Expected Policy Gradients for Reinforcement Learning

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

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 quadric critics and then extend it to an analytical method for the universal case, covering a broad class of actors and critics, including Gaussian, exponential families, and reparameterised policies with bounded support. For Gaussian policies, we show that it is optimal to explore using covariance proportional to e^H, where H is the scaled Hessian of the critic with respect to the actions. EPG also provides a general framework for reasoning about policy gradient methods, which we use to 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 show that EPG outperforms existing approaches on six challenging domains involving the simulated control of physical systems.

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

1. Introduction

In reinforcement learning, an agent aims to learn an optimal behaviour policy from trajectories sampled from the environment. In settings where it is feasible to explicitly represent the policy, *policy gradient* methods (Sutton et al., 2000; Peters and Schaal, 2006, 2008b; Silver et al., 2014), which optimise policies by gradient ascent, have enjoyed great success, especially with large or continuous action spaces. The archetypal algorithm optimises 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., 2000) or *deterministic policy gradients* (DPG) (Silver et al., 2014). The theory 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 is undesirable when we want learning to take the cost of exploration into account. 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, which extends our previous work (Ciosek and Whiteson, 2018), 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.

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 principled 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. We present empirical results (Section 6) confirming that this new approach to exploration substantially outperforms DPG with Ornstein-Uhlenbeck exploration in six challenging MuJoCo domains. 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 reparameterised 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.

2. Background

A *Markov decision process* (Puterman, 2014) is a tuple $(S, A, R, p, p_0, \gamma)$ where S is a set of states, A is a set of actions (in practice either $A = \mathbb{R}^d$ or A is finite), $R(s, a)$ is a reward function, $p(s' \mid a, s)$ 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 $s_0 \sim p_0$, $a_t \sim \pi(\cdot \mid s_{t-1})$ and r_t is a sample 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 $V^\pi = \mathbb{E}_\tau \left[\sum_i \gamma_i r_i \right]$ where actions are sampled from π. The Q-function is

---

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

$Q^\pi(a \mid s) = \mathbb{E}_R[r \mid s, a] + \gamma \mathbb{E}_{p_\pi(s' \mid s)}[V^\pi(s') \mid s]$ and the advantage function is $A^\pi(a \mid s) = Q^\pi(a \mid s) - V^\pi(s)$. An optimal policy maximises 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 parameterised by θ, then *stochastic policy gradients* (SPG) (Sutton et al., 2000; Peters and Schaal, 2006, 2008b) perform gradient ascent on ∇J, the gradient of J with respect to θ (gradients without a subscript are always with respect to θ). For stochastic policies, we have:

$$\nabla J = \int_s d\rho(s) \int_a d\pi(a \mid s) \nabla \log \pi(a \mid s)(Q(a, s) + b(s)), \tag{1}$$

where ρ is the discounted-ergodic occupancy measure, defined in the Appendix, and $b(s)$ is a baseline, which can be any function that depends on the state but not the action, since $\int_a d\pi(a \mid s) \nabla \log \pi(a \mid s) b(s) = 0$. Typically, because of ergodicity and Lemma 18 (see Appendix), we can approximate (1) from samples from a trajectory τ of length T:

$$\hat{\nabla} J = \sum_{t=0}^{T} \gamma^t \nabla \log \pi(a_t \mid s_t)(\hat{Q}(a_t, s_t) + b(s_t)), \tag{2}$$

where $\hat{Q}(a_t, s_t)$ 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:

$$\nabla J = \int_s d\rho(s) \nabla \pi(s) [\nabla_a Q(a, s)]_{a=\pi(s)}. \tag{3}$$

This update is then approximated using samples:

$$\hat{\nabla} J = \sum_{t=0}^{T} \gamma^t \nabla \pi(s) \left[\nabla_a \hat{Q}(a, s_t)\right]_{a=\pi(s_t)}. \tag{4}$$

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

$$n_i \leftarrow -n_{i-1}\psi + \mathcal{N}(0, \sigma I) \quad a \sim \pi(s) + n_i. \tag{5}$$

In (2) and (4), $\hat{Q}$ is a *critic* that approximates Q and can be learned by *sarsa* (Rummery and Niranjan, 1994; Sutton, 1996):

$$\hat{Q}(a_t, s_t) \leftarrow \hat{Q}(a_t, s_t) + \alpha[r_{t+1} + \gamma \hat{Q}(s_{t+1}, a_{t+1}) - \hat{Q}(a_t, s_t)]. \tag{6}$$

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

$$\hat{Q}(s_t, a_t) \leftarrow \hat{Q}(a_t, s_t) + \alpha[r_{t+1} + \gamma \int_a d\pi(a \mid s) \hat{Q}(s_{t+1}, a) - \hat{Q}(a_t, s_t)]. \tag{7}$$

We could also use advantage learning (Baird et al., 1995) or LSTDQ (Lagoudakis and Parr, 2003). If the critic's function approximator is *compatible*, then the actor, i.e., π, converges (Sutton et al., 2000).

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

$$\hat{\nabla} J = \sum_{t=0}^{T} \gamma^t \nabla \log \pi(a_t \mid s_t)(r + \gamma \hat{V}(s') - \hat{V}(s)), \tag{8}$$

where $\hat{V}(s)$ is an approximate value function learned using any policy evaluation algorithm. (8) works because $\mathbb{E}\left[\delta(r, s', s) \mid a, s\right] = A(s, a)$, 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, the TD error is noisy, introducing variance in the gradient.

To cope with this variance, we can reduce the learning rate when the variance of the gradient would otherwise explode, using, e.g., *Adam* (Kingma and Ba, 2014), *natural policy gradients* (Kakade, 2002; Amari, 1998; Peters and Schaal, 2008a) or *Newton's method* (Furmston and Barber, 2012). However, this results in slow learning when the variance is high. See Section 7 for further discussion on variance reduction techniques.

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 (1):

$$\nabla J = \int_s d\rho(s) \underbrace{\int_a d\pi(a \mid s) \nabla \log \pi(a \mid s)(Q(a, s) + b(s))}_{I_\pi^Q(s)}$$

$$= \int_s d\rho(s) \int_a d\pi(a \mid s) \nabla \log \pi(a \mid s) Q(a, s)$$

$$= \int_s d\rho(s) I_\pi^Q(s). \tag{9}$$

This suggests a new way to write the approximate gradient[2], using Lemma 18 (see Appendix):

$$\hat{\nabla} J = \sum_{t=0}^{T} \underbrace{\gamma^t I_\pi^{\hat{Q}}(s_t)}_{g_t}, \quad \text{where} \quad I_\pi^{\hat{Q}}(s) = \int_a d\pi(a \mid s) \nabla \log \pi(a \mid s) \hat{Q}(a, s). \tag{10}$$

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.

