Journal of Machine Learning Research 21 (2020) 1-51 Submitted 1/18; Revised 2/20; Published 2/20

Expected Policy Gradients for Reinforcement Learning

Kamil Ciosek* KAMIL.CIOSEK@MICROSOFT.COM
Microsoft Research Cambridge,
21 Station Road, Cambridge CB1 2FB, United Kingdom

Shimon Whiteson SHIMON.WHITESON@CS.OX.AC.UK
Department of Computer Science, University of Oxford
Wolfson Building, Parks Road, Oxford OX1 3QD United Kingdom

Editor: Jan Peters

Abstract

We propose *expected policy gradients* (EPG), which unify stochastic policy gradients (SPG) and deterministic policy gradients (DPG) for reinforcement learning. Inspired by *expected sarsa*, EPG integrates (or sums) across actions when estimating the gradient, instead of relying only on the action in the sampled trajectory. For continuous action spaces, we first derive a practical result for Gaussian policies and quadratic critics and then extend it to a universal analytical method, covering a broad class of actors and critics, including Gaussian, exponential families, and policies with bounded support. For Gaussian policies, we introduce an exploration method that uses covariance proportional to e^H, where H is the scaled Hessian of the critic with respect to the actions. For discrete action spaces, we derive a variant of EPG based on softmax policies. We also establish a new *general policy gradient theorem*, of which the stochastic and deterministic policy gradient theorems are special cases. Furthermore, we prove that EPG reduces the variance of the gradient estimates without requiring deterministic policies and with little computational overhead. Finally, we provide an extensive experimental evaluation of EPG and show that it outperforms existing approaches on multiple challenging control domains.

Keywords: policy gradients, exploration, bounded actions, reinforcement learning, Markov decision process (MDP)

1. Introduction

In reinforcement learning, an agent aims to learn an optimal behavior policy from trajectories sampled from the environment. In settings where it is feasible to explicitly represent the policy, *policy gradient* methods (Sutton et al., 2000a; Peters and Schaal, 2006, 2008b; Silver et al., 2014), which optimize policies by gradient ascent, have enjoyed great success, especially with large or continuous action spaces. The archetypal algorithm optimizes an *actor*, i.e., a policy, by following a policy gradient that is estimated using a *critic*, i.e., a value function.

The policy can be stochastic or deterministic, yielding *stochastic policy gradients* (SPG) (Sutton et al., 2000a) or *deterministic policy gradients* (DPG) (Silver et al., 2014). The theory

∗. Work on this paper started when the first author was a post-doc at the Department of Computer Science, University of Oxford.

underpinning these methods is quite fragmented, as each approach has a separate policy gradient theorem guaranteeing the policy gradient is unbiased under certain conditions.

Furthermore, both approaches have significant shortcomings. For SPG, variance in the gradient estimates means that many trajectories are usually needed for learning. Since gathering trajectories is typically expensive, there is a great need for more sample efficient methods.

DPG's use of deterministic policies mitigates the problem of variance in the gradient but raises other difficulties. The theoretical support for DPG is limited since it assumes a critic that approximates $\nabla_a Q$ when in practice it approximates Q instead. In addition, DPG learns *off-policy*,[1] which means that, unless specifically designed otherwise, it explores in a way that is oblivious to the reward signal. More importantly, learning off-policy necessitates designing a suitable *exploration policy*, which is difficult in practice. In fact, efficient exploration in DPG is an open problem and most applications simply use independent Gaussian noise or the Ornstein-Uhlenbeck heuristic (Uhlenbeck and Ornstein, 1930; Lillicrap et al., 2015).

This article proposes a new approach called *expected policy gradients* (EPG) that unifies policy gradients in a way that yields both theoretical and practical insights. Inspired by *expected sarsa* (Sutton and Barto, 1998; van Seijen et al., 2009), the main idea is to integrate across the action selected by the stochastic policy when estimating the gradient, instead of relying only on the action selected during the sampled trajectory. While the idea of summing over discrete actions and calculating analytic integrals has been proposed previously (Sutton et al., 2000b; Bahdanau et al., 2016; Kakade, 2002) and concurrently (Asadi et al., 2017) in some specific settings, EPG is the first method to treat the technique in a unified way for both discrete and continuous action space on top of a single theoretical framework. The detailed differences between EPG and these approaches are given in Section 7.7.

The contributions of this paper are threefold. First, EPG enables two general theoretical contributions (Section 3.1): 1) a new *general policy gradient theorem*, of which the stochastic and deterministic policy gradient theorems are special cases, and 2) a proof that (Section 3.2) EPG reduces the variance of the gradient estimates without requiring deterministic policies and, for the Gaussian case, with no computational overhead over SPG. Second, we define practical policy gradient methods. For the Gaussian case (Section 4), the EPG solution is not only analytically tractable but also leads to a successful exploration strategy (Section 4.2) for continuous problems, with an exploration covariance that is proportional to e^H, where H is the scaled Hessian of the critic with respect to the actions. In Section 5.5, we derive a version of EPG for discrete control problems. We present empirical results (Section 6) confirming that this new approach to exploration substantially outperforms DPG with Ornstein-Uhlenbeck exploration in MuJoCo continuous control tasks. Third, we provide a way of deriving tractable EPG methods for the general case of policies coming from a certain exponential family (Section 5) and for critics that can be reparameterized as polynomials, thus yielding analytic EPG solutions that are tractable for a broad class of problems and essentially making EPG a universal method. Finally, in Section 7, we relate EPG to other RL approaches, including entropy-based methods and value gradient methods.

This paper is a revised and extended version of our AAAI conference submission (Ciosek and Whiteson, 2018). On the theoretical side, we have added an analysis of softmax (Gibbs)

1. We show in this article that, in certain settings, off-policy DPG is equivalent to EPG, our on-policy method.

policies. For continuous actions, we have analyzed a more general policy class (exponential families) and a critic class general enough to approximate any function (polynomials). We provide an analysis of the off-policy version of EPG. We also compare EPG with methods that adapt to the geometry of the policy space, entropy-based methods and value gradients. In addition, we have greatly expanded the experimental section, which now includes a comparison to reparameterized policy gradients and several ablations, in addition to results about EPG with numerical quadrature.

2. Background

A *Markov decision process* (Puterman, 2014) is a tuple $(S, A, R_d, p, p_0, \gamma)$ where S is a set of states, A is a set of actions (in practice either $A = \mathbb{R}^d$ or A is finite), $R_d(a, s)$ is a reward distribution (we introduce the notation $R(a, s) = \mathbb{E}_{R_d}[r \,|\, a, s]$ for the mean state-action reward), $p(s' \mid a, s)$ is a transition kernel, p_0 is an initial state distribution, and $\gamma \in [0, 1)$ is a discount factor. A policy $\pi(a \mid s)$ is a distribution over actions given a state. We denote trajectories as $\tau^\pi = (s_0, a_0, r_0, s_1, a_1, r_1, \dots)$, where $s_0 \sim p_0$, $a_t \sim \pi(\cdot \mid s_t)$, $s_{t+1} \sim p(\cdot \mid s_t, a_t)$ and r_t is the sampled reward. A policy π induces a Markov process with transition kernel $p_\pi(s' \mid s) = \int_a d\pi(a \mid s) p(s' \mid a, s)$ where we use the symbol $d\pi(a \mid s)$ to denote Lebesgue integration against the measure $\pi(a \mid s)$ where s is fixed. We assume the induced Markov process is ergodic with a single invariant measure defined for the whole state space. The value function is $V^\pi(s) = \mathbb{E}_{\tau : s_0 = s}\left[\sum_{i=0}^\infty \gamma_i r_i\right]$ where actions are sampled from π. The Q-function is $Q^\pi(a, s) = \mathbb{E}_{R_d}[r \,|\, a, s] + \gamma \mathbb{E}_{p(s'|a,s)}[V^\pi(s') \,|\, s]$ and the advantage function is $A^\pi(a \mid s) = Q^\pi(a, s) - V^\pi(s)$. An optimal policy maximizes the total return $J = \int_s dp_0(s) V^\pi(s)$. Since we consider only on-policy learning with just one current policy, we drop the π super/subscript where it is redundant.

If π is parametrized by θ, then *stochastic policy gradients* (SPG) (Sutton et al., 2000a; Peters and Schaal, 2006, 2008b) perform gradient ascent on $\nabla_\theta J$, the gradient of J with respect to θ. For stochastic policies, we have

$$\nabla_\theta J = \int_s d\rho(s) \int_a d\pi(a \mid s) \nabla_\theta \log \pi(a \mid s) (Q^{\pi_\theta}(a, s) + b(s)),$$

where ρ is the discounted-ergodic occupancy measure, defined in the Appendix, and $b(s)$ is a baseline,[2] which can be any function that depends on the state but not the action, since $\int_a d\pi(a \mid s) \nabla_\theta \log \pi(a \mid s) b(s) = 0$. Typically, because of ergodicity and Lemma 19 (see Appendix), we can approximate (2) from samples from a trajectory τ of length T, giving

$$\hat{\nabla}_\theta J = \sum_{t=0}^T \gamma^t \nabla_\theta \log \pi(a_t \mid s_t) (\hat{Q}(a_t, s_t) + b(s_t)).$$

Here, $\hat{Q}(a_t, s_t)$ is a critic, discussed below. If the policy is deterministic (we denote it $\pi(s)$), we can use *deterministic policy gradients* (Silver et al., 2014) instead, so that the gradient becomes

$$\nabla_\theta J = \int_s d\rho(s) \nabla_\theta \pi(s) \left[\nabla_a Q(a, s)\right]_{a = \pi(s)}.$$

2. See also the work on action-dependent baselines by Thomas and Brunskill (2017) and Wu et al. (2018).

This update is then approximated using samples, giving the estimate

$$\hat{\nabla}_\theta J = \sum_{t=0}^{T} \gamma^t \nabla_\theta \pi(s) \left[\nabla_a \hat{Q}(a, s_t)\right]_{a=\pi(s_t)}.$$

Since the policy is deterministic, the problem of exploration is addressed using an external source of noise, typically modeled using a zero-mean Ornstein-Uhlenbeck (OU) process (Uhlenbeck and Ornstein, 1930; Lillicrap et al., 2015) parameterized by ψ and σ and generated as

$$n_t \leftarrow -n_{t-1}\psi + \mathcal{N}(0, \sigma I) \quad \text{and} \quad a_t \sim \pi(s_t) + n_t.$$

In (2) and (2), $\hat{Q}$ is a *critic* that approximates Q and can be learned by *sarsa* (Rummery and Niranjan, 1994; Sutton, 1996), using the update

$$\hat{Q}(a_t, s_t) \leftarrow \hat{Q}(a_t, s_t) \; + \; \alpha\big[r_{t+1} + \gamma \hat{Q}(s_{t+1}, a_{t+1}) - \hat{Q}(a_t, s_t)\big].$$

Alternatively, we can use *expected sarsa* (Sutton and Barto, 1998; van Seijen et al., 2009), which marginalizes out a_{t+1}, the distribution over which is specified by the known policy, to reduce the variance, using the update

$$\hat{Q}(a_t, s_t) \leftarrow \hat{Q}(a_t, s_t) \; + \; \alpha\big[r_{t+1} + \gamma \int_a d\pi(a \mid s) \hat{Q}(a, s_{t+1}) - \hat{Q}(a_t, s_t)\big].$$

Instead, we could also use advantage learning (Baird et al., 1995) or LSTDQ (Lagoudakis and Parr, 2003). This update also has the advantage that actions a_t sampled from a policy different than π can be used during learning, giving an off-policy algorithm.

The theory behind policy gradient methods says that the actor follows the gradient of the total discounted return J (Sutton et al., 2000a) and therefore finds its local maximum if the critic's function approximator is *compatible*.[3]

Instead of learning $\hat{Q}$, we can set $b(s) = -V^{\pi_\theta}(s)$ so that $Q^{\pi_\theta}(a, s) + b(s) = A(a, s)$ and then use the TD error $\delta(r, s', s) = r + \gamma V^{\pi_\theta}(s') - V^{\pi_\theta}(s)$ as an estimate of $A(a, s)$ (Bhatnagar et al., 2008). The policy gradient estimate then becomes

$$\hat{\nabla}_\theta J = \sum_{t=0}^{T} \gamma^t \nabla_\theta \log \pi(a_t \mid s_t)(r + \gamma \hat{V}(s_{t+1}) - \hat{V}(s_t)), \tag{1}$$

where $\hat{V}(s)$ is an approximate value function learned using any policy evaluation algorithm. Equation (1) works because $\mathbb{E}\left[\delta(r, s', s) \,|\, a, s\right] = A(a, s)$, i.e., the TD error is an unbiased estimate of the advantage function. The benefit of this approach is that it is sometimes easier to approximate V than Q and that the return in the TD error is unprojected, i.e., it is not distorted by function approximation. However, for stochastic MDPs, the TD error is noisy, introducing variance in the gradient.

To cope with this variance, we can use an optimizer that adaptively reduces the learning rate when the variance of the gradient is high, using, e.g., *Adam* (Kingma and Ba, 2015) or RMSprop (Tieleman and Hinton, 2012). However, this results in slow learning when the variance is high. We discuss other variance reduction techniques in Section 7.

3. This holds under the idealized setting where the critic is run to convergence and minimizes a weighted L_2-loss. In practice, the critic is not run until convergence and, even if it were, it is not guaranteed to minimize the weighted L_2 loss. Compatible critics are rarely used in practice.

3. Expected Policy Gradients

In this section, we propose *expected policy gradients* (EPG). First, we introduce $I_\pi^Q(s)$ to denote the inner integral in (2), rewriting the policy gradient as

$$\begin{aligned}\nabla_\theta J &= \int_s d\rho(s) \underbrace{\int_a d\pi(a \mid s) \nabla_\theta \log \pi(a \mid s)(Q^{\pi_\theta}(a,s) + b(s))}_{I_\pi^Q(s)} \\ &= \int_s d\rho(s) \int_a d\pi(a \mid s) \nabla_\theta \log \pi(a \mid s) Q^{\pi_\theta}(a,s) \\ &= \int_s d\rho(s) I_\pi^Q(s). \end{aligned} \tag{2}$$

This suggests a new way to write the approximate gradient, giving[4]

$$\hat{\nabla}_\theta J = \sum_{t=0}^{T} \underbrace{\gamma^t I_\pi^{\hat{Q}}(s_t)}_{g_t}, \quad \text{where} \quad I_\pi^{\hat{Q}}(s) = \int_a d\pi(a \mid s) \nabla_\theta \log \pi(a \mid s) \hat{Q}(a,s).$$

Here, we used Lemma 19 in the Appendix to sample from $\rho(s)$. This approach makes explicit that one step in estimating the gradient is to evaluate an integral included in the term $I_\pi^{\hat{Q}}(s)$. The main insight behind EPG is that, given a state, $I_\pi^{\hat{Q}}(s)$ is expressed fully in terms of known quantities. Hence we can manipulate it analytically to obtain a formula or we can just compute the integral using numerical quadrature if an analytical solution is impossible (in Section 5.1 we show that this is rare). For a discrete action space, $I_\pi^{\hat{Q}}(s_t)$ becomes a sum over actions (see Section 5.5 for more details).

SPG as given in (2) performs this quadrature using a simple one-sample Monte Carlo method, using the action $a_t \sim \pi(\cdot \mid s_t)$. It uses the update

$$I_\pi^{\hat{Q}}(s) = \int_a d\pi(a \mid s) \nabla_\theta \log \pi(a \mid s) \hat{Q}(a,s) \approx \nabla_\theta \log \pi(a_t \mid s_t)(\hat{Q}(a_t, s_t) + b(s_t)).$$

Moreover, SPG assumes that the action a_t used in the above estimation is the same action that is executed in the environment. However, relying on such a method is unnecessary. In fact, the actions used to interact with the environment need not be used at all in the evaluation of $\hat{I}_\pi^Q(s)$ since a is a bound variable in the definition of $I_\pi^Q(s)$. The motivation is thus similar to that of expected sarsa but applied to the actor's gradient estimate instead of the critic's update rule. EPG, shown in Algorithm 1, uses (3) to form a policy gradient algorithm that repeatedly estimates $\hat{I}_\pi^Q(s)$ with an integration subroutine.

One of the motivations of DPG was precisely that the simple one-sample Monte-Carlo quadrature implicitly used by SPG often yields high variance gradient estimates, even with a good baseline. To see why, consider the setting in Figure 1, where we use the parametrization $\theta = \mu$. On the left, a simple Monte Carlo method evaluates the integral by sampling one or more times from $\pi(a \mid s)$ (blue) and evaluating $\nabla_\mu \log \pi(a \mid s) Q^{\pi_\theta}(a,s)$ (red) as a function

4. The idea behind EPG was also independently and concurrently developed as Mean Actor Critic (Asadi et al., 2017), though only for discrete actions and without a supporting theoretical analysis.

Algorithm 1 Expected policy gradients

$s \leftarrow s_0$, $t \leftarrow 0$
initialize optimizer, initialize policy π parameterized by θ
while not converged **do**
    $g_t \leftarrow \gamma^t$ DO-INTEGRAL$(\hat{Q}, s, \pi_\theta)$ ▷ g_t is the estimated policy gradient as per (3)
    $\theta \leftarrow \theta +$ optimizer.UPDATE(g_t)
    $a \sim \pi(\cdot \mid s)$
    $s', r \leftarrow$ simulator.PERFORM-ACTION(a)
    $\hat{Q}$.UPDATE(s, a, r, s')
    $t \leftarrow t + 1$
    $s \leftarrow s'$
end while

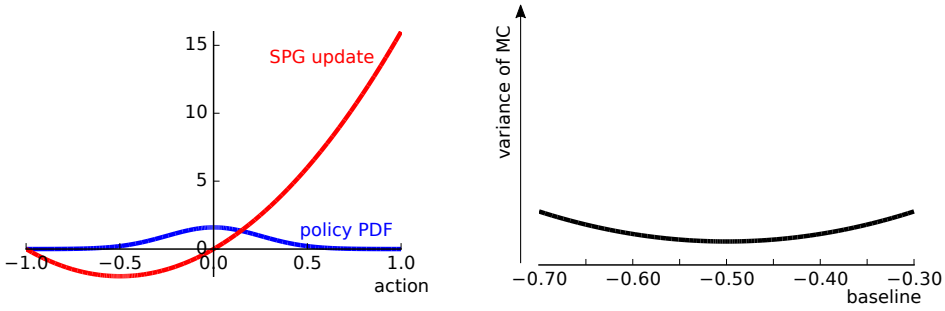


Figure 1: At left, $\pi(a \mid s)$ for a Gaussian policy with mean $\mu = \theta = 0$ at a given state and constant σ^2 (blue) and $\nabla_\theta \log \pi(a \mid s) Q^{\pi_\theta}(a, s)$ for $Q = \frac{1}{2} + \frac{1}{2}a$ (red). At right, the variance of a simple single-sample Monte Carlo estimator as a function of the baseline. In a simple multi-sample Monte Carlo method, the variance would go down as the number of samples.

of a. A baseline can decrease the variance by adding a multiple of $\nabla_\mu \log \pi(a \mid s)$ to the red curve. However, whatever the baseline, substantial variance persists, even with a simple linear Q-function, as shown in Figure 1 (right). DPG addressed this problem for deterministic policies but EPG extends it to stochastic ones. We show in Section 5 that an analytical EPG solution, and thus the corresponding reduction in the variance, is possible for a wide array of critics. We also discuss the rare case where numerical quadrature is necessary in Section 5.4.

