HIGH-DIMENSIONAL CONTINUOUS CONTROL USING GENERALIZED ADVANTAGE ESTIMATION

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan and Pieter Abbeel
Department of Electrical Engineering and Computer Science
University of California, Berkeley
{joschu,pcmoritz,levine,jordan,pabbeel}@eecs.berkeley.edu

ABSTRACT

Policy gradient methods are an appealing approach in reinforcement learning because they directly optimize the cumulative reward and can straightforwardly be used with nonlinear function approximators such as neural networks. The two main challenges are the large number of samples typically required, and the difficulty of obtaining stable and steady improvement despite the nonstationarity of the incoming data. We address the first challenge by using value functions to substantially reduce the variance of policy gradient estimates at the cost of some bias, with an exponentially-weighted estimator of the advantage function that is analogous to TD(λ). We address the second challenge by using trust region optimization procedure for both the policy and the value function, which are represented by neural networks.

Our approach yields strong empirical results on highly challenging 3D locomotion tasks, learning running gaits for bipedal and quadrupedal simulated robots, and learning a policy for getting the biped to stand up from starting out lying on the ground. In contrast to a body of prior work that uses hand-crafted policy representations, our neural network policies map directly from raw kinematics to joint torques. Our algorithm is fully model-free, and the amount of simulated experience required for the learning tasks on 3D bipeds corresponds to 1-2 weeks of real time.

1 INTRODUCTION

The typical problem formulation in reinforcement learning is to maximize the expected total reward of a policy. A key source of difficulty is the long time delay between actions and their positive or negative effect on rewards; this issue is called the *credit assignment problem* in the reinforcement learning literature (Minsky, 1961; Sutton & Barto, 1998), and the *distal reward problem* in the behavioral literature (Hull, 1943). Value functions offer an elegant solution to the credit assignment problem—they allow us to estimate the goodness of an action before the delayed reward arrives. Reinforcement learning algorithms make use of value functions in a variety of different ways; this paper considers algorithms that optimize a parameterized policy and use value functions to help estimate how the policy should be improved.

When using a parameterized *stochastic* policy, it is possible to obtain an unbiased estimate of the gradient of the expected total returns (Williams, 1992; Sutton et al., 1999; Baxter & Bartlett, 2000); these noisy gradient estimates can be used in a stochastic gradient ascent algorithm. Unfortunately, the variance of the gradient estimator scales unfavorably with the time horizon, since the effect of an action is confounded with the effects of past and future actions. Another class of policy gradient algorithms, called actor-critic methods, use a value function rather than the empirical returns, obtaining an estimator with lower variance at the cost of introducing bias (Konda & Tsitsiklis, 2003; Hafner & Riedmiller, 2011). But while high variance necessitates using more samples, bias is more pernicious—even with an unlimited number of samples, bias can cause the algorithm to fail to converge, or to converge to a poor solution that is not even a local optimum.

We propose a family of policy gradient estimators that significantly reduce variance while maintaining a tolerable level of bias. We call this estimation scheme, parameterized by $\gamma \in [0, 1]$ and

$\lambda \in [0, 1]$, the generalized advantage estimator (GAE). Related methods have been proposed in the context of online actor-critic methods (Kimura & Kobayashi, 1998; Wawrzyński, 2009). We provide a more general analysis, which is applicable in both the online and batch settings, and discuss an interpretation of our method as an instance of reward shaping (Ng et al., 1999), where the approximate value function is used to shape the reward.

We present experimental results on a number of highly challenging 3D locomotion tasks, where we show that our approach can learn complex gaits using high-dimensional, general purpose neural network function approximators for both the policy and the value function, each with over 10^4 parameters. The policies perform torque-level control of simulated 3D robots with up to 33 state dimensions and 10 actuators.

The contributions of this paper are summarized as follows:

1. We provide justification and intuition for an effective variance reduction scheme for policy gradients, which we call generalized advantage estimation (GAE). While the formula has been proposed in prior work (Kimura & Kobayashi, 1998; Wawrzyński, 2009), our analysis is novel and enables GAE to be applied with a more general set of algorithms, including the batch trust-region algorithm we use for our experiments.
2. We propose the use of a trust region optimization method for the value function, which we find is a robust and efficient way to train neural network value functions with thousands of parameters.
3. By combining (1) and (2) above, we obtain an algorithm that empirically is effective at learning neural network policies for challenging control tasks. The results extend the state of the art in using reinforcement learning for high-dimensional continuous control. Videos are available at https://sites.google.com/site/gaepapersupp.

2 PRELIMINARIES

We consider an undiscounted formulation of the policy optimization problem. The initial state s_0 is sampled from distribution ρ_0. A trajectory $(s_0, a_0, s_1, a_1, \dots)$ is generated by sampling actions according to the policy $a_t \sim \pi(a_t \mid s_t)$ and sampling the states according to the dynamics $s_{t+1} \sim P(s_{t+1} \mid s_t, a_t)$, until a terminal (absorbing) state is reached. A reward $r_t = r(s_t, a_t, s_{t+1})$ is received at each timestep. The goal is to maximize the expected total reward $\sum_{t=0}^{\infty} r_t$, which is assumed to be finite for all policies. Note that we are not using a discount as part of the problem specification; it will appear below as an algorithm parameter that adjusts a bias-variance tradeoff. But the discounted problem (maximizing $\sum_{t=0}^{\infty} \gamma^t r_t$) can be handled as an instance of the undiscounted problem in which we absorb the discount factor into the reward function, making it time-dependent.

Policy gradient methods maximize the expected total reward by repeatedly estimating the gradient $g := \nabla_\theta \mathbb{E}\left[\sum_{t=0}^{\infty} r_t\right]$. There are several different related expressions for the policy gradient, which have the form

$$g = \mathbb{E}\left[\sum_{t=0}^{\infty} \Psi_t \nabla_\theta \log \pi_\theta(a_t \mid s_t)\right], \tag{1}$$

where Ψ_t may be one of the following:

1. $\sum_{t=0}^{\infty} r_t$: total reward of the trajectory.
2. $\sum_{t'=t}^{\infty} r_{t'}$: reward following action a_t.
3. $\sum_{t'=t}^{\infty} r_{t'} - b(s_t)$: baselined version of previous formula.
4. $Q^\pi(s_t, a_t)$: state-action value function.
5. $A^\pi(s_t, a_t)$: advantage function.
6. $r_t + V^\pi(s_{t+1}) - V^\pi(s_t)$: TD residual.

