{E}_a[\mathbb{1}_{\alpha < a < \beta} Q^{\pi_\theta}(s,a) \nabla_\theta \log \pi_\theta(a|s)] \\ &\quad\quad + Q^{\pi_\theta}(s, \beta) \mathbb{E}_a[\mathbb{1}_{\beta \leq a} \nabla_\theta \log \pi_\theta(a|s)]. \end{aligned} \quad (7)$$

Meanwhile, the following useful lemma holds for compatible conditional distributions.

Lemma 3.1. *Suppose $\pi_\theta(a|s)$ is a compatible conditional distribution whose cumulative distribution function is*

$\Pi_\theta(a|s)$. *Then, the following equations hold.*

$$\mathbb{E}_a[\mathbb{1}_{a\le\alpha}\nabla_\theta \log \pi_\theta(a|s)] = \mathbb{E}_a[\mathbb{1}_{a\le\alpha}\nabla_\theta \log \Pi_\theta(\alpha|s)],$$
$$\mathbb{E}_a[\mathbb{1}_{\beta\le a}\nabla_\theta \log \pi_\theta(a|s)] = \mathbb{E}_a[\mathbb{1}_{\beta\le a}\nabla_\theta \log(1-\Pi_\theta(\beta|s))].$$

See the appendix for the proof.

By applying Lemma 3.1 to (7), we can construct an alternative estimator

$$\begin{aligned}\mathbb{E}_a&[Q^{\pi_\theta}(s,a)\nabla_\theta \log \pi_\theta(a|s)]\\ &= Q^{\pi_\theta}(s,\alpha)\mathbb{E}_a[\mathbb{1}_{a\le\alpha}\nabla_\theta \log \Pi_\theta(\alpha|s)]\\ &\quad + \mathbb{E}_a[\mathbb{1}_{\alpha<a<\beta}Q^{\pi_\theta}(s,a)\nabla_\theta \log \pi_\theta(a|s)]\\ &\quad + Q^{\pi_\theta}(s,\beta)\mathbb{E}_a[\mathbb{1}_{\beta\le a}\nabla_\theta \log(1-\Pi_\theta(\beta|s))]\\ &= \mathbb{E}_a[Q^{\pi_\theta}(s,a)\bar{\psi}(s,a)], \end{aligned} \tag{8}$$

where

$$\bar{\psi}(s,a) = \begin{cases} \nabla_\theta \log \Pi_\theta(\alpha|s) & \text{if } a\le\alpha \\ \nabla_\theta \log \pi_\theta(a|s) & \text{if } \alpha<a<\beta \\ \nabla_\theta \log(1-\Pi_\theta(\beta|s)) & \text{if } \beta\le a \end{cases}. \tag{9}$$

The right-hand side of (8) can be estimated using the sample average for given s. This estimator, which we call clipped action policy gradient (CAPG), is better than the original gradient estimator (2) in the sense that it has lower variance while being unbiased.

It is obvious that the difference between the original estimator (7) and CAPG (8) comes from the endpoints. $\pi_\theta(a|s)$ of $\nabla_\theta \log \pi_\theta(a|s)$ is replaced with $\Pi_\theta(\alpha|s)$ and $1-\Pi_\theta(\beta|s)$ at $a\le\alpha$ and $\beta\le a$, respectively. Intuitively speaking, since either of $\Pi_\theta(\alpha|s)$ or $1-\Pi_\theta(\beta|s)$ is not a random variable given s, the variance should be decreased; in fact, this is true.

To show this, we need to decompose the variance. The variance of X that is a function of random variable a, can be decomposed by utilizing the fact $X = \mathbb{1}_{a\le\alpha}X + \mathbb{1}_{\alpha<a<\beta}X + \mathbb{1}_{\beta\le a}X$.

$$\begin{aligned}\mathbb{V}_a[X] = \mathbb{V}_a[\mathbb{1}_{a\le\alpha}X] &+ \mathbb{V}_a[\mathbb{1}_{\alpha<a<\beta}X] + \mathbb{V}_a[\mathbb{1}_{\beta\le a}X]\\ &- 2\mathbb{E}_a[\mathbb{1}_{a\le\alpha}X]\mathbb{E}_a[\mathbb{1}_{\alpha<a<\beta}X]\\ &- 2\mathbb{E}_a[\mathbb{1}_{\alpha<a<\beta}X]\mathbb{E}_a[\mathbb{1}_{\beta\le a}X]\\ &- 2\mathbb{E}_a[\mathbb{1}_{\beta\le a}X]\mathbb{E}_a[\mathbb{1}_{a\le\alpha}X].\end{aligned}$$

Let us compare each term of the right-hand side between the cases $X = Q^{\pi_\theta}(s,a)\nabla_\theta \log \pi_\theta(a|s)$ and $X = Q^{\pi_\theta}(s,a)\bar{\psi}(s,a)$. By using Lemma 3.1, we will see that $\mathbb{V}_a[\mathbb{1}_{\alpha<a<\beta}X]$, $\mathbb{E}_a[\mathbb{1}_{a\le\alpha}X], \mathbb{E}_a[\mathbb{1}_{a\le\alpha}X]$ and $\mathbb{E}_a[\mathbb{1}_{a\le\alpha}X]$ are not different, and the difference arises only from the terms $\mathbb{V}_a[\mathbb{1}_{a\le\alpha}X]$ and $\mathbb{V}_a[\mathbb{1}_{\beta\le a}X]$.

Actually, both of the terms become smaller when CAPG is used.

Lemma 3.2. *Suppose $\pi_\theta(a|s)$ is a compatible conditional distribution whose cumulative distribution function is $\Pi_\theta(a|s)$. Then, the following inequalities hold:*

$$\mathbb{V}_a[\mathbb{1}_{a\le\alpha}\nabla_\theta \log \pi_\theta(a|s)] \ge \mathbb{V}_a[\mathbb{1}_{a\le\alpha}\nabla_\theta \log \Pi_\theta(\alpha|s)],$$
$$\mathbb{V}_a[\mathbb{1}_{\beta\le a}\nabla_\theta \log \pi_\theta(a|s)] \ge \mathbb{V}_a[\mathbb{1}_{\beta\le a}\nabla_\theta \log(1-\Pi_\theta(\beta|s))].$$

See the appendix for the proof.

Therefore, we can conclude that CAPG has lower variance than the original estimator while being unbiased. In general, we have the following result.