3.1. General Policy Gradient Theorem

We begin by stating our most general result, showing that EPG can be seen as a generalization of both SPG and DPG. To do this, we first state a new general policy gradient theorem.

Theorem 1 (General Policy Gradient Theorem)
If the value function $V(s)$ is bounded, continuously differentiable in the policy parameters θ

and measurable in s then

$$\nabla_\theta J = \int_s d\rho(s) \underbrace{\left[\nabla_\theta V^{\pi_\theta}(s) - \int_a d\pi(a|s) \nabla_\theta Q^{\pi_\theta}(a,s)\right]}_{I_G(s)} = \int_s d\rho(s) I_G(s).$$

Proof We start with the expression on the left and begin expanding. We have

$$\begin{aligned}
&\textstyle\int_s d\rho(s) \int_a d\pi(a|s) \nabla_\theta Q^{\pi_\theta}(a,s) \\
&\qquad = \textstyle\int_s d\rho(s) \int_a d\pi(a|s) \nabla_\theta (R(a,s) + \gamma \int_{s'} dp(s' \mid a,s) V^{\pi_\theta}(s')) \\
&\qquad = \textstyle\int_s d\rho(s) \int_a d\pi(a|s) (\underbrace{\nabla_\theta R(a,s)}_{0} + \gamma \int_{s'} dp(s' \mid a,s) \nabla_\theta V^{\pi_\theta}(s')) \\
&\qquad = \textstyle\gamma \int_s d\rho(s) \int_{s'} dp_\pi(s' \mid s) \nabla_\theta V^{\pi_\theta}(s')) \\
&\qquad = \textstyle\int_s d\rho(s) \nabla_\theta V^{\pi_\theta}(s) - \underbrace{\int_s dp_0(s) \nabla_\theta V^{\pi_\theta}(s)}_{\nabla_\theta J} \\
&\qquad = \textstyle\int_s d\rho(s) \nabla_\theta V^{\pi_\theta}(s) - \nabla_\theta J.
\end{aligned}$$

In the above, we used the notation $p_\pi(s' \mid s) = \int_a d\pi(a \mid s) p(s' \mid a, s)$. The first equality follows by expanding the definition of Q and the penultimate one follows from Lemma 20 in the Appendix. Then the theorem follows by rearranging terms. ■

The crucial benefit of Theorem 1 is that it works for all policies, both stochastic and deterministic, unifying previously separate derivations for the two settings. To show this, in the following two corollaries, we use Theorem 1 to recover the *stochastic policy gradient theorem* (Sutton et al., 2000a) and the *deterministic policy gradient theorem* (Silver et al., 2014), in each case by introducing additional assumptions to obtain a formula for $I_G(s)$ expressible in terms of known quantities.

Corollary 2 (Stochastic Policy Gradient Theorem) *If $\pi(a \mid s)$, considered as a probability density function, is continuous in s and continuously differentiable in θ and if $R(a,s)$ is continuous in s and bounded then*

$$\nabla_\theta J = \textstyle\int_s d\rho(s) I_G(s) = \int_s d\rho(s) \int_a d\pi(a \mid s) \nabla_\theta \log \pi(a \mid s) Q^{\pi_\theta}(a,s).$$

Proof We expand $\nabla_\theta V$, obtaining

$$\nabla_\theta V = \nabla_\theta \textstyle\int_a d\pi(a|s) Q^{\pi_\theta}(a,s) = \int_a da (\nabla_\theta \pi(a|s)) Q^{\pi_\theta}(a,s) + \int_a d\pi(a|s) (\nabla_\theta Q^{\pi_\theta}(a,s)). \quad (3)$$

We obtain $I_G(s) = \int_a d\pi(a \mid s) \nabla_\theta \log \pi(a \mid s) Q^{\pi_\theta}(a,s) = I_\pi^Q(s)$ by plugging (3) into the definition of $I_G(s)$ as given in (1). We obtain $\nabla_\theta J$ by invoking Theorem 1, plugging in the above expression for $I_G(s)$ and observing that the regularity conditions on V follow from the regularity conditions on $\pi(\cdot \mid s)$ and $R(a,s)$. ■

Corollary 3 (Deterministic Policy Gradient Theorem) *If $\pi(\cdot \mid s)$ is a Dirac-delta measure, and $\pi(s)$ is continuously differentiable in θ and continuous in s, if $R(a,s)$ is continuous in s, differentiable in a and bounded and the transition kernel $p(s' \mid a, s)$ is continuous in s and differentiable in a, then*

$$\nabla_\theta J = \int_s d\rho(s) I_G(s) = \int_s d\rho(s) \nabla_\theta \pi(s) \left[\nabla_a Q^{\pi_\theta}(a,s)\right]_{a=\pi(s)}.$$

Here, we overload the notation of π slightly. We denote by $\pi(s)$ the action taken at state s, i.e. $\pi(s) = \int_a a d\pi(a \mid s)$, where $\pi(\cdot \mid s)$ is the corresponding Dirac-delta measure.

Proof We begin by expanding the term for $\nabla_\theta V^{\pi_\theta}(s)$, which will be useful later on. We have

$$\nabla_\theta V^{\pi_\theta}(s) = \nabla_\theta Q(\pi(s), s) = \left[\nabla_\theta Q^{\pi_\theta}(a,s)\right]_{a=\pi(s)} + \nabla_\theta \pi(s) \left[\nabla_a Q^{\pi_\theta}(a,s)\right]_{a=\pi(s)}.$$

The above results from applying the multivariate chain rule—observe that both $\pi(s)$ and $Q^{\pi_\theta}(a,s)$ depend on the policy parameters θ; hence, the dependency appears twice in $Q(\pi(s), s)$.

We proceed to obtain an expression for $I_G(s)$. We have

$$\begin{aligned} I_G(s) &= \nabla_\theta V^{\pi_\theta}(s) - \int_a d\pi(a|s) \nabla_\theta Q^{\pi_\theta}(a,s) \\ &= \nabla_\theta V^{\pi_\theta}(s) - \left[\nabla_\theta Q^{\pi_\theta}(a,s)\right]_{a=\pi(s)} \\ &= \nabla_\theta \pi(s) \left[\nabla_a Q^{\pi_\theta}(a,s)\right]_{a=\pi(s)}. \end{aligned}$$

Here, the second equality follows by observing that the policy is a Dirac-delta and the third one follows from using (3.1). We can then obtain $\nabla_\theta J$ by invoking Theorem 1 and plugging in the above expression for $I_G(s)$. The regularity conditions on V follow from the regularity conditions on $\pi(s)$, $R(a,s)$ and $p(s' \mid a, s)$. ■

These corollaries show that the choice between deterministic and stochastic policy gradients is fundamentally a choice of quadrature method. Hence, the empirical success of DPG relative to SPG (Silver et al., 2014; Lillicrap et al., 2015) can be understood in a new light. In particular, it can be attributed, not to a fundamental limitation of stochastic policies (indeed, stochastic policies are sometimes preferred), but instead to superior quadrature. DPG integrates over Dirac-delta measures, which is known to be easy, while SPG typically relies on simple Monte Carlo integration. Thanks to EPG, a deterministic approach is no longer required to obtain a method with low variance.

3.2. Variance Analysis

In stochastic policy gradients, there are two reasons why the actor update can be inaccurate (Gu et al., 2017). First, using an approximation in place of the original quadrature means that the integral estimate is stochastic and has nonzero variance. In, particular, the value of the integral

$$\int_a d\pi(a \mid s) \nabla_\theta \log \pi(a \mid s) (\hat{Q}(a,s) + b(s))$$

is different from its one-sample Monte-Carlo approximation

$$\nabla_\theta \log \pi(a_t \mid s_t)(\hat{Q}(a_t, s_t) + b(s_t)), \quad \text{where} \quad a_t \sim \pi(\cdot \mid s_t).$$

Second, the learned critic value $\hat{Q}$ itself can be inaccurate – if we use a neural net to learn $\hat{Q}$, it will have some approximation error, while if we use an advantage estimate as in (1), there will be additional variance which comes from this advantage estimate.

We now prove that EPG completely eliminates the first kind of variance. In particular, for any policy, the EPG estimator of (3) has lower variance than the SPG estimator of (2).

Lemma 4 *If for all $s \in S$, the random variable $\nabla_\theta \log \pi(a \mid s)\hat{Q}(a, s)$ where $a \sim \pi(\cdot|s)$ has nonzero variance, then*

$$\mathbb{V}_\tau \left[\sum_{t=0}^{\infty} \gamma^t \nabla_\theta \log \pi(a_t \mid s_t)(\hat{Q}(a_t, s_t) + b(s_t)) \right] > \mathbb{V}_\tau \left[\sum_{t=0}^{\infty} \gamma^t I_\pi^{\hat{Q}}(s_t) \right].$$

Proof Both random variables have the same mean so we need only show that

$$\mathbb{E}_\tau \left[\left(\sum_{t=0}^{\infty} \gamma^t \nabla_\theta \log \pi(a_t \mid s_t)(\hat{Q}(a_t, s_t) + b(s_t)) \right)^2 \right] > \mathbb{E}_\tau \left[\left(\sum_{t=0}^{\infty} \gamma^t I_\pi^{\hat{Q}}(s_t) \right)^2 \right].$$

We start by applying Lemma 22 to the left-hand side and setting

$$X = X_1(s_t) = \nabla_\theta \log \pi(a_t \mid s_t)(\hat{Q}(a_t, s_t) + b(s_t))$$

where $a_t \sim \pi(a_t|s_t)$. This shows that

$$\mathbb{E}_\tau \left[\left(\sum_{t=0}^{\infty} \gamma^t \nabla_\theta \log \pi(a_t \mid s_t)(\hat{Q}(a_t, s_t) + b(s_t)) \right)^2 \right]$$

is the total return of the Markov reward process (MRP)[5] (p, p_0, u_1, γ^2), where

$$u_1(s) = \mathbb{V}_{X_1(x|s)}[x] + \left(\mathbb{E}_{X_1(x|s)}[x] \right)^2 + 2\gamma \mathbb{E}_{X_1(x|s)}[x] \, \mathbb{E}_{p(s'|s)} \left[V^{\pi_\theta}(s') \right].$$

Likewise, applying Lemma 22 again to the right-hand side, instantiating X as a deterministic random variable $X_2(s_t) = I_\pi^{\hat{Q}}(s_t)$, we have that $\mathbb{E}_\tau \left[\sum_{t=0}^{\infty} \left(\gamma^t I_\pi^{\hat{Q}}(s_t) \right)^2 \right]$ is the total return of the MRP (p, p_0, u_2, γ^2), where

$$u_2(s) = \left(\mathbb{E}_{X_2(x|s)}[x] \right)^2 + 2\gamma \mathbb{E}_{X_2(x|s)}[x] \, \mathbb{E}_{p(s'|s)} \left[V^{\pi_\theta}(s') \right].$$

Note that $\mathbb{E}_{X_1(x|s)}[x] = \mathbb{E}_{X_2(x|s)}[x]$ and therefore $u_1(s) \geq u_2(s)$ for all states s. Furthermore, by assumption of the lemma, the inequality is strict. The lemma then follows by applying Observation 23. ■

5. See Definition 21 in the Appendix.

For convenience, Lemma 4 also assumes infinite length trajectories. However, this is not a practical limitation since all policy gradient methods implicitly assume trajectories are long enough to be modeled as infinite. Furthermore, a finite trajectory variant also holds, though the proof is messier.

Lemma 4's assumption is reasonable since the only way a random variable

$$\nabla_\theta \ \log \ \pi(a \ \mid \ s) \ \hat{Q}(a, s)$$

could have zero variance is if it were the same for all actions in the policy's support (except for sets of measure zero), in which case optimizing the policy would be unnecessary. Since we know that both the estimators of (2) and (3) are unbiased,[6] the estimator with lower variance has lower MSE. Moreover, we observe that Lemma 4 holds for the case where the computation of $I_\pi^{\hat{Q}}$ is exact. Section 5 shows that this is often possible.

4. Expected policy gradients for Gaussian Policies

EPG is particularly useful when we make the common assumption of a Gaussian policy: we can then perform the integration analytically under reasonable conditions. We show below (see Corollary 7) that the update to the policy mean computed by EPG is equivalent to the DPG update. Moreover, we derive a simple formula for the covariance (see Lemma 6). Algorithms 2 and 3 show the resulting special case of EPG, which we call *Gaussian policy gradients* (GPG).

Algorithm 2 Gaussian policy gradients

$s \leftarrow s_0$, $t \leftarrow 0$
initialize optimizer
while not converged **do**
  $g_t \leftarrow \gamma^t$ DO-INTEGRAL-GAUSS$(\hat{Q}, s, \pi_\theta)$
  $\theta \leftarrow \theta +$ optimiser.UPDATE(g_t) ▷ policy parameters θ are updated using gradient
  $\Sigma_s^{1/2} \leftarrow$ GET-COVARIANCE$(\hat{Q}, s, \pi_\theta)$ ▷ $\Sigma_s^{1/2}$ computed from scratch
  $a \sim \pi(\cdot \mid s)$ ▷ $\pi(\cdot \mid s) = N(\mu_s, \Sigma_s)$
  $s', r \leftarrow$ simulator.PERFORM-ACTION(a)
  $\hat{Q}$.UPDATE(s, a, r, s')
  $t \leftarrow t + 1$
  $s \leftarrow s'$
end while

Surprisingly, GPG is on-policy but nonetheless fully equivalent to DPG, an off-policy method, with a particular form of exploration. Hence, GPG, by specifying the policy's covariance, can be seen as a derivation of an exploration strategy for DPG. In this way, GPG addresses an important open question. As we show in Section 6, this leads to improved performance in practice.

The computational cost of GPG is small: while it must store a Hessian matrix $H(a, s) = \nabla_a^2 \hat{Q}(a, s)$, its size is only $d \times d$, where $A = \mathbb{R}^d$, which is typically small, e.g., $d = 6$ for

6. They provide an unbiased estimate of an integral that includes $\hat{Q}$. Of course the gradient can still be biased if $\hat{Q}$ itself is biased.

Algorithm 3 Gaussian integrals

function DO-INTEGRAL-GAUSS($\hat{Q}, s, \pi_\theta$)
    $I^{\hat{Q}}_{\pi(s),\mu_s} \leftarrow (\nabla_\theta \mu_s) \nabla_a \hat{Q}(a = \mu_s, s)$ ▷ Use Lemma 5
    return $I^{\hat{Q}}_{\pi(s),\mu_s}$
end function

function GET-COVARIANCE($\hat{Q}, s, \pi_\theta$)
    $H \leftarrow$ COMPUTE-HESSIAN($\hat{Q}(\mu_s, s)$)
    return $\sigma_0 e^{cH}$ ▷ Use Lemma 6
end function

HalfCheetah, one of the MuJoCo tasks we use for our experiments in Section 6. This Hessian is the same size as the policy's covariance matrix, which any policy gradient must store anyway, and should not be confused with the Hessian with respect to the parameters of the neural network, as used with Newton's or natural gradient methods (Peters and Schaal, 2008a; Furmston et al., 2016), which can easily have thousands of entries. Hence, GPG obtains EPG's variance reduction essentially for free.

4.1. Analytical Quadrature for Gaussian Policies

We now derive a lemma supporting GPG.

Lemma 5 (Gaussian Policy Gradients) *For Gaussian policies, i.e.* $\pi(\cdot|s) \sim \mathcal{N}(\mu_s, \Sigma_s)$ *with* μ_s *and* $\Sigma_s^{1/2}$ *parametrized by* θ*, where* $\Sigma_s^{1/2}$ *is symmetric,* $\Sigma_s^{1/2}\Sigma_s^{1/2} = \Sigma_s$ *and the critic is of the form* $\hat{Q}(a, s) = a^\top A(s)a + a^\top B(s) + const$ *where* $A(s)$ *is symmetric for every* s*, then* $I^{\hat{Q}}_\pi(s) = I^{\hat{Q}}_{\pi(s),\mu_s} + I^{\hat{Q}}_{\pi(s),\Sigma_s^{1/2}}$*, where the mean and covariance components are given by*

$$\begin{aligned} I^{\hat{Q}}_{\pi(s),\mu_s} &= (\nabla_\theta \mu_s)(2A(s)\mu + B(s)) \quad \textit{and} \\ I^{\hat{Q}}_{\pi(s),\Sigma_s^{1/2}} &= (\nabla_\theta \Sigma_s^{1/2}) 2A(s)\Sigma_s^{1/2}. \end{aligned} \tag{4}$$

Proof

First, we observe that the critic $\hat{Q}$ defined in the statement of the lemma does not depend on the policy parameters θ because $\hat{Q}$ is an approximation to the Q-function maintained by the algorithm, as opposed to the true Q-function, which is defined with respect to the policy and does depend on it.

We can hence move the differentiation outside of the integral, obtaining

$$I^{\hat{Q}}_\pi(s) = \nabla_\theta \int_a \pi(a|s)\hat{Q}(a, s)da = \nabla_\theta \mathbb{E}_\pi \left[\hat{Q}(a, s)\right].$$

We now expand the expectation using the expression $\mathbb{E}_\pi\left[\hat{Q}(a,s)\right] = \text{trace}(A(s)\Sigma) + \mu^\top A(s)\mu + B(s)^\top\mu$ for the expectation of a quadratic form. This yields the derivatives

$$\nabla_{\Sigma^{1/2}}\mathbb{E}_\pi\left[Q(a,s)\right] = \nabla_{\Sigma^{1/2}}(\text{trace}(A(s)\Sigma) + \mu^\top A(s)\mu + B(s)^\top\mu) = 2A(s)\Sigma^{1/2} \quad \text{and}$$
$$\nabla_{\mu}\mathbb{E}_\pi\left[\hat{Q}(a,s)\right] = \nabla_{\mu}(\text{trace}(A(s)\Sigma) + \mu^\top A(s)\mu + B(s)^\top\mu) = 2A(s)\mu + B(s).$$

We now obtain the result by applying the chain rule, giving

$$I_\pi^{\hat{Q}}(s) = I^{\hat{Q}}_{\pi(s),\mu_s} + I^{\hat{Q}}_{\pi(s),\Sigma_s^{1/2}} = (\nabla_\theta\mu)(2A(s)\mu + B(s)) + (\nabla_\theta\Sigma^{1/2})(2A(s)\Sigma^{1/2}).$$

■

While Lemma 5 requires the critic to be quadratic in the actions, this assumption is not very restrictive since the coefficients $B(s)$ and $A(s)$ can be arbitrary continuous functions of the state, e.g., a neural network.

