Smoothed Action Value Functions for Learning Gaussian Policies

Ofir Nachum [1]   Mohammad Norouzi [1]   George Tucker [1]   Dale Schuurmans [1 2]

Abstract

State-action value functions (*i.e.,* Q-values) are ubiquitous in reinforcement learning (RL), giving rise to popular algorithms such as SARSA and Q-learning. We propose a new notion of action value defined by a Gaussian smoothed version of the expected Q-value. We show that such smoothed Q-values still satisfy a Bellman equation, making them learnable from experience sampled from an environment. Moreover, the gradients of expected reward with respect to the mean and covariance of a parameterized Gaussian policy can be recovered from the gradient and Hessian of the smoothed Q-value function. Based on these relationships, we develop new algorithms for training a Gaussian policy directly from a learned smoothed Q-value approximator. The approach is additionally amenable to proximal optimization by augmenting the objective with a penalty on KL-divergence from a previous policy. We find that the ability to learn both a mean and covariance during training leads to significantly improved results on standard continuous control benchmarks.

1. Introduction

Model-free reinforcement learning algorithms often alternate between two concurrent but interacting processes: (1) *policy evaluation*, where an *action value function* (*i.e.,* a Q-value) is updated to obtain a better estimate of the return associated with taking a specific action, and (2) *policy improvement*, where the policy is updated aiming to maximize the current value function. In the past, different notions of Q-value have led to distinct but important families of RL methods. For example, SARSA (Rummery & Niranjan, 1994; Sutton & Barto, 1998; Van Seijen et al., 2009) uses the *expected* Q-value, defined as the expected return of following the current policy. Q-learning (Watkins, 1989) exploits a

hard-max notion of Q-value, defined as the expected return of following an optimal policy. Soft Q-learning (Haarnoja et al., 2017) and PCL (Nachum et al., 2017) both use a *soft-max* form of Q-value, defined as the future return of following an optimal entropy regularized policy. Clearly, the choice of Q-value function has a considerable effect on the resulting algorithm; for example, restricting the types of policies that can be expressed, and determining the type of exploration that can be naturally applied. In each case, the Q-value at a state s and action a answers the question,

"What would my future value from s be if I were to take an initial action a?"

Such information about a hypothetical action is helpful when learning a policy; we want to nudge the policy distribution to favor actions with potentially higher Q-values.

In this work, we investigate the practicality and benefits of answering a more difficult, but more relevant, question:

"What would my future value from s be if I were to sample my initial action from a distribution centered at a?"

We focus our efforts on Gaussian policies and thus the counterfactual posited by the Q-value inquires about the expected future return of following the policy when changing the mean of the initial Gaussian distribution. Thus, our new notion of Q-values maps a state-action pair (s, a) to the expected return of first taking an action sampled from a normal distribution $N(\cdot|a, \Sigma(s))$ centered at a, and following actions sampled from the current policy thereafter. In this way, the Q-values we introduce may be interpreted as a Gaussian-smoothed version of the expected Q-value, hence we term them *smoothed* Q-values.

We show that smoothed Q-values possess a number of important properties that make them attractive for use in RL algorithms. It is clear from the definition of smoothed Q-values that, if known, their structure is highly beneficial for learning the mean of a Gaussian policy. We are able to show more than this: although the smoothed Q-values are not a direct function of the covariance, one can surprisingly use knowledge of the smoothed Q-values to derive updates to the covariance of a Gaussian policy. Specifically, the gradient of the standard expected return objective with respect to the mean and covariance of a Gaussian policy is equivalent to the gradient and Hessian of the smoothed Q-value func-

---

[1]Google Brain [2]Department of Computing Science, University of Alberta. Correspondence to: Ofir Nachum <ofirnachum@google.com>.

tion, respectively. Moreover, we show that the smoothed Q-values satisfy a single-step Bellman consistency, which allows bootstrapping to be used to train them via function approximation.

These results lead us to propose an algorithm, *Smoothie*, which, in the spirit of (Deep) Deterministic Policy Gradient (DDPG) (Silver et al., 2014; Lillicrap et al., 2016), trains a policy using the derivatives of a trained (smoothed) Q-value function to learn a Gaussian policy. Crucially, unlike DDPG, which is restricted to deterministic policies and is well-known to have poor exploratory behavior (Haarnoja et al., 2017), the approach we develop is able to utilize a non-deterministic Gaussian policy parameterized by both a mean and a covariance, thus allowing the policy to be exploratory by default and alleviating the need for excessive hyperparameter tuning. On the other hand, compared to standard policy gradient algorithms (Williams & Peng, 1991; Konda & Tsitsiklis, 2000), Smoothie's utilization of the derivatives of a Q-value function to train a policy avoids the high variance and sample inefficiency of stochastic updates.

Furthermore, we show that Smoothie can be easily adapted to incorporate proximal policy optimization techniques by augmenting the objective with a penalty on KL-divergence from a previous version of the policy. The inclusion of a KL-penalty is not feasible in the standard DDPG algorithm, but we show that it is possible with our formulation, and it significantly improves stability and overall performance. On standard continuous control benchmarks, our results are competitive with or exceed state-of-the-art, especially for more difficult tasks in the low-data regime.

2. Formulation

We consider the standard model-free RL problem represented a Markov decision process (MDP), consisting of a state space $\mathcal{S}$ and an action space $\mathcal{A}$. At iteration t the agent encounters a state $s_t \in \mathcal{S}$ and emits an action $a_t \in \mathcal{A}$, after which the environment returns a scalar reward $r_t \sim R(s_t, a_t)$ and places the agent in a new state $s_{t+1} \sim P(s_t, a_t)$.

We focus on continuous control tasks, where the actions are real-valued, *i.e.,* $\mathcal{A} \equiv \mathbb{R}^{d_a}$. Our observations at a state s are denoted $\Phi(s) \in \mathbb{R}^{d_s}$. We parameterize the behavior of the agent using a stochastic policy $\pi(a \,|\, s)$, which takes the form of a Gaussian density at each state s. The Gaussian policy is parameterized by a mean and a covariance function, $\mu(s) : \mathbb{R}^{d_s} \to \mathbb{R}^{d_a}$ and $\Sigma(s) : \mathbb{R}^{d_s} \to \mathbb{R}^{d_a} \times \mathbb{R}^{d_a}$ so that $\pi(a \,|\, s) = \mathcal{N}(a \,|\, \mu(s), \Sigma(s))$, where

$$N(a \,|\, \mu, \Sigma) \;=\; |2\pi\Sigma|^{-1/2} \exp\left\{-\frac{1}{2}\|a - \mu\|^2_{\Sigma^{-1}}\right\}, \quad (1)$$

here using the notation $\|v\|^2_A = v^\mathsf{T} A v$.

2.1. Policy Gradient for Generic Stochastic Policies

The optimization objective (expected discounted return), as a function of a generic stochastic policy, is expressed in terms of the expected action value function $Q^\pi(s, a)$ by,

$$O_{\mathrm{ER}}(\pi) \;=\; \int_{\mathcal{S}} \int_{\mathcal{A}} \pi(a \,|\, s) Q^\pi(s, a) \,\mathrm{d}a \,\mathrm{d}\rho^\pi(s) \,, \quad (2)$$

where $\rho^\pi(s)$ is the state visitation distribution under π, and $Q^\pi(s, a)$ is recursively defined using the Bellman equation,

$$Q^\pi(s, a) = \mathbb{E}_{r,s'}\left[r + \gamma \int_{\mathcal{A}} Q^\pi(s', a')\pi(a' \,|\, s') \,\mathrm{d}a'\right] \,, \quad (3)$$

where $\gamma \in [0, 1)$ is the discount factor. For brevity, we suppress explicit denotation of the distribution R over immediate rewards and P over state transitions.