---

2. 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.

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

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

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 (10) to form a policy gradient algorithm that repeatedly estimates $\hat{I}_\pi^Q(s)$ with an integration subroutine.

Algorithm 1 Expected policy gradients

1: $s \leftarrow s_0$, $t \leftarrow 0$
2: initialise optimiser, initialise policy π parameterised by θ
3: **while** not converged **do**
4:     $g_t \leftarrow \gamma^t$ DO-INTEGRAL$(\hat{Q}, s, \pi_\theta)$ ▷ g_t is the estimated policy gradient as per (10)
5:     $\theta \leftarrow \theta +$ optimiser.UPDATE(g_t)
6:     $a \sim \pi(\cdot, s)$
7:     $s', r \leftarrow$ simulator.PERFORM-ACTION(a)
8:     $\hat{Q}$.UPDATE(s, a, r, s')
9:     $t \leftarrow t + 1$
10:     $s \leftarrow s'$
11: **end while**

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 Figure 1 (left). A simple Monte Carlo method evaluates the integral by sampling one or more times from $\pi(a \mid s)$ (blue) and evaluating $\nabla_\mu \log \pi(a \mid s) Q(a, s)$ (red) as a function of a. A baseline can decrease the variance by adding a multiple of $\nabla_\mu \log \pi(a \mid s)$ to the red curve, but the problem remains that the red curve has high values where the blue curve is almost zero. Consequently, substantial variance persists, whatever the baseline, 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.

We now provide our most general results, which apply to EPG in *any* setting.

3.1 General Policy Gradient Theorem

We begin by stating our most general result, showing that EPG can be seen as a generalisation of both SPG and DPG. To do this, we first state a new general policy gradient theorem.
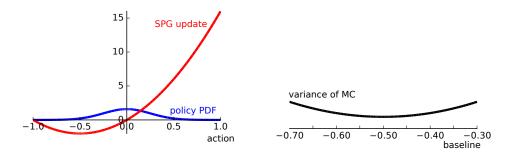
Figure 1: At left, $\pi(a \mid s)$ for a Gaussian policy with mean $\mu = \theta = 0$ at a given state and constant σ^2 (blue) and $\nabla_\theta \log \pi(a \mid s) Q(a, s)$ for $Q = \frac{1}{2} + \frac{1}{2}a$ (red). At right, the variance of a simple single-sample Monte Carlo estimator as a function of the baseline. In a simple multi-sample Monte Carlo method, the variance would go down as the number of samples.

Theorem 1 (General Policy Gradient Theorem) *If $\pi(\cdot, s)$ is a normalised Lebesgue measure for all s, then*

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

Proof We begin by expanding the following expression.

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

The first equality follows by expanding the definition of Q and the penultimate one follows from Lemma 19 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., 2000) 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 $\pi(\cdot \mid s)$ is differentiable, then*

$$\nabla J = \int_s d\rho(s) I_G(s) = \int_s d\rho(s) \int_a d\pi(a \mid s) \nabla \log \pi(a \mid s) Q(a, s).$$

Proof We obtain the following by expanding ∇V.

$$\nabla V = \nabla \int_a d\pi(a|s)Q(a,s) = \int_a da(\nabla\pi(a|s))Q(a,s) + \int_a d\pi(a|s)(\nabla Q(a,s))$$

We obtain $I_G(s) = \int_a d\pi(a \mid s)\nabla \log \pi(a \mid s)Q(a,s) = I_\pi^Q(s)$ by plugging this into the definition of $I_G(s)$. We obtain ∇J by invoking Theorem 1 and plugging in the above expression for $I_G(s)$. ∎

We now recover the DPG update introduced in (3).

Corollary 3 (Deterministic Policy Gradient Theorem) *If $\pi(\cdot \mid s)$ is a Dirac-delta measure (i.e., a deterministic policy) and $Q(\cdot, s)$ is differentiable, then*

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

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

Proof We begin by expanding the term for $\nabla V(s)$, which will be useful later on.

$$\nabla V(s) = \nabla Q(\pi(s), a) = \left[\nabla Q(a,s)\right]_{a=\pi(s)} + \left[\nabla_a Q(a,s)\right]_{a=\pi(s)}\nabla\pi(s) \tag{11}$$

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

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

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

Here, the second equality follows by observing that the policy is a Dirac-delta and the third one follows from using (11). We can then obtain ∇J by invoking Theorem 1 and plugging in the above expression for $I_G(s)$. ∎

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

We now prove that for any policy, the EPG estimator of (10) has lower variance than the SPG estimator of (2).

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

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

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

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

We start by applying Lemma 21 to the lefthand side and setting $X = X_1(s_t) = \gamma^t \nabla \log \pi(a_t \mid s_t)(\hat{Q}(a_t, s_t) + b(s_t))$ where $a_t \sim \pi(a_t|s_t)$. This shows that

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

is the total return of the MRP (p, p_0, u_1, γ^2), where

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

Likewise, applying Lemma 21 again to the righthand 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

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

Note that $\mathbb{E}_{X_1(x|s)}[x] = \mathbb{E}_{X_2(x|s)}[x]$ and therefore $u_1 \geq u_2$. Furthermore, by assumption of the lemma, the inequality is strict. The lemma then follows by applying Observation 22. ∎

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 modelled 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 $\nabla \log \pi(a \mid s)\hat{Q}(a, s)$ 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 optimising the policy would be unnecessary. Since we know that both the estimators of (2) and (10) are unbiased, 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

---

Algorithm 2 Gaussian policy gradients

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

Algorithm 3 Gaussian integrals

1: **function** DO-INTEGRAL-GAUSS$(\hat{Q}, s, \pi_\theta)$
2:     $I^Q_{\pi(s),\mu_s} \leftarrow (\nabla \mu_s) \nabla_a \hat{Q}(a = \mu_s, s)$ ▷ Use Lemma 5
3:     **return** $I^Q_{\pi(s),\mu_s}$
4: **end function**
5:
6: **function** GET-COVARIANCE$(\hat{Q}, s, \pi_\theta)$
7:     $H \leftarrow$ COMPUTE-HESSIAN$(\hat{Q}(\mu_s, s))$
8:     **return** $\sigma_0 e^{cH}$ ▷ Use Lemma 6
9: **end function**

(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).