4.2. Exploration using the Hessian

Equation (4) suggests that we can include the covariance in the actor network and learn it along with the mean, using the update rule

$$\Sigma_s^{1/2} \leftarrow \Sigma_s^{1/2} + \alpha H(s)\Sigma_s^{1/2}.$$

In practice, this update has two disadvantages. First, our policy network must include outputs for the covariance. Second, we must constrain our covariance matrix to be positive semi-definite, which is not trivial in a deep learning setup without influencing the gradient norm in a way that slows learning. Instead, we sidestep the issue by not performing incremental covariance updates at all. Instead, we analytically compute the matrix that would have been obtained if we ran the update (4.2) for long enough.

The following lemma derives the covariance from scratch at each iteration by analytically computing the result of applying (4.2) infinitely many times.

Lemma 6 (Exploration Limit) *For any fixed total number of iterations n and the same constant learning rate $\alpha = 1/n$ applied at every iteration, the iterative procedure defined by* (4.2) *applied n times yields $(\Sigma_s^{1/2})_{n+1} = U(I + \frac{1}{n}\Lambda)^n U^\top\sigma_0$. Moreover, the limit as $n \to \infty$, is $\Sigma_s^{1/2} \propto e^{H(s)}$.*

Proof First, let n be fixed. Consider the sequence $(\Sigma_s^{1/2})_1 = \sigma_0 I$, $(\Sigma_s^{1/2})_n = (\Sigma_s^{1/2})_{n-1} + \frac{1}{n}H(s)(\Sigma_s^{1/2})_{n-1}$. We diagonalize the Hessian as $H(s) = U\Lambda U^\top$ for some orthonormal matrix U and obtain the following expression for the n-th element of the sequence

$$(\Sigma_s^{1/2})_{n+1} = \left(I + \frac{1}{n}H(s)\right)^n \sigma_0 = U\left(I + \frac{1}{n}\Lambda\right)^n U^\top\sigma_0.$$

We now let $n \to \infty$. Since we have $\lim_{n\to\infty}(1 + \frac{1}{n}\lambda)^n = e^\lambda$ for each eigenvalue of the Hessian, we obtain the identity

$$\lim_{n\to\infty} U\left(I + \frac{1}{n}\Lambda\right)^n U^\top\sigma_0 = \sigma_0 e^{H(s)}.$$

■

The practical implication of Lemma 6 is that, in a policy gradient method, it is justified[7] to use Gaussian exploration with covariance proportional to e^{cH} for some reward scaling constant c, as in

$$\Sigma = \sigma_0 e^H = \sigma_0 U e^\Lambda U^\top \quad \text{where} \quad H(s) = U\Lambda U^\top.$$

Thus, by exploring with (scaled) covariance e^{cH}, we obtain a critic-driven alternative to the Ornstein-Uhlenbeck heuristic of (2). Our results below show that it also performs much better in practice.

In order to show that exponentiating the eigenvalues is really necessary, we also implemented a simpler version, which we call 1-step EPG. It approximates the matrix exponential as

$$\sigma_0 e^H \approx \sigma_0 U \max(1 + \Lambda, 0) U^\top \quad \text{where} \quad H(s) = U\Lambda U^\top.$$

Here, the max operator applies to each entry in the diagonal matrix separately. This process corresponds to truncating the Taylor series of an exponential function after the linear term and then constraining the eigenvalues to be positive. It can also be interpreted as performing just one iteration of (4.2), starting with initial covariance $\sigma_0 I$.

Lemma 6 has an intuitive interpretation. If $H(s)$ has a large positive eigenvalue λ, then $\hat{Q}(\cdot, s)$ has a sharp minimum along the corresponding eigenvector, and the corresponding eigenvalue of $\Sigma^{1/2}$ is e^λ, i.e., also large. This is easiest to see with a one-dimensional action space, where the Hessian and its only eigenvalue are scalar. The exploration mechanism in the one-dimensional case is illustrated in Figure 2. If λ is positive, we have a minimum and apply exploration noise. If, on the other hand, λ is negative, then $\hat{Q}(\cdot, s)$ has a maximum and so e^λ is small, leading to a policy that does not deviate too much from the existing maximum.

In the multi-dimensional case, the critic can have saddle points, as shown in Figure 3. For the case shown in the figure, we explore little along the blue eigenvector (since the intersection of $Q(\cdot, s)$ with the blue plane shows a maximum) and much more along the red eigenvector (since the intersection of $Q(\cdot, s)$ with the red plane shows a minimum, which we want to escape). In essence, we apply the one-dimensional reasoning shown in Figure 2 to each plane separately, where the planes are spanned by the corresponding eigenvector and the z-axis. This way, we can escape saddle points and minima.[8] In practice, we show experimentally that good performance is obtained if we consider quadratic functions constrained to have a diagonal Hessian, i.e., where eigenvectors are axis-aligned.

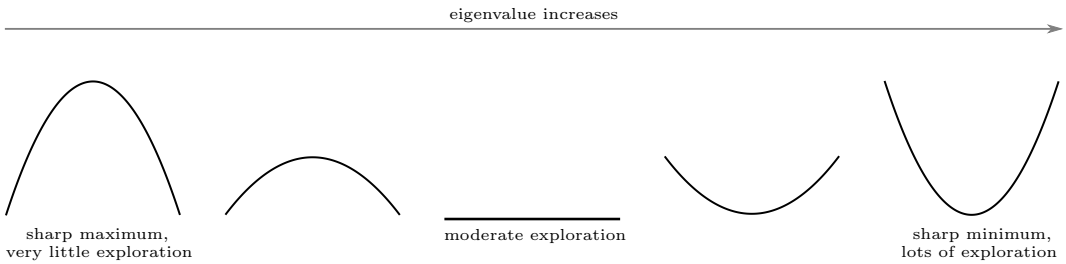


Figure 2: The parabolas show different possible curvatures of the critic $\hat{Q}(\cdot, s)$. We set exploration to be the strongest for sharp mimima, on the left side of the figure. The exploration strength then increases as we move towards the right. There is almost no exploration to the far left, where we have a sharp maximum.

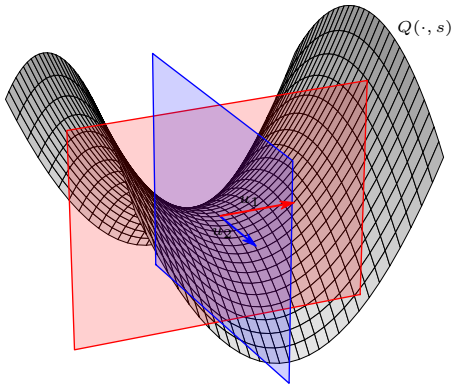
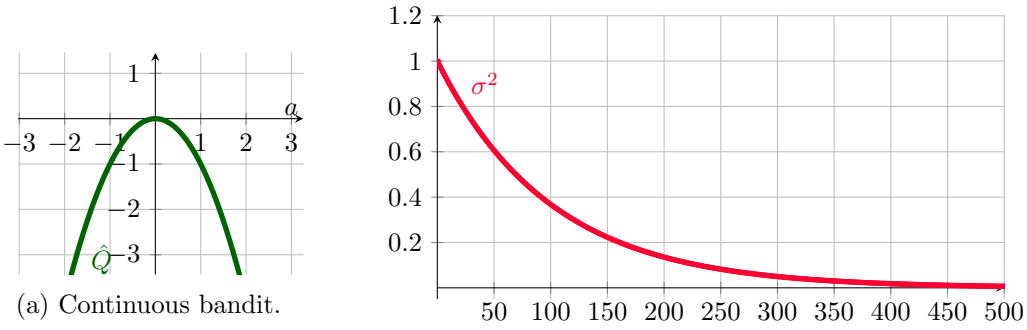


Figure 3: In multi-dimensional action spaces, the critic $\hat{Q}(\cdot, s)$ can have saddle points. In this case, we define exploration along each eigenvector separately.

4.3. Behavior of Policy Gradients across Optimization Time

To provide additional intuition about Lemma 6, we now analyze how Gaussian Policy Gradients behave across optimization time. Consider a simple bandit task shown in Figure 4a, where there is only one state and the critic is defined by $\hat{Q}(a) = -a^2$. Applying equation (4), we have that, as the policy gradient update is applied again and again, the variance evolves according to the equation $\dot{\sigma} = -\sigma$. Assuming the initial value of σ is 1, the standard deviation at time t is given by e^{-t}. In this equation, t is optimization time, which is external to the bandit problem.

7. Lemma 6 relies crucially on the use of special constant step sizes that diminish as we consider longer and longer trajectories. This step sequence serves as a useful intermediate stage between simply taking *one* PG step of (4.2) and using conventional step sizes, which would mean that the covariance would either converge to zero or diverge to infinity.
8. Of course the optimization is still local and there is no guarantee of finding a global optimum—we can merely increase our chances.



(a) Continuous bandit.

(b) Policy variance as a function of optimization time.

Figure 4: Policy Gradient on a bandit problem across optimization time.

Figure 4b plots the standard deviation as a function of optimization time. The plot confirms the intuition that policy gradient algorithms try to find the policy with the maximum expected value of $\hat{Q}$. In particular, σ^2 decays to zero because the critic $\hat{Q}$ has one maximum $a = 0$, where the optimal policy puts all the probability mass. This is also consistent with Lemma 6, which corresponds to fixing $t = 1$.

4.4. Action Clipping

We now describe how GPG works in environments where the action space has bounded support[9]. This setting occurs frequently in practice, since real-world systems often have physical constraints such as a bound on how fast a robot arm can accelerate. The typical solution to this problem is simply to start with a policy π_b with unbounded support and then, when an action is to be taken, clip it to the desired range, so that sampling an action

$$a \sim \pi(a \mid s) \quad \text{is equivalent to} \quad a = \max(\min(b, 1), 0) \;\; \text{with} \;\; b \sim \pi_b(b \mid s).$$

The justification for this process is that we can simply treat the clipping operation $\max(\min(b, 1), 0)$ as part of the environment specification. Formally, this means that we transform the original MDP M defined as $M = (S, A, R_d, p, p_0, \gamma)$ with $A = [0, 1]^d$ into another MDP $M' = (S, A', R'_d, p', p_0, \gamma)$, where $A' = \mathbb{R}^d$ and p' and R'_d are defined as

$$p'(s'|b, s) \;=\; p\left(s' | \max(\min(b, 1), 0), s\right) \quad \text{and} \quad R'_d(r|b, s) \;=\; R_d\left(r | \max(\min(b, 1), 0), s\right).$$

Since M' has an unbounded action space, we can use the RL machinery for unbounded actions to solve it. Since any MDP is guaranteed to have an optimal deterministic policy, we call this deterministic solution $\pi^\star_D : S \to A$. Now, $\pi^\star_D$ can be transformed into a policy for M of the form $\max(\min(\pi^\star_D(s), 1), 0)$. In practice, the MDP M' is never constructed explicitly—the described process in equivalent to using an RL algorithm meant for $A = \mathbb{R}^d$ and then, when the action is generated, simply clipping it (Algorithm 4).

However, while such an algorithm does not introduce new bias in the sense that reward obtained in M and M' will be the same, it can lead to problems with slow convergence in the policy gradient settings. To see why, consider the one-dimensional example in Figure

9. We assume without loss of generality that the support interval is $[0, 1]$.

Algorithm 4 Policy gradients with clipped actions.

$s \leftarrow s_0$, $t \leftarrow 0$
initialize optimizer, initialize policy π parameterized by θ
while not converged **do**
    $g_t \leftarrow \gamma^t$ DO-INTEGRAL$(\hat{Q}_b, s, \pi_\theta)$
    $\theta \leftarrow \theta +$ optimizer.UPDATE(g_t)
    $b \sim \pi(\cdot \mid s)$
    $a = c(b)$ ▷ Clipping function $c(b) = \max(\min(\pi^\star_D(s), 1), 0)$.
    $s', r \leftarrow$ simulator.PERFORM-ACTION(a)
    $\hat{Q}_b$.UPDATE(s, b, r, s') ▷ Update using the unclipped action b.
    $t \leftarrow t + 1$
    $s \leftarrow s'$
end while

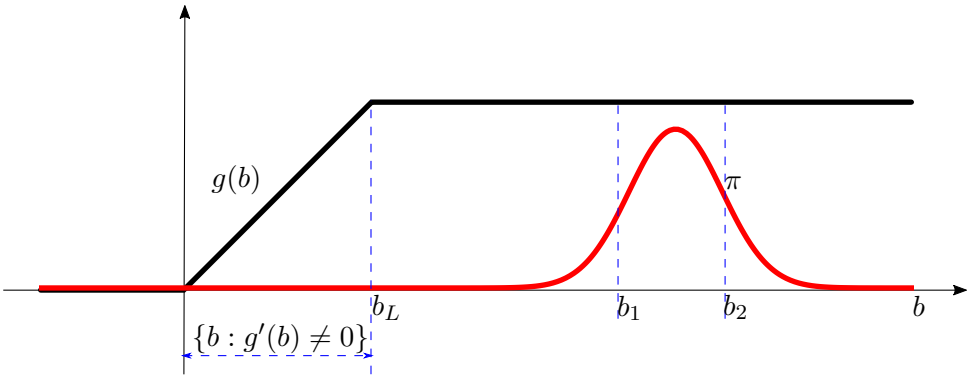


Figure 5: Vanishing gradients when using hard clipping. The agent cannot determine whether b is too small or too large from b_1 and b_2 alone. It is necessary to sample from the interval $\{b : g'(b) \neq 0\}$ in order to obtain a meaningful policy update but this is unlikely for the current policy (shown as the red curve).

5, where the policy mean is located far away from the clipping boundary. This can arise due to a combination of random initialization of the policy network and generalization error across states.

With hard clipping, the agent cannot distinguish between b_1 and b_2 since squashing reduces them both to the same value, i.e., $g(b_1) = g(b_2)$. Hence, the corresponding Q values are identical and, based on trajectories using b_1 and b_2, there is no way of knowing how the mean of the policy should be adjusted. In order to get a useful gradient that moves the distribution into the interval $[0, b_L]$, a sample $b_\star < b_L$ has to be chosen. Since the b's are samples from a Gaussian with infinite support, it will eventually happen, yielding a nonzero gradient. However, if this interval falls into a distant part of the tail of π_b, convergence will be slow.

GPG mitigates this problem because it always explores with standard deviation σ_0 in the flat region. This is shown in Figure 6. For actions $b > b_L$, the critic is constant. Since a

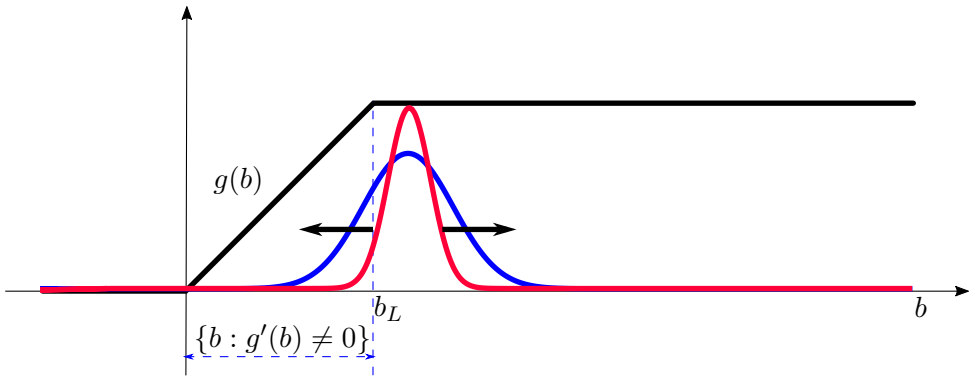


Figure 6: GPG avoids the vanishing gradient problem. Once a policy with a small variance (denoted in red) enters the flat area where $b > b_L$, exploration noise is set to σ_0 (the new distribution is in blue).

constant critic has a zero Hessian, the standard deviation of the policy is set to $\sigma_0 e^0 = \sigma_0$, which makes it likely that a point $b < b_L$ is sampled and a useful gradient is obtained for typical values of σ_0. In contrast, consider the case where stochastic policy gradients were used to tune the standard deviation. It follows from setting $A(s) = 0$ in (4)) that for SPG, the policy gradient would be zero in expectation. Then, if the policy standard deviation is initialized to a small value, it likely to to remain stuck and take prohibitively long to leave the plateau. Another way of mitigating the hard clipping problem is to use a differentiable squashing function, which we describe in Section 5.

4.5. Quadratic Critics and their Approximations

Gaussian policy gradients require a quadratic critic *given the state*. This assumption, which is different from assuming a quadratic dependency *on the state*, is typically sufficient for two reasons. First, discrete-time linear quadratic regulators (LQR) with time-varying feedback, a class of problems widely studied in classical control[10] theory, are known to have a Q-function that is quadratic in the action vector given the state (Bradtke 1993, Equation 1; ten Hagen, S.H.G. et al. 1998, Equation 10; Peters et al. 2003; Crassidis and Junkins 2011, Equation 8.81). Second, it is often assumed (Li and Todorov, 2004) that a quadratic critic (or a quadratic approximation to a general critic) is enough to capture enough local structure to perform a policy optimization step, in much the same way as Newton's method for deterministic unconstrained optimization, which locally approximates a function with a quadratic, can be used to optimize a non-quadratic function across several iterations. In Corollary 7 below, we describe such an approximation method applied to GPG where we approximate Q with a quadratic function in the neighborhood of the policy mean.