Lemma 3.3. *Suppose $\pi_\theta(a|s)$ is a compatible conditional distritbuion whose cumulative distribution function is $\Pi_\theta(\cdot|s)$. Let $\psi(s,a) = \nabla_\theta \log \pi_\theta(a|s)$, and $\bar{\psi}(s,a)$ be its CAPG version (9). Let $f(s,a)$ be a function such that*

$$f(s,a) = \begin{cases} f(s,\alpha) & \text{if } a\le\alpha \\ f(s,a) & \text{if } \alpha<a<\beta \\ f(s,\beta) & \text{if } \beta\le a \end{cases}.$$

Then, the following equality and inequality hold:

$$\mathbb{E}_a[f(s,a)\bar{\psi}(s,a)] = \mathbb{E}_a[f(s,a)\psi(s,a)],$$
$$\mathbb{V}_a[f(s,a)\bar{\psi}(s,a)] \le \mathbb{V}_a[f(s,a)\psi(s,a)].$$

3.2. The case of vector actions

The results in the previous subsection can be extended to the case of vector actions as long as each action dimension of the policy is conditionally independent, i.e.,

$$\pi_\theta(a|s) = \pi_\theta(a_1|s)\pi_\theta(a_2|s)\cdots\pi_\theta(a_d|s),$$

which includes the case for a multivariate Gaussian policy with a diagonal covariance. Before going to the details, let us use $\psi(s,a) = \nabla_\theta \log \pi_\theta(a|s)$ and $Q(s,a) = Q^{\pi_\theta}(s,a)$ for notational simplicity. Since each action is conditionally independent, we can decompose the variance as

$$\begin{aligned}\mathbb{V}_a&[Q(s,a)\psi(s,a)]\\ &= \sum_{i=1}^{d}\mathbb{V}_a[Q(s,a)\psi(s,a_i)]\\ &\quad + 2\sum_{1\le i<j\le d}\mathrm{Cov}[Q(s,a)\psi(s,a_i), Q(s,a)\psi(s,a_j)],\end{aligned}$$

where

$$\begin{aligned}\mathrm{Cov}&[Q(s,a)\psi(s,a_i), Q(s,a)\psi(s,a_j)]\\ &= \mathbb{E}_a[(Q(s,a))^2\psi(s,a_i)\psi(s,a_j)]\\ &\quad - \mathbb{E}_a[Q(s,a)\psi(s,a_i)]\mathbb{E}_a[Q(s,a)\psi(s,a_j)].\end{aligned}$$

We need to show that the variance will not increase by using $\bar{\psi}(s,a_i)$ instead of $\psi(s,a_i)$ for all $1 \le i \le d$. Applying Lemma 3.3, we have

$$\begin{aligned}\mathbb{V}_a[Q(s,a)\bar{\psi}(s,a_i)] &\le \mathbb{V}_a[Q(s,a)\psi(s,a_i)],\\ \mathbb{E}_a[Q(s,a)\bar{\psi}(s,a_i)] &= \mathbb{E}_a[Q(s,a)\psi(s,a_i)].\end{aligned}$$

Using the law of total expectation,

$$\begin{aligned}&\mathbb{E}_a[(Q(s,a))^2\psi(s,a_i)\psi(s,a_j)]\\ &\quad= \mathbb{E}_{a\setminus i,j}[\mathbb{E}_{a_i}[\mathbb{E}_{a_j}[(Q(s,a))^2\psi(s,a_i)\psi(s,a_j)]]]\\ &\quad= \mathbb{E}_{a\setminus i,j}[\mathbb{E}_{a_i}[\psi(s,a_i)\mathbb{E}_{a_j}[(Q(s,a))^2\psi(s,a_j)]]],\end{aligned}$$

where $a \setminus j$ denotes a vector a excluding the element j.

Noting the fact that $(Q(s,a))^2$ is a function of a_j conditioned on s and $a \setminus j$, we can immediately have the following equation by applying Lemma 3.3.

$$\mathbb{E}_{a_j}[(Q(s,a))^2\bar{\psi}(s,a_j)] = \mathbb{E}_{a_j}[(Q(s,a))^2\psi(s,a_j)].$$

Similarly, we can derive the following equations based on the fact that $\mathbb{E}_{a_j}[(Q(s,a))^2\psi(s,a_j)]$ is a function of a_i conditioned on s and $a \setminus i,j$.

$$\begin{aligned}&\mathbb{E}_{a_i}[\bar{\psi}(s,a_i)\mathbb{E}_{a_j}[(Q(s,a))^2\bar{\psi}(s,a_j)]]\\ &\quad= \mathbb{E}_{a_i}[\psi(s,a_i)\mathbb{E}_{a_j}[(Q(s,a))^2\psi(s,a_j)]],\\ &\mathbb{E}_a[(Q(s,a))^2\bar{\psi}(s,a_i)\bar{\psi}(s,a_j)]\\ &\quad= \mathbb{E}_a[(Q(s,a))^2\psi(s,a_i)\psi(s,a_j)]\end{aligned}$$

Therefore, the following inequality holds for vector actions as well.

$$\mathbb{V}_a[Q(s,a)\bar{\psi}(s,a)] \le \mathbb{V}_a[Q(s,a)\psi(s,a)].$$

3.3. Implementation

CAPG can be easily incorporated into existing policy gradient methods. In short, we only have to replace the computation of $\nabla_\theta \log \pi_\theta(a|s)$ with that of $\bar{\psi}(s,a)$ to use CAPG. When $\nabla_\theta \log \pi_\theta(a|s)$ is computed using any automatic differentiation library, we can instead replace $\log \pi_\theta(a|s)$ with

$$f_\theta(s,a) = \begin{cases}\log \Pi_\theta(\alpha|s) & \text{if } a \le \alpha\\ \log \pi_\theta(a|s) & \text{if } \alpha < a < \beta\\ \log(1-\Pi_\theta(\beta|s)) & \text{if } \beta \le a\end{cases}.$$

3.4. Extensions

Although we have used standard notations of MDPs, our results do not rely on the Markov property. CAPG works as an unbiased and low-variance policy gradient estimator in non-Markovian environments, in the same way that the REINFORCE algorithm (Williams, 1992) works in such environments.