The policy gradient theorem (Sutton et al., 2000) expresses the gradient of $O_{\mathrm{ER}}(\pi_\theta)$ *w.r.t.* θ, the tunable parameters of a policy π_θ, as,

$$\nabla_\theta O_{\mathrm{ER}}(\pi_\theta) = \int_{\mathcal{S}} \int_{\mathcal{A}} \nabla_\theta \pi_\theta(a \,|\, s) Q^\pi(s, a) \,\mathrm{d}a \,\mathrm{d}\rho^\pi(s)$$

$$= \int_{\mathcal{S}} \mathbb{E}_{a \sim \pi_\theta(a|s)}\left[\nabla_\theta \log \pi_\theta(a \,|\, s) Q^\pi(s, a)\right] \mathrm{d}\rho^\pi(s). \quad (4)$$

In order to approximate the expectation on the RHS of (4), one often resorts to an empirical average over on-policy samples from $\pi_\theta(a \,|\, s)$. This sampling scheme results in a gradient estimate with high variance, especially when $\pi_\theta(a \,|\, s)$ is not concentrated. Many policy gradient algorithms, including actor-critic variants, trade off variance and bias, *e.g.,* by attempting to estimate $Q^\pi(s, a)$ accurately using function approximation and the Bellman equation.

2.2. Deterministic Policy Gradient

Silver et al. (2014) study the policy gradient for the specific class of Gaussian policies in the limit where the policy's covariance approaches zero. In this scenario, the policy becomes deterministic and samples from the policy approach the Gaussian mean. Under a deterministic policy $\pi \equiv (\mu, \Sigma \to 0)$, one can estimate the expected future return from a state s as,

$$\lim_{\Sigma \to 0} \int_{\mathcal{A}} \pi(a \,|\, s) Q^\pi(s, a) \,\mathrm{d}a \;=\; Q^\pi(s, \mu(s)) \,. \quad (5)$$

Accordingly, Silver et al. (2014) express the gradient of the expected discounted return objective for $\pi_\theta \equiv \delta(\mu_\theta)$ as,

$$\nabla_\theta O_{\mathrm{ER}}(\pi_\theta) = \int_{\mathcal{S}} \frac{\partial Q^\pi(s, a)}{\partial a}\Big|_{a=\mu_\theta(s)} \nabla_\theta \mu_\theta(s) \mathrm{d}\rho^\pi(s). \quad (6)$$

This characterization of the policy gradient theorem for deterministic policies is called *deterministic policy gradient*

(DPG). Since no Monte Carlo sampling is required for estimating the gradient, the variance of the estimate is reduced. On the other hand, the deterministic nature of the policy can lead to poor exploration and training instability in practice.

In the limit of $\Sigma \to 0$, one can also re-express the Bellman equation (3) as,

$$Q^\pi(s, a) = \mathbb{E}_{r,s'} \left[r + Q^\pi(s', \mu(s')) \right] . \tag{7}$$

Therefore, a value function approximator Q_w^π can be optimized by minimizing the Bellman error,

$$E(w) = \sum_{(s,a,r,s') \in \mathcal{D}} (Q_w^\pi(s,a) - r - \gamma Q_w^\pi(s', \mu(s')))^2 , \tag{8}$$

for transitions (s, a, r, s') sampled from a dataset $\mathcal{D}$ of interactions of the agent with the environment. The deep variant of DPG known as DDPG (Lillicrap et al., 2016) alternates between improving the action value estimate by gradient descent on (8) and improving the policy based on (6).

To improve sample efficiency, Degris et al. (2012) and Silver et al. (2014) replace the state visitation distribution $\rho^\pi(s)$ in (6) with an off-policy visitation distribution $\rho^\beta(s)$ based on a *replay buffer*. This subsitition introduces some bias in the gradient estimate (6), but previous work has found that it works well in practice and improves the sample efficiency of the policy gradient algorithms. We also adopt a similar heuristic in our method to make use of off-policy data.

In practice, DDPG exhibits improved sample efficiency over standard policy gradient algorithms: using off-policy data to train Q-values while basing policy updates on their gradients significantly improves stochastic policy updates dictated by (4), which require a large number of samples to reduce noise. On the other hand, the deterministic nature of the policy learned by DDPG leads to poor exploration and instability in training. In this paper, we propose an algorithm which, like DDPG, utilizes derivative information of learned Q-values for better sample-efficiency, but which, unlike DDPG, is able to learn a Gaussian policy and imposes a KL-penalty for better exploration and stability.

3. Idea

Before giving a full exposition, we use a simplified scenario to illustrate the key intuitions behind the proposed approach and how it differs fundamentally from previous methods.

Consider a one-shot decision making problem over a one dimensional action space with a single state. Here the expected reward is given by a function over the real line, which also corresponds to the optimal Q-value function; Figure 1 gives a concrete example. We assume the policy π is specified by a Gaussian distribution parameterized by a scalar mean μ and standard deviation σ. The goal is to optimize the policy parameters to maximize expected reward.
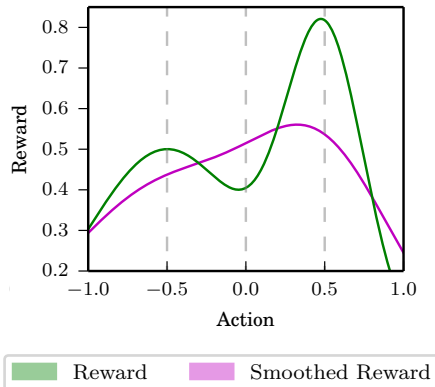

Figure 1. A simple expected reward function, shown in green, with a Gaussian-smoothed version, shown in magenta.

A naive policy gradient method updates the parameters by sampling $a \sim \pi$, observing reward r_i, then adjusting μ and σ in directions $\Delta\mu = \frac{d \log \pi(a_i)}{d\mu} r_i = \frac{\mu - a_i}{\sigma^2} r_i$ and $\Delta\sigma = \frac{d \log \pi(a_i)}{d\sigma} r_i = \left(\frac{(\mu - a_i)^2}{\sigma^2} - \frac{1}{\sigma} \right) r_i$. Note that such updates suffer from large variance, particularly when σ is small.

To reduce the variance of direct policy gradient, deterministic policy gradient methods leverage a value function approximator Q_w^π, parameterized by w, to approximate Q^π. For example, in this scenario, vanilla DPG would sample an action $a_i = \mu + \varepsilon_i$ with exploration noise $\varepsilon_i \sim N(0, \sigma^2)$, then update μ using $\Delta\mu = \frac{\partial Q_w^\pi(a)}{\partial a} \big|_{a=\mu}$ and Q_w^π using $\Delta w = (r_i - Q_w^\pi(a_i)) \nabla_w Q_w^\pi(a_i)$. Clearly, this update exhibits reduced variance, but requires Q_w^π to approximate Q^π (the green curve in Figure 1) to control bias. Unfortunately, DPG is not able to learn the exploration variance σ^2. Variants of DPG such as SVG (Heess et al., 2015) and EPG (Ciosek & Whiteson, 2018) have been proposed to work with stochastic policies. However they either have restrictive assumptions on the form of the true Q-value, introduce a noise into the policy updates, or require an approximate integral, thus losing the advantage of deterministic gradient updates.

Note, however, that the expected value at any given location is actually given by a *convolution* of the Gaussian policy with the underlying expected reward function. Such a process inherently smooths the landscape, as shown in the magenta curve in Figure 1. Unfortunately, DPG completely ignores this smoothing effect by trying to approximate Q^π, while policy gradient methods only benefit from it indirectly through sampling. A key insight is that this smoothing effect can be captured directly in the value function approximator itself, bypassing any need for sampling or approximating Q^π. That is, instead of using an approximator to model Q^π, one can directly approximate the smoothed version given by $\tilde{Q}^\pi(a) = \int_{\mathcal{A}} N(\tilde{a}|a, \sigma^2) Q^\pi(\tilde{a}) \, d\tilde{a}$ (the magenta curve in Figure 1), which, crucially, satisfies $O_{ER}(\pi) = \tilde{Q}^\pi(\mu)$.