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 HalfCheetah-v1, 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) *If the policy is Gaussian, i.e. $\pi(\cdot|s) \sim \mathcal{N}(\mu_s, \Sigma_s)$ with μ_s and $\Sigma_s^{1/2}$ parameterised by θ, and the critic is of the form $Q(a, s) = a^\top A(s)a + a^\top B(s) + const$, then*

$$I_\pi^Q(s) = \left[I_{\pi(s),\mu_s}^Q \,\middle|\, I_{\pi(s),\Sigma_s^{1/2}}^Q \right]^\top,$$

where the mean and covariance components are given by:

$$I_{\pi(s),\mu_s}^Q = (\nabla\mu_s)B(s),$$
$$I_{\pi(s),\Sigma_s^{1/2}}^Q = (\nabla\Sigma_s^{1/2})\Sigma_s^{1/2}A(s). \tag{12}$$

Proof For ease of presentation, we prove the lemma for a one-dimensional action space, where $\mu_s, a \in \mathbb{R}$ and $\Sigma_s^{1/2} = \sigma_s$ is the standard deviation (we drop the suffix s in μ_s and σ_s in the subsequent formulae). First, note that the constant term in the critic does not influence the value of $I_\pi^Q(s)$ since it depends only on the state and not on the action and can be treated as a baseline. Observe that

$$I_{\pi(s),\mu}^Q = (\nabla\mu)\mathbb{E}_\pi \left[\nabla_\mu \log \pi(a|s)Q(a, s) \right]$$
$$= (\nabla\mu) \left(\mathbb{E}_\pi \left[\nabla_\mu \log \pi(a|s)a^\top B(s) \right] \mathbb{E}_\pi \left[\nabla_\mu \log \pi(a|s)a^\top A(s)a \right] \right).$$

We consider the linear term and the quadric term separately. For the linear term we have:

$$\mathbb{E}_\pi \left[\nabla_\mu \log \pi(a|s)aB(s) \right] = \mathbb{E}_\pi \left[\frac{a - \mu}{\sigma^2} B(s)a \right]$$
$$= \frac{1}{\sigma^2}\mathbb{E}_\pi \left[B(s)a^2 - B(s)a\mu \right] =$$
$$= \frac{1}{\sigma^2} \left(B(s)\mathbb{E}_\pi \left[a^2 \right] - B(s)\mu\mathbb{E}_\pi \left[a \right] \right)$$
$$= \frac{1}{\sigma^2} \left(B(s) \left(\sigma^2 + \mu^2 \right) - B(s)\mu\mu \right) = B(s).$$

For the quadric term we have:

$$\mathbb{E}_\pi \left[\nabla_\mu \log \pi(a|s)a^2 A(s) \right] = \frac{1}{\sigma^2}\mathbb{E}_\pi \left[a^2 A(s)(a - \mu) \right] =$$
$$= \frac{A(s)}{\sigma^2}\mathbb{E}_\pi \left[a^3 - a^2\mu \right] =$$
$$= \frac{A(s)}{\sigma^2} \left(\mu^3 + 3\mu\sigma^2 - (\mu^2 + \sigma^2)\mu \right) =$$
$$= 2A(s)\mu$$

Summing the two terms yields:

$$I_{\pi(s),\mu}^Q = \nabla\mu(2A(s)\mu + B(s)).$$

We now calculate the integrals for the standard deviation, again beginning with the linear term:

$$
\begin{aligned}
\mathbb{E}_\pi \left[\nabla_\sigma \log \pi(a|s) a B(s) \right] &= \mathbb{E}_\pi \left[B(s) a \frac{(a-\mu)^2}{\sigma^3} - B(s) a \frac{1}{\sigma} \right] \\
&= \frac{B(s)}{\sigma} \mathbb{E}_\pi \left[\frac{1}{\sigma^2} \left(a^3 - 2a^2\mu + \mu^2 a \right) - a \right] \\
&= \frac{B(s)}{\sigma} \left(\frac{1}{\sigma^2} \left(\mu^3 + 3\mu\sigma^2 - 2(\mu^2 + \sigma^2)\mu + \mu^3 \right) - \mu \right) \\
&= \frac{B(s)}{\sigma} (\mu - \mu) = 0.
\end{aligned}
$$

For the quadric term we have:

$$
\begin{aligned}
& \mathbb{E}_\pi \left[\nabla_\sigma \log \pi(a|s) A(s) a^2 \right] \\
&= \mathbb{E}_\pi \left[A(s) a^2 \frac{(a-\mu)^2}{\sigma^3} - A(s) a^2 \frac{1}{\sigma} \right] \\
&= \frac{A(s)}{\sigma} \mathbb{E}_\pi \left[\frac{1}{\sigma^2} (a^4 - 2a^3\mu + \mu^2 a^2) - a^2 \right] \\
&= \frac{A(s)}{\sigma} \left[\frac{1}{\sigma^2} (\mu^4 + 6\mu^2\sigma^2 + 3\sigma^4 - 2\mu(\mu^3 + 3\mu\sigma^2) + \mu^2(\mu^2 + \sigma^2)) - (\mu^2 + \sigma^2) \right] \\
&= 2A(s)\sigma.
\end{aligned}
$$

Summing the two terms yields:

$$
I^Q_{\pi(s),\sigma} = \nabla\sigma(2A(s)\sigma).
$$

The multivariate case (i.e., with a multi-dimensional action space) can be obtained using the method developed in Section 5 later in the paper by observing that the multivariate normal distribution is in the parametric family given by (15) with the sufficient statistic vector T containing the vector a and the vectorised matrix $aa^\top$, both of which are polynomial in a, and hence Lemma 8 is applicable. ∎

While Lemma 5 requires the critic to be quadric 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 (12) suggests that we can include the covariance in the actor network and learn it along with the mean, using the update rule:

$$
\Sigma_s^{1/2} \leftarrow \Sigma_s^{1/2} + \alpha \Sigma_s^{1/2} H(s). \tag{13}
$$

However, another option is to compute the covariance from scratch at each iteration by analytically computing the result of applying (13) infinitely many times, as in the following lemma.