We assumed (4) so that $Q^{\pi_\theta}(s,a)$ becomes a constant outside the action bounds. However, sometimes it makes sense to use a reward function that depends on out-of-bound actions even when the state transition dynamics does not, e.g., to penalize the norm of actions in order to prevent the policy from going too far out of the bounds. With such a reward function, (5) no longer holds. Instead, we can use the recursive structure of $Q^{\pi_\theta}(s,a)$ to obtain

$$\begin{aligned}&\mathbb{E}_a[Q^{\pi_\theta}(s,a)\psi(s,a)]\\ &\quad= \mathbb{E}_a[r(s,a)\psi(s,a)]\\ &\qquad+ \mathbb{E}_a[\gamma\mathbb{E}_{s',a'}[Q^{\pi_\theta}(s',a')]\psi(s,a)]],\end{aligned} \tag{10}$$

where $\mathbb{E}_{s',a'}[\cdot]$ denotes an expected value with respect to $s' \sim P(s, \text{clip}(a,\alpha,\beta), \cdot)$, $a' \sim \pi_\theta(\cdot|s')$. We can apply CAPG to the second term of the right-hand side of (10) because $\gamma\mathbb{E}_{s',a'}[Q^{\pi_\theta}(s',a')]$ only depends on a via $\text{clip}(\cdot,\alpha,\beta)$.

3.5. Clipped distribution

So far we have derived CAPG as a better policy gradient estimator. We now argue that CAPG can be interpreted as estimating the policy gradient of a different policy.

Given a policy π_θ and action bounds $[\alpha,\beta]$, we can consider a policy π^c_θ modeled as a probability distribution with bounded support whose cumulative distribution function is defined as $\Pi^c_\theta(a|s) = \mathbb{1}_{\alpha\le a<\beta}\Pi_\theta(a|s) + \mathbb{1}_{\beta\le a}$, which is a mixture of two degenerate distributions at $\{\alpha,\beta\}$ and a truncated version of π_θ. The corresponding probability density function with respect to the measure generated by the mixture [1] is given by

$$\pi^c_\theta(a|s) = \begin{cases}\Pi_\theta(\alpha|s) & \text{if } a = \alpha\\ \pi_\theta(a|s) & \text{if } \alpha < a < \beta\\ 1-\Pi_\theta(\beta|s) & \text{if } a = \beta\end{cases}.$$

We call this distribution a clipped distribution. Seeing that $\nabla_\theta \log \pi^c_\theta(a|s) = \bar{\psi}(s,a)$ for $a \in [\alpha,\beta]$, CAPG applied to π_θ is in fact estimating the policy gradient of π^c_θ. If we see Gaussian policies used with action bounds as clipped Gaussian policies, then CAPG is the straightforward policy gradient estimator for them, whereas the conventional estimator has unnecessarily high variance.

While a clipped distribution resembles a truncated distribution, it can be multimodal even when its underlying distribution is unimodal because it has peaks at the action bounds. In contrast, a truncated distribution is always unimodal when its underlying distribution is unimodal, which implies its inferior representational power in modeling a policy.

[1] The probability measure P corresponding to $\Pi^c_\theta(a|s)$, defined over the measurable space $([\alpha,\beta],\mathcal{B}([\alpha,\beta]))$, is such that $P \ll \lambda + \delta_\alpha + \delta_\beta$, where $\mathcal{B}$ is the Borel σ-algebra, λ is the Lebesgue measure and δ_x is a Dirac measure at x.

4. Experiments

In this section, we evaluate the performance of CAPG compared to the standard policy gradient estimator, which we call PG, in problems with action bounds.

4.1. Continuum-armed bandit problems

To demonstrate how CAPG works and how it interacts with each aspect of problems separately, we used continuum-armed bandit problems (Agrawal, 1995), i.e., MDPs with continuous action spaces and no state transitions. State-independent policies were optimized by policy gradients to maximize action-dependent immediate rewards.

The action space was $[-1, 1]^d, d \in \mathbb{N}$ and the reward function was defined as $r(a) = -\frac{1}{d}\sum_i |a_i|$ so that only choosing the optimal action of zeros achieves the maximum, zero reward.

Each policy was modeled as a multivariate Gaussian distribution with a diagonal covariance matrix and parameterized by $\theta = \{\theta_\mu, \theta_\Sigma\}$, where $\theta_\mu \in \mathbb{R}^n$ is the mean vector and $\theta_\Sigma \in \mathbb{R}^n$ is the main diagonal of the covariance matrix.

The following experimental settings were used unless otherwise stated. Actions were scalars, i.e., $d = 1$. The parameters of a policy were initialized as zero mean and unit variance for each dimension. Each policy update used a batch of 5 (action, reward) pairs. The average reward in a batch was used as a baseline that was subtracted from each reward. Adam (Kingma & Ba, 2015) with its default hyperparameters was used to update the parameters.

To quantify variance reduction achieved by CAPG, we repeatedly estimated policy gradients using new samples without updating a policy. Figure 1 shows mean and standard deviation of policy gradient estimates obtained by CAPG and PG with varying mean and variance of the fixed policy. For both θ_μ and θ_Σ in all settings, CAPG consistently achieved lower variance than PG without introducing visible bias. These results numerically corroborate CAPG's variance reduction ability as well as its unbiasedness. The efficacy of CAPG diminished at $\sigma^2 = 0.1$, where sampled actions rarely go outside the bounds.

Figure 2 shows the training curves of CAPG and PG with four different aspects separately controlled: the variance of the initial policy, the mean of the initial policy, the number of dimensions of actions and the batch size. CAPG consistently achieved faster learning across the settings. Larger initial variance and more distant initial mean tend to make the gap more visible. CAPG's gain scales even for 100 dimensions, which implies its utility for more challenging, complex continuous control tasks. Using smaller batch sizes benefits more from CAPG, which is expected because smaller batch sizes are more affected by the variance of gradient estimation. With the batch size of 100, the training curve of CAPG is difficult to distinguish from that of PG. It should be noted that in these experiments all the actions are sampled from the same state. In practical model-free RL scenarios, you cannot sample more than one action from the same state.