The latter formulas use the definitions

$$V^\pi(s_t) := \mathbb{E}_{\substack{s_{t+1:\infty},\\ a_{t:\infty}}}\left[\sum_{l=0}^{\infty} r_{t+l}\right] \qquad Q^\pi(s_t, a_t) := \mathbb{E}_{\substack{s_{t+1:\infty},\\ a_{t+1:\infty}}}\left[\sum_{l=0}^{\infty} r_{t+l}\right] \tag{2}$$

$$A^\pi(s_t, a_t) := Q^\pi(s_t, a_t) - V^\pi(s_t), \quad \text{(Advantage function)}. \tag{3}$$

Here, the subscript of $\mathbb{E}$ enumerates the variables being integrated over, where states and actions are sampled sequentially from the dynamics model $P(s_{t+1} \mid s_t, a_t)$ and policy $\pi(a_t \mid s_t)$, respectively. The colon notation $a : b$ refers to the inclusive range $(a, a+1, \ldots, b)$. These formulas are well known and straightforward to obtain; they follow directly from Proposition 1, which will be stated shortly.

The choice $\Psi_t = A^\pi(s_t, a_t)$ yields almost the lowest possible variance, though in practice, the advantage function is not known and must be estimated. This statement can be intuitively justified by the following interpretation of the policy gradient: that a step in the policy gradient direction should increase the probability of better-than-average actions and decrease the probability of worse-than-average actions. The advantage function, by it's definition $A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$, measures whether or not the action is better or worse than the policy's default behavior. Hence, we should choose Ψ_t to be the advantage function $A^\pi(s_t, a_t)$, so that the gradient term $\Psi_t \nabla_\theta \log \pi_\theta(a_t \mid s_t)$ points in the direction of increased $\pi_\theta(a_t \mid s_t)$ if and only if $A^\pi(s_t, a_t) > 0$. See Greensmith et al. (2004) for a more rigorous analysis of the variance of policy gradient estimators and the effect of using a baseline.

We will introduce a parameter γ that allows us to reduce variance by downweighting rewards corresponding to delayed effects, at the cost of introducing bias. This parameter corresponds to the discount factor used in discounted formulations of MDPs, but we treat it as a variance reduction parameter in an undiscounted problem; this technique was analyzed theoretically by Marbach & Tsitsiklis (2003); Kakade (2001b); Thomas (2014). The discounted value functions are given by:

$$V^{\pi,\gamma}(s_t) := \mathbb{E}_{\substack{s_{t+1:\infty}, \\ a_{t:\infty}}} \left[\sum_{l=0}^{\infty} \gamma^l r_{t+l} \right] \qquad Q^{\pi,\gamma}(s_t, a_t) := \mathbb{E}_{\substack{s_{t+1:\infty}, \\ a_{t+1:\infty}}} \left[\sum_{l=0}^{\infty} \gamma^l r_{t+l} \right] \tag{4}$$

$$A^{\pi,\gamma}(s_t, a_t) := Q^{\pi,\gamma}(s_t, a_t) - V^{\pi,\gamma}(s_t). \tag{5}$$

The discounted approximation to the policy gradient is defined as follows:

$$g^\gamma := \mathbb{E}_{\substack{s_{0:\infty} \\ a_{0:\infty}}} \left[\sum_{t=0}^{\infty} A^{\pi,\gamma}(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t \mid s_t) \right]. \tag{6}$$

The following section discusses how to obtain biased (but not too biased) estimators for $A^{\pi,\gamma}$, giving us noisy estimates of the discounted policy gradient in Equation (6).

Before proceeding, we will introduce the notion of a γ-just estimator of the advantage function, which is an estimator that does not introduce bias when we use it in place of $A^{\pi,\gamma}$ (which is not known and must be estimated) in Equation (6) to estimate g^γ.[1] Consider an advantage estimator $\hat{A}_t(s_{0:\infty}, a_{0:\infty})$, which may in general be a function of the entire trajectory.

Definition 1. *The estimator $\hat{A}_t$ is γ-just if*

$$\mathbb{E}_{\substack{s_{0:\infty} \\ a_{0:\infty}}} \left[\hat{A}_t(s_{0:\infty}, a_{0:\infty}) \nabla_\theta \log \pi_\theta(a_t \mid s_t) \right] = \mathbb{E}_{\substack{s_{0:\infty} \\ a_{0:\infty}}} \left[A^{\pi,\gamma}(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t \mid s_t) \right]. \tag{7}$$

It follows immediately that if $\hat{A}_t$ is γ-just for all t, then

$$\mathbb{E}_{\substack{s_{0:\infty} \\ a_{0:\infty}}} \left[\sum_{t=0}^{\infty} \hat{A}_t(s_{0:\infty}, a_{0:\infty}) \nabla_\theta \log \pi_\theta(a_t \mid s_t) \right] = g^\gamma \tag{8}$$

One sufficient condition for $\hat{A}_t$ to be γ-just is that $\hat{A}_t$ decomposes as the difference between two functions Q_t and b_t, where Q_t can depend on any trajectory variables but gives an unbiased estimator of the γ-discounted Q-function, and b_t is an arbitrary function of the states and actions sampled before a_t.

Proposition 1. *Suppose that $\hat{A}_t$ can be written in the form $\hat{A}_t(s_{0:\infty}, a_{0:\infty}) = Q_t(s_{t:\infty}, a_{t:\infty}) - b_t(s_{0:t}, a_{0:t-1})$ such that for all (s_t, a_t), $\mathbb{E}_{s_{t+1:\infty}, a_{t+1:\infty} \mid s_t, a_t} \left[Q_t(s_{t:\infty}, a_{t:\infty}) \right] = Q^{\pi,\gamma}(s_t, a_t)$. Then $\hat{A}$ is γ-just.*

[1] Note, that we have already introduced bias by using $A^{\pi,\gamma}$ in place of A^π; here we are concerned with obtaining an unbiased estimate of g^γ, which is a biased estimate of the policy gradient of the undiscounted MDP.