Lemma 6 (Exploration Limit) *The iterative procedure defined by (13) applied n times using the diminishing learning rate $\alpha = 1/n$ converges to $\Sigma_s^{1/2} \propto e^{H(s)}$ as $n \to \infty$.*

eigenvalue increases

sharp maximum,
very little exploration

moderate exploration

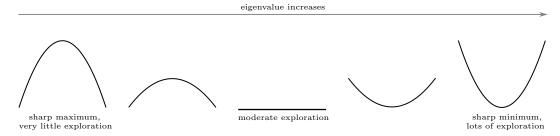sharp minimum,
lots of exploration

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 decreases as we move towards the right. There is almost no exploration to the far right, where we have a sharp maximum.

Proof Consider the sequence $(\Sigma_s^{1/2})_1 = \sigma_0 I$, $(\Sigma_s^{1/2})_n = (\Sigma_s^{1/2})_{n-1} + \frac{1}{n}(\Sigma_s^{1/2})_{n-1}H(s)$. We diagonalise 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:

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

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:

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

$\blacksquare$

The practical implication of Lemma 6 is that, in a policy gradient method, it is justified[3] to use Gaussian exploration with covariance proportional to e^{cH} for some reward scaling constant c. Thus, by exploring with (scaled) covariance e^{cH}, we obtain a principled alternative to the Ornstein-Uhlenbeck heuristic of (5). Our results below show that it also performs much better in practice.

Lemma 6 has an intuitive interpretation. If $H(s)$ has a large positive eigenvalue λ, then $\hat{Q}(s, \cdot)$ 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 just the same scalar. The exploration mechanism in the one-dimensional case is illustrated in Figure 2. The idea is simple: the larger the eigenvalue the worse the minimum we are in and the more exploration we need to leave it. On the other hand, if λ is negative, then $\hat{Q}(s, \cdot)$ has a maximum and so e^λ is small, since exploration is not needed.

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

---

3. Lemma 6 relies crucially on the use of step sizes diminishing in the length of the trajectory, rather than finite step sizes. Therefore, the step sequence serves as a useful intermediate stage between simply taking *one* PG step of (13) and using finite step sizes, which would mean that the covariance would converge either to zero or diverge to infinity.
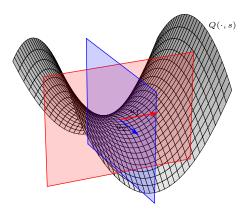
Figure 3: In multi-dimensional action spaces, the critic $\hat{Q}(\cdot, s)$ can have saddle points. In this case, we define exploration along each eigenvector separately.

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.[4]

4.3 Action Clipping

We now describe how GPG works in environments where the action space has bounded support[5]. 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 as follows

$$a \sim \pi(a \mid s) \quad \text{equivalent to} \quad a = \max(\min(b, 1), 0) \ \text{ with } \ b \sim \pi_b(b \mid s). \tag{14}$$

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 $M = (S, A, R, p, p0, \gamma)$ with $A = [0, 1]^d$ into another MDP $M' = (S, A', R, p', p0, \gamma)$, where $A' = \mathbb{R}^d$ and p' is defined as

$$p'(s'|b, s) \ = \ p(s'| \max(\min(b, 1), 0), s).$$

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 $\pi_D^\star : S \to A$. Now, $\pi_D^\star$ can be transformed into a policy for M of the form $\max(\min(\pi_D^\star(s), 1), 0)$. 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).

---

4. Of course the optimisation is still local and there is no guarantee of finding a global optimum—we can merely increase our chances.

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

Algorithm 4 Policy gradients with clipped actions.

1: $s \leftarrow s_0$, $t \leftarrow 0$
2: initialise optimiser, initialise policy π parameterised by θ
3: **while** not converged **do**
4:   $g_t \leftarrow \gamma^t$ DO-INTEGRAL$(\hat{Q}_b, s, \pi_\theta)$
5:   $\theta \leftarrow \theta +$ optimiser.UPDATE(g_t)
6:   $b \sim \pi(\cdot, s)$
7:   $a = c(b)$ $\triangleright$ Clipping function $c(b) = \max(\min(\pi_D^\star(s), 1), 0)$.
8:   $s', r \leftarrow$ simulator.PERFORM-ACTION(a)
9:   $\hat{Q}_b$.UPDATE(s, b, r, s') $\triangleright$ Update using the pre-clipping action b.
10:   $t \leftarrow t + 1$
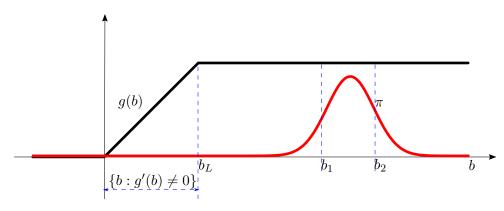11: $s \leftarrow s'$
12: **end while**

Figure 4: 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).

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 Figure 4.

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, a $b_\star$ has to be chosen which falls into the interval $(-\infty, b_L]$. Since the b's are samples from a Gaussian with infinite support, it will eventually happen and a nonzero gradient will be obtained. However, if this interval falls into a distant part of the tail of π_b, convergence will be slow.

However, this problem is mitigated with GPG. To see why, consider Figure 5. Once the policy shifts into the flat area, the critic becomes constant. A constant critic has a zero Hessian, generating a boost to exploration by increasing the standard deviation of the policy, making it much more likely that a point $b < b_L$ is sampled and a useful gradient is obtained.
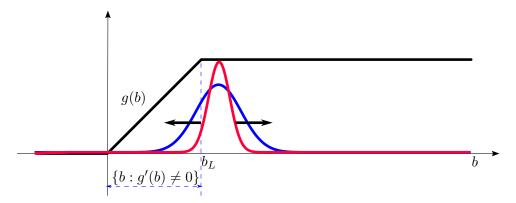
Figure 5: GPG avoids the vanishing gradient problem. Once a policy (denoted in red) enters the flat area entering the flat area $b > b_L$, exploration immediately increases (the new distribution is in blue).

Another way of mitigating the hard clipping problem is to use a differentiable squashing function, which we describe in Section 5.

4.4 Quadric Critics and their Approximations

Gaussian policy gradients require a quadric critic *given the state*. This assumption, which is different from assuming a quadric 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 theory, are known to have a Q-function that is quadric in the action vector given the state (Crassidis and Junkins, 2011, Equation 8.81)[6]. Second, it is often assumed (Li and Todorov, 2004) that a quadric critic (or a quadric approximation to a general critic) is enough to capture enough local structure to preform a policy optimisation step, in much the same way as Newton's method for deterministic unconstrained optimisation, which locally approximates a function with a quadric, can be used to optimise a non-quadric function across several iterations. In Corollary 7 below, we describe such an approximation method applied to GPG where we approximate Q with a quadric function in the neighbourhood of the policy mean.