Corollary 7 (Approximate Gaussian Policy Gradients with Arbitrary Critic) *If the policy is Gaussian, i.e.* $\pi(\cdot|s) \sim \mathcal{N}(\mu_s, \Sigma_s^{1/2})$ *with* μ_s *and* $\Sigma_s^{1/2}$ *parameterized by* θ *as in Lemma 5 and any critic* $\hat{Q}(a, s)$ *doubly differentiable with respect to actions for each*

10. Indeed, the Hessian discussed in Section 4.2 can be considered a type of reward model.

state, then $I^{\hat{Q}}_{\pi(s),\mu_s} \approx (\nabla_\theta \mu_s)\nabla_a \hat{Q}(a=\mu_s, s)$ *and* $I^{\hat{Q}}_{\pi(s),\Sigma_s^{1/2}} \approx (\nabla_\theta \Sigma_s^{1/2}) H(\mu_s, s)\Sigma_s^{1/2}$*, where* $H(\mu_s, s)$ *is the Hessian of* $\hat{Q}$ *with respect to* a*, evaluated at* μ_s *for a fixed* s*.*

Proof We begin by approximating the critic (for a given s) using the first two terms of the Taylor expansion of $\hat{Q}$ in μ_s. This allows us to approximate the critic as

$$\begin{aligned}\hat{Q}(a,s) &\approx \hat{Q}(\mu_s, s) + (a-\mu_s)^\top \left[\nabla_a \hat{Q}(a,s)\right]_{a=\mu_s} + \tfrac{1}{2}(a-\mu_s)^\top H(\mu_s,s)(a-\mu_s) \\ &= \tfrac{1}{2} a^\top H(\mu_s,s) a + a^\top \left(\left[\nabla_a \hat{Q}(a,s)\right]_{a=\mu_s} - H(\mu_s,s)\mu_s\right) + \text{const}_s.\end{aligned}$$

We used the notation const_s to denote a term constant in the action (but dependent on the state). Because of the series truncation, the function on the right-hand side is quadratic and we can then use Lemma 5, obtaining

$$\begin{aligned} I^{\hat{Q}}_{\pi(s),\mu_s} &= \nabla_\theta \mu_s (2\tfrac{1}{2} H(\mu_s,s)\mu_s + \left[\nabla_a \hat{Q}(a,s)\right]_{a=\mu_s} - H(\mu_s,s)\mu_s) \\ &= \nabla_\theta \mu_s \left[\nabla_a \hat{Q}(a,s)\right]_{a=\mu_s}, \text{ and} \\ I^{\hat{Q}}_{\pi(s),\Sigma_s^{1/2}} &= \nabla_{\Sigma_s^{1/2}} (2\tfrac{1}{2} H(\mu_s,s)\Sigma_s^{1/2}) = \nabla_{\Sigma_s^{1/2}} H(\mu_s,s)\Sigma_s^{1/2}. \end{aligned}$$

■

To actually obtain the Hessian, we could use automatic differentiation to compute it analytically. Sometimes this may not be possible—for example when ReLU units are used, the Hessian is always zero or undefined. In these cases, we can approximate the Hessian by generating a number of random action-values around μ_s, computing the $\hat{Q}$ values, and (locally) fitting a quadratic, akin to sigma-point methods in control (Roth et al., 2016).

5. Universal Expected Policy Gradients

Having covered the most common case of continuous Gaussian policies, we now extend the analysis to other policy classes. We provide two cases of such results in the following sections: exponential family policies with multivariate polynomial critics (of arbitrary order) and arbitrary policies (possessing a mean) with linear critics. Our main claim is that an analytic solution to the EPG integral is possible for almost any system; hence we describe EPG as a *universal* method.[11]

5.1. Exponential Family Policies and Polynomial Critics

We now describe a general technique to obtain analytic EPG updates for the case when the policy belongs to a certain exponential family and the critic is an arbitrary polynomial.

11. Of course no method can be truly universal for a *completely arbitrary* problem. Our claim is that EPG is universal *for the class of systems* arising from lemmas in this section. However, this class is so broad that we feel the term 'universal' is justified. This is similar to the claim that neural networks based on sigmoid nonlinearities are universal, even though then can only approximate continuous functions, as opposed to completely arbitrary ones.

This result is significant since polynomials can approximate any continuous function on a bounded interval with arbitrary accuracy (Weierstrass, 1885; Stone, 1948). Since our result holds for a nontrivial class of distributions in the exponential family, it implies that analytic solutions for EPG can almost always be obtained in practice and hence that the Monte Carlo sampling to estimate the inner integral that is typical in SPG is not necessary in many cases.

Lemma 8 (EPG for Exponential Families w. Polynomial Sufficient Statistics) *Consider the class of policies parameterized by θ defined by the formula*

$$\pi(a \mid s) = e^{(\eta_\theta^s)^\top T^s(a) - U^s_{\eta_\theta^s} + W^s(a)},$$

where each entry in the vector $T^s(a)$ is a (possibly multivariate) polynomial in the entries of the vector a. Moreover, assume that the critic $\hat{Q}(a, s)$ is (a possibly multivariate) polynomial in the entries of a. Then, the policy gradient update is a closed form expression in terms of the uncentered moments of $\pi(\cdot \mid s)$ and can be written as

$$I_\pi^Q(s) = (\nabla_\theta(\eta_\theta^s)^\top)((C^s_{TQ})^\top m_\pi) - (\nabla_\theta U^s_{\eta_\theta^s})((C^s_Q)^\top m_\pi),$$

where C^s_Q is the vector containing the coefficients of the polynomial $\hat{Q}(\cdot, s)$, C^s_{TQ} is the vector containing the coefficients of the polynomial $T^s(a)\hat{Q}(a, s)$, i.e., a multiplication of T^s and $\hat{Q}(a, s)$. Moreover, m_π is a vector of uncentered moments of π (in the order matching the polynomials).

The lemma is proved in the Appendix. The cross-moments themselves can be obtained from the *moment generating function* (MGF) of π. Indeed, for a distribution of the form of (8), the MGF of $T^s(a)$ is guaranteed to exist and has a closed form (Bickel and Doksum, 2006). Hence, the computation of the moments reduces to the computation of derivatives. See details in Appendix A.3.

In Lemma 8, the assumption that T^s and $\hat{Q}(a, s)$ are polynomial is with respect to the action a. The dependence on the state can be arbitrary, e.g., a multi-layered neural network.

Of course, while polynomials are universal approximators, they may not be the most efficient or stable ones. The importance of Lemma 8 is currently mainly conceptual—analytic EPG is possible for a universal class of approximators (polynomials) which shows that EPG is analytically tractable in principle for any continuous Q-function.[12] It is an open research question whether more suitable universal approximators admitting analytic EPG solutions can be identified.

5.2. Reparameterized Exponential Families and Reparameterized Critics

In Lemma 8, we assumed that the function $T^s(a)$ (called the sufficient statistic of the exponential family) is polynomial. We now relax this assumption. Our approach is to start

12. The universality of polynomials holds only for bounded intervals (Weierstrass, 1885), while the support of the policy may be unbounded. We do not address the unbounded approximation case here other than by saying that, in practice, the critic is learned from samples and is thus typically only accurate on a bounded interval anyway.

with a policy π_b which *does* have a polynomial sufficient statistic and then introduce a suitable reparameterization function $g : \mathbb{R}^d \to A$. The policy is then defined so that sampling an action

$$a \sim \pi(a \mid s) \quad \text{is equivalent to} \quad a = g(b) \;\text{ with }\; b \sim \pi_b(b \mid s) = e^{(\eta_\theta^s)^\top T^s(b) - U^s_{\eta_\theta^s} + W^s(b)},$$

where b is the random variable representing the action before the squashing. Assuming that g^{-1} exists and the Jacobian[13] $\mathbf{Jb}^\theta g$ is non-singular almost everywhere, the PDF[14] of the policy π can be written as

$$\pi(a \mid s) = \pi_b(g^{-1}(a) \mid s) \frac{1}{\det \mathbf{Jb}^\theta g(g^{-1}(a))} = \pi_b(b \mid s) \frac{1}{\det \mathbf{Jb}^\theta g(b)}.$$

The following lemma develops an EPG method for such policies.

Lemma 9 *Consider an invertible and differentiable function g. Define a policy π as in* (5.2)*. Assume that the Jacobian of g is non-singular except on a set of π_b-measure zero. Consider a critic $\hat{Q}$. Denote as $\hat{Q}_b$ a reparameterized critic such that for all a, $\hat{Q}_b(g^{-1}(a), s) = \hat{Q}(a, s)$. Then the policy gradient update is given by the formula $I_\pi^{\hat{Q}}(s) = I_{\pi_b}^{\hat{Q}_b}(s)$.*

We are now ready to state our universality result. The idea is to obtain a reparameterized version of EPG (and Lemma 8) by reparametrizing the critic and the policy using the same transformation g. We do so in the following corollary, which is the most general constructive result in this article.

Corollary 10 (EPG for Exponential Families with Reparametrization) *Consider the class of policies, parameterized by θ, defined as in* (8)*. Consider reparametrization function g and define T_b^s, V_b^s and $\hat{Q}_b^s$ as $T_b^s(g^{-1}(a)) = T^s(a)$, $W_b^s(g^{-1}(a)) = W^s(a)$ and $\hat{Q}_b(g^{-1}(a), s) = \hat{Q}(a, s)$ for every a. Assume the following:*

1. *g is invertible;*
2. *The Jacobian of g exists and is non-singular except on a set of π_b-measure zero, where π_b is the reparameterized policy as in* (5.2)*; and*
3. *T_b^s and $\hat{Q}_b^s$ are polynomial as in Lemma 8.*

Then a closed-form policy gradient update can be obtained as

$$I_\pi^{\hat{Q}}(s) = (\nabla_\theta \eta_\theta^\top)((C^s_{T_b Q_b})^\top m^s_{\pi_b}) - (\nabla_\theta U^s_{\eta_\theta})((C^s_{Q_b})^\top m^s_{\pi_b}).$$

Proof Apply Lemmas 9 and then 8. ■

Lemma 9 also has a practical application in case we want to deal with bounded action spaces. As we discussed in Section 4.4, hard clipping can cause the problem of vanishing gradients and the default solution should be to use GPG. In case we can't use GPG, for

13. We avoid the standard Jacobian notation $\mathbf{J}$ because it is too similar to the total RL return J.
14. We abuse notation slightly by using $\pi(a \mid s)$ for both the probability distribution and its PDF.

instance when the dimensionality of the action space is so large that computing the covariance of the policy is too costly, we can alleviate the vanishing gradients problem by using a strictly monotonic squashing function g. One implication of Lemma 9 is that, if we set π_b to be Gaussian, we can invoke Lemma 5 to obtain exact analytic updates for useful policy classes such as Log-Normal and Logit-Normal (obtained by setting g to the sigmoid and the exponential function respectively), as long as we choose our critic $\hat{Q}^s$ to be quadratic in $g^{-1}(a)$, i.e., $\hat{Q}^s_b$ is quadratic in b. The reparameterized version of EPG is the same as Algorithm 4 except it uses a squashing function g instead of the clipping function c.

5.3. Arbitrary Policies and Linear Critics

Next, we consider the case where the stochastic policy is almost completely arbitrary, i.e., it only has to possess a mean and need not even be in the already general exponential family of policies used in Lemma 8 and Corollary 10, but the critic is constrained to be linear in the actions. We have the following lemma, which is a slight modification of an observation made in connection with the Q-Prop algorithm (Gu et al., 2016a, Eq. 7).

Lemma 11 (EPG for Arbitrary Stochastic Policies and Linear Critics) *Consider an arbitrary (non-degenerate) probability distribution $\pi(\cdot \mid s)$ which has a mean. Assume that the critic $\hat{Q}(a, s)$ is of the form $A_s^\top a$ for some coefficient vector A_s. Then the policy gradient update is given by $I_\pi^{\hat{Q}}(s) = A_s^\top \nabla_\theta \mu_{\pi(\cdot|s)}$ where $\mu_{\pi(\cdot|s)}$ denotes the integral $\int_a a d\pi(a \mid s)$ (the mean).*

Proof The lemma is proven by rewriting the policy gradient update as

$$\begin{aligned} I_\pi^Q(s) &= \int_a \nabla_\theta \pi(a \mid s) \hat{Q}(a, s) da = \int_a \nabla_\theta \pi(a \mid s) A_s^\top a da \\ &= A_s^\top \nabla_\theta \underbrace{\int_a \pi(a \mid s) a da}_{\mu_{\pi(\cdot|s)}} = A_s^\top \nabla_\theta (\mu_{\pi(\cdot|s)}). \end{aligned}$$

■

Since DPG already provides the same result for Dirac-delta policies (see Corollary 3), we conclude that using linear critics means we can have an analytic solution for any reasonable policy class.

To see why the above lemma is useful, first consider systems that arise as a discretization of continuous time systems with a fine enough time scale and differentiable dynamics and rewards. If we assume that the true Q is smooth in the actions and that the magnitude of the allowed action goes to zero as the time step decreases, then a linear critic is sufficient as an approximation of Q because we can approximate any smooth function with a linear function in any sufficiently small neighborhood of a given point and then choose the time step to be small enough so an action does not leave that neighborhood. We can then use Lemma 11 to perform policy gradients with any policy.[15]

15. Of course the update derived in Lemma 11 only provides a direction in which to change the policy mean (which means that exploration has to be performed using some other mechanism). This is because a linear critic does not contain enough information to determine exploration.

5.4. If All Else Fails: EPG with Numerical Quadrature

If, despite the broad framework shown in this article, an analytical solution is impossible, we can still perform integration numerically. EPG can still be beneficial in these cases: if the action space is low dimensional, numerical quadrature is cheap; if it is high dimensional, it is still often worthwhile to balance the expense of simulating the system with the cost of quadrature. Actually, even in the extreme case of expensive quadrature but cheap simulation, the limited resources available for quadrature could still be better spent on EPG with smart quadrature than SPG with simple Monte Carlo.

The crucial insight behind numerical EPG is that the integral given as[16]

$$I_{\pi}^{\hat{Q}} = \int_a d\pi(a \mid s) \nabla_\theta \log \pi(a \mid s) \hat{Q}(a, s) = \int_a d\pi(a \mid s) \nabla_\theta \hat{Q}(\pi(a_i), s)$$

only depends on two fully known quantities: the current policy π and the current approximate critic $\hat{Q}$. Therefore, we can use any standard numerical integration method to compute it. For example, multi-sample Monte-Carlo quadrature for the policy gradient with respect to the mean is given by

$$I_{\pi(s),\mu_s}^{\hat{Q}} = \frac{1}{m} \sum_{i=1}^{m} (\nabla_\theta \hat{Q}(\pi(a_i), s)) \quad \text{where} \quad a_i \sim \pi(\cdot|s).$$

The actions at which the integrand is evaluated do not have to be sampled—one can also use a method such as the Gauss-Legendre quadrature where the abscissae are designed.

5.5. Probability Distributions over Discrete Actions and Softmax Policies

The main idea of EPG can also be applied to the setting of discrete actions. In this case, the integral in $I_{\pi}^{Q}(s)$ becomes a sum, and the policy gradient update takes the form

$$I_{\pi}^{Q}(s) = \sum_a \pi(a \mid s) \nabla \log \pi(a \mid s) Q(a, s) = \sum_a \nabla \pi(a \mid s) Q(a, s).$$

In this case, a softmax parametrization of the discrete policy, also known as a Gibbs or Boltzmann policy, is often chosen. The following observation provides a slightly optimized formula for the sum.

Observation 12 (Expected Policy Gradients for Discrete Softmax Policies) *If the action space is discrete, and the policy is a discrete softmax distribution, i.e., $\pi(\cdot|s) \propto e^{h_\theta(\cdot,s)}$ for some function h_θ parametrized by θ, then*

$$I_{\pi}^{Q}(s) = (\nabla h) u.$$

Here, ∇h is a Jacobian matrix and u is a vector whose elements are defined as

$$\{u\}_i = \pi(a_i|s) \left(\sum_j \pi(a_j|s)(Q(a_i, s) - Q(a_j, s)) \right) = \pi(a_i|s) Q(a_i, s) - V(s).$$

16. The second expression is known as the reparameterized gradient and was introduced by Heess et al. (2015) in the context of RL.

Proof Denote by $Z(s)$ the normalization factor of the policy, i.e. $Z(s) = \sum_i e^{h_\theta(a_i,s)}$ and $\pi(a_i|s) = \frac{e^{h_\theta(a_i,s)}}{Z(s)}$. First, we expand the term $\nabla\pi(a_i|s)$ as

$$\nabla\pi(a_i|s) = \frac{e^{h_\theta(a_i,s)}}{Z(s)}\nabla h_\theta(a_i,s) - \frac{e^{h_\theta(a_i,s)}(\sum_j e^{h_\theta(a_j,s)}\nabla h_\theta(a_j,s))}{Z(s)^2} = \\ = \pi(a_i|s)\nabla h_\theta(a_i,s) - \pi(a_i|s)\sum_j \pi(a_j|s)\nabla h_\theta(a_j,s).$$

We now plug this into the definition of $I_\pi^Q(s)$, obtaining

$$\begin{aligned} I_\pi^Q(s) &= \textstyle\sum_i \nabla\pi(a_i|s)Q(a_i,s) = \\ &= \left(\textstyle\sum_i \pi(a_i|s)\nabla h_\theta(a_i,s)Q(a_i,s)\right) - \left(\textstyle\sum_j \pi(a_j|s)\nabla h_\theta(a_j,s)\right)\left(\textstyle\sum_i \pi(a_i|s)Q(a_i,s)\right) = \\ &= \left(\textstyle\sum_i \pi(a_i|s)\nabla h_\theta(a_i,s)Q(a_i,s)\right) - \left(\textstyle\sum_i \pi(a_i|s)\nabla h_\theta(a_i,s)\right)\left(\textstyle\sum_j \pi(a_j|s)Q(a_j,s)\right) = \\ &= \textstyle\sum_{ij} \pi(a_i|s)\pi(a_j|s)\nabla h_\theta(a_i,s)(Q(a_i,s) - Q(a_j,s)) \\ &= \textstyle\sum_i \pi(a_i|s)\nabla h_\theta(a_i,s)(Q(a_i,s) - V(s)). \end{aligned}$$

The last two lines give the desired result. ■

We include this observation because the required simplifications, leading to (12) may not always be performed when using automatic differentiation software. Also, (12) makes clear that only differences between Q-values matter, not absolute values.

To apply Observation 12 in practice, Q and V are replaced by their approximations $\hat{Q}$ and $\hat{V}$ respectively, similarly to the continuous case. However, the use of EPG for discrete policies did not improve performance for a task we tried, a result we discuss in Section 6.5.

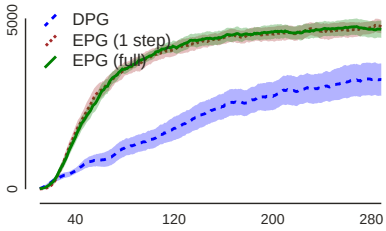
6. Experiments

While EPG has many potential uses, we focus on empirically evaluating its applications to exploration and variance reduction in the actor. To benchmark our algorithms, we use five continuous-action domains, modeled with the MuJoCo physics simulator (Todorov et al., 2012): HalfCheetah-v2, InvertedPendulum-v2, Reacher2d-v2, Walker2d-v2, and InvertedDoublePendulum-v2, as well as one discrete-action domain: Atari Pong. In Section 6.1, we evaluate the benefits of exploring using a covariance matrix that comes from the Hessian exponential. In Section 6.2, we compare the types of quadrature typically used in connection with policy gradients. In Section 6.3, we discuss hyperparameter tuning. In Section 6.4, we compare EPG with *proximal policy optimization* (PPO) (Schulman et al., 2017). In Section 6.5, we apply EPG to the discrete-action domain, Atari Pong.