Based on this observation, we propose a novel actor-critic strategy below that uses a function approximator $\tilde{Q}_w^\pi$ to model $\tilde{Q}^\pi$. Although approximating $\tilde{Q}^\pi$ instead of Q^π might appear to be a subtle change, it is a major alteration to existing actor-critic approaches. Not only is approximating the magenta curve in Figure 1 far easier than the green curve, modeling $\tilde{Q}^\pi$ allows the policy parameters to be updated *deterministically* for any given action. In particular, in the simple scenario above, if one sampled an action from the current policy, $a_i \sim \pi$, and observed r_i, then μ could be updated using $\Delta\mu = \frac{\partial \tilde{Q}_w^\pi(a)}{\partial a}\big|_{a=\mu}$, σ using $\Delta\sigma = \frac{\partial^2 \tilde{Q}_w^\pi(a)}{\partial a^2}\big|_{a=\mu}$ (a key result we establish below), and $\tilde{Q}_w^\pi$ using $\Delta w = (r_i - \tilde{Q}_w^\pi(\mu))\nabla_w \tilde{Q}_w^\pi(\mu)$.

Such a strategy combines the best aspects of DPG and policy gradient while conferring additional advantages: (1) the smoothed value function $\tilde{Q}^\pi$ cannot add but can only remove local minima from Q^π; (2) $\tilde{Q}^\pi$ is smoother than Q^π hence easier to approximate; (3) approximating $\tilde{Q}^\pi$ allows deterministic gradient updates for π; (4) approximating $\tilde{Q}^\pi$ allows gradients to be computed for both the mean and variance parameters. Among these advantages, DPG shares only 3 and policy gradient only 1. We will see below that the new strategy we propose significantly outperforms existing approaches, not only in the toy scenario depicted in Figure 1, but also in challenging benchmark problems.

4. Smoothed Action Value Functions

Moving beyond a simple illustrative scenario, the key contribution of this paper is to introduce the general notion of a *smoothed action value function*, the gradients of which provide an effective signal for optimizing the parameters of a Gaussian policy. Smoothed Q-values, which we denote $\tilde{Q}^\pi(s, a)$, differ from ordinary Q-values $Q^\pi(s, a)$ by not assuming the first action of the agent is fully specified; instead, they assume only that a Gaussian centered at the action is known. Thus, to compute $\tilde{Q}^\pi(s, a)$, one has to perform an expectation of $Q^\pi(s, \tilde{a})$ for actions $\tilde{a}$ drawn in the vicinity of a. More formally, smoothed action values are defined as,

$$\tilde{Q}^\pi(s, a) = \int_{\mathcal{A}} N(\tilde{a} \,|\, a, \Sigma(s))\, Q^\pi(s, \tilde{a})\, \mathrm{d}\tilde{a}\,. \quad (9)$$

With this definition of $\tilde{Q}^\pi$, one can re-express the gradient of the expected reward objective (Equation (4)) for a Gaussian policy $\pi \equiv (\mu, \Sigma)$ as,

$$\nabla_{\mu,\Sigma}\, O_{\mathrm{ER}}(\pi) = \int_{\mathcal{S}} \nabla_{\mu,\Sigma}\, \tilde{Q}^\pi(s, \mu(s))\, \mathrm{d}\rho^\pi(s)\,. \quad (10)$$

The insight that differentiates this approach from prior work (Heess et al., 2015; Ciosek & Whiteson, 2018) is that instead of learning a function approximator for Q^π then drawing samples to approximate the expectation in (9) and its derivative, we directly learn a function approximator for $\tilde{Q}^\pi$.

One of the key observations that enables learning a function approximator for $\tilde{Q}^\pi$ is that smoothed Q-values satisfy a notion of Bellman consistency. First, note that for Gaussian policies $\pi \equiv (\mu, \Sigma)$ we have the following relation between the expected and smoothed Q-values:

$$Q^\pi(s, a) = \mathbb{E}_{r,s'}[r + \gamma \tilde{Q}^\pi(s', \mu(s'))]\,. \quad (11)$$

Then, combining (9) and (11), one can derive the following one-step Bellman equation for smoothed Q-values,

$$\tilde{Q}^\pi(s, a) = \int_{\mathcal{A}} N(\tilde{a} \,|\, a, \Sigma(s))\, \mathbb{E}_{\tilde{r},\tilde{s}'}\Big[\tilde{r} + \gamma \tilde{Q}^\pi(\tilde{s}', \mu(\tilde{s}'))\Big]\, \mathrm{d}\tilde{a}, \quad (12)$$

where $\tilde{r}$ and $\tilde{s}'$ are sampled from $R(s, \tilde{a})$ and $P(s, \tilde{a})$. Below, we elaborate on how one can make use of the derivatives of $\tilde{Q}^\pi$ to learn μ and Σ, and how the Bellman equation in (12) enables direct optimization of $\tilde{Q}^\pi$.

4.1. Policy Improvement

We assume a Gaussian policy $\pi_{\theta,\phi} \equiv (\mu_\theta, \Sigma_\phi)$ parameterized by θ and ϕ for the mean and the covariance respectively. The gradient of the objective *w.r.t.* the mean parameters follows from the policy gradient theorem in conjunction with (10) and is almost identical to (6),

$$\nabla_\theta O_{\mathrm{ER}}(\pi_{\theta,\phi}) = \int_{\mathcal{S}} \frac{\partial \tilde{Q}^\pi(s, a)}{\partial a}\Big|_{a=\mu_\theta(s)} \nabla_\theta \mu_\theta(s) \mathrm{d}\rho^\pi(s). \quad (13)$$

Estimating the derivative of the objective *w.r.t.* the covariance parameters is not as straightforward, since $\tilde{Q}^\pi$ is not a direct function of Σ. However, a key result is that the second derivative of $\tilde{Q}^\pi$ *w.r.t.* actions is sufficient to exactly compute the derivative of $\tilde{Q}^\pi$ *w.r.t.* Σ.

Theorem 1. $\quad \dfrac{\partial \tilde{Q}^\pi(s, a)}{\partial \Sigma(s)} = \dfrac{1}{2} \cdot \dfrac{\partial^2 \tilde{Q}^\pi(s, a)}{\partial a^2} \quad \forall s, a\,. \quad (14)$

A proof of this identity is provided in the Appendix. The full derivative *w.r.t.* ϕ can then be shown to take the form,

$$\nabla_\phi O_{\mathrm{ER}}(\pi_{\theta,\phi}) = \frac{1}{2}\int_{\mathcal{S}} \frac{\partial^2 \tilde{Q}^\pi(s, a)}{\partial a^2}\Big|_{a=\mu_\theta(s)} \nabla_\phi \Sigma_\phi(s) \mathrm{d}\rho^\pi(s). \quad (15)$$

4.2. Policy Evaluation

There are two natural ways to optimize $\tilde{Q}_w^\pi$. The first approach leverages (9) to update $\tilde{Q}^\pi$ based on the expectation of Q^π. In this case, one first trains a parameterized model Q_w^π to approximate the standard Q^π function using conventional methods (Rummery & Niranjan, 1994; Sutton & Barto, 1998; Van Seijen et al., 2009), then fits $\tilde{Q}_w^\pi$ to Q_w^π based on (9). In particular, given transitions (s, a, r, s') sampled from interactions with the environment, one can train

Q_w^π to minimize the Bellman error

$$(Q_w^\pi(s,a) - r - \gamma Q_w^\pi(s',a'))^2, \qquad (16)$$

where $a' \sim N(\mu(s'), \Sigma(s'))$. Then, $\tilde{Q}_w^\pi$ can be optimized to minimize the squared error

$$(\tilde{Q}_w^\pi(s,a) - \mathbb{E}_{\tilde{a}}[Q_w^\pi(s,\tilde{a})])^2, \qquad (17)$$

where $\tilde{a} \sim N(a, \Sigma(s))$, using several samples. When the target values in the residuals are treated as fixed (*i.e.,* using a target network), these updates will reach a fixed point when $\tilde{Q}_w^\pi$ satisfies the recursion in the Bellman equation (9).