Corollary 7 (Approximate Gaussian Policy Gradients with an 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 $\Sigma_s^{1/2}$ parameterised by θ as in Lemma 5 and any critic $Q(a, s)$ doubly differentiable with respect to actions for each state, then $I_{\pi(s),\mu_s}^Q \approx (\nabla \mu_s)\nabla_a Q(a = \mu_s, s)$ and $I_{\pi(s),\Sigma_s^{1/2}}^Q \approx (\nabla \Sigma_s^{1/2})\Sigma_s^{1/2} H(\mu_s, s)$, where $H(\mu_s, s)$ is the Hessian of 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 Q in μ_s.

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

---

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

Because of the series truncation, the function on the righthand side is quadric and we can then use Lemma 5:

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

∎

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. In these cases, we can approximate the Hessian by generating a number of random action-values around μ_s, computing the Q values, and (locally) fitting a quadric, 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.[7]

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. 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 rarely necessary.

Lemma 8 (EPG for Exponential Families with Polynomial Sufficient Statistics)
Consider the class of policies parameterised by θ where:

$$
\pi(a \mid s) = e^{\eta_\theta^\top T(a) - U_{\eta_\theta} + W(a)}, \tag{15}
$$

where each entry in the vector $T(a)$ is a (possibly multivariate) polynomial in the entries of the vector a. Moreover, assume that the critic $\hat{Q}(a)$ is (a possibly multivariate) polynomial

---

7. 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.

in the entries of a. Then, the policy gradient update is a closed form expression in terms of the uncentered moments of $\pi(\cdot \mid s)$:

$$I_\pi^Q(s) = (\nabla \eta_\theta^\top)(C_{TQ}^\top m_\pi) - (\nabla U_{\eta_\theta})(C_Q^\top m_\pi), \tag{16}$$

where C_Q is the vector containing the coefficients of the polynomial Q, C_{TQ} is the vector containing the coefficients of the polynomial $T(a)Q(a)$ (i.e., a multiplication of T and Q) and 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.

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

Since $T(a)$ and $Q(a)$ 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 Q is a polynomial, it is a sum of monomial terms:

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

On the right, the terms $\mathbb{E}_{a \sim \pi}\left[\prod_{j=1}^{d} a_j^{p_i(j)}\right]$, 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_Q and the cross-moments into the vector m_π, we obtain the right term in (16). We can apply the same reasoning to the product of T and Q to obtain the left term. ∎

The cross-moments themselves can be obtained from the *moment generating function* (MGF) of π. Indeed, for a distribution of the form of (15), the MGF of $T(a)$ 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.2.

Note that the assumption that T and Q are polynomial is with respect to the action a. The dependence on the state only appears in η_θ and U_{η_θ} and 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.[8] It is an open research question whether more suitable universal approximators admitting analytic EPG solutions can be identified.

5.2 Reparameterised Exponential Families and Reparameterised Critics

In Lemma 8, we assumed that the function $T(a)$ (called the sufficient statistic of the exponential family) is polynomial. We now relax this assumption. Our approach is to start with a policy π_b which *does* have a polynomial sufficient statistic and then introduce a suitable reparameterisation function $g : \mathbb{R}^d \to A$. The policy is then defined as:

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

where b is the random variable representing the action before the squashing. Assuming that g^{-1} exists and the Jacobian ∇g is non-singular almost everywhere, the PDF[9] of the policy π can be written as:

$$\pi(a \mid s) = \pi_b(g^{-1}(a) \mid s) \frac{1}{\det \nabla g(g^{-1}(a))} = \pi_b(b \mid s) \frac{1}{\det \nabla g(b)}. \tag{17}$$

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 (17). Assume that the Jacobian of g is nonsingular except on a set of π_b-measure zero. Consider a critic Q. Denote as Q_b a reparameterised critic such that for all a, $Q_b(g^{-1}(a)) = Q(a)$. Then the policy gradient update is given by the formula $I_\pi^Q(s) = I_{\pi_b}^{Q_b}(s)$.*

Proof

$$\begin{aligned}
I_\pi^Q(s) &= \int_A d\pi(a \mid s) \nabla \log \pi(a \mid s) Q(a) \\
&= \int_{\mathbb{R}^d} d\pi(g(b) \mid s) \nabla \log \pi(g(b) \mid s) Q(g(b)) \det \nabla g(b) \\
&= \int_{\mathbb{R}^d} d\pi_b(b \mid s) \nabla \log \pi(g(b) \mid s) Q_b(b) \\
&= \int_{\mathbb{R}^d} d\pi_b(b \mid s)(\nabla \log \pi_b(b \mid s) - \underbrace{\nabla \log \det \nabla g(b)}_{0}) Q_b(b) = I_{\pi_b}^{Q_b}(s).
\end{aligned}$$

In the second equality, we perform the variable substitution $a = g(b)$. In the third equality we use (17) and the fact that $Q_b(g^{-1}(a)) = Q(a)$. In the fourth equality we again use (17) and the fact that $\log \det \nabla g(b) = 0$ since g is not parameterised by θ. ∎

We are now ready to state our universality result. The idea is to obtain a reparameterised version of EPG (and Lemma 8) by reparameterising 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 Reparameterisation) *Consider the class of policies, parameterised by θ, defined as in* (15). *Consider reparameterisation function g and define T_b, V_b and Q_b as $T_b(g^{-1}(a)) = T(a)$, $W_b(g^{-1}(a)) = W(a)$ and $Q_b(g^{-1}(a)) = Q(a)$ for every a. Assume the following:*

1. *g is invertible;*

2. *The Jacobian of g exists and is nonsingluar except on a set of π_b-measure zero, where π_b is the reparameterised policy as in* (17); *and*

3. *T_b and Q_b are polynomial as in Lemma 8.*

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

$$I_\pi^Q(s) = (\nabla \eta_\theta^\top)(C_{T_b Q_b}^\top m_{\pi_b}) - (\nabla U_{\eta_\theta})(C_{Q_b}^\top m_{\pi_b}). \tag{18}$$

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 4.3, 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 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 Q to be quadric in $g^{-1}(a)$, i.e., Q_b is quadric in b. The reparameterised version of EPG is the same as Algorithm 4 except it uses a squashing function g instead of the clipping function c.