The proof is provided in Appendix B. It is easy to verify that the following expressions are γ-just advantage estimators for $\hat{A}_t$:

- $\sum_{l=0}^{\infty} \gamma^l r_{t+l}$
- $Q^{\pi,\gamma}(s_t, a_t)$
- $A^{\pi,\gamma}(s_t, a_t)$
- $r_t + \gamma V^{\pi,\gamma}(s_{t+1}) - V^{\pi,\gamma}(s_t)$.

3 ADVANTAGE FUNCTION ESTIMATION

This section will be concerned with producing an accurate estimate $\hat{A}_t$ of the discounted advantage function $A^{\pi,\gamma}(s_t, a_t)$, which will then be used to construct a policy gradient estimator of the following form:

$$\hat{g} = \frac{1}{N} \sum_{n=1}^{N} \sum_{t=0}^{\infty} \hat{A}_t^n \nabla_\theta \log \pi_\theta(a_t^n \mid s_t^n) \tag{9}$$

where n indexes over a batch of episodes.

Let V be an approximate value function. Define $\delta_t^V = r_t + \gamma V(s_{t+1}) - V(s_t)$, i.e., the TD residual of V with discount γ (Sutton & Barto, 1998). Note that δ_t^V can be considered as an estimate of the advantage of the action a_t. In fact, if we have the correct value function $V = V^{\pi,\gamma}$, then it is a γ-just advantage estimator, and in fact, an unbiased estimator of $A^{\pi,\gamma}$:

$$\begin{aligned} \mathbb{E}_{s_{t+1}} \left[\delta_t^{V^{\pi,\gamma}} \right] &= \mathbb{E}_{s_{t+1}} \left[r_t + \gamma V^{\pi,\gamma}(s_{t+1}) - V^{\pi,\gamma}(s_t) \right] \\ &= \mathbb{E}_{s_{t+1}} \left[Q^{\pi,\gamma}(s_t, a_t) - V^{\pi,\gamma}(s_t) \right] = A^{\pi,\gamma}(s_t, a_t). \end{aligned} \tag{10}$$

However, this estimator is only γ-just for $V = V^{\pi,\gamma}$, otherwise it will yield biased policy gradient estimates.

Next, let us consider taking the sum of k of these δ terms, which we will denote by $\hat{A}_t^{(k)}$

$$\hat{A}_t^{(1)} := \delta_t^V \qquad = -V(s_t) + r_t + \gamma V(s_{t+1}) \tag{11}$$

$$\hat{A}_t^{(2)} := \delta_t^V + \gamma \delta_{t+1}^V \qquad = -V(s_t) + r_t + \gamma r_{t+1} + \gamma^2 V(s_{t+2}) \tag{12}$$

$$\hat{A}_t^{(3)} := \delta_t^V + \gamma \delta_{t+1}^V + \gamma^2 \delta_{t+2}^V = -V(s_t) + r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 V(s_{t+3}) \tag{13}$$

$$\hat{A}_t^{(k)} := \sum_{l=0}^{k-1} \gamma^l \delta_{t+l}^V = -V(s_t) + r_t + \gamma r_{t+1} + \cdots + \gamma^{k-1} r_{t+k-1} + \gamma^k V(s_{t+k}) \tag{14}$$

These equations result from a telescoping sum, and we see that $\hat{A}_t^{(k)}$ involves a k-step estimate of the returns, minus a baseline term $-V(s_t)$. Analogously to the case of $\delta_t^V = \hat{A}_t^{(1)}$, we can consider $\hat{A}_t^{(k)}$ to be an estimator of the advantage function, which is only γ-just when $V = V^{\pi,\gamma}$. However, note that the bias generally becomes smaller as $k \to \infty$, since the term $\gamma^k V(s_{t+k})$ becomes more heavily discounted, and the term $-V(s_t)$ does not affect the bias. Taking $k \to \infty$, we get

$$\hat{A}_t^{(\infty)} = \sum_{l=0}^{\infty} \gamma^l \delta_{t+l}^V = -V(s_t) + \sum_{l=0}^{\infty} \gamma^l r_{t+l}, \tag{15}$$

which is simply the empirical returns minus the value function baseline.

The generalized advantage estimator $\mathrm{GAE}(\gamma, \lambda)$ is defined as the exponentially-weighted average of these k-step estimators:

$$\begin{aligned}
\hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)} &:= (1-\lambda)\left(\hat{A}_t^{(1)} + \lambda \hat{A}_t^{(2)} + \lambda^2 \hat{A}_t^{(3)} + \dots\right) \\
&= (1-\lambda)\left(\delta_t^V + \lambda(\delta_t^V + \gamma\delta_{t+1}^V) + \lambda^2(\delta_t^V + \gamma\delta_{t+1}^V + \gamma^2\delta_{t+2}^V) + \dots\right) \\
&= (1-\lambda)(\delta_t^V(1 + \lambda + \lambda^2 + \dots) + \gamma\delta_{t+1}^V(\lambda + \lambda^2 + \lambda^3 + \dots) \\
&\qquad + \gamma^2\delta_{t+2}^V(\lambda^2 + \lambda^3 + \lambda^4 + \dots) + \dots) \\
&= (1-\lambda)\left(\delta_t^V\left(\frac{1}{1-\lambda}\right) + \gamma\delta_{t+1}^V\left(\frac{\lambda}{1-\lambda}\right) + \gamma^2\delta_{t+2}^V\left(\frac{\lambda^2}{1-\lambda}\right) + \dots\right) \\
&= \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}^V
\end{aligned} \tag{16}$$

From Equation (16), we see that the advantage estimator has a remarkably simple formula involving a discounted sum of Bellman residual terms. Section 4 discusses an interpretation of this formula as the returns in an MDP with a modified reward function. The construction we used above is closely analogous to the one used to define TD(λ) (Sutton & Barto, 1998), however TD(λ) is an estimator of the value function, whereas here we are estimating the advantage function.