The second approach requires a single function approximator for $\tilde{Q}_w^\pi$, resulting in a simpler implementation; hence we use this approach in our experimental evaluation. Suppose one has access to a tuple $(s, \tilde{a}, \tilde{r}, \tilde{s}')$ sampled from a replay buffer with knowledge of the sampling probability $q(\tilde{a} \,|\, s)$ (possibly unnormalized) with full support. Then we draw a *phantom* action $a \sim N(\tilde{a}, \Sigma(s))$ and optimize $\tilde{Q}_w^\pi(s,a)$ by minimizing a weighted Bellman error

$$\frac{1}{q(\tilde{a}|s)}(\tilde{Q}_w^\pi(s,a) - \tilde{r} - \gamma\tilde{Q}_w^\pi(\tilde{s}', \mu(\tilde{s}'))^2 . \qquad (18)$$

In this way, for a specific pair of state and action (s,a) the expected objective value is,

$$\mathbb{E}_{q(\tilde{a}\,|\,s),\tilde{r},\tilde{s}'}\left[\delta \cdot (\tilde{Q}_w^\pi(s,a) - \tilde{r} - \gamma\tilde{Q}_w^\pi(\tilde{s}', \mu(\tilde{s}')))^2\right], \quad (19)$$

where $\delta = \frac{N(a\,|\,\tilde{a},\Sigma(s))}{q(\tilde{a}\,|\,s)}$. Note that the denominator of δ counter-acts the expectation over $\tilde{a}$ in (19) and that the numerator of δ is $N(a|\tilde{a},\Sigma(s)) = N(\tilde{a}|a,\Sigma(s))$. Therefore, when the target value $\tilde{r} + \gamma\tilde{Q}_w^\pi(\tilde{s}', \mu(\tilde{s}'))$ is treated as fixed (*i.e.,* using a target network) this training procedure reaches an optimum when $\tilde{Q}_w^\pi(s,a)$ takes on the target value provided in the Bellman equation (12).

In practice, we find that it is unnecessary to keep track of the probabilities $q(\tilde{a} \,|\, s)$, and assume the replay buffer provides a near-uniform distribution of actions conditioned on states. Other recent work has also benefited from ignoring or heavily damping importance weights (Munos et al., 2016; Wang et al., 2017; Schulman et al., 2017). However, it is possible when interacting with the environment to save the probability of sampled actions along with their transitions, and thus have access to $q(\tilde{a} \,|\, s) \approx N(\tilde{a} \,|\, \mu_{\text{old}}(s), \Sigma_{\text{old}}(s))$.

4.3. Proximal Policy Optimization

Policy gradient algorithms are notoriously unstable, particularly in continuous control problems. Such instability has motivated the development of trust region methods that constrain each gradient step to lie within a trust region (Schulman et al., 2015), or augment the expected reward objective with a penalty on KL-divergence from a previous policy (Nachum et al., 2018; Schulman et al., 2017). These

---

Algorithm 1 Smoothie

Input: Environment ENV, learning rates η_π, η_Q, discount factor γ, KL-penalty λ, batch size B, number of training steps N, target network lag τ.

Initialize θ, ϕ, w, set $\theta' = \theta, \phi' = \phi, w' = w$.
for $i = 0$ **to** $N - 1$ **do**
    // Collect experience
    Sample action $a \sim N(\mu_\theta(s), \Sigma_\phi(s))$ and apply to ENV to yield r and s'.
    Insert transition (s, a, r, s') to replay buffer.

    // Train μ, Σ
    Sample batch $\{(s_k, a_k, r_k, s'_k)\}_{k=1}^B$ from replay buffer.
    Compute gradients $g_k = \frac{\partial\tilde{Q}_w^\pi(s_k,a)}{\partial a}\big|_{a=\mu_\theta(s_k)}$.
    Compute Hessians $H_k = \frac{\partial^2\tilde{Q}_w^\pi(s_k,a)}{\partial a^2}\big|_{a=\mu_\theta(s_k)}$.
    Compute penalties $KL_k = KL(\mu_\theta, \Sigma_\phi || \mu_{\theta'}, \Sigma_{\phi'})$.
    Compute updates
      $\Delta\theta = \frac{1}{B}\sum_{k=1}^B g_k\nabla_\theta\mu_\theta(s_k) - \lambda\nabla_\theta KL_k$,
      $\Delta\phi = \frac{1}{B}\sum_{k=1}^B \frac{1}{2}H_k\nabla_\phi\Sigma_\phi(s_k) - \lambda\nabla_\phi KL_k$.
    Update $\theta \leftarrow \theta + \eta_\pi\Delta\theta$, $\phi \leftarrow \phi + \eta_\pi\Delta\phi$.

    // Train $\tilde{Q}^\pi$
    Sample batch $\{(s_k, \tilde{a}_k, \tilde{r}_k, \tilde{s}'_k)\}_{k=1}^B$ from replay buffer.
    Sample phantom actions $a_k \sim N(\tilde{a}_k, \Sigma_\phi(s_k))$.
    Compute loss
    $\mathcal{L}(w) = \frac{1}{B}\sum_{k=1}^B(\tilde{Q}_w^\pi(s,a) - r - \gamma\tilde{Q}_{w'}^\pi(\tilde{s}', \mu_{\theta'}(\tilde{s}')))^2$.
    Update $w \leftarrow w - \eta_Q\nabla_w\mathcal{L}(w)$.

    // Update target variables
    Update $\theta' \leftarrow (1-\tau)\theta' + \tau\theta$; $\phi' \leftarrow (1-\tau)\phi' + \tau\phi$;
    $w' \leftarrow (1-\tau)w' + \tau w$.
end for

---

stabilizing techniques have thus far not been applicable to algorithms like DDPG, since the policy is deterministic. The formulation we propose above, however, is easily amenable to trust region optimization. Specifically, we may augment the objective (10) with a penalty

$$O_{\text{TR}}(\pi) = O_{\text{ER}}(\pi) - \lambda\int_{\mathcal{S}} KL\left(\pi \,\|\, \pi_{\text{old}}\right) d\rho^\pi(s), \quad (20)$$

where $\pi_{\text{old}} \equiv (\mu_{\text{old}}, \Sigma_{\text{old}})$ is a previous version of the policy. The optimization is straightforward, since the KL-divergence of two Gaussians can be expressed analytically.

This concludes the technical presentation of the proposed algorithm *Smoothie*: pseudocode for the full training procedure, including policy improvement, policy evaluation, and proximal policy improvement is given in Algorithm 1. The reader may also refer to the appendix for additional implementation details.

4.4. Compatible Function Approximation

The approximator $\tilde{Q}_w^\pi$ for $\tilde{Q}^\pi$ should be sufficiently accurate so that updates for μ_θ and Σ_ϕ are not affected by substituting $\frac{\partial \tilde{Q}_w^\pi(s,a)}{\partial a}$ and $\frac{\partial^2 \tilde{Q}_w^\pi(s,a)}{\partial a^2}$ for $\frac{\partial \tilde{Q}^\pi(s,a)}{\partial a}$ and $\frac{\partial^2 \tilde{Q}^\pi(s,a)}{\partial a^2}$ respectively. Define $\epsilon_k(s,\theta,w)$ as the difference between the true k-th derivative of $\tilde{Q}^\pi$ and the k-th derivative of the approximated $\tilde{Q}_w^\pi$ at $a = \mu_\theta(s)$:

$$\epsilon_k(s,\theta,w) = \nabla_a^k \tilde{Q}_w^\pi(s,a)\big|_{a=\mu_\theta(s)} - \nabla_a^k \tilde{Q}^\pi(s,a)\big|_{a=\mu_\theta(s)}. \tag{21}$$

We claim that a $\tilde{Q}_w^\pi$ is compatible with respect to μ_θ if

1. $\nabla_a \tilde{Q}_w^\pi(s,a)\big|_{a=\mu_\theta(s)} = \nabla_\theta \mu_\theta(s)^T w$,

2. $\nabla_w \int_{\mathcal{S}} ||\epsilon_1(s,\theta,w)||^2 d\rho^\pi(s) = 0$ (*i.e.*, w minimizes the expected squared error of the gradients).