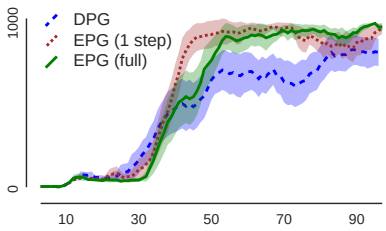
6.1. Continuous Control: Exploration

The EPG framework can be used to derive a new policy gradient algorithm (see Algorithm 2 and Lemma 6), where the covariance of the exploration policy is obtained by either taking the matrix exponent of the critic as given in (4.2) or its approximation, which we call 1-step EPG, as given in (4.2). In practice, both versions of EPG differ from deep DPG (Lillicrap

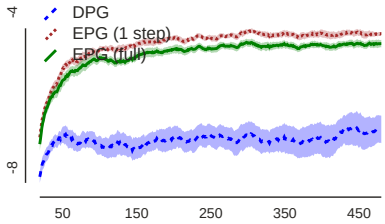
(a) HalfCheetah-v2



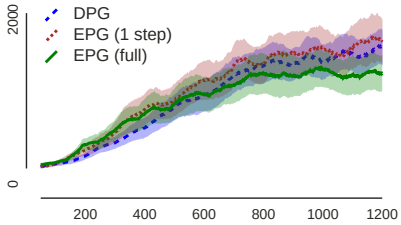
(b) InvertedPendulum-v2



(c) Reacher-v2



(d) Walker2d-v2



(e) InvertedDoublePendulum-v2

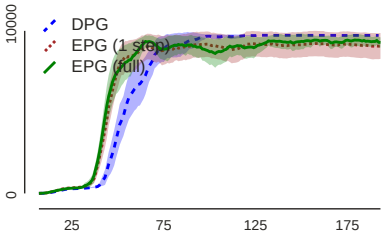
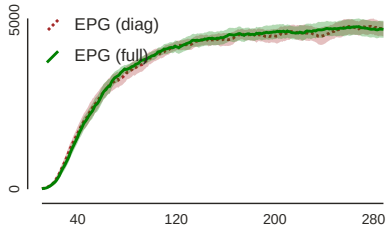


Figure 7: Learning curves (mean and 90% interval) showing exploration using EPG, 1-step EPG and the OU noise used in DPG. Returns for Reacher-v2 are clipped at -10. All results are obtained from 20 runs. Horizontal axis shows thousands of steps.

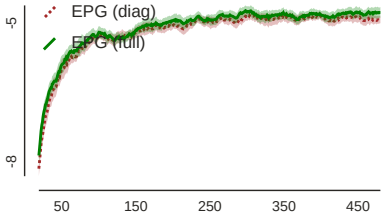
et al., 2015; Silver et al., 2014) only in the exploration strategy, though their theoretical underpinnings are also different.

The Hessian is obtained using a sigma-point method, as follows. At each step, the agent samples 100 action values from $\hat{Q}(\cdot, s)$ and a quadratic is fit to them in the L_2 norm. Since this is a least-squares problem, it can be accomplished by solving a linear system. The Hessian computation could be greatly sped up by using an approximate method, or even skipped completely if we use a quadratic critic. However, we did not optimize this part of

(a) HalfCheetah-v2



(b) Reacher-v2



(c) Walker2d-v2

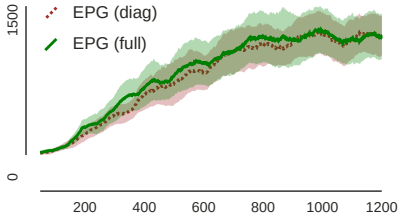


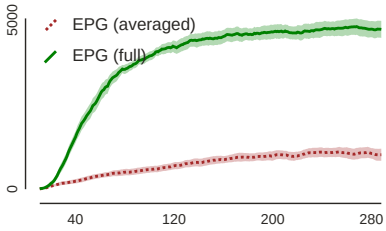
Figure 8: Learning curves (mean and 90% interval) comparing EPG with diagonal and full Hessian. Returns for Reacher-v2 are clipped at -10. All results are obtained from 20 runs. Horizontal axis shows thousands of steps. Results for the pendulum domains are not show since they only have one action dimension.

the algorithm since it is orthogonal to the core insight of GPG that the Hessian is useful for exploration.

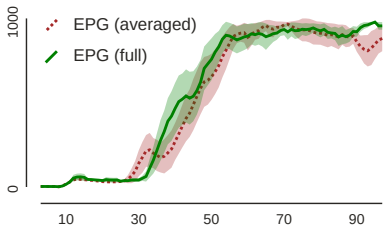
We now evaluate both EPG and 1-step EPG, as an alternative to the standard Ornstein-Uhlenbeck (OU) exploration used by Deterministic Policy Gradients. The results in Figure 7 show that EPG's exploration strategy yields much better performance than DPG with OU. 1-step EPG performs just well, or even slightly better than the version with the exponent. This is not surprising – the function $\max(0, 1 + \lambda)$ is a good approximation of e^{λ} in for the range of eigenvalues seen during training.

In order to benchmark more accurately which aspects of EPG are important for performance, we performed two additional ablations. Figure 8 compares EPG with a diagonal version, which uses a diagonal Hessian. The diagonal Hessian is fitted using the same process as the full Hessian, except that the local quadratic approximation is constrained so that the off-diagonal entries are zero. The performance of diagonal EPG is similar to the full version, showing that the critic curvature that has practical significance for exploration can be estimated component-wise. This result is consistent with most recent work on the topic (Haarnoja et al., 2018; Fujimoto et al., 2018).

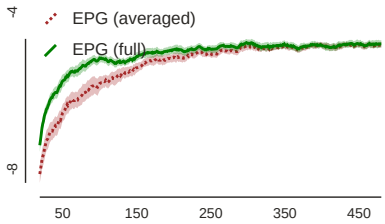
(a) HalfCheetah-v2



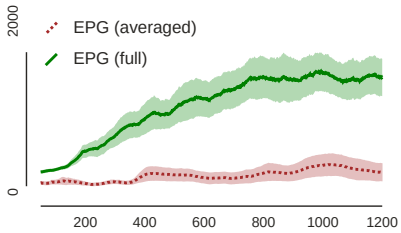
(b) InvertedPendulum-v2



(c) Reacher-v2



(d) Walker2d-v2



(e) InvertedDoublePendulum-v2

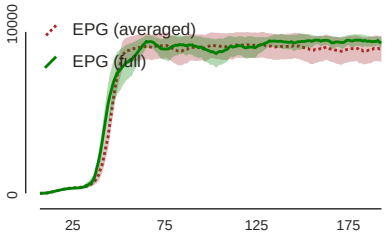
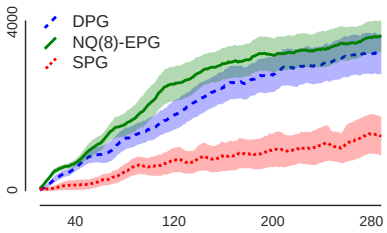


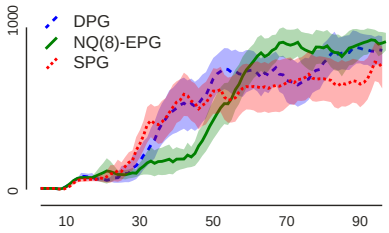
Figure 9: Learning curves (mean and 90% interval) showing exploration using vanilla EPG and a variant that uses a averaged covariance (i.e. one that is constant for every state). Returns for Reacher-v2 are clipped at -10. All results are obtained from 20 runs. Horizontal axis shows thousands of steps.

To make an even more minimal version of EPG, we tried to estimate the Hessian globally, i.e., using the same estimate for every state. The results in Figure 9 show using such a global Hessian is suboptimal.

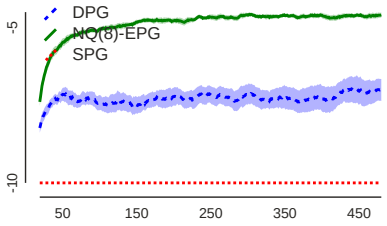
(a) HalfCheetah-v2



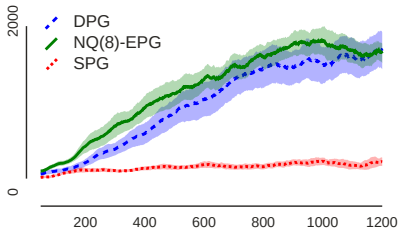
(b) InvertedPendulum-v2



(c) Reacher-v2



(d) Walker2d-v2



(e) InvertedDoublePendulum-v2

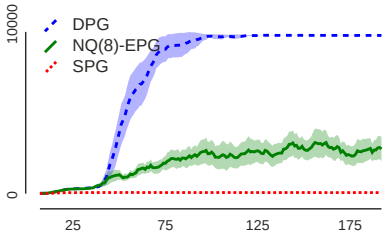
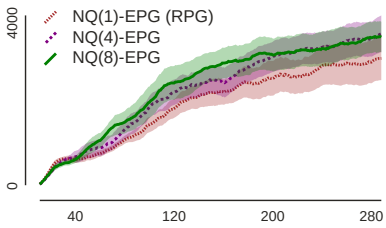


Figure 10: Learning curves (mean and 90% interval) showing learning performance when updating the policy mean using Expected Policy Gradients with numerical quadrature, as compared to deterministic policy gradients and stochastic policy gradients. Returns for Reacher-v2 are clipped at -10. All results are obtained from 20 runs. Horizontal axis shows thousands of steps.

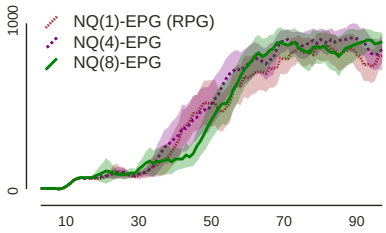
6.2. Continuous Control: Updates for the Policy Mean

In this section, we compare different ways of updating the policy mean. We test an expected policy gradient method based on numerical quadrature NQ(m)-EPG, which computes estimates of the policy gradient using multiple samples of the policy gradient integral and compare it to DPG and SPG. The quadrature is performed as in (5.4).

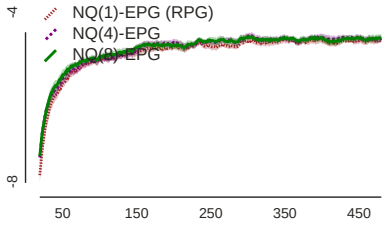
(a) HalfCheetah-v2



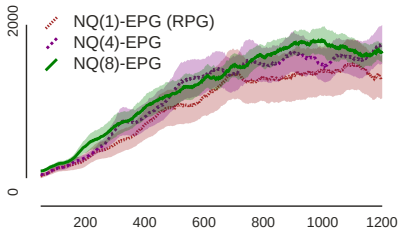
(b) InvertedPendulum-v2



(c) Reacher-v2



(d) Walker2d-v2



(e) InvertedDoublePendulum-v2

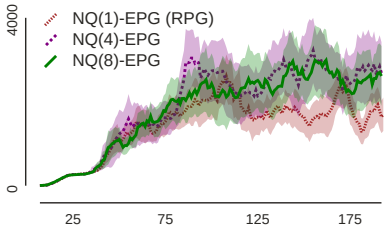


Figure 11: Learning curves (mean and 90% interval) showing learning performance when updating the policy mean using multi-sample quadrature for variants with 1, 4 and 8 samples. Quadrature with 1 sample is equivalent to reparametrized policy gradients (RPG). Returns for Reacher-v2 are clipped at -10. All results are obtained from 20 runs. Horizontal axis shows thousands of steps.

In order to capture the effect of the actor update only, we used OU exploration with noise 0.2 for both NQ(m)-EPG and DPG. We used Gaussian exploration for SPG. Details of hyperparameters are given in Appendix A.4. Our results in Figure 10 show that using numerical quadrature as in (5.4) generally improves performance. The results in Figure 11 show that adding more samples to numerical quadrature does not change performance much, at least for the simple type of quadrature in (5.4). This is unsurprising given that

NQ(1)-EPG (with one sample) corresponds to Reparameterized Policy Gradients, which are known to perform well (Haarnoja et al., 2018). To obtain a larger improvement on these tasks one would need to use either more sophisticated quadrature or a local parametric approximation similar to the sigma-point method in Section 6.1.

The behavior of the InvertedDoublePendulum domain, where numerical quadrature performs significantly worse than standard deterministic policy gradient update, is interesting. We believe this behavior is due to the task's inherent instability – any deviation from the true value leads to sub-optimal updates. One way of addressing this problem would be to reduce the exploration noise. However, in order to get meaningful comparisons, we elected to run our experiments with the same hyperparameters across tasks.

Furthermore, SPG does poorly, solving only the easiest domain (InvertedPendulum-v2) in reasonable time, achieving slow progress on HalfCheetah-v2, and failing entirely on the other domains. This is not surprising since DPG was introduced precisely to solve the problem of high variance SPG estimates on this type of task. In InvertedPendulum-v2, SPG initially learns quickly, outperforming the other methods, because noisy gradient updates provide a crude, indirect form of exploration that happens to suit this problem. Clearly, this is inadequate for more complex domains: even for this simple domain it leads to sub-par performance late in learning.

6.3. Sensitivity of EPG to hyperparameters

The hyperparameters for DPG and those of EPG that are not related to exploration were taken from an existing benchmark (Islam et al., 2017; Brockman et al., 2016). They are detailed in Appendix A.4. Our EPG exploration technique has just one hyperparameter σ_0 while OU has two (standard deviation and mean reversion constant). We optimized σ_0 on the HalfCheetah domain (Figure 12) and settled on the value $\sigma_0 = 0.5$.

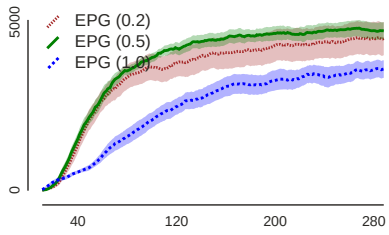


Figure 12: Learning curves (mean and 90% interval) for HalfCheetah-v2 showing different values of σ_0 for EPG. All results are obtained from 20 runs. Horizontal axis shows thousands of steps.

To ensure a fair comparison, we also optimized hyperparameters for DPG; details and learning curves are in Appendix A.4.

6.4. Comparison with PPO.

One feature of EPG is stability in the learning outcome, i.e., low variance across runs. EPG's stability raises the question whether the instability of an algorithm (i.e., an inverted or oscillating learning curve) is caused primarily by inefficient exploration or by excessively large differences between subsequent policies. To address this question, we compare our results with *proximal policy optimization* (PPO) (Schulman et al., 2017), a policy gradient algorithm that explicitly penalizes large differences between successive policies. The results are shown in Figure 13, where we allowed to run PPO until its performance plateaus. On one hand, the results show that EPG is indeed more stable (represented by a narrower confidence interval). On the other hand, PPO performed better overall on the Walker task. This suggests that both the stability relating to exploration and the stability relating to changes in policy space can play a role in policy gradients. In principle, it would be possible to achieve *both* kinds of stability by exploiting the curvature of the critic to obtain the covariance of the policy while at the same time constraining the sequence of policies to be close to one another. Due to the amount of engineering involved in tuning such an algorithm, we leave this idea to future work.

(a) HalfCheetah-v2 (b) Walker2d-v2

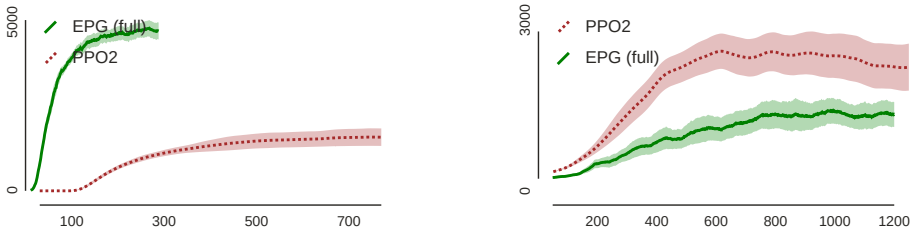


Figure 13: Learning curves (mean and 90% interval) showing learning performance of PPO (using the default parameters) as compared to EPG. All results are obtained from 20 runs. Horizontal axis shows thousands of steps.

6.5. EPG for Discrete Action Spaces

To evaluate the usefulness of Expected Policy Gradients for discrete action spaces, we applied it to the Atari version of Pong. The results are presented in Figure 14. We used the OpenAI version of A2C (Mnih et al., 2016) for stochastic policy gradients (labeled SPG in the plot) as a baseline, together with the default hyperparameters of the OpenAI implementation (Dhariwal et al., 2017). To make a discrete version of EPG, we modified the A2C critic to also learn $\hat{Q}$. We also modified the policy gradient update to EPG as in (5.5) and kept the remaining settings of the algorithm the same. Figure 14 demonstrates that for Pong, there is no measurable benefit from introducing a sum over actions to the policy gradient estimate. In discrete domains like Pong, the learned critic is often inaccurate, which can be

a much greater problem than the variance of the policy estimator. Hence, even reducing the variance in the policy gradient estimator to zero, as EPG does, does not help performance.

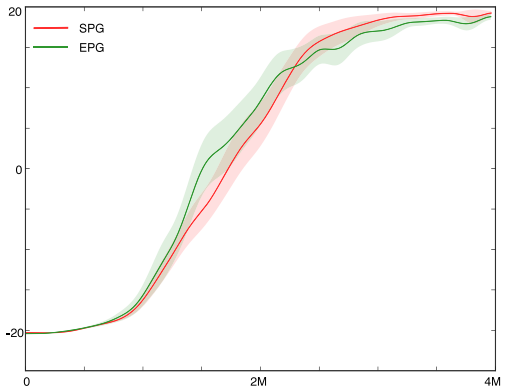


Figure 14: EPG vs SPG on a discrete action domain (Atari Pong). The curves show the mean and 90% confidence interval and were obtained by 24 runs of each algorithm. Horizontal scale is in millions of steps.

7. Related Work

In this section, we discuss the relationship between EPG and several other methods.

7.1. Sampling Methods for SPG

EPG has some similarities with VINE sampling (Schulman et al., 2015), which uses an (intrinsically noisy) Monte Carlo quadrature with many samples. However, there are important differences. First, VINE relies entirely on reward rollouts and does not use an explicit critic. This means that VINE has to perform many independent rollouts of $Q(\cdot, s)$ for each s, requiring a simulator with reset. A second, related difference is that VINE uses the *same* actions in the estimation of $I_\pi^{\hat{Q}}$ that it executes in the environment. While this is necessary with purely Monte Carlo rollouts, Section 5.4 shows that there is no such need in general if we have an explicit critic. Ultimately, the main weakness of VINE is that it is a purely Monte Carlo method. However, the example in Figure 1 (Section 3) shows that even with a computationally expensive many-sample Monte Carlo method, the problem of variance in the gradient estimator remains, regardless of the baseline.

EPG is also related to variance minimization techniques that interpolate between two estimators (Gu et al., 2016a). However, EPG uses a quadratic (not linear) critic, which is crucial for exploration. Furthermore, it completely eliminates variance in the inner integral, as opposed to just reducing it.