There are two notable special cases of this formula, obtained by setting $\lambda = 0$ and $\lambda = 1$.

$$\mathrm{GAE}(\gamma, 0): \quad \hat{A}_t := \delta_t \qquad = r_t + \gamma V(s_{t+1}) - V(s_t) \tag{17}$$

$$\mathrm{GAE}(\gamma, 1): \quad \hat{A}_t := \sum_{l=0}^{\infty} \gamma^l \delta_{t+l} = \sum_{l=0}^{\infty} \gamma^l r_{t+l} - V(s_t) \tag{18}$$

$\mathrm{GAE}(\gamma, 1)$ is γ-just regardless of the accuracy of V, but it has high variance due to the sum of terms. $\mathrm{GAE}(\gamma, 0)$ is γ-just for $V = V^{\pi,\gamma}$ and otherwise induces bias, but it typically has much lower variance. The generalized advantage estimator for $0 < \lambda < 1$ makes a compromise between bias and variance, controlled by parameter λ.

We've described an advantage estimator with two separate parameters γ and λ, both of which contribute to the bias-variance tradeoff when using an approximate value function. However, they serve different purposes and work best with different ranges of values. γ most importantly determines the scale of the value function $V^{\pi,\gamma}$, which does not depend on λ. Taking $\gamma < 1$ introduces bias into the policy gradient estimate, regardless of the value function's accuracy. On the other hand, $\lambda < 1$ introduces bias only when the value function is inaccurate. Empirically, we find that the best value of λ is much lower than the best value of γ, likely because λ introduces far less bias than γ for a reasonably accurate value function.

Using the generalized advantage estimator, we can construct a biased estimator of g^γ, the discounted policy gradient from Equation (6):

$$g^\gamma \approx \mathbb{E}\left[\sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)}\right] = \mathbb{E}\left[\sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}^V\right], \tag{19}$$

where equality holds when $\lambda = 1$.

4 Interpretation as Reward Shaping

In this section, we discuss how one can interpret λ as an extra discount factor applied after performing a reward shaping transformation on the MDP. We also introduce the notion of a response function to help understand the bias introduced by γ and λ.

Reward shaping (Ng et al., 1999) refers to the following transformation of the reward function of an MDP: let $\Phi : \mathcal{S} \to \mathbb{R}$ be an arbitrary scalar-valued function on state space, and define the transformed reward function $\tilde{r}$ by

$$\tilde{r}(s, a, s') = r(s, a, s') + \gamma\Phi(s') - \Phi(s), \tag{20}$$

which in turn defines a transformed MDP. This transformation leaves the discounted advantage function $A^{\pi,\gamma}$ unchanged for any policy π. To see this, consider the discounted sum of rewards of a trajectory starting with state s_t:

$$\sum_{l=0}^{\infty} \gamma^l \tilde{r}(s_{t+l}, a_t, s_{t+l+1}) = \sum_{l=0}^{\infty} \gamma^l r(s_{t+l}, a_{t+l}, s_{t+l+1}) - \Phi(s_t). \tag{21}$$

Letting $\tilde{Q}^{\pi,\gamma}, \tilde{V}^{\pi,\gamma}, \tilde{A}^{\pi,\gamma}$ be the value and advantage functions of the transformed MDP, one obtains from the definitions of these quantities that

$$\tilde{Q}^{\pi,\gamma}(s,a) = Q^{\pi,\gamma}(s,a) - \Phi(s) \tag{22}$$

$$\tilde{V}^{\pi,\gamma}(s,a) = V^{\pi,\gamma}(s) - \Phi(s) \tag{23}$$

$$\tilde{A}^{\pi,\gamma}(s,a) = (Q^{\pi,\gamma}(s,a) - \Phi(s)) - (V^{\pi,\gamma}(s) - \Phi(s)) = A^{\pi,\gamma}(s,a). \tag{24}$$

Note that if Φ happens to be the state-value function $V^{\pi,\gamma}$ from the original MDP, then the transformed MDP has the interesting property that $\tilde{V}^{\pi,\gamma}(s)$ is zero at every state.

Note that (Ng et al., 1999) showed that the reward shaping transformation leaves the policy gradient and optimal policy unchanged when our objective is to maximize the discounted sum of rewards $\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, s_{t+1})$. In contrast, this paper is concerned with maximizing the undiscounted sum of rewards, where the discount γ is used as a variance-reduction parameter.

Having reviewed the idea of reward shaping, let us consider how we could use it to get a policy gradient estimate. The most natural approach is to construct policy gradient estimators that use discounted sums of shaped rewards $\tilde{r}$. However, Equation (21) shows that we obtain the discounted sum of the original MDP's rewards r minus a baseline term. Next, let's consider using a "steeper" discount $\gamma\lambda$, where $0 \leq \lambda \leq 1$. It's easy to see that the shaped reward $\tilde{r}$ equals the Bellman residual term δ^V, introduced in Section 3, where we set $\Phi = V$. Letting $\Phi = V$, we see that

$$\sum_{l=0}^{\infty} (\gamma\lambda)^l \tilde{r}(s_{t+l}, a_t, s_{t+l+1}) = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta^V_{t+l} = \hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)}. \tag{25}$$

Hence, by considering the $\gamma\lambda$-discounted sum of shaped rewards, we exactly obtain the generalized advantage estimators from Section 3. As shown previously, $\lambda = 1$ gives an unbiased estimate of g^γ, whereas $\lambda < 1$ gives a biased estimate.

To further analyze the effect of this shaping transformation and parameters γ and λ, it will be useful to introduce the notion of a response function χ, which we define as follows:

$$\chi(l; s_t, a_t) = \mathbb{E}\left[r_{t+l} \mid s_t, a_t\right] - \mathbb{E}\left[r_{t+l} \mid s_t\right]. \tag{26}$$

Note that $A^{\pi,\gamma}(s,a) = \sum_{l=0}^{\infty} \gamma^l \chi(l; s, a)$, hence the response function decomposes the advantage function across timesteps. The response function lets us quantify the temporal credit assignment problem: long range dependencies between actions and rewards correspond to nonzero values of the response function for $l \gg 0$.