Additionally, $\tilde{Q}_w^\pi$ is compatible with respect to Σ_ϕ if

1. $\nabla_a^2 \tilde{Q}_w^\pi(s,a)\big|_{a=\mu_\theta(s)} = \nabla_\phi \Sigma_\phi(s)^T w$,

2. $\nabla_w \int_{\mathcal{S}} ||\epsilon_2(s,\theta,w)||^2 d\rho^\pi(s) = 0$ (*i.e.*, w minimizes the expected squared error of the Hessians).

We provide a proof of these claims in the Appendix. One possible parameterization of $\tilde{Q}_w^\pi$ may be achieved by taking $w = [w_0, w_1, w_2]$ and parameterizing

$$\tilde{Q}_w^\pi(s,a) = V_{w_0}(s) + (a - \mu_\theta(s))^T \nabla_\theta \mu_\theta(s)^T w_1$$
$$+ (a - \mu_\theta(s))^T \nabla_\phi \Sigma_\phi(s)^T w_2 (a - \mu_\theta(s)). \tag{22}$$

Similar conditions and parameterizations exist for DDPG (Silver et al., 2014), in terms of Q^π. While it is reassuring to know that there exists a class of function approximators which are compatible, this fact has largely been ignored in practice. At first glance, it seems impossible to satisfy the second set of conditions without access to derivative information of the true $\tilde{Q}^\pi$ (for DDPG, Q^π). Indeed, the methods for training Q-value approximators (equation (8) and Section 4.2) only train to minimize an error between raw scalar values. For DDPG, we are unaware of any method that allows one to train Q_w^π to minimize an error with respect to the derivatives of the true Q^π. However, the case is different for the smoothed Q-values $\tilde{Q}^\pi$. In fact, it is possible to train $\tilde{Q}_w^\pi$ to minimize an error with respect to the derivatives of the true $\tilde{Q}^\pi$. We provide an elaboration in the Appendix. In brief, it is possible to use (12) to derive Bellman-like equations which relate a derivative $\frac{\partial^k \tilde{Q}^\pi(s,a)}{\partial a^k}$ of any degree k to an integral over the raw Q-values at the next time step $\tilde{Q}^\pi(\tilde{s}', \mu(\tilde{s}'))$. Thus, it is possible to devise a training scheme in the spirit of the one outlined in Section 4.2, which optimizes $\tilde{Q}_w^\pi$ to minimize the squared error with the derivatives of the true $\tilde{Q}^\pi$. This theoretical property of the smoothed Q-values is unique and provides added benefits to its use over the standard Q-values.

5. Related Work

This paper follows a long line of work that uses Q-value functions to stably learn a policy, which in the past has been used to either approximate expected (Rummery & Niranjan, 1994; Van Seijen et al., 2009; Gu et al., 2017) or optimal (Watkins, 1989; Silver et al., 2014; Nachum et al., 2017; Haarnoja et al., 2017; Metz et al., 2017) future value.

Work that is most similar to what we present are methods that exploit gradient information from the Q-value function to train a policy. Deterministic policy gradient (Silver et al., 2014) is perhaps the best known of these. The method we propose can be interpreted as a generalization of the deterministic policy gradient. Indeed, if one takes the limit of the policy covariance $\Sigma(s)$ as it goes to 0, the proposed Q-value function becomes the deterministic value function of DDPG, and the updates for training the Q-value approximator and the policy mean are identical.

Stochastic Value Gradient (SVG) (Heess et al., 2015) also trains stochastic policies using an update that is similar to DDPG (*i.e.,* SVG(0) with replay). The key differences with our approach are that SVG does not provide an update for the covariance, and the mean update in SVG estimates the gradient with a noisy Monte Carlo sample, which we avoid by estimating the smoothed Q-value function. Although a covariance update could be derived using the same reparameterization trick as in the mean update, that would also require a noisy Monte Carlo estimate. Methods for updating the covariance along the gradient of expected reward are essential for applying the subsequent trust region and proximal policy techniques.

More recently, Ciosek & Whiteson (2018) introduced expected policy gradients (EPG), a generalization of DDPG that provides updates for the mean and covariance of a stochastic Gaussian policy using gradients of an estimated Q-value function. In that work, the expected Q-value used in standard policy gradient algorithms such as SARSA (Sutton & Barto, 1998; Rummery & Niranjan, 1994; Van Seijen et al., 2009) is estimated. The updates in EPG therefore require approximating an integral of the expected Q-value function, or assuming the Q-value has a simple form that allows for analytic computation. Our analogous process directly estimates an integral (via the smoothed Q-value function) and avoids approximate integrals, thereby making the updates simpler and generally applicable. Moreover, while Ciosek & Whiteson (2018) rely on a quadratic Taylor expansion of the estimated Q-value function, we instead rely on the strength of neural network function approximators to directly estimate the smoothed Q-value function.

The novel training scheme we propose for learning the covariance of a Gaussian policy relies on properties of Gaussian integrals (Bonnet, 1964; Price, 1958). Similar identities
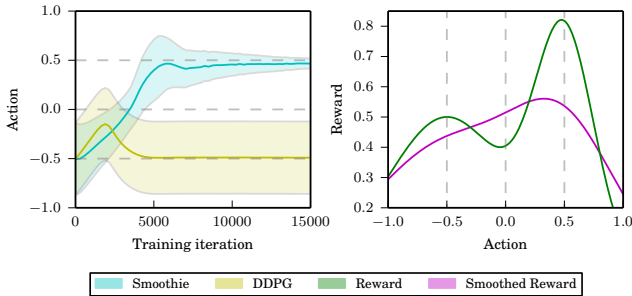
Figure 2. Left: The learnable policy mean and standard deviation during training for Smoothie and DDPG on the simple synthetic task introduced in Section 3. The standard deviation for DDPG is the exploratory noise kept constant during training. Right: Copy of Figure 1 showing the reward function and its Gaussian-smoothed version. Smoothie successfully escapes the lower-reward local optimum, while increasing then decreasing its policy variance as the convexity/concavity of the smoothed reward function changes.

have been used in the past to derive updates for variational auto-encoders (Kingma & Welling, 2014) and Gaussian back-propagation (Rezende et al., 2014).

Finally, the perspective presented in this paper, where Q-values represent the averaged return of a distribution of actions rather than a single action, is distinct from recent advances in distributional RL (Bellemare et al., 2017). Those approaches focus on the distribution of returns of a single action, whereas we consider the single average return of a distribution of actions. Although we restrict our attention in this paper to Gaussian policies, an interesting topic for further investigation is to study the applicability of this new perspective to a wider class of policy distributions.

6. Experiments

We utilize the insights from Section 4 to introduce a new RL algorithm, *Smoothie*. Smoothie maintains a parameterized $\tilde{Q}_w^\pi$ trained via the procedure described in Section 4.2. It then uses the gradient and Hessian of this approximation to train a Gaussian policy $\pi_{\theta,\phi} \equiv (\mu_\theta, \Sigma_\phi)$ using the updates stated in (13) and (15). See Algorithm 1 for a simplified pseudocode of the algorithm.

We perform a number of evaluations of Smoothie to compare to DDPG. We choose DDPG as a baseline because it (1) utilizes gradient information of a Q-value approximator, much like the proposed algorithm; and (2) is a standard algorithm well-known to have achieve good, sample-efficient performance on continuous control benchmarks.