	Obs. space	Action space
InvertedPendulum-v1	$\mathbb{R}^4$	$[-3.0, 3.0]^1$
InvertedDoublePendulum-v1	$\mathbb{R}^{11}$	$[-1.0, 1.0]^1$
Reacher-v1	$\mathbb{R}^{11}$	$[-1.0, 1.0]^2$
Hopper-v1	$\mathbb{R}^{11}$	$[-1.0, 1.0]^3$
HalfCheetah-v1	$\mathbb{R}^{17}$	$[-1.0, 1.0]^6$
Swimmer-v1	$\mathbb{R}^8$	$[-1.0, 1.0]^2$
Walker2d-v1	$\mathbb{R}^{17}$	$[-1.0, 1.0]^6$
Ant-v1	$\mathbb{R}^{111}$	$[-1.0, 1.0]^8$
Humanoid-v1	$\mathbb{R}^{376}$	$[-0.4, 0.4]^{17}$
HumanoidStandup-v1	$\mathbb{R}^{376}$	$[-0.4, 0.4]^{17}$

Table 1. MuJoCo-simulated environments used in the experiments and their observation and action spaces.

4.2. Simulated control problems

CAPG can also be easily incorporated into existing deep RL algorithms that rely on policy gradients. To evaluate its effectiveness in deep RL settings, we used the following two popular deep RL algorithms for continuous control:

- Proximal Policy Optimization (PPO) with clipped surrogate objective (Schulman et al., 2017)
- Trust Region Policy Optimization (TRPO) (Schulman et al., 2015) with Generalized Advantage Estimation (GAE) (Schulman et al., 2016)

For each of the two algorithms, we implemented the variant that uses CAPG as well as the original one that uses PG (1). The only difference between these two is whether CAPG or PG is used.

We used 10 MuJoCo-simulated environments implemented in OpenAI Gym for our experiments, which are widely used as benchmark tasks for deep RL algorithms (Schulman et al., 2017; Henderson et al., 2018; Ciosek & Whiteson, 2018; Gu et al., 2017b; Duan et al., 2016; Dhariwal et al., 2017). The names of the environments are listed along with their observation and action spaces in Table 1. All the environments have bounded action spaces; hence, actions are clipped before being sent to the environments.

We considered all the combinations of {PPO, TRPO} $\times$ {CAPG, PG} $\times$ 10 environments, each of which is trained for 1 million timesteps. Each combination is tried 50 times with different random seeds. Since we found it is difficult to obtain reasonable performance within 1 million timesteps

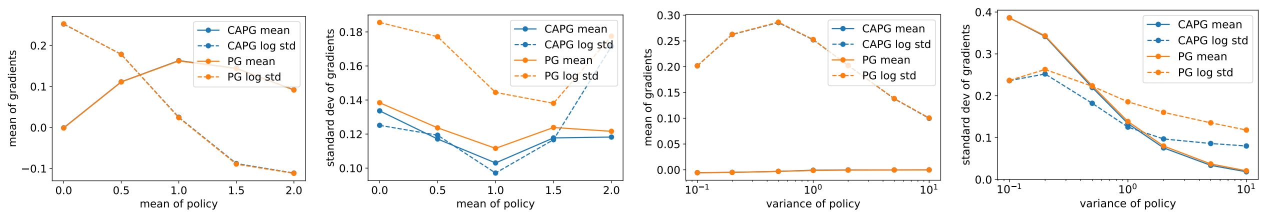

Figure 1. Mean and standard deviation of policy gradient estimates obtained by CAPG and PG on a continuum-armed bandit problem with varying mean (left half) and variance (right half) of the fixed policy. For each data points, policy gradients with respect to θ_μ and θ_Σ are estimated 10,000 times using 10,000 different batches of 5 (action, reward) pairs. The CAPG and PG plots of mean of gradients almost overlap each other, so only the PG plots are visible.

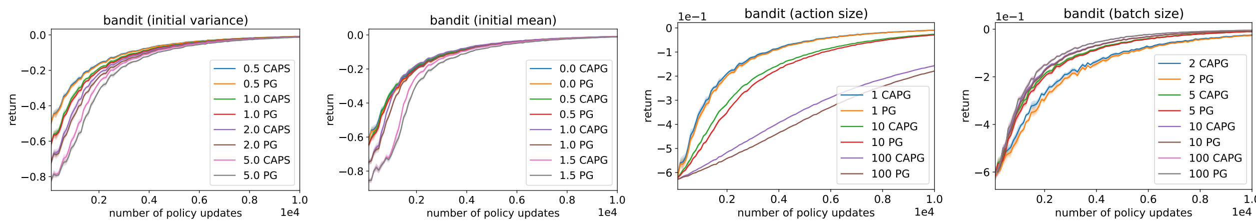

Figure 2. Training curves on continuum-armed bandit problems with four different aspects controlled: from left to right, the variance of the initial policy, the mean of the initial policy, the number of dimensions of actions and the batch size. For each run, the last reward before every policy update is sampled and then averaged over the last 100 updates to obtain a smoothed curve. The smoothed curves are then aggregated to compute the mean curves with 68% and 95% bootstrapped confidence intervals drawn as shaded areas.

on Ant-v1, Humanoid-v1, and HumanoidStandup-v1, we also tried training for 10 million timesteps on these environments.

We followed the hyperparameter settings used in (Henderson et al., 2018), except that the learning rate of Adam used by PPO was reduced to 3e-5 for 10 million timesteps training in order to obtain reasonable performance with PG. We used separate neural networks with two hidden layers, each of which has 64 hidden units with tanh nonlinearities, for both a policy and a state value function. The policy network outputs the mean of a multivariate Gaussian distribution with a diagonal covariance matrix. The main diagonal of the covariance matrix were separately parameterized as a logarithm of the standard deviation for each dimension.

Table 2 summarizes the comparison between CAPG and PG, combined with TRPO and PPO. We used areas under the learning curves (AUCs) as an evaluation measure since these can measure not only the final performance but also the learning speed and stability.

For PPO and TRPO with 1 million training timesteps, CAPG significantly ($p < 0.025$, i.e., $> 95\%$ significance) improved AUCs on 3 and 7 out of the 10 environments, respectively. It also significantly helped in training for 10 million timesteps on two out of the three harder environments for both PPO and TRPO. On other environments, it kept almost the same level of AUCs on other tasks, although there seemed to be slight decreases in some environments. These results indicate that CAPG can safely replace PG in many cases.

Figures 3,4 show the smoothed learning curves of all the experiments. In some cases, the improvements were small but consistent, e.g., TRPO on InvertedDoublePendulum-v1, TRPO on HumanoidStandup-v1 (10 million). In some other cases, the large improvements were achieved, e.g., PPO on Swimmer-v1, TRPO on Humanoid-v1 (10 million).