Next, let us revisit the discount factor γ and the approximation we are making by using $A^{\pi,\gamma}$ rather than $A^{\pi,1}$. The discounted policy gradient estimator from Equation (6) has a sum of terms of the form

$$\nabla_\theta \log \pi_\theta(a_t \mid s_t) A^{\pi,\gamma}(s_t, a_t) = \nabla_\theta \log \pi_\theta(a_t \mid s_t) \sum_{l=0}^{\infty} \gamma^l \chi(l; s_t, a_t). \tag{27}$$

Using a discount $\gamma < 1$ corresponds to dropping the terms with $l \gg 1/(1-\gamma)$. Thus, the error introduced by this approximation will be small if χ rapidly decays as l increases, i.e., if the effect of an action on rewards is "forgotten" after $\approx 1/(1-\gamma)$ timesteps.

If the reward function $\tilde{r}$ were obtained using $\Phi = V^{\pi,\gamma}$, we would have $\mathbb{E}\left[\tilde{r}_{t+l} \mid s_t, a_t\right] = \mathbb{E}\left[\tilde{r}_{t+l} \mid s_t\right] = 0$ for $l > 0$, i.e., the response function would only be nonzero at $l = 0$. Therefore, this shaping transformation would turn temporally extended response into an immediate response. Given that $V^{\pi,\gamma}$ completely reduces the temporal spread of the response function, we can hope that a good approximation $V \approx V^{\pi,\gamma}$ partially reduces it. This observation suggests an interpretation of Equation (16): reshape the rewards using V to shrink the temporal extent of the response function, and then introduce a "steeper" discount $\gamma\lambda$ to cut off the noise arising from long delays, i.e., ignore terms $\nabla_\theta \log \pi_\theta(a_t \mid s_t) \delta^V_{t+l}$ where $l \gg 1/(1-\gamma\lambda)$.

5 VALUE FUNCTION ESTIMATION

A variety of different methods can be used to estimate the value function (see, e.g., Bertsekas (2012)). When using a nonlinear function approximator to represent the value function, the simplest approach is to solve a nonlinear regression problem:

$$\underset{\phi}{\text{minimize}} \sum_{n=1}^{N} \|V_\phi(s_n) - \hat{V}_n\|^2, \tag{28}$$

where $\hat{V}_t = \sum_{l=0}^{\infty} \gamma^l r_{t+l}$ is the discounted sum of rewards, and n indexes over all timesteps in a batch of trajectories. This is sometimes called the Monte Carlo or TD(1) approach for estimating the value function (Sutton & Barto, 1998).[2]

For the experiments in this work, we used a trust region method to optimize the value function in each iteration of a batch optimization procedure. The trust region helps us to avoid overfitting to the most recent batch of data. To formulate the trust region problem, we first compute $\sigma^2 = \frac{1}{N}\sum_{n=1}^{N} \|V_{\phi_{\text{old}}}(s_n) - \hat{V}_n\|^2$, where ϕ_{old} is the parameter vector before optimization. Then we solve the following constrained optimization problem:

$$\begin{aligned} \underset{\phi}{\text{minimize}} \quad & \sum_{n=1}^{N} \|V_\phi(s_n) - \hat{V}_n\|^2 \\ \text{subject to} \quad & \frac{1}{N}\sum_{n=1}^{N} \frac{\|V_\phi(s_n) - V_{\phi_{\text{old}}}(s_n)\|^2}{2\sigma^2} \le \epsilon. \end{aligned} \tag{29}$$

This constraint is equivalent to constraining the average KL divergence between the previous value function and the new value function to be smaller than ϵ, where the value function is taken to parameterize a conditional Gaussian distribution with mean $V_\phi(s)$ and variance σ^2.

We compute an approximate solution to the trust region problem using the conjugate gradient algorithm (Wright & Nocedal, 1999). Specifically, we are solving the quadratic program

$$\begin{aligned} \underset{\phi}{\text{minimize}} \quad & g^T(\phi - \phi_{\text{old}}) \\ \text{subject to} \quad & \frac{1}{N}\sum_{n=1}^{N} (\phi - \phi_{\text{old}})^T H (\phi - \phi_{\text{old}}) \le \epsilon. \end{aligned} \tag{30}$$

where g is the gradient of the objective, and $H = \frac{1}{N}\sum_n j_n j_n^T$, where $j_n = \nabla_\phi V_\phi(s_n)$. Note that H is the "Gauss-Newton" approximation of the Hessian of the objective, and it is (up to a σ^2 factor) the Fisher information matrix when interpreting the value function as a conditional probability distribution. Using matrix-vector products $v \to Hv$ to implement the conjugate gradient algorithm, we compute a step direction $s \approx -H^{-1}g$. Then we rescale $s \to \alpha s$ such that $\frac{1}{2}(\alpha s)^T H(\alpha s) = \epsilon$ and take $\phi = \phi_{\text{old}} + \alpha s$. This procedure is analogous to the procedure we use for updating the policy, which is described further in Section 6 and based on Schulman et al. (2015).

6 EXPERIMENTS

We designed a set of experiments to investigate the following questions:

1. What is the empirical effect of varying $\lambda \in [0, 1]$ and $\gamma \in [0, 1]$ when optimizing episodic total reward using generalized advantage estimation?
2. Can generalized advantage estimation, along with trust region algorithms for policy and value function optimization, be used to optimize large neural network policies for challenging control problems?

[2] Another natural choice is to compute target values with an estimator based on the TD(λ) backup (Bertsekas, 2012; Sutton & Barto, 1998), mirroring the expression we use for policy gradient estimation: $\hat{V}_t^\lambda = V_{\phi_{\text{old}}}(s_n) + \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}$. While we experimented with this choice, we did not notice a difference in performance from the $\lambda = 1$ estimator in Equation (28).