A more direct way of coping with variance in policy gradients is to simply reduce the learning rate when the variance of the gradient would otherwise explode, using, e.g., *Adam* (Kingma and Ba, 2015) or the adaptive step size method (Pirotta et al., 2013). However, this results in slow learning when the variance is high. Another way of reducing variance is

by defining new estimators. Parmas (2018) introduces a framework for doing that based on a graphical representation that also allows for including model-based approaches. This framework provides another way of generalizing the deterministic and the stochastic policy gradient theorem by looking them as different ways of estimating the same quantity. Unlike EPG, it does not perform analytic integration.

7.2. Sarsa and Q-Learning

It has been known since the introduction of policy gradient methods (Sutton et al., 2000a) that they represent a kind of slow-motion policy improvement as opposed to a greedy improvement performed by methods such as (expected) sarsa with action maximization or Q-learning. The two main reasons for the slow-motion improvement are that a greedy maximization operator may not be available (e.g., for continuous or large discrete action spaces) and that a greedy step may be too large because the critic only approximates the value function locally. The argument for a sarsa-like method is that it may converge faster and does not need an additional optimization for the actor. Recently, approaches combining the features of both methods have been investigated. One-step Newton's method for Q-functions that are quadratic in the actions has been used to produce a sarsa-like algorithm for continuous domains (Gu et al., 2016b), previously only tractable with policy gradient methods. For discrete action spaces, softmax Q-learning, a family of methods with a hybrid loss combining sarsa and Q-learning, has recently been linked to policy gradients via an entropy term (O'Donoghue et al., 2017). In this paper, GPG with Hessian-based exploration (Section 4.2) can be seen as another kind of hybrid. Specifically, it changes the mean of the policy slowly, similar to a vanilla policy gradient method, and computes the covariance greedily, similar to sarsa.

7.3. DPG

The update for the policy mean obtained in Corollary 7 is the same as the DPG update, linking the two methods

$$I_{\pi}^{Q}(s) = \left[\nabla_a Q^{\pi_\theta}(a,s)\right]_{a=\mu_s} \nabla_\theta \mu_s.$$

We now formalize the equivalences between EPG and DPG. First, any EPG method with a linear critic (or an arbitrary critic approximated by the first term in the Taylor expansion) is equivalent to DPG with actions from a given state s drawn from an exploration policy of the form

$$a \sim \pi(s) + n(a|s), \quad \text{where} \quad \mathbb{E}_{a\sim n}\left[a \,|\, s\right] = 0.$$

Here, the PDF of the zero-mean exploration noise $n(\cdot|s)$ must not depend on the policy parameters. This fact follows directly from Lemma 11, which says that, in essence, a linear critic only gives information on how to shift the mean of the policy and no information about other moments. Second, on-policy GPG with a quadratic critic (or an arbitrary critic approximated by the first two terms in the Taylor expansion) is equivalent to DPG with a Gaussian exploration policy where the covariance is computed as in Section 4.2. This follows from Corollary 7. Third, and most generally, for any critic at all (not necessarily quadratic), DPG is a kind of EPG for a particular choice of quadrature (using a Dirac measure). This follows from Theorem 1.

Surprisingly, this means that DPG, normally considered to be off-policy, can also be seen as on-policy when exploring with Gaussian noise defined as above for the quadratic critic or any noise for the linear critic. Furthermore, the compatible critic for DPG (Silver et al., 2014) is indeed linear in the actions. Hence, this relationship holds whenever DPG uses a compatible critic.[17] Furthermore, Lemma 5 lends new legitimacy to the common practice of replacing the critic required by the DPG theory, which approximates $\nabla_a Q$, with one that approximates Q itself, as done in SPG and EPG.

7.4. Entropy-Based Methods

On-policy SPG sometimes includes an entropy term (Peters et al., 2010) in the gradient in order to aid exploration by making the policy more stochastic. The gradient of the differential entropy $\mathcal{H}(s)$ of the policy at state s is defined as[18]

$$\begin{aligned}
-\nabla_\theta \mathcal{H}(s) &= \nabla_\theta \textstyle\int_a d\pi(a|s) \log \pi(a|s) \\
&= \textstyle\int_a da \nabla_\theta \pi(a|s) \log \pi(a|s) + \int_a d\pi(a|s) \nabla_\theta \log \pi(a|s) \\
&= \textstyle\int_a da \nabla_\theta \pi(a|s) \log \pi(a|s) + \int_a d\pi(a|s) \frac{1}{\pi(a|s)} \nabla_\theta \pi(a|s) \\
&= \textstyle\int_a da \nabla_\theta \pi(a|s) \log \pi(a|s) + \nabla_\theta \underbrace{\int_a d\pi(a|s)}_{1} \\
&= \textstyle\int_a da \nabla_\theta \pi(a|s) \log \pi(a|s) = \int_a d\pi(a|s) \nabla_\theta \log \pi(a|s) \log \pi(a|s).
\end{aligned}$$

Typically, we add the entropy update to the policy gradient update with a weight α, obtaining

$$\begin{aligned}
I_G^E(s) &= I_G(s) + \alpha \nabla_\theta \mathcal{H}(s) \\
&= \textstyle\int_a d\pi(a|s) \nabla_\theta \log \pi(a|s) (Q^{\pi_\theta}(a,s) - \alpha \log \pi(a|s)).
\end{aligned} \tag{5}$$

This equation makes clear that performing entropy regularization is equivalent to using a different critic with Q-values shifted by $\alpha \log \pi(a|s)$. This holds for both EPG and SPG, including SPG with discrete actions where the integral over actions is replaced with a sum. This follows because adding entropy regularization to the objective of optimizing the total discounted reward in an RL setting corresponds to shifting the reward function by a term proportional to $\log \pi(a|s)$ (Neu et al., 2017; Nachum et al., 2017). Indeed, the *path consistency learning algorithm* (Nachum et al., 2017) contains a formula similar to (5), though we obtained ours independently.

Next, we derive a further specialization of (5) for the case where the parameters θ are shared between the actor and the critic. We start with the policy gradient identity given by (2) and replace the true critic Q with the approximate critic $\hat{Q}$. Since this holds for any stochastic policy, we choose one of the form

$$\pi(a|s) = \frac{1}{Z(s)} e^{\hat{Q}(a,s)}, \quad \text{where} \quad Z(s) = \int_a e^{\hat{Q}(a,s)} da.$$

For the continuous case, we assume that the integral in (7.4) converges for each state. Here, we assume that the approximate critic is parameterized by θ. Because of the form of (7.4),

17. The notion of compatibility of a critic is different for stochastic and deterministic policy gradients.
18. For discrete action spaces, the same derivation with integrals replaced by sums holds for the entropy.

the policy is parameterized by θ as well. Now, for the policy class given by (7.4), we can simplify the gradient update even further, obtaining

$$\begin{aligned} I_G^E(s) &= \textstyle\int_a d\pi(a|s)\nabla_\theta \log \pi(a|s)(\hat{Q}(a,s) - \alpha \log \pi(a|s)) \\ &= \textstyle\int_a \pi(a|s)\nabla_\theta \log \pi(a|s)(\hat{Q}(a,s) - \alpha \underbrace{\log e^{\hat{Q}(a,s)}}_{\hat{Q}(a,s)} - \alpha \log Z(s)) \\ &= (1-\alpha)\textstyle\int_a \pi(a|s)\nabla_\theta \log \pi(a|s)\hat{Q}(a,s) \\ &= -(1-\alpha)\nabla_\theta \mathcal{H}(s). \end{aligned}$$

In the above derivation, we could drop the term $\log Z(s)$ since it does not depend on a, as with a baseline. This shows that, in the case of sharing parameters between the critic and the policy as above, methods such as A3C (Mnih et al., 2016), which have both an entropy loss and a policy gradient loss, are redundant since entropy regularization does nothing except scale the learning rate.[19] Alternatively, for this shared parameterization, a policy gradient method simply subtracts entropy from the policy. In practice, this means that a policy gradient method with this kind of parameter sharing is quite similar to learning the critic alone and simply acting according to the argmax of the Q values rather than representing the policy explicitly, producing a method similar to sarsa.

Another family of entropy-based methods are soft actor-critic methods (Haarnoja et al., 2018). They combine a policy learned with an entropy bonus similar to (5), but with $\alpha = 1$ and a different critic. In particular, for stochastic policies the gradient of the soft actor loss (Haarnoja et al., 2018, Equation 12), can be written as

$$\begin{aligned} I_G^{\text{soft}} &= \textstyle\int_a d\pi(a|s)\nabla_\theta \log \pi(a|s)(\hat{Q}^{\text{soft}}(a,s) - \log \pi(a|s)) = \\ &= \nabla_\theta \left[\int_a d\pi(a|s) \log \pi(a|s)\hat{Q}^{\text{soft}}(a,s) - \mathcal{H}(s) \right]. \end{aligned}$$

Here, $\hat{Q}^{\text{soft}}$ is a separate critic, learned in a way analogous to the regular critic $\hat{Q}$, but based off a reward function $R(a,s)^{\text{soft}} = R(a,s) + \gamma\mathcal{H}(s)$, where the term $\gamma\mathcal{H}(s)$ is an intrinsic entropy bonus. The formula above assumes that $\hat{Q}^{\text{soft}}$ is a fixed learned approximation, i.e., it does not depend on the policy parameters. The rationale for using such a critic is largely orthogonal to this paper and given by Haarnoja et al. (2018). Transforming the integral above by a change of variables, we obtain the formula

$$I_G^{\text{soft}} = \textstyle\int_\epsilon dN(\epsilon)\nabla_\theta \log \pi(a|s) + (\nabla_a \log \pi(a|s) - \nabla_a Q^{\text{soft}}(a,s))\nabla_\theta f(\epsilon, s).$$

Here, the reparameterization function f is chosen such that the pdf of π matches the pdf of $f(\epsilon)$ and $\epsilon \sim N$ is sampled from the standard normal. Soft actor-critic uses a one-sample Monte Carlo estimate of the integral above, which can be written as

$$I_G^{\text{soft}} \approx \nabla_\theta \log \pi(a|s) + (\nabla_a \log \pi(a|s) - \nabla_a Q^{\text{soft}}(a,s))\nabla_\theta f(\epsilon, s),$$

where $\epsilon \sim N(\epsilon)$ and $a = f(\epsilon)$.

19. In this argument, we ignore the effects of sampling on exploration.

7.5. Off-Policy Actor-Critic

Off-policy learning with policy gradients typically follows the framework of *off-policy actor-critic* (Degris et al., 2012). Denote the behavior policy as $b(a \mid s)$ and the corresponding discounted-ergodic measure as ρ_b. The method uses the reweighing approximation

$$\begin{aligned}\nabla_\theta J &= \int_s d\rho(s) \int_a d\pi(a \mid s) \nabla_\theta \log \pi(a \mid s) (Q^{\pi_\theta}(a, s)) \\ &\approx \int_s d\rho_b(s) \int_a d\pi(a \mid s) \nabla_\theta \log \pi(a \mid s) (Q^{\pi_\theta}(a, s)). \end{aligned} \tag{6}$$

The approximation is necessary since, as the samples are generated using the policy b, it is not known how to approximate the integral with ρ from samples, while it is easy to do so for an integral with ρ_b. A natural off-policy version of EPG emerges from this approximation (see Algorithm 5), which simply replaces the inner integral with I^Q_π, giving

$$\int_s d\rho_b(s) \int_a d\pi(a \mid s) \nabla_\theta \log \pi(a \mid s) (Q^{\pi_\theta}(a, s)) = \int_s d\rho_b(s) I^Q_\pi(s).$$

Here, we use an analytic solution to $I^Q_\pi(s)$ as before. The importance sampling term $\frac{\pi(a|s)}{b(a|s)}$ does not appear because, as the integral is computed analytically, there is no sampling in $I^Q_\pi(s)$, much less sampling with an importance correction. Of course, the algorithm also requires an off-policy critic for which an importance sampling correction is typically necessary. Indeed, (7.5) makes clear that off-policy actor-critic differs from SPG in two places: the use of ρ_b as in (6) and the use of an importance-sampled Monte Carlo estimator, rather than regular Monte Carlo, for the inner integral.

Algorithm 5 Off-policy expected policy gradients with reweighing approximation

$s \leftarrow s_0$, $t \leftarrow 0$
initialize optimizer, initialize policy π parameterized by θ
while not converged **do**
    $g_t \leftarrow \gamma^t$ DO-INTEGRAL$(\hat{Q}, s, \pi_\theta)$ ▷ g_t is the estimated policy gradient as per (3)
    $\theta \leftarrow \theta +$ optimizer.UPDATE(g_t)
    $a \sim b(\cdot, s)$
    $s', r \leftarrow$ simulator.PERFORM-ACTION(a)
    $\hat{Q}$.UPDATE(s, a, r, s', π, b) ▷ Off-policy critic algorithm
    $t \leftarrow t + 1$
    $s \leftarrow s'$
end while

7.6. Value Gradient Methods

Value gradient methods (Fairbank, 2014; Fairbank and Alonso, 2012; Heess et al., 2015) assume the same parametrization of the policy as policy gradients, i.e., π is parameterized by θ, and maximize J by recursively computing the gradient of the value function. In our notation, the policy gradient has the following connection with the value gradient of the

initial state

$$\nabla_\theta J = \int_{s_0} dp_0(s_0) \nabla_\theta V^{\pi_\theta}(s)_0.$$

Value gradient methods use a recursive equation that computes $\nabla_\theta V^{\pi_\theta}(s)$ using $\nabla_\theta V^{\pi_\theta}(s)'$ where s' is the successor state. In practice, this means that a trajectory is truncated and the computation goes backward from the last state all the way to s_0, where (7.6) is applied, so that the resulting estimate of $\nabla_\theta J$ can be used to update the policy. The recursive formulae for $\nabla_\theta V^{\pi_\theta}(s)$ are based on the differentiated Bellman equation

$$\nabla_\theta V = \nabla_\theta \int_a d\pi(a|s) \left(R(a,s) + \gamma \int_{s'} p(s'|a,s) V^{\pi_\theta}(s)' \right).$$

Different value gradient methods differ in the form of the recursive update for the value gradient obtained from (7.6). For example, *stochastic value gradients* (SVG) introduce a reparameterization both of π and $p(s'|a,s)$ i.e.

$$\begin{aligned} s' \sim p(\cdot|a,s) \quad &\Leftrightarrow \quad s' = f(a,s,\xi) \text{ with } \xi \sim \mathcal{B}_1, \\ a \sim \pi(\cdot|s) \quad &\Leftrightarrow \quad a = h(s,\eta) \text{ with } \eta \sim \mathcal{B}_2. \end{aligned}$$

Here, we denote the base noise distributions as $\mathcal{B}_1$ and $\mathcal{B}_2$, while f and h are deterministic functions. The function f can be thought of as an MDP transition model. SVG rewrites (7.6) using this reparametrization, to obtain

$$\begin{aligned} \nabla_\theta V &= \nabla_\theta \textstyle\int_\eta d\mathcal{B}_2(\eta) \left(R(s,h(s,\eta)) + \gamma \int_{s'} d\mathcal{B}_1(\xi) V(f(h(s,\eta),s,\xi)) \right) = \\ &= \textstyle\int_\eta d\mathcal{B}_2(\eta) \left(\nabla_\theta R(s,h(s,\eta)) + \gamma \int_\xi d\mathcal{B}_1(\xi) \nabla_\theta V(\underbrace{f(h(s,\eta),s,\xi)}_{s'}) \right). \end{aligned} \tag{7}$$

Here, the quantities $\nabla_\theta R(s,h(s,\eta))$ and $\nabla_\theta V(f(h(s,\eta),s,\xi))$ can be computed by the chain rule from the known reward model R and a transition model f. SVG learns the approximate model $\hat{f}, \hat{R}, \hat{\xi}, \hat{\eta}$ from samples uses a sample-based approximation to (7) to obtain the value gradient recursion.

By contrast, we now derive a related but simpler value gradient method that does not require a model or a reparameterized policy,[20] starting with (7.6). The value gradient is given by

$$\begin{aligned} \nabla_\theta V^{\pi_\theta}(s) &= \nabla_\theta \textstyle\int_a d\pi(a|s) \left(R(a,s) + \gamma \int_{s'} p(s'|a,s) \right) \\ &= \textstyle\int_a da \nabla_\theta \pi(a|s) R(a,s) + \gamma \nabla_\theta \pi(a|s) \left(\int_{s'} p(s'|a,s) V^{\pi_\theta}(s)' \right) && (8) \\ &\qquad + \pi(a|s) \nabla_\theta \left(\textstyle\int_{s'} p(s'|a,s) V^{\pi_\theta}(s)' \right) \\ &= \textstyle\int_{a,s'} \pi(a|s) p(s'|a,s) \big(\nabla_\theta \log \pi(a|s) R(a,s) && (9) \\ &\qquad + \nabla_\theta V^{\pi_\theta}(s)' + \nabla_\theta \log \pi(a|s) V^{\pi_\theta}(s)' \big). && (10) \end{aligned}$$

20. SVG(∞) and SVG(1) require a model and a policy reparameterization while SVG(0) requires only a policy reparameterization. In fact, SVG(0) can be thought of as a direct analog of DPG or reparametrized gradient methods in the value gradient setting.

We can use random sampling to approximate (10)

$$\hat{\nabla}_\theta V^{\pi_\theta}(s) \approx \left(\nabla_\theta \log \pi(a|s) R(a,s) + \hat{\nabla}_\theta V^{\pi_\theta}(s)') + \nabla_\theta \log \pi(a|s) \hat{V}(s')\right).$$

Here, the pair a corresponds to the action taken at s and s' to the successor state. This method requires learning a critic, while SVG requires a model.

An additional connection between value gradient methods and policy gradients is that, since the quantity $I_G(s)$ in Theorem 1 can be written as $I_G(s) = \nabla_\theta V^{\pi_\theta}(s) - \gamma \int_{s'} dp_\pi(s' \mid s) \nabla_\theta V^{\pi_\theta}(s)')$, we can think of this theorem as showing how to obtain a policy gradient from a value gradient without backwards iteration.

7.7. Methods using Sums or Integrals over Actions

Several methods that involve summations over actions pre-date EPG. For discrete actions, it was first explicitly introduced in an unrefereed draft by Sutton et al. (2000b) and independently developed by Bahdanau et al. (2016). The same idea was also independently developed as Mean Actor Critic (Asadi et al., 2017), concurrently with EPG. Our work differs from these other efforts in that we treat both continuous and discrete action spaces and analyze the algorithm in a rigorous theoretical framework. We believe this contribution is significant because, experimentally, the performance benefit of EPG occurs for systems with continuous action spaces.