Although we used the hyperparameters tuned by (Henderson et al., 2018) for PG, the best hyperparameters can be different for CAPG since it affects the variance of policy gradient estimation. It is possible that separate hyperparameter tuning can further improve the performance of CAPG.

Comparing the results of PPO and TRPO, PPO was more affected than TRPO by the difference in estimators, suggesting that PPO is more vulnerable to high variance in gradient estimation. TRPO is likely to be more robust against variance for the following reasons.

- TRPO uses a large batch of 5000 actions for every policy update. PPO uses minibatches of 64 actions, resulting in more noisy updates.

- TRPO solves a constrained optimization problem for every policy update so that the change in KL divergence is close to a constant, thus it is robust to changes in the scale of gradients. PPO also adapts its step size

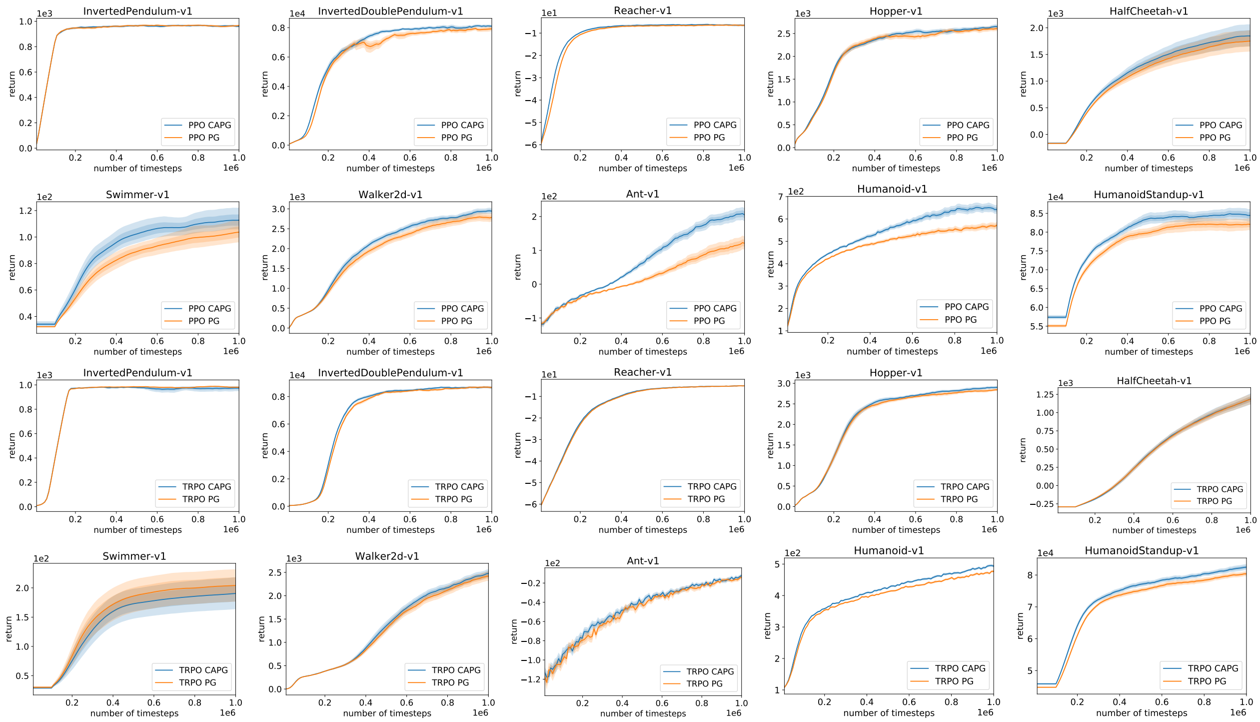

Figure 3. Training curves of PPO (upper half) and TRPO (lower half) on the 10 MuJoCo-simulated environments. For each run, the average of the last 100 training episodes after every training episode is computed and then linearly interpolated between episodes to obtain a smoothed curve. The smoothed curves are then aggregated to compute the mean curves with 68% and 95% bootstrapped confidence intervals drawn as shaded areas.

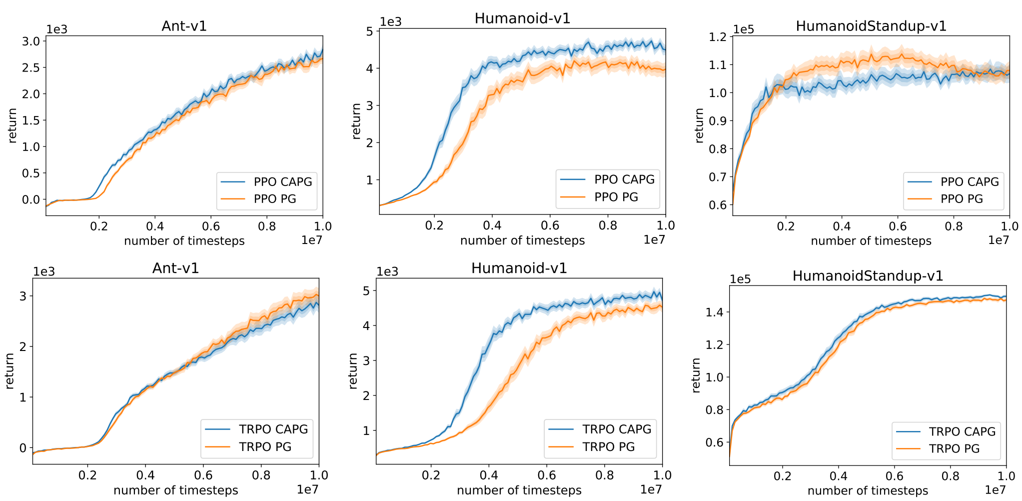

Figure 4. Training curves of PPO (upper half) and TRPO (lower half) on the three harder MuJoCo-simulated environments. For each run, the average of last 100 training episodes after every training episode is computed and then linearly interpolated between episodes to obtain a smoothed curve. The smoothed curves are then aggregated to compute the mean curves with 68% and 95% bootstrapped confidence intervals drawn as shaded areas.

using Adam, but only much more slowly, based on the statistics of accumulated past gradients.

Since we find that even TRPO can benefit from CAPG, we expect other algorithms with noisier updates to also benefit from it.