6.1 POLICY OPTIMIZATION ALGORITHM

While generalized advantage estimation can be used along with a variety of different policy gradient methods, for these experiments, we performed the policy updates using trust region policy optimization (TRPO) (Schulman et al., 2015). TRPO updates the policy by approximately solving the following constrained optimization problem each iteration:

$$\begin{aligned}
&\underset{\theta}{\text{minimize}}\, L_{\theta_{old}}(\theta) \\
&\text{subject to } \overline{D}_{\text{KL}}^{\theta_{old}}(\pi_{\theta_{old}}, \pi_\theta) \leq \epsilon \\
&\text{where } L_{\theta_{old}}(\theta) = \frac{1}{N}\sum_{n=1}^{N} \frac{\pi_\theta(a_n \mid s_n)}{\pi_{\theta_{old}}(a_n \mid s_n)} \hat{A}_n \\
&\qquad \overline{D}_{\text{KL}}^{\theta_{old}}(\pi_{\theta_{old}}, \pi_\theta) = \frac{1}{N}\sum_{n=1}^{N} D_{KL}(\pi_{\theta_{old}}(\cdot \mid s_n) \parallel \pi_\theta(\cdot \mid s_n))
\end{aligned} \tag{31}$$

As described in (Schulman et al., 2015), we approximately solve this problem by linearizing the objective and quadraticizing the constraint, which yields a step in the direction $\theta - \theta_{old} \propto -F^{-1}g$, where F is the average Fisher information matrix, and g is a policy gradient estimate. This policy update yields the same step direction as the natural policy gradient (Kakade, 2001a) and natural actor-critic (Peters & Schaal, 2008), however it uses a different stepsize determination scheme and numerical procedure for computing the step.

Since prior work (Schulman et al., 2015) compared TRPO to a variety of different policy optimization algorithms, we will not repeat these comparisons; rather, we will focus on varying the γ, λ parameters of policy gradient estimator while keeping the underlying algorithm fixed.

For completeness, the whole algorithm for iteratively updating policy and value function is given below:

Initialize policy parameter θ_0 and value function parameter ϕ_0.
for i = 0, 1, 2, ... **do**
  Simulate current policy π_{θ_i} until N timesteps are obtained.
  Compute δ_t^V at all timesteps $t \in \{1, 2, \ldots, N\}$, using $V = V_{\phi_i}$.
  Compute $\hat{A}_t = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}^V$ at all timesteps.
  Compute θ_{i+1} with TRPO update, Equation (31).
  Compute ϕ_{i+1} with Equation (30).
end for

Note that the policy update $\theta_i \to \theta_{i+1}$ is performed using the value function V_{ϕ_i} for advantage estimation, not $V_{\phi_{i+1}}$. Additional bias would have been introduced if we updated the value function first. To see this, consider the extreme case where we overfit the value function, and the Bellman residual $r_t + \gamma V(s_{t+1}) - V(s_t)$ becomes zero at all timesteps—the policy gradient estimate would be zero.

6.2 EXPERIMENTAL SETUP

We evaluated our approach on the classic cart-pole balancing problem, as well as several challenging 3D locomotion tasks: (1) bipedal locomotion; (2) quadrupedal locomotion; (3) dynamically standing up, for the biped, which starts off laying on its back. The models are shown in Figure 1.

6.2.1 ARCHITECTURE

We used the same neural network architecture for all of the 3D robot tasks, which was a feedforward network with three hidden layers, with 100, 50 and 25 tanh units respectively. The same architecture was used for the policy and value function. The final output layer had linear activation. The value function estimator used the same architecture, but with only one scalar output. For the simpler cart-pole task, we used a linear policy, and a neural network with one 20-unit hidden layer as the value function.

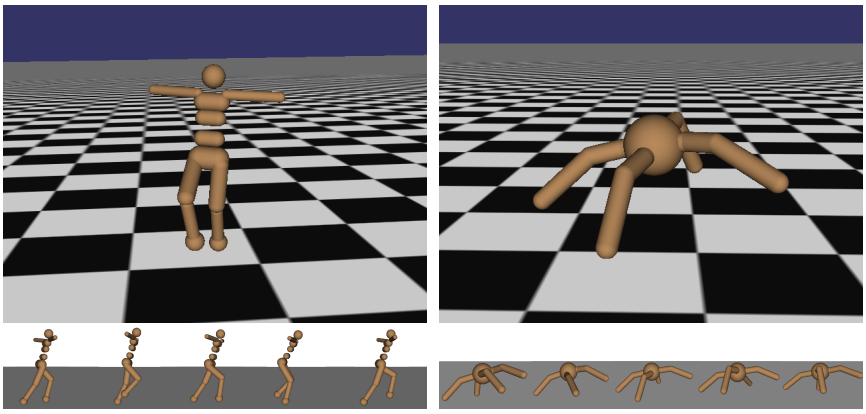

Figure 1: Top figures: robot models used for 3D locomotion. Bottom figures: a sequence of frames from the learned gaits. Videos are available at https://sites.google.com/site/gaepapersupp.

6.2.2 TASK DETAILS

For the cart-pole balancing task, we collected 20 trajectories per batch, with a maximum length of 1000 timesteps, using the physical parameters from Barto et al. (1983).

The simulated robot tasks were simulated using the MuJoCo physics engine (Todorov et al., 2012). The humanoid model has 33 state dimensions and 10 actuated degrees of freedom, while the quadruped model has 29 state dimensions and 8 actuated degrees of freedom. The initial state for these tasks consisted of a uniform distribution centered on a reference configuration. We used 50000 timesteps per batch for bipedal locomotion, and 200000 timesteps per batch for quadrupedal locomotion and bipedal standing. Each episode was terminated after 2000 timesteps if the robot had not reached a terminal state beforehand. The timestep was 0.01 seconds.

The reward functions are provided in the table below.