5.3 Aribtrary 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 (nondegenerate) probability distribution $\pi(\cdot \mid s)$ which has a mean. Assume that the critic $\hat{Q}(a)$ is of the form $A_s^\top a$ for some coefficient vector A_s. Then the policy gradient update is given by $I_\pi^Q(s) = A_s^\top \nabla \mu_{\pi(\cdot|s)}$ where $\mu_{\pi(\cdot|s)}$ denotes the integral $\int_a a\, d\pi(a \mid s)$ (the mean).*

Proof

$$I_\pi^Q(s) = \int_a \nabla\pi(a \mid s)Q(a \mid s)da$$

$$= \int_a \nabla\pi(a \mid s)A_s^\top a\, da$$

$$= A_s^\top \nabla \underbrace{\int_a \pi(a \mid s)a\, da}_{\mu_{\pi(\cdot|s)}}$$

$$= A_s^\top \nabla(\mu_{\pi(\cdot|s)}).$$

∎

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 discretisation of continuous time systems with a fine-enough time scale. 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 neighbourhood of a given point and then choose the time step to be small enough so an action does not leave that neighbourhood. We can then use Lemma 11 to perform policy gradients with any policy.[10]

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

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

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. 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.

---

10. 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.

6. Experiments

While EPG has many potential uses, we focus on empirically evaluating one particular application: exploration driven by the Hessian exponential (as introduced in Algorithm 2 and Lemma 6), replacing the standard Ornstein-Uhlenbeck (OU) exploration in continuous action domains. To this end, we apply EPG to five domains modelled with the MuJoCo physics simulator (Todorov et al., 2012): HalfCheetah-v1, InvertedPendulum-v1, Reacher2d-v1, Walker2d-v1, and InvertedDoublePendulum-v1 and compare its performance to DPG and SPG. The experiments described here extend our previous conference work (Ciosek and Whiteson, 2018) in two ways: we added the InvertedDoublePendulum-v1 domain and used it for a detailed comparison with the PPO algorithm (Schulman et al., 2017).

In practice, EPG differs from deep DPG (Lillicrap et al., 2015; Silver et al., 2014) only in the exploration strategy, though their theoretical underpinnings are also different. 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). The exploration hyperparameters for EPG were $\sigma_0 = 0.2$ and $c = 1.0$ where the exploration covariance is $\sigma_0 e^{cH}$. These values were obtained using a grid search from the set $\{0.2, 0.5, 1\}$ for σ_0 and $\{0.5, 1.0, 2.0\}$ for c over the HalfCheetah-v1 domain. Since c is just a constant scaling the rewards, it is reasonable to set it to 1.0 whenever reward scaling is already used. Hence, our exploration strategy has just one hyperparameter σ_0 as opposed specifying a pair of parameters (standard deviation and mean reversion constant) for OU. We used the same learning parameters for the other domains. For SPG[11], we used OU exploration and a constant diagonal covariance of 0.2 in the actor update (this approximately corresponds to the average variance of the OU process over time). The other parameters for SPG are the same as for the rest of the algorithm. For the learning curves, we obtained 90% confidence intervals and show results of independent evaluation runs that used actions generated by the policy mean without any exploration noise.

The Hessian in GPG 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 quadric is fit to them in the L2 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 used a quadric critic. However, we did not optimise this part of the algorithm since the core message of GPG is that a Hessian is useful, not how to compute it efficiently.

The results in Figure 6 show that EPG's exploration strategy yields much better performance than DPG with OU. Furthermore, SPG does poorly, solving only the easiest domain (InvertedPendulum-v1) reasonably quickly, achieving slow progress on HalfCheetah-v1, 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-v1, SPG initially learns quickly, outperforming the other methods. This is 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 subpar performance late in learning.

---

11. We tried learning the covariance for SPG but the covariance estimate was unstable; no regularisation hyperparameters we tested matched SPG's performance with OU even on the simplest domain.
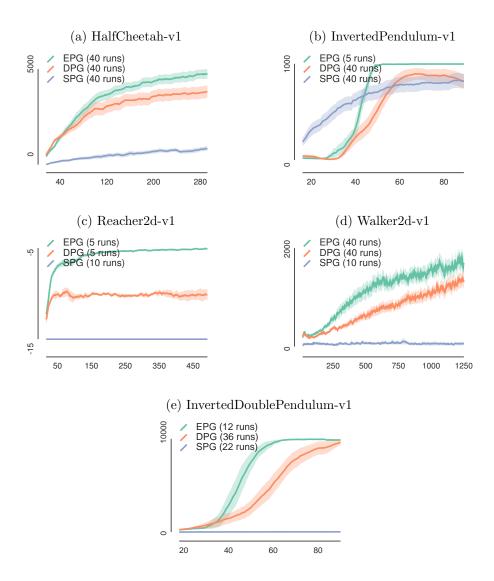
Figure 6: Learning curves (mean and 90% interval). Returns for Reacher2d-v1 are clipped at -14. The number of independent training runs is in parentheses. Horizontal axis is scaled in thousands of steps.

In addition, EPG typically learns more consistently than DPG with OU. In three tasks, the empirical standard deviation across runs of EPG ($\hat{\sigma}_{\mathrm{EPG}}$) was substantially lower than that of DPG ($\hat{\sigma}_{\mathrm{DPG}}$) at the end of learning, as shown in Table 1. For the other two domains, the confidence intervals around the empirical standard deviations for DPG and EPG were too wide to draw conclusions.