	PPO CAPG	PPO PG	p-value	TRPO CAPG	TRPO PG	p-value
InvertedPendulum-v1	955.30±1.12	955.68±0.84	7.88e-01	915.08±5.23	919.94±0.79	3.63e-01
InvertedDoublePendulum-v1	**7239.24±23.01**	6991.40±43.01	2.67e-06	**7108.54±18.17**	7007.32±18.95	2.07e-04
Reacher-v1	**-10.67±0.15**	-11.60±0.17	8.71e-05	-14.66±0.13	-14.93±0.13	1.41e-01
Hopper-v1	2320.49±11.49	2288.50±17.91	1.37e-01	2313.33±16.14	2283.55±16.03	1.94e-01
HalfCheetah-v1	1219.54±60.94	1144.53±58.81	3.78e-01	502.05±18.36	499.99±18.57	9.37e-01
Swimmer-v1	**92.56±3.48**	82.45±2.75	2.49e-02	148.86±11.44	161.18±11.92	4.58e-01
Walker2d-v1	**2185.63±26.23**	2060.95±38.92	9.41e-03	1436.38±30.31	1390.69±27.60	2.68e-01
Ant-v1	**56.85±5.19**	-33.32±7.26	2.01e-16	**-204.68±1.84**	-212.15±1.92	6.04e-03
Humanoid-v1	**547.64±5.90**	493.39±3.89	2.49e-11	**415.88±0.79**	402.19±0.75	3.96e-22
HumanoidStandup-v1	**79414.10±496.59**	76845.37±512.67	5.03e-04	**73592.94±292.50**	71796.93±265.64	1.58e-05
Ant-v1 (10m)	**1579.54±10.64**	1476.51±15.21	2.98e-07	1395.50±28.44	1449.61±32.05	2.10e-01
Humanoid-v1 (10m)	**3650.00±33.98**	3107.34±59.01	1.06e-11	**3353.08±23.57**	2743.53±40.29	1.99e-21
HumanoidStandup-v1 (10m)	101826.33±1012.21	105289.56±1173.48	2.78e-02	**123777.22±383.77**	120994.09±403.91	2.57e-06

Table 2. Performance comparison of CAPG and PG on the 10 MuJoCo-simulated environments. Performance is evaluated with the average area under the learning curve (AUC) ± standard error over 1 million timesteps. For each training run, its AUC is computed by linearly interpolating returns between training episodes. For each combination of $\{\text{TRPO, PPO}\} \times \{\text{CAPG, PG}\} \times 10$ environments, from 50 training runs with different random seeds, the average AUC and standard error is computed. p-values are also computed between CAPG and PG versions by Welch's t-test. Bold numbers indicate that they are better than their counterparts by 95% significance.

5. Related Work

A variety of techniques have been proposed to reduce the variance of policy gradient estimation since its introduction. The control variate method, namely subtracting some baseline from approximate returns, is widely used to reduce the variance while avoiding the introduction of bias into the estimation (Williams, 1992; Sutton et al., 1999; Greensmith et al., 2004; Gu et al., 2017a;b). Relying on predicted values instead of sampled returns is also popular despite the bias it often introduces (Degris et al., 2012; Mnih et al., 2016; Schulman et al., 2016; Ciosek & Whiteson, 2018). Our approach reduces the variance in a different way from these two common approaches. Therefore, it can be easily combined with the existing techniques to further reduce the variance while not introducing additional bias.

The problem of using probability distributions with unbounded support for control problems with bounded action spaces was pointed out in (Chou et al., 2017), which proposed modeling policies as beta distributions as a solution. While they reported performance improvements by using beta policies across multiple continuous control environments, Gaussian policies still nearly dominate the deep RL literature (Dhariwal et al., 2017; Henderson et al., 2018; Tassa et al., 2018). Truncated distributions have also been used to deal with bounded action spaces in prior work (Nakano et al., 2012; Shariff & Dick, 2013; Zimmer et al., 2016). In contrast, our approach allows us to keep using the same policy parameterizations, typically Gaussians, and still exploit action bounds. In addition, as we argued in Section 3.5, CAPG itself can also be seen as using a distribution with bounded support, and it can be multimodal, whereas beta policies and truncated Gaussian policies are unimodal. For example, a clipped Gaussian policy can easily learn to choose end-values of the action bounds with high probability by moving its mean toward the corresponding end, while beta and truncated Gaussian policies need to be near deterministic in order to choose near-end values with high probability.

Exploiting the integrated form of stochastic policy gradients to reduce the variance has also been proposed in (Ciosek & Whiteson, 2018; Asadi et al., 2017). They directly evaluated the integral over the whole action space, which can be analytically computed for limited classes of action value approximators and policies. Their method can reduce the variance by eliminating the need for Monte-Carlo estimation of policy gradients while introducing bias from action value approximation. Our method only evaluates the integral outside the action bounds, i.e., where action values are constant, and thus is unbiased.

6. Discussion

We have shown that variance of policy gradient estimation can be reduced by exploiting the fact that actions are clipped before they are sent to the environment. An unbiased and lower-variance policy gradient estimator, which we call clipped action policy gradient, has been proposed based on our analysis. CAPG is easy to implement and can be combined with existing variance reduction techniques, such as control variates and value function approximations.

We numerically analyzed CAPG's behavior on simple continuum-armed bandit problems, confirming its efficacy in variance reduction. When incorporated into existing deep RL algorithms, CAPG generally achieved the same or better performance on challenging simulated control benchmark tasks, indicating its promise as an alternative to the standard estimator.

While a Gaussian policy is the most common choice in policy gradient based continuous control, distributions with bounded support may sound more suitable for bounded action spaces. Prior work has proposed beta and truncated distributions to this end. We argued that CAPG can be seen as estimating the policy gradient of a different distribution with bounded support, termed a clipped distribution. Further studies are needed on the behaviors of different kinds of distributions as policy representations, including the ease of optimization and the representational power.

Acknowledgements

We thank Toshiki Kataoka, Kenta Oono, Masaki Watanabe and many others at Preferred Networks for insightful comments and discussions.

References

Agrawal, Rajeev. The Continuum-Armed Bandit Problem. *SIAM Journal on Control and Optimization*, 33(6):1926–1951, 1995.

Asadi, Kavosh, Allen, Cameron, Roderick, Melrose, Mohamed, Abdel-rahman, Konidaris, George, and Littman, Michael. Mean Actor Critic. *ArXiv e-prints*, 2017.