Task	Reward
3D biped locomotion	$v_{\text{fwd}} - 10^{-5}\|u\|^2 - 10^{-5}\|f_{\text{impact}}\|^2 + 0.2$
Quadruped locomotion	$v_{\text{fwd}} - 10^{-6}\|u\|^2 - 10^{-3}\|f_{\text{impact}}\|^2 + 0.05$
Biped getting up	$-(h_{\text{head}} - 1.5)^2 - 10^{-5}\|u\|^2$

Here, $v_{\text{fwd}} :=$ forward velocity, $u :=$ vector of joint torques, $f_{\text{impact}} :=$ impact forces, $h_{\text{head}} :=$ height of the head.

In the locomotion tasks, the episode is terminated if the center of mass of the actor falls below a predefined height: $.8\,\text{m}$ for the biped, and $.2\,\text{m}$ for the quadruped. The constant offset in the reward function encourages longer episodes; otherwise the quadratic reward terms might lead lead to a policy that ends the episodes as quickly as possible.

6.3 EXPERIMENTAL RESULTS

All results are presented in terms of the cost, which is defined as negative reward and is minimized. Videos of the learned policies are available at https://sites.google.com/site/gaepapersupp. In plots, "No VF" means that we used a time-dependent baseline that did not depend on the state, rather than an estimate of the state value function. The time-dependent baseline was computed by averaging the return at each timestep over the trajectories in the batch.

6.3.1 CART-POLE

The results are averaged across 21 experiments with different random seeds. Results are shown in Figure 2, and indicate that the best results are obtained at intermediate values of the parameters: $\gamma \in [0.96, 0.99]$ and $\lambda \in [0.92, 0.99]$.

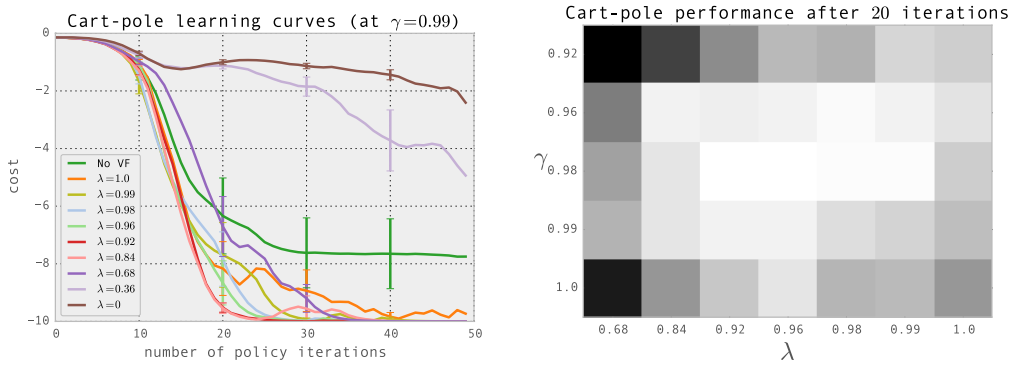

Figure 2: Left: learning curves for cart-pole task, using generalized advantage estimation with varying values of λ at $\gamma = 0.99$. The fastest policy improvement is obtain by intermediate values of λ in the range $[0.92, 0.98]$. Right: performance after 20 iterations of policy optimization, as γ and λ are varied. White means higher reward. The best results are obtained at intermediate values of both.

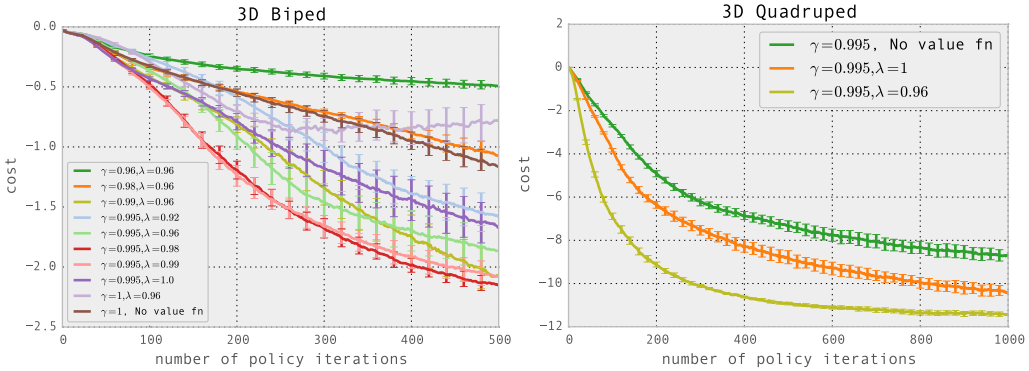

Figure 3: Left: Learning curves for 3D bipedal locomotion, averaged across nine runs of the algorithm. Right: learning curves for 3D quadrupedal locomotion, averaged across five runs.

6.3.2 3D BIPEDAL LOCOMOTION

Each trial took about 2 hours to run on a 16-core machine, where the simulation rollouts were parallelized, as were the function, gradient, and matrix-vector-product evaluations used when optimizing the policy and value function. Here, the results are averaged across 9 trials with different random seeds. The best performance is again obtained using intermediate values of $\gamma \in [0.99, 0.995], \lambda \in [0.96, 0.99]$. The result after 1000 iterations is a fast, smooth, and stable gait that is effectively completely stable. We can compute how much "real time" was used for this learning process: $0.01\,{}^{\text{seconds}}/_{\text{timestep}} \times 50000\,{}^{\text{timesteps}}/_{\text{batch}} \times 1000\,\text{batches}/3600 \cdot 24\,{}^{\text{seconds}}/_{\text{day}} = 5.8\,\text{days}$. Hence, it is plausible that this algorithm could be run on a real robot, or multiple real robots learning in parallel, if there were a way to reset the state of the robot and ensure that it doesn't damage itself.

6.3.3 OTHER 3D ROBOT TASKS

The other two motor behaviors considered are quadrupedal locomotion and getting up off the ground for the 3D biped. Again, we performed 5 trials per experimental condition, with different random seeds (and initializations). The experiments took about 4 hours per trial on a 32-core machine. We performed a more limited comparison on these domains (due to the substantial computational resources required to run these experiments), fixing $\gamma = 0.995$ but varying $\lambda = \{0, 0.96\}$, as well as an experimental condition with no value function. For quadrupedal locomotion, the best results are obtained using a value function with $\lambda = 0.96$ Section 6.3.2. For 3D standing, the value function always helped, but the results are roughly the same for $\lambda = 0.96$ and $\lambda = 1$.