For continuous action spaces, the idea of numerical integration over actions is first mentioned by Kakade (2002), where a continuous Linear-Quadratic-Gaussian (LQC) control with positive-definite quadratic critic is solved. However, this method is computationally inefficient.[21] Peters et al. (2003) suggested extending (Sutton et al., 2000b) to continuous action spaces for linear critics. They apply quadrature to pre-compute a matrix, which can then be used to obtain the policy gradient by matrix-vector multiplication. However, their method crucially depends on using a linear critic. The idea of performing integration over actions is further explored in Interpolated Policy Gradients (Gu et al., 2017). The IPG interpolation scheme between deterministic and stochastic gradients can be viewed as a specific type of Monte-Carlo quadrature that could be used in EPG. In addition, both the concept of the Normalized Advantage Function control variate (Gu et al., 2017, Appendix, Section 11.3) and the quadratic assumption in MORE/MOTO (Abdolmaleki et al., 2015; Akrour et al., 2016) can be viewed as the special case of EPG with quadratic critics where the Hessian is positive/negative definite.[22] Bayesian Actor-Critic (Ghavamzadeh et al., 2016) also employs a similar idea in the context of learning a critic as a Gaussian Process, where the kernel is chosen such that the mean estimate of the actor GP roughly corresponds to choosing a specific subset of quadratic critics with positive-definite Hessians. The advantage of EPG is that it works with *any* Hessian, and we derive an explicit formula for the exploration covariance, for which indefinite Hessians are crucial for good performance. EPG also allows families of critics more general than quadratic functions.

21. Kakade (2002) uses a numerical quadrature ran to convergence to evaluate the policy gradient exactly. It does not describe the exact type of quadrature used.
22. MORE (Abdolmaleki et al., 2015) does cover some cases of indefinite Hessians, but at the cost of heuristically setting the value of a Lagrange multiplier, which may lead to constraint violations.

After EPG was originally published, Nachum et al. (2018a) introduced a similar approach to policy gradients, called the smoothed action-value function. Since the authors present it as novel, we provide an explicit comparison here. First, the concept of the smoothed action function is, up to differentiation, essentially the same as our definition of the inner integral I_G; in particular, we have the identity

$$I_\pi^Q(s) = \nabla_\theta \tilde{Q}^\pi(s, \mu).$$

Here, the expression to the right is the "smoothed action value" (Nachum et al., 2018a, Eq. 9), while the term $I_\pi^Q(s)$ is defined in (2) of this article as well as in (Ciosek and Whiteson, 2018, Equation 9). Nachum et al. (2018a, Section 5) claim that they can handle any critic while avoiding quadrature, as opposed to EPG which requires either numerical quadrature or a tractable analytical form of the critic. Nachum et al. (2018a, Section 5) describe the process of what they call "estimating an integral" as different from approximating an integral, which is what EPG with numerical quadrature does. However, the method they introduce for computing $\tilde{Q}$ given by Nachum et al. (2018a, Equation 19) in fact is a kind of numerical quadrature. Indeed, it requires computing averages over function evaluations at a number of what they call "phantom actions", i.e., actions used only to compute the value of the smoothed action-value function. These phantom actions are normally called *abscissae* in the numerical integration literature. Therefore, the only substantial difference between smoothed actor-critic (Nachum et al., 2018a) and our original work on EPG is the use of a different kind of quadrature and the fact that performing quadrature is integrated into the process of learning the critic – Nachum et al. (2018a) learn the *integrated critic* directly, rather than learning a regular critic and then integrating over it.

7.8. Methods Using the Geometry of the Policy Space

Policy gradient methods can be improved by adjusting the policy update in a way that respects distances in the space of probability distributions (Kakade, 2002; Amari, 1998; Peters and Schaal, 2008a; Furmston and Barber, 2012; Furmston et al., 2016; Schulman et al., 2015; Nachum et al., 2018b), leading to a *natural gradient* method. Trust Region Policy Optimization (Schulman et al., 2015), a representative recent method in this family, is based on an optimization problems of either the form

$$\theta^\star = \underset{\{\theta:\, \mathrm{KL}(\pi_\theta, \pi_{\mathrm{old}}) < \delta\}}{\arg\max} J_\theta, \quad \text{or}$$
$$\theta^\star = \arg\max_\theta \left(J_\theta - \mathrm{KL}\left(\pi_\theta, \pi_{\mathrm{old}}\right)\right).$$

Here, the total discounted return J is defined as in (2) and π_{old} is the policy from the previous time-step. (7.8) gives the version of TRPO as implemented, while (7.8) gives a version that comes form the theoretical analysis (Schulman et al., 2015). TRPO relies on Monte-Carlo approximation to the policy gradient integral in the gradient of J, while performing analytic integration in the KL term. In principle, one can combine EPG with TRPO by computing *both* integrals analytically. This is done by Model-Free Trajectory Optimization (MOTO) (Akrour et al., 2016), albeit only for critics that are both quadratic and negative-definite. Another example is the recent work of Abdolmaleki et al. (2018),

Policy Class	**Squashing**	$\hat{Q}$	**Analytic Update**
Normal, $a \in \mathbb{R}^d$	none	$a^\top A_s a + a^\top B_s$	$I^Q_{\pi(s),\mu_s} = (\nabla_\theta \mu_s)(2A_s\mu_s + B_s)$,
Logit-Normal; $a \in [0,1]^d$	$a = \text{expit}(b)$	$b^\top A_s b + b^\top B_s$	$I^Q_{\pi(s),\Sigma_s^{1/2}} = (\nabla_\theta \Sigma_s^{1/2}) A_s \Sigma_s^{1/2}$
Log-Normal; $a \in [0,\infty]^d$	$a = e^b$	$b^\top A_s b + b^\top B_s$	
any policy	none	$B_s^\top a$	$I^Q_\pi(s) = \nabla_\theta \mu_{\pi(\cdot\|s)} B_s$

Table 1: A summary of the most useful analytic results for expected policy gradients. For bounded action spaces, we assume that the bounding interval is $[0,1]$ or $[0,\infty]$.

where the KL constraint is introduced using the RL-as-inference framework. Because the idea of using a KL constraint is orthogonal to the main thrust of this article, we leave the study of such hybrid algorithms to future work.

In Section 6.4, we instead compare EPG with PPO (Schulman et al., 2017), an existing established approximation to natural gradients with good empirical performance.

8. Conclusions

This paper proposed a new framework for reasoning about policy gradient methods called *expected policy gradients* (EPG) that integrates across the action selected by the stochastic policy, thus reducing variance compared to existing stochastic policy gradient methods. We proved a new general policy gradient theorem subsuming the stochastic and deterministic policy gradient theorems, which covers any reasonable class of policies. We showed that analytical results for the policy update exist and, in the most common cases, lead to a practical algorithm (the analytic updates are summarized in Table 1). We also gave universality results which state that, under certain broad conditions, the quadrature required by EPG can be performed analytically. For Gaussian policies, we also developed a novel approach to exploration that infers the exploration covariance from the Hessian of the critic. The analysis of EPG yielded new insights about DPG and delineated the links between the two methods. We have also described a version of EPG that works with discrete policies. Finally, we discussed the connections between EPG and other common RL techniques, notably sarsa, Q-learning, entropy regularization and methods taking account of the geometry of the policy space. We performed an extensive empirical evaluation of versions of EPG based on analytic and numerical quadrature.

Acknowledgments

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement number 637713). Experiments performed at Oxford were made possible by a generous equipment grant from NVIDIA. Moreover, we appreciate the comments from the JMLR reviewers, which led to improvements in our original submission. We also thank Paavo Parmas and Rika Antonova for helpful feedback as well as Kaiqing Zhang and Zac Chen for pointing out typos in the original draft.

Appendix A.

A.1. Proofs and Detailed Definitions

First, we prove two lemmas concerning the discounted-ergodic measure $\rho(s)$, of the Markov chain induced by fixing a policy in an MDP. They have been implicitly realized for some time but as far as we could find, never proved explicitly in RL literature.

Definition 13 (Time-dependent occupancy) *The time-dependent state occupancy is defined as*

$$p(s \mid t=0) = p_0(s) \;\; \textit{and}$$
$$p(s' \mid t=i+1) = \int_s p(s' \mid s) p(s \mid t=i) \quad \textit{for} \quad i \geq 0.$$

Definition 14 (Truncated trajectory) *We define a trajectory truncated after N steps as $\tau_N = (s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_N)$.*

Observation 15 (Expectation with respect to truncated trajectory) *Since $\tau_N = (s_0, s_1, s_2, \ldots, s_N)$ is associated with the density $\prod_{i=0}^{N-1} p(s_{i+1} \mid s_i) p_0(s_0)$, we have*

$$\begin{aligned}
&\mathbb{E}_{\tau_N}\left[\textstyle\sum_{i=0}^{N} \gamma^i f(s_i)\right] = \\
&= \textstyle\int_{s_0,s_1,\ldots,s_N} \left(\prod_{i=0}^{N-1} p(s_{i+1} \mid s_i)\right) p_0(s_0) \left(\sum_{i=0}^{N} \gamma^i f(s_i)\right) ds_0 ds_1 \ldots ds_N = \\
&= \textstyle\sum_{i=0}^{N} \int_{s_0,s_1,\ldots,s_N} \left(p_0(s_0) \prod_{i=0}^{N-1} p(s_{i+1} \mid s_i)\right) \gamma^i f(s_i) ds_0 ds_1 \ldots ds_N = \\
&= \textstyle\sum_{i=0}^{N} \int_s p(s \mid t=i) \gamma^i f(s) ds
\end{aligned}$$

for any function f.

Definition 16 (Expectation with respect to infinite trajectory) *For any bounded function f, we have*

$$\mathbb{E}_{\tau}\left[\sum_{i=0}^{\infty} \gamma^i f(s_i)\right] \triangleq \lim_{N\to\infty} \mathbb{E}_{\tau_N}\left[\sum_{i=0}^{N} \gamma^i f(s_i)\right].$$

Here, the sum on the left-hand side is part of the symbol being defined.

Observation 17 (Property of expectation with respect to infinite trajectory) *We have*

$$\begin{aligned}
\mathbb{E}_{\tau}\left[\textstyle\sum_{i=0}^{\infty} \gamma^i f(s_i)\right] &= \lim_{N\to\infty} \mathbb{E}_{\tau_N}\left[\textstyle\sum_{i=0}^{N} \gamma^i f(s_i)\right] = \\
&= \lim_{N\to\infty} \textstyle\sum_{i=0}^{N} \int_s p(s \mid t=i) \gamma^i f(s) ds = \\
&= \sum_{i=0}^{\infty} \int_s dp(s \mid t=i) \gamma^i f(s)
\end{aligned}$$

for any bounded function f.

Definition 18 (Discounted-ergodic occupancy measure ρ) *We define the discounted-ergodic occupancy measure as*

$$\rho(s) = \sum_{i=0}^{\infty} \gamma^i p(s \mid t = i).$$

The measure ρ is not normalized in general. Intuitively, it can be thought of as 'marginalizing out' the time in the system dynamics.

Lemma 19 (Discounted-ergodic property) *For any bounded function f, we have*

$$\int_s \rho(s) f(s) = \mathbb{E}_\tau \left[\sum_{i=0}^{\infty} \gamma^i f(s_i) \right].$$

Proof We re-write the expectation as

$$\mathbb{E}_\tau \left[\sum_{i=0}^{\infty} \gamma^i f(s_i) \right] = \sum_{i=0}^{\infty} \gamma^i \int_s p(s \mid t = i) f(s) ds = \int_s \underbrace{\left[\sum_{i=0}^{\infty} \gamma^i p(s \mid t = i) \right]}_{\rho(s)} f(s) ds.$$

Here, the first equality follows from Observation 17. ■

This property is useful since the expression on the left can be easily manipulated while the expression on the right can be estimated from samples using Monte Carlo.

Lemma 20 (Generalized eigenfunction property) *For any bounded function f, we have*

$$\gamma \int_s d\rho(s) \int_{s'} dp(s' \mid s) f(s') = \left(\int_s d\rho(s) f(s) \right) - \left(\int_s dp_0(s) f(s) \right).$$

Proof We rewrite the expression in the left as

$$\begin{aligned}
\gamma \textstyle\int_s d\rho(s) \int_{s'} dp(s' \mid s) f(s') &= \gamma \textstyle\sum_{i=0}^{\infty} \gamma^i \int_{s,s'} p(s \mid t = i) p(s' \mid s) f(s') ds ds' = \\
&= \textstyle\sum_{i=0}^{\infty} \gamma^{i+1} \int_{s'} dp(s' \mid t = i+1) f(s') \\
&= \textstyle\sum_{i=1}^{\infty} \gamma^i \int_{s'} dp(s' \mid t = i) f(s') \\
&= \left(\textstyle\sum_{i=0}^{\infty} \gamma^i \int_{s'} dp(s' \mid t = i) f(s') \right) - \left(\int_s dp_0(s) f(s) \right) \\
&= \left(\textstyle\int_s d\rho(s) f(s) \right) - \left(\int_s dp_0(s) f(s) \right).
\end{aligned}$$

Here, the first equality follows form definition 18, the second one from definition 13. The last equality follows again from definition 18. ■

Definition 21 (Markov Reward Process)
A Markov Reward Process is a tuple (p, p_0, R, γ), where $p(s'|s)$ is a transition kernel, p_0 is the distribution over initial states, $R(\cdot|s)$ is a reward distribution conditioned on the state and γ is the discount constant.

An MRP can be thought of as an MDP with a fixed policy and dynamics given by marginalizing out the actions $p_\pi(s' \mid s) = \int_a d\pi(a \mid s) p(s' \mid a, s)$. Since this paper considers the case of one policy, we abuse notation slightly by using the same symbol τ to denote trajectories including actions, i.e. $(s_0, a_0, r_0, s_1, a_1, r_1, \dots)$ and without them $(s_0, r_0, s_1, r_1, \dots)$.

Lemma 22 (Second Moment Bellman Equation) *Consider a Markov Reward Process (p, p_0, X, γ) where $p(s' \mid s)$ is a Markov process and $X(\cdot \mid s)$ is some probability density function.*[23] *Denote the value function of the MRP as V. Denote the second moment function S as*

$$S(s) = \mathbb{E}_\tau \left[\left(\sum_{t=0}^{\infty} \gamma^t x_t \right)^2 \middle| s_0 = s \right], \quad \text{where} \quad x_t \sim X(\cdot \mid s_t).$$

Then S is the value function of the MRP: (p, p_0, u, γ^2), where $u(s)$ is a deterministic random variable given by

$$u(s) = \mathbb{V}_{X(x|s)}[x] + \left(\mathbb{E}_{X(x|s)}[x] \right)^2 + 2\gamma \mathbb{E}_{X(x|s)}[x] \, \mathbb{E}_{p(s'|s)} \left[V^{\pi_\theta}(s)' \right].$$

Proof We rewrite $S(s)$ as

$$\begin{aligned} S(s) &= \mathbb{E}_\tau \left[\left(x_0 + \textstyle\sum_{t=1}^{\infty} \gamma^t x_t \right)^2 \middle| s_0 = s \right] \\ &= \mathbb{E}_\tau \left[x_0^2 + 2x_0 \left(\textstyle\sum_{t=1}^{\infty} \gamma^t x_t \right) + \left(\textstyle\sum_{t=1}^{\infty} \gamma^t x_t \right)^2 \middle| s_0 = s \right] \\ &= \underbrace{\mathbb{E}_\tau \left[x_0^2 \middle| s_0 = s \right] + \mathbb{E}_\tau \left[2x_0 \left(\textstyle\sum_{t=1}^{\infty} \gamma^t x_t \right) \middle| s_0 = s \right]}_{u(s)} + \underbrace{\mathbb{E}_\tau \left[\left(\textstyle\sum_{t=1}^{\infty} \gamma^t x_t \right)^2 \middle| s_0 = s \right]}_{\gamma^2 \mathbb{E}_{p(s'|s)}[S(s')]}. \end{aligned}$$

This is exactly the Bellman equation of the MRP (p, p_0, u, γ^2). The theorem follows since the Bellman equation uniquely determines the value function. ■

Observation 23 (Dominated Value Functions)
Consider two Markov Reward Processes (p, p_0, X_1, γ) and (p, p_0, X_2, γ), where $p(s' \mid s)$ is a Markov process (common to both MRPs) and $X_1(s)$, $X_2(s)$ are some deterministic random variables meeting the condition $X_1(s) \leq X_2(s)$ for every s. Then the value functions V_1 and V_2 of the respective MRPs satisfy $V_1(s) \leq V_2(s)$ for every s. Moreover, if we have that $X_1(s) < X_2(s)$ for all states, then the inequality between value functions is strict.

Proof Follows trivially by expanding the value function as a series and comparing series elementwise. ■

23. Note that while X occupies a place in the definition of the MRP usually called 'reward distribution', we are using the symbol X, not R since we shall apply the lemma to Xes which are constructions distinct from the reward of the MDP we are solving.

A.2. Proofs about Exponential Families

Lemma 8 (EPG for Exponential Families w. Polynomial Sufficient Statistics) *Consider the class of policies parameterized by θ defined by the formula*

$$\pi(a \mid s) = e^{(\eta_\theta^s)^\top T^s(a) - U^s_{\eta_\theta^s} + W^s(a)},$$

where each entry in the vector $T^s(a)$ is a (possibly multivariate) polynomial in the entries of the vector a. Moreover, assume that the critic $\hat{Q}(a, s)$ is (a possibly multivariate) polynomial in the entries of a. Then, the policy gradient update is a closed form expression in terms of the uncentered moments of $\pi(\cdot \mid s)$ and can be written as

$$I_\pi^Q(s) = (\nabla_\theta(\eta_\theta^s)^\top)((C^s_{TQ})^\top m_\pi) - (\nabla_\theta U^s_{\eta_\theta^s})((C^s_Q)^\top m_\pi),$$

where C^s_Q is the vector containing the coefficients of the polynomial $\hat{Q}(\cdot, s)$, C^s_{TQ} is the vector containing the coefficients of the polynomial $T^s(a)\hat{Q}(a, s)$, i.e., a multiplication of T^s and $\hat{Q}(a, s)$. Moreover, m_π is a vector of uncentered moments of π (in the order matching the polynomials).