Brockman, Greg, Cheung, Vicki, Pettersson, Ludwig, Schneider, Jonas, Schulman, John, Tang, Jie, and Zaremba, Wojciech. OpenAI Gym. *ArXiv e-prints*, 2016.

Chou, Po-Wei, Maturana, Daniel, and Scherer, Sebastian. Improving Stochastic Policy Gradients in Continuous Control with Deep Reinforcement Learning using the Beta Distribution. In *ICML*, 2017.

Ciosek, Kamil and Whiteson, Shimon. Expected Policy Gradients. In *AAAI*, 2018.

Degris, Thomas, White, Martha, and Sutton, Richard S. Off-Policy Actor-Critic. In *ICML*, 2012.

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

Duan, Yan, Chen, Xi, Houthooft, Rein, Schulman, John, and Abbeel, Pieter. Benchmarking Deep Reinforcement Learning for Continuous Control. In *ICML*, 2016.

Greensmith, Evan, Bartlett, Pl, and Baxter, J. Variance Reduction Techniques for Gradient Estimates in Reinforcement Learning. *The Journal of Machine Learning Research*, 5:1471–1530, 2004.

Gu, Shixiang, Lillicrap, Timothy, Ghahramani, Zoubin, Turner, Richard E, and Levine, Sergey. Q-Prop: Sample-Efficient Policy Gradient with an Off-Policy Critic. In *ICLR*, 2017a.

Gu, Shixiang, Lillicrap, Timothy, Ghahramani, Zoubin, Turner, Richard E., Schölkopf, Bernhard, and Levine, Sergey. Interpolated Policy Gradient : Merging On-Policy and Off-Policy Gradient Estimation for Deep. In *NIPS*, 2017b.

Heess, Nicolas, TB, Dhruva, Sriram, Srinivasan, Lemmon, Jay, Merel, Josh, Wayne, Greg, Tassa, Yuval, Erez, Tom, Wang, Ziyu, Eslami, S. M. Ali, Riedmiller, Martin, and Silver, David. Emergence of Locomotion Behaviours in Rich Environments. *ArXiv e-prints*, 2017.

Henderson, Peter, Islam, Riashat, Bachman, Philip, Pineau, Joelle, Precup, Doina, and Meger, David. Deep Reinforcement Learning that Matters. In *AAAI*, 2018.

Kingma, Diederik P. and Ba, Jimmy Lei. Adam: a Method for Stochastic Optimization. In *ICLR*, 2015.

Levine, Sergey, Finn, Chelsea, Darrell, Trevor, and Abbeel, Pieter. End-to-End Training of Deep Visuomotor Policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

Mnih, Volodymyr, Kavukcuoglu, Koray, Silver, David, Rusu, Andrei a, Veness, Joel, Bellemare, Marc G, Graves, Alex, Riedmiller, Martin, Fidjeland, Andreas K, Ostrovski, Georg, Petersen, Stig, Beattie, Charles, Sadik, Amir, Antonoglou, Ioannis, King, Helen, Kumaran, Dharshan, Wierstra, Daan, Legg, Shane, and Hassabis, Demis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Mnih, Volodymyr, Badia, Adrià Puigdomènech, Mirza, Mehdi, Graves, Alex, Lillicrap, Timothy P, Harley, Tim, Silver, David, and Kavukcuoglu, Koray. Asynchronous Methods for Deep Reinforcement Learning. In *ICML*, 2016.

Nakano, Daichi, Maeda, Shin-ichi, and Ishii, Shin. Control of a Free-Falling Cat by Policy-Based Reinforcement Learning. In *ICANN*, 2012.

Schulman, John, Levine, Sergey, Moritz, Philipp, Jordan, Michael, and Abbeel, Pieter. Trust Region Policy Optimization. In *ICML*, 2015.

Schulman, John, Moritz, Philipp, Levine, Sergey, Jordan, Michael I, and Abbeel, Pieter. High-Dimensional Continuous Control Using Generalized Advantage Estimation. In *ICLR*, 2016.

Schulman, John, Wolski, Filip, Dhariwal, Prafulla, Radford, Alec, and Klimov, Oleg. Proximal Policy Optimization Algorithms. *ArXiv e-prints*, 2017.

Shariff, Roshan and Dick, Travis. Lunar Lander : A Continous-Action Case Study for Policy-Gradient Actor-Critic Algorithms. In *RLDM*, 2013.

Silver, David, Huang, Aja, Maddison, Chris J, Guez, Arthur, Sifre, Laurent, Driessche, George Van Den, Schrittwieser, Julian, Antonoglou, Ioannis, Panneershelvam, Veda, Lanctot, Marc, Dieleman, Sander, Grewe, Dominik, Nham, John, Kalchbrenner, Nal, Sutskever, Ilya, Lillicrap, Timothy, Leach, Madeleine, and Kavukcuoglu, Koray. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7585):484–489, 2016.

Silver, David, Schrittwieser, Julian, Simonyan, Karen, Antonoglou, Ioannis, Huang, Aja, Guez, Arthur, Hubert, Thomas, Baker, Lucas, Lai, Matthew, Bolton, Adrian, Chen, Yutian, Lillicrap, Timothy, Hui, Fan, and Sifre, Laurent. Mastering the game of Go without human knowledge. *Nature Publishing Group*, 550(7676):354–359, 2017.

Sutton, Richard S., Mcallester, David, Singh, Satinder, and Mansour, Yishay. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *NIPS*, 1999.

Tassa, Yuval, Doron, Yotam, Muldal, Alistair, Erez, Tom, Li, Yazhe, De, Diego, Casas, Las, Budden, David, Abdolmaleki, Abbas, Merel, Josh, Lefrancq, Andrew, Lillicrap, Timothy, and Riedmiller, Martin. DeepMind Control Suite. *ArXiv e-prints*, 2018.

Todorov, Emanuel, Erez, Tom, and Tassa, Yuval. MuJoCo: A physics engine for model-based control. In *IROS*, 2012.

Williams, RJ. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning*, 8(3-4):229–256, 1992.

Zimmer, Matthieu, Boniface, Yann, and Dutech, Alain. Off-policy Neural Fitted Actor-Critic. In *NIPS Deep Reinforcement Learning Workshop*, 2016.