Surprisingly, for InvertedPendulum-v1, DPG's learning curve declines late in learning. The reason can be seen in the individual runs shown in Figure 7: both DPG and SPG suffer from severe unlearning. This unlearning cannot be explained by exploration noise since the

Domain	$\hat{\sigma}_{\mathrm{DPG}}$	$\hat{\sigma}_{\mathrm{EPG}}$
HalfCheetah-v1	1336.39 [1107.85, 1614.51]	1056.15 [875.54, 1275.94]
InvertedPendulum-v1	291.26 [241.45, 351.88]	0.00 n/a
Reacher2d-v1	1.22 [0.63, 2.31]	0.13 [0.07, 0.26]
Walker2d-1	543.54 [450.58, 656.65]	762.35 [631.98, 921.00]
InvertedDoublePendulum-v1	921.73 [756.01, 1125.63]	226.07 [157.20, 326.00]

Table 1: Estimated standard deviation (mean and 90% interval) across runs after learning.
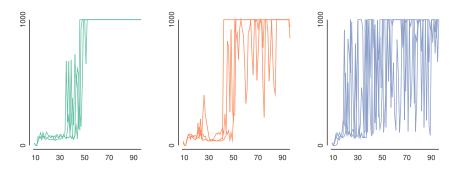

Figure 7: Three runs for EPG (left), DPG (middle) and SPG (right) for the InvertedPendulum-v1 domain, demonstrating that EPG shows much less unlearning.

evaluation runs just use the mean action, without exploring. Instead, OU exploration in DPG may be too coarse, causing the optimiser to exit good optima, while SPG unlearns due to noise in the gradients. The noise also helps speed initial learning, as described above, but this does not transfer to other domains. EPG avoids this problem by automatically reducing the noise when it finds a good optimum, i.e., a Hessian with large negative eigenvalues as described is Section 4.2.

The fact that EPG is stable in this way 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 excessivly large differences between subsequent policies. To address it, we compare our results with *proximal policy pptimisation* (PPO) (Schulman et al., 2017), a policy gradient algorithm designed specifically to include a term penalising the difference between successive policies. Comparing our EPG result for InvertedDoublePendulum-v1 in Figure 6e with PPO (Schulman et al., 2017, Figure 3, first row, third plot from left, blue PPO curve), it is clear that EPG is more stable. This suggests that efficient adaptive exploration of the type used by EPG is important for stability, even in this relatively simple domain.

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 minimisation techniques that interpolate between two estimators (Gu et al., 2016a). However, EPG uses a quadric (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, 2014), *natural policy gradients* (Kakade, 2002; Amari, 1998; Peters and Schaal, 2008a), trust region policy optimisation (Schulman et al., 2015), proximal policy optimisation (Schulman et al., 2017), the adaptive step size method (Pirotta et al., 2013) or *Newton's method* (Furmston and Barber, 2012; Furmston et al., 2016; Parisi et al., 2016). However, this results in slow learning when the variance is high.

7.2 Sarsa and Q-Learning

It has been known since the introduction of policy gradient methods (Sutton et al., 2000) that they represent a kind of slow-motion policy improvement as opposed to a greedy improvement performed by methods such as (expected) sarsa. The two main reasons for the slow-motion improvement are that a greedy maximisation 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. The argument for a sarsa-like method is that it may converge faster and does not need an additional optimisation for the actor. Recently, approaches combining the features of both methods have been investigated. One-step Newton's method for Q-functions that are quadric 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., 2016). 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:

$$I_\pi^Q(s) = [\nabla_a Q(a, s)]_{a=\mu_s} \nabla \mu_s.$$

We now formalise 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:

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

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 quadric 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 quadric), 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 quadric 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.[12] 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 follows.[13]

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

---

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

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

$$
\begin{aligned}
I_G^E(s) &= I_G(s) + \alpha \nabla \mathcal{H}(s) \\
&= \int_a d\pi(a|s) \nabla \log \pi(a|s)(Q(a,s) - \alpha \log \pi(a|s)). \quad (19)
\end{aligned}
$$

This equation makes clear that performing entropy regularisation 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 regularisation to the objective of optimising 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 (19), though we obtained ours independently.

Next, we derive a further specialisation of (19) for the case where the parameters θ are shared between the actor and the critic. We start with the policy gradient identity given by (9) 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:

$$
\pi(a|s) = \frac{1}{Z(s)} e^{\hat{Q}(a,s)}, \quad \text{where} \quad Z(s) = \int_a e^{\hat{Q}(a,s)} da. \quad (20)
$$

For the continuous case, we assume that the integral in (20) converges for each state. Here, we assume that the approximate critic is parameterised by θ. Because of the form of (20), the policy is parameterised by θ as well. Now, for the policy class given by (20), we can simplify the gradient update even further, obtaining:

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

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 regularisation does nothing except scale the learning rate.[14] Alternatively, for this shared parameterisation, 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.

---

14. 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 behaviour policy as $b(a \mid s)$ and the corresponding discounted-ergodic measure as ρ_b. The method uses the following reweighting approximation:

$$\nabla J = \int_s d\rho(s) \int_a d\pi(a \mid s) \nabla \log \pi(a \mid s)(Q(a,s))$$
$$\approx \int_s d\rho_b(s) \int_a d\pi(a \mid s) \nabla \log \pi(a \mid s)(Q(a,s)). \tag{21}$$

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_π^Q:

$$\int_s d\rho_b(s) \int_a d\pi(a \mid s) \nabla \log \pi(a \mid s)(Q(a,s)) = \int_s d\rho_b(s) I_\pi^Q(s). \tag{22}$$

Here, we use an analytic solution to $I_\pi^Q(s)$ 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 $I_\pi^Q(s)$, 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, (22) makes clear that off-policy actor-critic differs from SPG in two places: the use of ρ_b as in (21) 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 reweighting approximation

---

1: $s \leftarrow s_0$, $t \leftarrow 0$
2: initialise optimiser, initialise policy π parameterised by θ
3: **while** not converged **do**
4:      $g_t \leftarrow \gamma^t$ DO-INTEGRAL$(\hat{Q}, s, \pi_\theta)$ $\triangleright$ g_t is the estimated policy gradient as per (10)
5:      $\theta \leftarrow \theta +$ optimiser.UPDATE(g_t)
6:      $a \sim b(\cdot, s)$
7:      $s', r \leftarrow$ simulator.PERFORM-ACTION(a)
8:      $\hat{Q}$.UPDATE(s, a, r, s', π, b) $\triangleright$ Off-policy critic algorithm
9:      $t \leftarrow t + 1$
10:     $s \leftarrow s'$
11: **end while**

7.6 Value Gradient Methods

Value gradient methods (Fairbank, 2014; Fairbank and Alonso, 2012; Heess et al., 2015) assume the same parameterisation of the policy as policy gradients, i.e., π is parameterised by θ, and maximise 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:

$$\nabla J = \int_{s_0} dp_0(s_0) \nabla V(s_0). \tag{23}$$

Value gradient methods use a recursive equation that computes $\nabla V(s)$ using $\nabla V(s')$ 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 (23) is applied, so that the resulting estimate of ∇J can be used to update the policy. The recursive formulae for $\nabla V(s)$ are based on the differentiated Bellman equation:

$$\nabla V = \nabla \int_a d\pi(a|s) \left(R(s, a) + \gamma \int_{s'} p(s'|a, s) V(s') \right). \tag{24}$$