Proof We first rewrite the inner integral as an expectation, obtaining

$$\begin{aligned}
I_\pi^{\hat{Q}}(s) &= \int_A d\pi(a \mid s) \nabla_\theta \log \pi(a \mid s) \hat{Q}(a, s) \\
&= \mathbb{E}_{a\sim\pi}\left[\nabla_\theta \log \pi(a \mid s)\hat{Q}(a, s)\right] \\
&= \mathbb{E}_{a\sim\pi}\left[(\nabla_\theta(\eta_\theta^\top T^s(a) - U^s_{\eta_\theta} + W^s(a)))\hat{Q}(a, s)\right] \\
&= \mathbb{E}_{a\sim\pi}\left[(\nabla_\theta \eta_\theta^\top) T^s(a)\hat{Q}(a, s) - (\nabla_\theta U^s_{\eta_\theta})\hat{Q}(a, s)\right] \\
&= (\nabla_\theta \eta_\theta^\top)\mathbb{E}_{a\sim\pi}\left[T^s(a)\hat{Q}(a, s)\right] - (\nabla_\theta U^s_{\eta_\theta})\mathbb{E}_{a\sim\pi}\left[\hat{Q}(a, s)\right].
\end{aligned}$$

Since $T^s(a)$ and $\hat{Q}(a, s)$ are polynomials, and the multiplication of polynomials is still polynomial, both expectations are expectations of polynomials. To compute the second expectation, we exploit the fact that, since $\hat{Q}(a, s)$ is a polynomial on a, it equals a sum of monomial terms

$$\mathbb{E}_{a\sim\pi}\left[\hat{Q}(a, s)\right] = \mathbb{E}_{a\sim\pi}\left[\sum_{i=1}^{D} c_i \prod_{j=1}^{d} a_j^{p_i(j)}\right] = \sum_{i=1}^{D} c_i \underbrace{\mathbb{E}_{a\sim\pi}\left[\prod_{j=1}^{d} a_j^{p_i(j)}\right]}_{\text{cross-moment of } \pi}.$$

On the right, the terms $\mathbb{E}_{a\sim\pi}\left[\prod_{j=1}^{d} a_j^{p_i(j)}\right]$ (we do not explicitly denote the dependence on s for clarity), are the uncentered $(p_i(1), p_i(2), \ldots, p_i(d))$-cross-moments of π. If we arrange the coefficients c_i into the vector C^s_Q and the cross-moments into the vector m^s_π, we obtain the right term in (8). We can apply the same reasoning to the product of T^s and $\hat{Q}(\cdot, s)$ to obtain the left term. ■

Lemma 9 *Consider an invertible and differentiable function g. Define a policy π as in (5.2). Assume that the Jacobian of g is non-singular except on a set of π_b-measure zero. Consider a critic $\hat{Q}$. Denote as $\hat{Q}_b$ a reparameterized critic such that for all a, $\hat{Q}_b(g^{-1}(a), s) = \hat{Q}(a, s)$. Then the policy gradient update is given by the formula $I_\pi^{\hat{Q}}(s) = I_{\pi_b}^{\hat{Q}_b}(s)$.*

Proof We rewrite the policy gradient update

$$\begin{aligned}
I_\pi^Q(s) &= \int_A d\pi(a \mid s) \nabla_\theta \log \pi(a \mid s) \hat{Q}(a, s) \\
&= \int_{\mathbb{R}^d} d\pi(g(b) \mid s) \nabla_\theta \log \pi(g(b) \mid s) \hat{Q}(g(b), s) \det \mathbf{Jb}^\theta g(b) \\
&= \int_{\mathbb{R}^d} d\pi_b(b \mid s) \nabla_\theta \log \pi(g(b) \mid s) \hat{Q}_b(b, s) \\
&= \int_{\mathbb{R}^d} d\pi_b(b \mid s) (\nabla_\theta \log \pi_b(b \mid s) - \underbrace{\nabla_\theta \log \det \mathbf{Jb}^\theta g(b)}_{0}) \hat{Q}_b(b, s) = I_{\pi_b}^{\hat{Q}_b}(s).
\end{aligned}$$

In the second equality, we perform the variable substitution $a = g(b)$. In the third equality we use (5.2) and the fact that $\hat{Q}_b(g^{-1}(a), s) = \hat{Q}(a, s)$. In the fourth equality we again use (5.2) and the fact that $\log \det \mathbf{Jb}^\theta g(b) = 0$ since g is not parameterized by θ. ■

A.3. Computation of Moments for an Exponential Family

Consider the moment generating function of $T^s(a)$, which we denote as M_T, for the exponential family of the form given in equation (8), that is

$$M_T(v) = e^{U^s_{v+\eta^s_\theta} - U^s_{\eta^s_\theta}}.$$

It is well-known that M_T is finite in a neighborhood of the origin (Bickel and Doksum, 2006), and hence the cross moments can be obtained as

$$\mathbb{E}_{a\sim\pi}\left[\prod_{j=1}^{K} T^s(a)_j^{p(j)}\right] = \frac{\partial}{\partial^{p(1)} v_1, \partial^{p(2)} v_2, \ldots, \partial^{p(K)} v_K} M_T(v)\Big|_{v=0}.$$

Here, we denoted as K the size of the sufficient statistic (i.e. the length of the vector $T^s(a)$). However, we seek the cross-moments of a, not $T^s(a)$. If $T^s(a)$ contains a subset of indices which correspond to the vector a, then we can simply use the corresponding indices in the above equation. On the other hand, if this is not the case, we can introduce an extended distribution $\pi'(a \mid s) = e^{(\eta)'^\top T'(a) - U^s_{\eta^s_\theta} + W^s(a)}$, where T' is a vector concatenation of T^s and a. We can then use the MGF of $T'(a)$, restricted to a suitable set of indices to get the moments.

A.4. Experimental Details

The exploration hyperparameters for EPG were $\sigma_0 = 0.5$ and $c = 1.0$ where the exploration covariance is $\sigma_0 e^{cH}$. These values were obtained using a grid search from the set $\{0.2, 0.5, 1\}$

Shared parameters	
Target network update constant τ	0.01
Size of replay buffer	1000000
Method of sampling from buffer	uniform
Ignored steps at beginning of training	10000
Reward scale for InvertedPendulum-v2 and InvertedDoublePendulum-v2	0.1
Reward scale for other tasks	1 (no scaling)
Optimiser used for both actor and critic	Adam
Learning rate	1e-3
Batch size	64
Structure of critic network	hidden layers of 100, 100 neurons respectively, ReLU nonlinearities
Structure of actor network	hidden layers of 100, 50, 25 neurons respectively, ReLU nonlinearities
Target network update constant τ	0.01
Experiment-specific parameters	
DPG	$\sigma = 0.2, \psi = 0.15$
EPG	$\sigma_0 = 0.5$
SPG	$\sigma = 0.2$, fixed
NQ(m)-EPG	$\sigma = 0.2, \psi = 0.15$ (OU exploration)

Table 2: List of hyperparameters

for σ_0 over the HalfCheetah-v2 domain (see Figure 12). The remaining parameters were based on previous attempts to obtain good sample efficiency for deterministic policy gradients (Islam et al., 2017; Brockman et al., 2016). We provide a full list in Table 2.

The experiments described here extend our previous conference work (Ciosek and Whiteson, 2018). The experiments here are not directly comparable because they were performed on different versions of MuJoCo environment (version 2 instead of 1) and using PyTorch (Paszke et al., 2017) rather than TensorFlow (Abadi et al., 2015), which have minor differences in the implementation of optimisation algorithms. Despite these differences, our new results are very similar.

For the PPO algorithm, we simply ran the PPO2 version published by OpenAI (Dhariwal et al., 2017) with its default parameters. For the experiments with discrete action spaces we used the hyperparameters of the A2C algorithm (Dhariwal et al., 2017).

A.5. Parameter Tuning for Deterministic Policy Gradients

The original paper the combined Deterministic Policy Gradients with deep networks used $\sigma = 0.2$ (Lillicrap et al., 2015). To ensure a fair comparison, we tested the algorithm using three different settings for the exploration noise σ: 0.2, 0.5 and 1. The parameter for the memory of the OU noise was set to $\frac{15}{20}\sigma$, following Lillicrap et al. (2015). On the HalfCheetah domain (Figure 15), the performances were comparable, with the value 1 leading to the best performance (although the difference wasn't statistically significant). We initially wanted to use that value. However, exploring with such a large noise leads to catastrophic failure in the

(a) HalfCheetah-v2 (b) InvertedDoublePendulum-v2

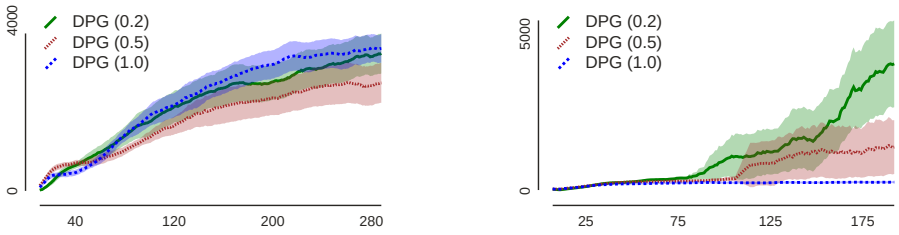


Figure 15: Learning curves (mean and 90% interval) showing the performance of Deterministic Policy Gradients for different values of the exploration noise. All results are obtained from 20 runs. Horizontal axis shows thousands of steps.

InvertedDoublePendulum task. We therefore settled on the value of 0.2, which performed reasonably on both tasks and was already widely used (Lillicrap et al., 2015).

References

Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, and Matthieu Devin. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015, 2015.

Abbas Abdolmaleki, Rudolf Lioutikov, Jan R. Peters, Nuno Lau, Luis Pualo Reis, and Gerhard Neumann. Model-based relative entropy stochastic search. In *Advances in Neural Information Processing Systems*, pages 3537–3545, 2015.

Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Rémi Munos, Nicolas Heess, and Martin A. Riedmiller. Maximum a posteriori policy optimisation. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

Riad Akrour, Gerhard Neumann, Hany Abdulsamad, and Abbas Abdolmaleki. Model-Free Trajectory Optimization for Reinforcement Learning. In *International Conference on Machine Learning*, pages 2961–2970, June 2016.

Shun-Ichi Amari. Natural Gradient Works Efficiently in Learning. *Neural computation*, 10 (2):251–276, 1998.

K. Asadi, C. Allen, M. Roderick, A.-r. Mohamed, G. Konidaris, and M. Littman. Mean Actor Critic. *ArXiv e-prints*, September 2017.

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. An Actor-Critic Algorithm for Sequence Prediction. *arXiv preprint arXiv:1607.07086*, 2016.

Leemon Baird et al. Residual Algorithms: Reinforcement Learning with Function Approximation. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 30–37, 1995.

Shalabh Bhatnagar, Mohammad Ghavamzadeh, Mark Lee, and Richard S Sutton. Incremental Natural Actor-Critic Algorithms. In *Advances in Neural Information Processing Systems*, pages 105–112, 2008.

Peter Bickel and Kjell Doksum. *Mathematical Statistics, Basic Ideas and Selected Topics, Vol. 1, (2nd Edition)*. Prentice Hall, 2 edition, 2006. ISBN 0-13-230637-9.

Steven J. Bradtke. Reinforcement Learning Applied to Linear Quadratic Regulation. In *In Advances in Neural Information Processing Systems 5*, pages 295–302. Morgan Kaufmann, 1993.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. *arXiv preprint arXiv:1606.01540*, 2016.

Kamil Ciosek and Shimon Whiteson. Expected Policy Gradients. In *AAAI 2018: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, February 2018.

John L Crassidis and John L Junkins. *Optimal Estimation of Dynamic Systems*. CRC press, 2011.

Thomas Degris, Martha White, and Richard S Sutton. Off-Policy Actor-Critic. *arXiv preprint arXiv:1205.4839*, 2012.

Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, Yuhuai Wu, and Peter Zhokhov. OpenAI Baselines. *GitHub repository*, 2017.

Michael Fairbank. *Value-Gradient Learning*. PhD thesis, City University London, 2014.

Michael Fairbank and Eduardo Alonso. Value-Gradient Learning. In *Neural Networks (IJCNN), The 2012 International Joint Conference On*, pages 1–8. IEEE, 2012.

Scott Fujimoto, Herke van Hoof, and Dave Meger. Addressing Function Approximation Error in Actor-Critic Methods. *arXiv preprint arXiv:1802.09477*, 2018.

Thomas Furmston and David Barber. A Unifying Perspective of Parametric Policy Search Methods for Markov Decision Processes. In *Advances in Neural Information Processing Systems*, pages 2717–2725, 2012.

Thomas Furmston, Guy Lever, and David Barber. Approximate Newton Methods for Policy Search in Markov Decision Processes. *Journal of Machine Learning Research*, 17:1–51, 2016.

Mohammad Ghavamzadeh, Yaakov Engel, and Michal Valko. Bayesian Policy Gradient and Actor-Critic Algorithms. *Journal of Machine Learning Research*, 17(66):1–53, 2016.

Shixiang Gu, Timothy Lillicrap, Zoubin Ghahramani, Richard E Turner, and Sergey Levine. Q-Prop: Sample-Efficient Policy Gradient with an off-Policy Critic. *arXiv preprint arXiv:1611.02247*, 2016a.

Shixiang Gu, Timothy Lillicrap, Ilya Sutskever, and Sergey Levine. Continuous Deep Q-Learning with Model-Based Acceleration. In *International Conference on Machine Learning*, pages 2829–2838, 2016b.

Shixiang Shane Gu, Timothy Lillicrap, Richard E. Turner, Zoubin Ghahramani, Bernhard Schölkopf, and Sergey Levine. Interpolated Policy Gradient: Merging on-Policy and off-Policy Gradient Estimation for Deep Reinforcement Learning. In *Advances in Neural Information Processing Systems*, pages 3846–3855, 2017.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *International Conference on Machine Learning*, pages 1861–1870, July 2018.

Nicolas Heess, Gregory Wayne, David Silver, Tim Lillicrap, Tom Erez, and Yuval Tassa. Learning Continuous Control Policies by Stochastic Value Gradients. In *Advances in Neural Information Processing Systems*, pages 2944–2952, 2015.

Riashat Islam, Peter Henderson, Maziar Gomrokchi, and Doina Precup. Reproducibility of Benchmarked Deep Reinforcement Learning Tasks for Continuous Control. *arXiv preprint arXiv:1708.04133*, 2017.

Sham M Kakade. A Natural Policy Gradient. In *Advances in Neural Information Processing Systems*, pages 1531–1538, 2002.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

Michail G Lagoudakis and Ronald Parr. Least-Squares Policy Iteration. *Journal of machine learning research*, 4(Dec):1107–1149, 2003.

Weiwei Li and Emanuel Todorov. Iterative Linear Quadratic Regulator Design for Nonlinear Biological Movement Systems. In *ICINCO (1)*, pages 222–229, 2004.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous Control with Deep Reinforcement Learning. *arXiv preprint arXiv:1509.02971*, 2015.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous Methods for Deep Reinforcement Learning. In *International Conference on Machine Learning*, pages 1928–1937, 2016.

Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. Bridging the Gap Between Value and Policy Based Reinforcement Learning. *arXiv preprint arXiv:1702.08892*, 2017.

Ofir Nachum, Mohammad Norouzi, George Tucker, and Dale Schuurmans. Smoothed Action Value Functions for Learning Gaussian Policies. *International Conference on Machine Learning*, 2018a.

Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. Trust-pcl: An off-policy trust region method for continuous control. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018b.

Gergely Neu, Anders Jonsson, and Vicenç Gómez. A Unified View of Entropy-Regularized Markov Decision Processes. *arXiv preprint arXiv:1705.07798*, 2017.

Brendan O'Donoghue, Rémi Munos, Koray Kavukcuoglu, and Volodymyr Mnih. Combining policy gradient and q-learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

Simone Parisi, Matteo Pirotta, and Marcello Restelli. Multi-Objective Reinforcement Learning through Continuous Pareto Manifold Approximation. *Journal of Artificial Intelligence Research*, 57:187–227, 2016.

Paavo Parmas. Total stochastic gradient algorithms and applications in reinforcement learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 10225–10235. Curran Associates, Inc., 2018.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS 2017 Workshop on Autodiff*, 2017. URL `https://openreview.net/forum?id=BJJsrmfCZ`.

Jan Peters and Stefan Schaal. Policy Gradient Methods for Robotics. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference On*, pages 2219–2225. IEEE, 2006.

Jan Peters and Stefan Schaal. Natural Actor-Critic. *Neurocomputing*, 71(7):1180–1190, 2008a.

Jan Peters and Stefan Schaal. Reinforcement Learning of Motor Skills with Policy Gradients. *Neural networks*, 21(4):682–697, 2008b.

Jan Peters, Sethu Vijaykumar, and Stefan Schaal. Policy Gradient Methods for Robot Control. *Report CS-03-787, University of Southern California*, 2003.

Jan Peters, Katharina Mülling, and Yasemin Altun. Relative Entropy Policy Search. In *AAAI*, pages 1607–1612. Atlanta, 2010.

Matteo Pirotta, Marcello Restelli, and Luca Bascetta. Adaptive Step-Size for Policy Gradient Methods. In *Advances in Neural Information Processing Systems*, pages 1394–1402, 2013.

Martin L Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 2014.

Michael Roth, Gustaf Hendeby, and Fredrik Gustafsson. Nonlinear Kalman Filters Explained: A Tutorial on Moment Computations and Sigma Point Methods. *Journal of Advances in Information Fusion*, 11(1):47–70, 2016.

Gavin A Rummery and Mahesan Niranjan. *On-Line Q-Learning Using Connectionist Systems*. University of Cambridge, Department of Engineering, 1994.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust Region Policy Optimization. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 1889–1897, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

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

Marshall H Stone. The Generalized Weierstrass Approximation Theorem. *Mathematics Magazine*, 21(5):237–254, 1948.

Richard S Sutton. Generalization in Reinforcement Learning: Successful Examples Using Sparse Coarse Coding. *Advances in neural information processing systems*, pages 1038–1044, 1996.

Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*, volume 1. MIT press Cambridge, 1998.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *Advances in Neural Information Processing Systems*, pages 1057–1063, 2000a.

Richard S Sutton, SP Singh, and DA McAllester. Comparing Policy-Gradient Algorithms, 2000b. URL `http://incompleteideas.net/papers/SSM-unpublished.pdf`.

ten Hagen, S.H.G., Kröse, B.J.A., van der Broek, W., Verdenius, F., and Amsterdam Machine Learning lab (IVI, FNWI). Linear Quadratic Regulation using Reinforcement Learning. In *Proc. of the 8th Belgian-dutch Conf. on machine learning BENELEARN-98*, pages 39–46, 1998.

Philip S. Thomas and Emma Brunskill. Policy Gradient Methods for Reinforcement Learning with Function Approximation and Action-Dependent Baselines. *arXiv preprint arXiv:1706.06643*, 2017.

Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. Technical report, COURSERA, 2012.

Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A Physics Engine for Model-Based Control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference On*, pages 5026–5033. IEEE, 2012.

George E Uhlenbeck and Leonard S Ornstein. On the Theory of the Brownian Motion. *Physical review*, 36(5):823, 1930.

Harm van Seijen, Hado van Hasselt, Shimon Whiteson, and Marco Wiering. A Theoretical and Empirical Analysis of Expected Sarsa. In *ADPRL 2009: Proceedings of the IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, pages 177–184, March 2009.

Karl Weierstrass. über Die Analytische Darstellbarkeit Sogenannter Willkürlicher Functionen Einer Reellen Veränderlichen. *Sitzungsberichte der Königlich Preußischen Akademie der Wissenschaften zu Berlin*, 2:633–639, 1885.

Cathy Wu, Aravind Rajeswaran, Yan Duan, Vikash Kumar, Alexandre M. Bayen, Sham Kakade, Igor Mordatch, and Pieter Abbeel. Variance Reduction for Policy Gradient with Action-Dependent Factorized Baselines. *ICLR*, 2018.