Appendix

Lemma 3.1. *Suppose $\pi_\theta(a|s)$ is a compatible conditional distribution whose cumulative distribution function is $\Pi_\theta(a|s)$. Then, the following equations hold.*

$$\mathbb{E}_a[\mathbb{1}_{a\leq\alpha}\nabla_\theta \log \pi_\theta(a|s)] = \mathbb{E}_a[\mathbb{1}_{a\leq\alpha}\nabla_\theta \log \Pi_\theta(\alpha|s)],$$
$$\mathbb{E}_a[\mathbb{1}_{\beta\leq a}\nabla_\theta \log \pi_\theta(a|s)] = \mathbb{E}_a[\mathbb{1}_{\beta\leq a}\nabla_\theta \log(1-\Pi_\theta(\beta|s))].$$

Proof. Noting that $\pi_\theta(a|s)$ allows the exchange of derivative and integral, we get

$$\begin{aligned}\mathbb{E}_a[\mathbb{1}_{a\leq\alpha}\nabla_\theta \log \pi_\theta(a|s)] &= \int_{-\infty}^{\alpha} \pi_\theta(a|s)\nabla_\theta \log \pi_\theta(a|s) da \\ &= \int_{-\infty}^{\alpha} \nabla_\theta \pi_\theta(a|s) da \\ &= \nabla_\theta \int_{-\infty}^{\alpha} \pi_\theta(a|s) da \\ &= \nabla_\theta \Pi_\theta(\alpha|s) \\ &= \Pi_\theta(\alpha|s)\nabla_\theta \log \Pi_\theta(\alpha|s) \\ &= \mathbb{E}_a[\mathbb{1}_{a\leq\alpha}\nabla_\theta \log \Pi_\theta(\alpha|s)].\end{aligned}$$

A similar calculation shows

$$\mathbb{E}_a[\mathbb{1}_{\beta\leq a}\nabla_\theta \log \pi_\theta(a|s)] = \mathbb{E}_a[\mathbb{1}_{\beta\leq a}\nabla_\theta \log(1-\Pi_\theta(\beta|s))],$$

where we used $\int_\beta^\infty \pi_\theta(a|s)da = 1-\Pi_\theta(\beta|s)$ instead of $\int_{-\infty}^\alpha \pi_\theta(a|s)da = \Pi_\theta(\alpha|s)$. □

Lemma 3.2. *Suppose $\pi_\theta(a|s)$ is a compatible conditional distribution whose cumulative distribution function is $\Pi_\theta(a|s)$. Then, the following inequalities hold:*

$$\mathbb{V}_a[\mathbb{1}_{a\leq\alpha}\nabla_\theta \log \pi_\theta(a|s)] \geq \mathbb{V}_a[\mathbb{1}_{a\leq\alpha}\nabla_\theta \log \Pi_\theta(\alpha|s)],$$
$$\mathbb{V}_a[\mathbb{1}_{\beta\leq a}\nabla_\theta \log \pi_\theta(a|s)] \geq \mathbb{V}_a[\mathbb{1}_{\beta\leq a}\nabla_\theta \log(1-\Pi_\theta(\beta|s))].$$

Proof. Since both $\mathbb{1}_{a\leq\alpha}\nabla_\theta \log \pi_\theta(a|s)$ and $\mathbb{1}_{a\leq\alpha}\nabla_\theta \log \Pi_\theta(\alpha|s)$ have the same expected value, the difference of their variances is written as follows:

$$\begin{aligned}&\mathbb{V}_a[\mathbb{1}_{a\leq\alpha}\nabla_\theta \log \pi_\theta(a|s)] - \mathbb{V}_a[\mathbb{1}_{a\leq\alpha}\nabla_\theta \log \Pi_\theta(\alpha|s)] \\ &\quad = \mathbb{E}_a[\mathbb{1}_{a\leq\alpha}(\nabla_\theta \log \pi_\theta(a|s))^2] - \mathbb{E}_a[\mathbb{1}_{a\leq\alpha}(\nabla_\theta \log \Pi_\theta(\alpha|s))^2].\end{aligned}$$

The difference above is non-negative because

$$\begin{aligned}
\mathbb{E}_a[\mathbb{1}_{a\le\alpha}(\nabla_\theta \log \pi_\theta(a|s))^2] &= \int_{-\infty}^{\alpha} \pi_\theta(a|s)(\nabla_\theta \log \pi_\theta(a|s))^2 da \\
&= \Pi_\theta(\alpha|s) \int_{-\infty}^{\infty} \mathbb{1}_{a\le\alpha} \frac{\pi_\theta(a|s)}{\Pi_\theta(\alpha|s)} (\nabla_\theta \log \pi_\theta(a|s))^2 da \\
&\ge \Pi_\theta(\alpha|s) \Big(\int_{-\infty}^{\infty} \mathbb{1}_{a\le\alpha} \frac{\pi_\theta(a|s)}{\Pi_\theta(\alpha|s)} \nabla_\theta \log \pi_\theta(a|s) da \Big)^2 \\
&= \Pi_\theta(\alpha|s) \Big(\frac{1}{\Pi_\theta(\alpha|s)} \int_{-\infty}^{\alpha} \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s) da \Big)^2 \\
&= \Pi_\theta(\alpha|s) \Big(\frac{1}{\Pi_\theta(\alpha|s)} \int_{-\infty}^{\alpha} \nabla_\theta \pi_\theta(a|s) da \Big)^2 \\
&= \Pi_\theta(\alpha|s) \Big(\frac{1}{\Pi_\theta(\alpha|s)} \nabla_\theta \int_{-\infty}^{\alpha} \pi_\theta(a|s) da \Big)^2 \\
&= \Pi_\theta(\alpha|s) \Big(\frac{1}{\Pi_\theta(\alpha|s)} \nabla_\theta \Pi_\theta(\alpha|s) \Big)^2 \\
&= \Pi_\theta(\alpha|s)(\nabla_\theta \log \Pi_\theta(\alpha|s))^2 \\
&= \mathbb{E}_a[\mathbb{1}_{a\le\alpha}(\nabla_\theta \log \Pi_\theta(\alpha|s))^2],
\end{aligned}$$

where the equality holds only when $\nabla_\theta \log \pi_\theta(a|s)$ is a constant for $a \le \alpha$.

$\mathbb{V}_a[\mathbb{1}_{\beta\le a} \nabla_\theta \log \pi_\theta(a|s)] \ge \mathbb{V}_a[\mathbb{1}_{\beta\le a} \nabla_\theta \log(1 - \Pi_\theta(\beta|s))]$ is guaranteed by a similar calculation. $\square$