Different value gradient methods differ in the form of the recursive update for the value gradient obtained from (24). For example, *stochastic value gradients* (SVG) introduce a reparameterisation both of π and $p(s'|a, s)$:

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

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 (24) using the reparameterisation as follows:

$$\nabla V = \nabla \int_\eta d\mathcal{B}_2(\eta) \left(R(s, h(s, \eta)) + \gamma \int_{s'} d\mathcal{B}_1(\xi) V(f(h(s, \eta), s, \xi)) \right) =$$
$$= \int_\eta d\mathcal{B}_2(\eta) \left(\nabla R(s, h(s, \eta)) + \gamma \int_\xi d\mathcal{B}_1(\xi) \nabla V(\underbrace{f(h(s, \eta), s, \xi)}_{s'}) \right). \tag{25}$$

Here, the quantities $\nabla R(s, h(s, \eta))$ and $\nabla V(f(h(s, \eta), s, \xi))$ can be computed by the chain rule from the known reward model R, transition model f. SVG learns the approximate model $\hat{f}, \hat{R}, \hat{\xi}, \hat{\eta}$ from samples and using a sample-based approximation to (25) 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 reparameterised policy[15], starting with (24).

$\nabla V(s) = \nabla \int_a d\pi(a|s) \left(R(s, a) + \gamma \int_{s'} p(s'|a, s) V(s') \right)$
$\quad = \int_a da \nabla \pi(a|s) R(s, a) + \gamma \nabla \pi(a|s) \left(\int_{s'} p(s'|a, s) V(s') \right) + \pi(a|s) \nabla \left(\int_{s'} p(s'|a, s) V(s') \right)$
$\quad = \int_{a, s'} \pi(a|s) p(s'|a, s) \left(\nabla \log \pi(a|s) R(s, a) + \nabla V(s') + \nabla \log \pi(a|s) V(s') \right). \tag{26}$

Now (26) can be approximated from samples:

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

---

15. SVG(∞) and SVG(1) require a model an a policy reparameterisation while SVG(0) requires only a policy reparameterisation. However, SVG(0) is inefficient since it does not directly use the reward in the computation of the value gradient.

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

Table 2: 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 $[0, \infty]$.

Here, the pair (a, s') corresponds to the action taken at s and 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 $I_G(s) = \nabla V(s) - \gamma \int_{s'} dp_\pi(s' \mid s) \nabla V(s')$, we can think of this theorem as showing how to obtain a policy gradient from a value gradient without backwards iteration.

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 summarised in Table 2). 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 also discussed the connections between EPG and other common RL techniques, notably sarsa, Q-learning and entropy regularisation. Finally, we evaluated the GPG algorithm in six practical domains, showing that it outperforms existing techniques.

Acknowledgments

Appendix A.

A.1 Proofs and Detailed Definitions

First, we prove two lemmas concerning the discounted-ergodic measure $\rho(s)$ which have been implicitly realised for some time but as far as we could find, never proved explicitly.

Definition 12 (Time-dependent occupancy)

$$p(s \mid t = 0) = p_0(s)$$

$$p(s' \mid t = i + 1) = \int_s p(s' \mid s)p(s \mid t = i) \quad for \quad i \geq 0$$

Definition 13 (Truncated trajectory) *Define the trajectory truncated after N steps as* $\tau^N = (s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_N)$.

Observation 14 (Expectation wrt. 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 that*

$$\mathbb{E}_{\tau_N} \left[\sum_{i=0}^{N} \gamma^i f(s_i) \right] =$$

$$= \int_{s_0, s_1, \ldots, s_N} \left(\prod_{i=0}^{N-1} p(s_{i+1} \mid s_i) \right) p_0(s_0) \left(\sum_{i=0}^{N} \gamma^i f(s_i) \right) ds_0 ds_1 \ldots ds_N =$$

$$= \sum_{i=0}^{N} \int_{s_0, s_1, \ldots, s_N} \left(p_0(s_0) \prod_{i=0}^{N-1} p(s_{i+1} \mid s_i) \right) \gamma^i f(s_i) ds_0 ds_1 \ldots ds_N =$$

$$= \sum_{i=0}^{N} \int_s p(s \mid t = i)\gamma^i f(s) ds$$

for any function f.

Definition 15 (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 16 (Property of expectation with respect to infinite trajectory)

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

$$= \lim_{N \to \infty} \sum_{i=0}^{N} \int_s p(s \mid t = i)\gamma^i f(s) ds =$$

$$= \sum_{i=0}^{\infty} \int_s dp(s \mid t = i)\gamma^i f(s)$$

for any bounded function f.

Definition 17 (Discounted-ergodic occupancy measure ρ)

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

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

Lemma 18 (Discounted-ergodic property) *For any bounded function f:*

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

Proof

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

Here, the first equality follows from Observation 16. ∎

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 19 (Generalised eigenfunction property) *For any bounded function f:*

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

Proof

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

Here, the first equality follows form definition 17, the second one from definition 12. The last equality follows again from definition 17. ∎

Definition 20 (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 marginalising 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 21 (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*[16]. *Denote the value function of the MRP as* V. *Denote the second moment function* S *as*

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

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*

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

Proof

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

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 22 (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. ∎

A.2 Computation of Moments for an Exponential Family

Consider the moment generating function of $T(a)$, which we denote as M_T, for the exponential family of the form given in Equation 15.

$$M_T(v) = e^{U_{v+\eta_\theta} - U_{\eta_\theta}}$$

---

16. 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.

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

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

Here, we denoted as K the size of the sufficient statistic (i.e. the length of the vector $T(a)$). However, we seek the cross-moments of a, not $T(a)$. If $T(a)$ 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 $\pi'(a \mid s) = e^{\eta'^{\top}_{\theta}T'(a)-U_{\eta_{\theta}}+W(a)}$., where T' is a vector concatenation of T and a. We can then use the MGF of $T'(a)$, restricted to a suitable set of indices, to get the moments.

References

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.

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 0132306379.

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.

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.

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.

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.

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 Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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.

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, Remi Munos, Koray Kavukcuoglu, and Volodymyr Mnih. Combining policy gradient and q-learning. 2016.

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.

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, 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, 2000.

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. URL http://www.cs.ox.ac.uk/people/shimon.whiteson/pubs/vanseijenadprl09.pdf.

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.