6.1. Synthetic Task

Before investigating benchmark problems, we first briefly revisit the simple task introduced in Section 3 and reproduced in Figure 2 (Right). Here, the reward function is a
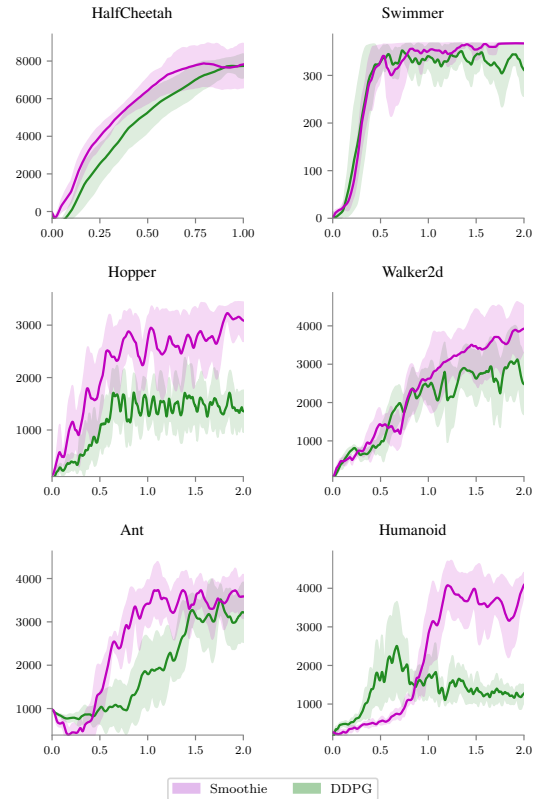

Figure 3. Results of Smoothie and DDPG on continuous control benchmarks. The x-axis is in millions of environment steps. Each plot shows the average reward and standard deviation clipped at the min and max of six randomly seeded runs after choosing best hyperparameters. We see that Smoothie is competitive with DDPG even when DDPG uses a hyperparameter-tuned noise scale, and Smoothie learns the optimal noise scale (the covariance) during training. Moreoever, we observe significant advantages in terms of final reward performance, especially in the more difficult tasks like Hopper, Walker2d, and Humanoid.

mixture of two Gaussians, one better than the other, and we initialize the policy mean to be centered on the worse of the two. We plot the learnable policy mean and standard deviation during training for Smoothie and DDPG in Figure 2 (Left). Smoothie learns both the mean and variance, while DDPG learns only the mean and the variance plotted is the exploratory noise, whose scale is kept fixed during training.

As expected, we observe that DDPG cannot escape the local optimum. At the beginning of training it exhibits some movement away from the local optimum (likely due to the initial noisy approximation given by Q_w^π). However, it is unable to progress very far from the initial mean. Note that this is not an issue of exploration. The exploration scale is high enough that Q_w^π is aware of the better Gaussian. The issue is in the update for μ_θ, which is only with regard to the derivative of Q_w^π at the current mean.

On the other hand, we find Smoothie is easily able to solve the task. This is because the smoothed reward function approximated by $\tilde{Q}_w^\pi$ has a derivative that clearly points μ_θ toward the better Gaussian. We also observe that Smoothie is able to suitably adjust the covariance Σ_ϕ during training. Initially, Σ_ϕ decreases due to the concavity of the smoothed reward function. As a region of convexity is entered, it begins to increase, before again decreasing to near-zero as μ_θ approaches the global optimum. This example clearly shows the ability of Smoothie to exploit the smoothed action value landscape.

6.2. Continuous Control

Next, we consider standard continuous control benchmarks available on OpenAI Gym (Brockman et al., 2016) utilizing the MuJoCo environment (Todorov et al., 2012).

Our implementations utilize feed forward neural networks for policy and Q-values. We parameterize the covariance Σ_ϕ as a diagonal given by e^ϕ. The exploration for DDPG is determined by an Ornstein-Uhlenbeck process (Uhlenbeck & Ornstein, 1930; Lillicrap et al., 2016). Additional implementation details are provided in the Appendix.

We compare the results of Smoothie and DDPG in Figure 3. For each task we performed a hyperparameter search over actor learning rate, critic learning rate and reward scale, and plot the average of six runs for the best hyperparameters. For DDPG we extended the hyperparameter search to also consider the scale and damping of exploratory noise provided by the Ornstein-Uhlenbeck process. Smoothie, on the other hand, contains an additional hyperparameter to determine the weight on KL-penalty.

Despite DDPG having the advantage of its exploration decided by a hyperparameter search while Smoothie must learn its exploration without supervision, we find that Smoothie performs competitively or better across all tasks, exhibiting a slight advantage in Swimmer and Ant, while showing more dramatic improvements in Hopper, Walker2d, and Humanoid. The improvement is especially dramatic for Hopper, where the average reward is doubled. We also highlight the results for Humanoid, which as far as we know, are the best published results for a method that only trains on the order of millions of environment steps. In contrast, TRPO, which to the best of our knowledge is the only other algorithm that can achieve competitive performance, requires on the order of tens of millions of environment steps to achieve comparable reward. This gives added evidence to the benefits of using a learnable covariance and not restricting a policy to be deterministic.

Empirically, we found the introduction of a KL-penalty to improve performance of Smoothie, especially on harder tasks. We present a comparison of results of Smoothie
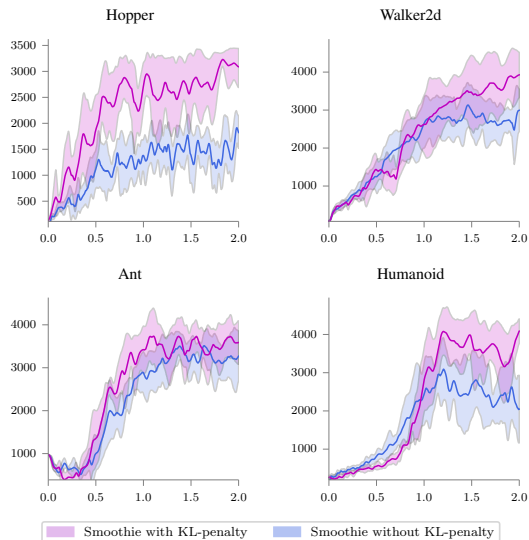

Figure 4. Results of Smoothie with and without a KL-penalty. The x-axis is in millions of environment steps. We observe benefits of using a proximal policy optimization method, especially in Hopper and Humanoid, where the performance improvement is significant without sacrificing sample efficiency.

with and without the KL-penalty on the four harder tasks in Figure 4. A KL-penalty to encourage stability is not possible in DDPG. Thus, Smoothie provides a much needed solution to the inherent instability in DDPG training.

7. Conclusion

We have presented a new Q-value function concept, $\tilde{Q}^\pi$, that is a Gaussian-smoothed version of the standard expected Q-value, Q^π. The advantage of $\tilde{Q}^\pi$ over Q^π is that its gradient and Hessian possess an intimate relationship with the gradient of expected reward with respect to mean and covariance of a Gaussian policy. The resulting algorithm, Smoothie, is able to successfully learn both mean and covariance during training, leading to performance that surpasses that of DDPG, especially when incorporating a penalty on divergence from a previous policy.

The success of $\tilde{Q}^\pi$ is encouraging. Intuitively it appears advantageous to learn $\tilde{Q}^\pi$ instead of Q^π. The smoothed Q-values by definition make the true reward surface smoother, thus possibly easier to learn; moreover the smoothed Q-values have a more direct relationship with the expected discounted return objective. We encourage further investigation of these claims and techniques for applying the underlying motivations for $\tilde{Q}^\pi$ to other types of policies.

8. Acknowledgments

References

Bellemare, M. G., Dabney, W., and Munos, R. A distributional perspective on reinforcement learning. In *ICML*, pp. 449–458, 2017.

Bonnet, G. Transformations des signaux aléatoires a travers les systemes non linéaires sans mémoire. *Annals of Telecommunications*, 19(9):203–220, 1964.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. OpenAI Gym. arXiv:1606.01540, 2016.

Ciosek, K. and Whiteson, S. Expected policy gradients. *AAAI*, 2018.

Degris, T., White, M., and Sutton, R. S. Off-policy actor-critic. *ICML*, 2012.

Gu, S., Lillicrap, T., Ghahramani, Z., Turner, R. E., Schölkopf, B., and Levine, S. Interpolated policy gradient: Merging on-policy and off-policy gradient estimation for deep reinforcement learning. *NIPS*, 2017.

Haarnoja, T., Tang, H., Abbeel, P., and Levine, S. Reinforcement learning with deep energy-based policies. *ICML*, 2017.

Heess, N., Wayne, G., Silver, D., Lillicrap, T., Erez, T., and Tassa, Y. Learning continuous control policies by stochastic value gradients. In *NIPS*, 2015.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *ICLR*, 2014.

Konda, V. R. and Tsitsiklis, J. N. Actor-critic algorithms, 2000.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *ICLR*, 2016.

Metz, L., Ibarz, J., Jaitly, N., and Davidson, J. Discrete sequential prediction of continuous actions for deep RL. *CoRR*, abs/1705.05035, 2017. URL http://arxiv.org/abs/1705.05035.

Munos, R., Stepleton, T., Harutyunyan, A., and Bellemare, M. Safe and efficient off-policy reinforcement learning. In *NIPS*, 2016.

Nachum, O., Norouzi, M., Xu, K., and Schuurmans, D. Bridging the gap between value and policy based reinforcement learning. *NIPS*, 2017.

Nachum, O., Norouzi, M., Xu, K., and Schuurmans, D. Trust-pcl: An off-policy trust region method for continuous control. *ICLR*, 2018.

Price, R. A useful theorem for nonlinear devices having gaussian inputs. *IRE Transactions on Information Theory*, 4(2):69–72, 1958.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, pp. 1278–1286, 2014.

Rummery, G. A. and Niranjan, M. *On-line Q-learning using connectionist systems*, volume 37. 1994.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *ICML*, 2015.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. Deterministic policy gradient algorithms. In *ICML*, 2014.

Sutton, R. S. and Barto, A. G. *Introduction to Reinforcement Learning*. MIT Press, 1998.

Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. *NIPS*, 2000.

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

Uhlenbeck, G. E. and Ornstein, L. S. On the theory of the brownian motion. *Physical review*, 36(5):823, 1930.

Van Seijen, H., Van Hasselt, H., Whiteson, S., and Wiering, M. A theoretical and empirical analysis of expected sarsa. In *Adaptive Dynamic Programming and Reinforcement Learning, 2009. ADPRL'09. IEEE Symposium on*, pp. 177–184. IEEE, 2009.

Wang, Z., Bapst, V., Heess, N., Mnih, V., Munos, R., Kavukcuoglu, K., and de Freitas, N. Sample efficient actor-critic with experience replay. *ICLR*, 2017.

Watkins, C. J. C. H. *Learning from delayed rewards*. PhD thesis, University of Cambridge England, 1989.

Williams, R. J. and Peng, J. Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, 1991.

A. Proof of Theorem 1

We want to show that for any s, a,

$$\frac{\partial \tilde{Q}^\pi(s,a)}{\partial \Sigma(s)} = \frac{1}{2} \cdot \frac{\partial^2 \tilde{Q}^\pi(s,a)}{\partial a^2} \tag{23}$$

We note that similar identities for Gaussian integrals exist in the literature (Price, 1958; Rezende et al., 2014) and point the reader to these works for further information.

Proof. The specific identity we state may be derived using standard matrix calculus. We make use of the fact that

$$\frac{\partial}{\partial A}|A|^{-1/2} = -\frac{1}{2}|A|^{-3/2}\frac{\partial}{\partial A}|A| = -\frac{1}{2}|A|^{-1/2}A^{-1}, \tag{24}$$

and for symmetric A,

$$\frac{\partial}{\partial A}||v||^2_{A^{-1}} = -A^{-1}vv^T A^{-1}. \tag{25}$$

We omit s from $\Sigma(s)$ in the following equations for succinctness. The LHS of (23) is

$$\int_\mathcal{A} Q^\pi(s,\tilde{a})\frac{\partial}{\partial\Sigma}N(\tilde{a}|a,\Sigma)\mathrm{d}\tilde{a}$$

$$= \int_\mathcal{A} Q^\pi(s,\tilde{a})\exp\left\{-\frac{1}{2}||\tilde{a}-a||^2_{\Sigma^{-1}}\right\}\left(\frac{\partial}{\partial\Sigma}|2\pi\Sigma|^{-1/2} - \frac{1}{2}|2\pi\Sigma|^{-1/2}\frac{\partial}{\partial\Sigma}||\tilde{a}-a||^2_{\Sigma^{-1}}\right)\mathrm{d}\tilde{a}$$

$$= \frac{1}{2}\int_\mathcal{A} Q^\pi(s,\tilde{a})N(\tilde{a}|a,\Sigma)\left(-\Sigma^{-1} + \Sigma^{-1}(\tilde{a}-a)(\tilde{a}-a)^T\Sigma^{-1}\right)\mathrm{d}\tilde{a}.$$

Meanwhile, towards tackling the RHS of (23) we note that

$$\frac{\partial \tilde{Q}^\pi(s,a)}{\partial a} = \int_\mathcal{A} Q^\pi(s,\tilde{a})N(\tilde{a}|a,\Sigma)\Sigma^{-1}(\tilde{a}-a)\mathrm{d}\tilde{a} . \tag{26}$$

Thus we have

$$\frac{\partial^2 \tilde{Q}^\pi(s,a)}{\partial a^2} = \int_\mathcal{A} Q^\pi(s,\tilde{a})\left(\Sigma^{-1}(\tilde{a}-a)\frac{\partial}{\partial a}N(\tilde{a}|a,\Sigma) + N(\tilde{a}|a,\Sigma)\frac{\partial}{\partial a}\Sigma^{-1}(\tilde{a}-a)\right)\mathrm{d}\tilde{a}$$

$$= \int_\mathcal{A} Q^\pi(s,\tilde{a})N(\tilde{a}|a,\Sigma)(\Sigma^{-1}(\tilde{a}-a)(\tilde{a}-a)^T\Sigma^{-1} - \Sigma^{-1})\mathrm{d}\tilde{a} .$$

∎

B. Compatible Function Approximation

We claim that a $\tilde{Q}^\pi_w$ is compatible with respect to μ_θ if

1. $\nabla_a \tilde{Q}^\pi_w(s,a)\big|_{a=\mu_\theta(s)} = \nabla_\theta \mu_\theta(s)^T w$,

2. $\nabla_w \int_\mathcal{S} \left(\nabla_a\tilde{Q}^\pi_w(s,a)\big|_{a=\mu_\theta(s)} - \nabla_a\tilde{Q}^\pi(s,a)\big|_{a=\mu_\theta(s)}\right)^2 \mathrm{d}\rho^\pi(s) = 0$ (*i.e.,* w minimizes the expected squared error of the gradients).

Additionally, $\tilde{Q}^\pi_w$ is compatible with respect to Σ_ϕ if

1. $\nabla^2_a \tilde{Q}^\pi_w(s,a)\big|_{a=\mu_\theta(s)} = \nabla_\phi \Sigma_\phi(s)^T w$,

2. $\nabla_w \int_{\mathcal{S}} \left(\nabla_a^2 \tilde{Q}_w^\pi(s,a)\big|_{a=\mu_\theta(s)} - \nabla_a^2 \tilde{Q}^\pi(s,a)\big|_{a=\mu_\theta(s)} \right)^2 \mathrm{d}\rho^\pi(s) = 0$ (*i.e.,* w minimizes the expected squared error of the Hessians).

Proof. We shall show how the conditions stated for compatibility with respect to Σ_ϕ are sufficient. The reasoning for μ_θ follows via a similar argument. We also refer the reader to Silver et al. (2014) which includes a similar procedure for showing compatibility.

From the second condition for compatibility with respect to Σ_ϕ we have

$$\int_{\mathcal{S}} \left(\nabla_a^2 \tilde{Q}_w^\pi(s,a)\big|_{a=\mu_\theta(s)} - \nabla_a^2 \tilde{Q}^\pi(s,a)\big|_{a=\mu_\theta(s)} \right) \nabla_w \left(\nabla_a^2 \tilde{Q}_w^\pi(s,a)\big|_{a=\mu_\theta(s)} \right) \mathrm{d}\rho^\pi(s) \quad = \quad 0 \, .$$

We may combine this with the first condition to find

$$\int_{\mathcal{S}} \nabla_a^2 \tilde{Q}_w^\pi(s,a)\big|_{a=\mu_\theta(s)} \nabla_\phi \Sigma_\phi(s) \mathrm{d}\rho^\pi(s) \quad = \quad \int_{\mathcal{S}} \nabla_a^2 \tilde{Q}^\pi(s,a)\big|_{a=\mu_\theta(s)} \nabla_\phi \Sigma_\phi(s) \mathrm{d}\rho^\pi(s) \, ,$$

which is the desired property for compatibility.   ∎

C. Derivative Bellman Equations

The conditions for compatibility require training $\tilde{Q}_w^\pi$ to fit the true $\tilde{Q}^\pi$ with respect to derivatives. However, in RL contexts, one often does not have access to the derivatives of the true $\tilde{Q}^\pi$. In this section, we elaborate on a method to train $\tilde{Q}_w^\pi$ to fit the derivatives of the true $\tilde{Q}^\pi$ without access to true derivative information.

Our method relies on a novel formulation: *derivative Bellman equations*. We begin with the standard $\tilde{Q}^\pi$ Bellman equation presented in the main paper:

$$\tilde{Q}^\pi(s,a) = \int_{\mathcal{A}} N(\tilde{a}\,|\,a, \Sigma(s)) \, \mathbb{E}_{\tilde{r},\tilde{s}'} \left[\tilde{r} + \gamma \tilde{Q}^\pi(\tilde{s}', \mu(\tilde{s}')) \right] \mathrm{d}\tilde{a} \, . \tag{27}$$

One may take derivatives of both sides to yield the following identity for any k:

$$\frac{\partial^k \tilde{Q}^\pi(s,a)}{\partial a^k} = \int_{\mathcal{A}} \frac{\partial^k N(\tilde{a}\,|\,a, \Sigma(s))}{\partial a^k} \, \mathbb{E}_{\tilde{r},\tilde{s}'} \left[\tilde{r} + \gamma \tilde{Q}^\pi(\tilde{s}', \mu(\tilde{s}')) \right] \mathrm{d}\tilde{a} \, . \tag{28}$$

One may express the k-the derivative of a normal density for $k \leq 2$ simply as

$$\frac{\partial^k N(\tilde{a}\,|\,a, \Sigma(s))}{\partial a^k} = N(\tilde{a}\,|\,a, \Sigma(s)) \Sigma(s)^{-k/2} \cdot H_k(\Sigma(s)^{-1/2}(\tilde{a} - a)), \tag{29}$$

where H_k is a polynomial. Therefore, we have the following derivative Bellman equations for any $k \leq 2$:

$$\frac{\partial^k \tilde{Q}^\pi(s,a)}{\partial a^k} = \int_{\mathcal{A}} N(\tilde{a}\,|\,a, \Sigma(s)) \Sigma(s)^{-k/2} \cdot H_k(\Sigma(s)^{-1/2}(\tilde{a} - a)) \, \mathbb{E}_{\tilde{r},\tilde{s}'} \left[\tilde{r} + \gamma \tilde{Q}^\pi(\tilde{s}', \mu(\tilde{s}')) \right] \mathrm{d}\tilde{a} \, . \tag{30}$$

One may train a parameterized $\tilde{Q}_w^\pi$ to satisfy these consistencies in a manner similar to that described in Section 4.2. Specifically, suppose one has access to a tuple $(s, \tilde{a}, \tilde{r}, \tilde{s}')$ sampled from a replay buffer with knowledge of the sampling probability $q(\tilde{a}\,|\,s)$ (possibly unnormalized) with full support. Then we draw a *phantom action* $a \sim N(\tilde{a}, \Sigma(s))$ and optimize $\tilde{Q}_w^\pi(s,a)$ by minimizing a weighted derivative Bellman error

$$\frac{1}{q(\tilde{a}|s)} \left(\frac{\partial^k \tilde{Q}_w^\pi(s,a)}{\partial a^k} - \Sigma(s)^{-k/2} \cdot H_k(\Sigma(s)^{-1/2}(a - \tilde{a}))(\tilde{r} + \gamma \tilde{Q}_w^\pi(\tilde{s}', \mu(\tilde{s}'))) \right)^2 , \tag{31}$$

for $k = 0, 1, 2$. As in the main text, it is possible to argue that when using target networks, this training procedure reaches an optimum when $\tilde{Q}_w^\pi(s,a)$ satisfies the recursion in the derivative Bellman equations (30) for $k = 0, 1, 2$.

Hyperparameter	Range	Sampling
actor learning rate	[1e-6,1e-3]	log
critic learning rate	[1e-6,1e-3]	log
reward scale	[0.01,0.3]	log
OU damping	[1e-4,1e-3]	log
OU stddev	[1e-3,1.0]	log
λ	[1e-6, 4e-2]	log
discount factor	0.995	fixed
target network lag	0.01	fixed
batch size	128	fixed
clipping on gradients of Q	4.0	fixed
num gradient updates per observation	1	fixed
Huber loss clipping	1.0	fixed

Table 1. Random hyperparameter search procedure. We also include the hyperparameters which we kept fixed.

D. Implementation Details

We utilize feed forward networks for both policy and Q-value approximator. For $\mu_\theta(s)$ we use two hidden layers of dimensions $(400, 300)$ and relu activation functions. For $\tilde{Q}^\pi_w(s, a)$ and $Q^\pi_w(s, a)$ we first embed the state into a 400 dimensional vector using a fully-connected layer and $\tanh$ non-linearity. We then concatenate the embedded state with a and pass the result through a 1-hidden layer neural network of dimension 300 with $\tanh$ activations. We use a diagonal $\Sigma_\phi(s) = e^\phi$ for Smoothie, with ϕ initialized to -1.

To find optimal hyperparameters we perform a 100-trial random search over the hyperparameters specified in Table 1. The OU exploration parameters only apply to DDPG. The λ coefficient on KL-penalty only applies to Smoothie with a KL-penalty.

D.1. Fast Computation of Gradients and Hessians

The Smoothie algorithm relies on the computation of the gradients $\frac{\partial \tilde{Q}^\pi_w(s,a)}{\partial a}$ and Hessians $\frac{\partial^2 \tilde{Q}^\pi_w(s,a)}{\partial a^2}$. In general, these quantities may be computed through multiple backward passes of a computation graph. However, for faster training, in our implementation we take advantage of a more efficient computation. We make use of the following identities:

$$\frac{\partial}{\partial x} f(g(x)) = f'(g(x)) \frac{\partial}{\partial x} g(x), \tag{32}$$

$$\frac{\partial^2}{\partial x^2} f(g(x)) = \left(\frac{\partial}{\partial x} g(x) \right)^T f''(g(x)) \frac{\partial}{\partial x} g(x) + f'(g(x)) \frac{\partial^2}{\partial x^2} g(x). \tag{33}$$

Thus, during the forward computation of our critic network $\tilde{Q}^\pi_w$, we not only maintain the tensor output O_L of layer L, but also the tensor G_L corresponding to the gradients of O_L with respect to input actions and the tensor H_L corresponding to the Hessians of O_L with respect to input actions. At each layer we may compute $O_{L+1}, G_{L+1}, H_{L+1}$ given O_L, G_L, H_L. Moreover, since we utilize feed-forward fully-connected layers, the computation of $O_{L+1}, G_{L+1}, H_{L+1}$ may be computed using fast tensor products.