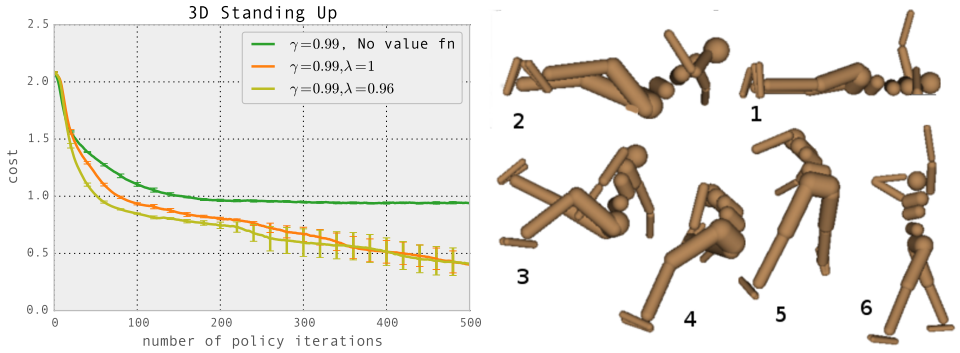

Figure 4: (a) Learning curve from quadrupedal walking, (b) learning curve for 3D standing up, (c) clips from 3D standing up.

7 Discussion

Policy gradient methods provide a way to reduce reinforcement learning to stochastic gradient descent, by providing unbiased gradient estimates. However, so far their success at solving difficult control problems has been limited, largely due to their high sample complexity. We have argued that the key to variance reduction is to obtain good estimates of the advantage function.

We have provided an intuitive but informal analysis of the problem of advantage function estimation, and justified the generalized advantage estimator, which has two parameters γ, λ which adjust the bias-variance tradeoff. We described how to combine this idea with trust region policy optimization and a trust region algorithm that optimizes a value function, both represented by neural networks. Combining these techniques, we are able to learn to solve difficult control tasks that have previously been out of reach for generic reinforcement learning methods.

Our main experimental validation of generalized advantage estimation is in the domain of simulated robotic locomotion. As shown in our experiments, choosing an appropriate intermediate value of λ in the range $[0.9, 0.99]$ usually results in the best performance. A possible topic for future work is how to adjust the estimator parameters γ, λ in an adaptive or automatic way.

One question that merits future investigation is the relationship between value function estimation error and policy gradient estimation error. If this relationship were known, we could choose an error metric for value function fitting that is well-matched to the quantity of interest, which is typically the accuracy of the policy gradient estimation. Some candidates for such an error metric might include the Bellman error or projected Bellman error, as described in Bhatnagar et al. (2009).

Another enticing possibility is to use a shared function approximation architecture for the policy and the value function, while optimizing the policy using generalized advantage estimation. While formulating this problem in a way that is suitable for numerical optimization and provides convergence guarantees remains an open question, such an approach could allow the value function and policy representations to share useful features of the input, resulting in even faster learning.

In concurrent work, researchers have been developing policy gradient methods that involve differentiation with respect to the continuous-valued action (Lillicrap et al., 2015; Heess et al., 2015). While we found empirically that the one-step return ($\lambda = 0$) leads to excessive bias and poor performance, these papers show that such methods can work when tuned appropriately. However, note that those papers consider control problems with substantially lower-dimensional state and action spaces than the ones considered here. A comparison between both classes of approach would be useful for future work.

Acknowledgements

We thank Emo Todorov for providing the simulator as well as insightful discussions, and we thank Greg Wayne, Yuval Tassa, Dave Silver, Carlos Florensa Campo, and Greg Brockman for insightful discussions. This research was funded in part by the Office of Naval Research through a Young

Investigator Award and under grant number N00014-11-1-0688, DARPA through a Young Faculty Award, by the Army Research Office through the MAST program.

A FREQUENTLY ASKED QUESTIONS

A.1 WHAT'S THE RELATIONSHIP WITH COMPATIBLE FEATURES?

Compatible features are often mentioned in relation to policy gradient algorithms that make use of a value function, and the idea was proposed in the paper *On Actor-Critic Methods* by Konda & Tsitsiklis (2003). These authors pointed out that due to the limited representation power of the policy, the policy gradient only depends on a certain subspace of the space of advantage functions. This subspace is spanned by the compatible features $\nabla_{\theta_i} \log \pi_\theta(a_t|s_t)$, where $i \in \{1, 2, \ldots, \dim \theta\}$. This theory of compatible features provides no guidance on how to exploit the temporal structure of the problem to obtain better estimates of the advantage function, making it mostly orthogonal to the ideas in this paper.

The idea of compatible features motivates an elegant method for computing the natural policy gradient (Kakade, 2001a; Peters & Schaal, 2008). Given an empirical estimate of the advantage function $\hat{A}_t$ at each timestep, we can project it onto the subspace of compatible features by solving the following least squares problem:

$$\underset{\mathbf{r}}{\text{minimize}} \sum_t \|\mathbf{r} \cdot \nabla_\theta \log \pi_\theta(a_t \mid s_t) - \hat{A}_t\|^2. \tag{32}$$

If $\hat{A}$ is γ-just, the least squares solution is the natural policy gradient (Kakade, 2001a). Note that any estimator of the advantage function can be substituted into this formula, including the ones we derive in this paper. For our experiments, we also compute natural policy gradient steps, but we use the more computationally efficient numerical procedure from Schulman et al. (2015), as discussed in Section 6.

A.2 WHY DON'T YOU JUST USE A Q-FUNCTION?

Previous actor critic methods, e.g. in Konda & Tsitsiklis (2003), use a Q-function to obtain potentially low-variance policy gradient estimates. Recent papers, including Heess et al. (2015); Lillicrap et al. (2015), have shown that a neural network Q-function approximator can used effectively in a policy gradient method. However, there are several advantages to using a state-value function in the manner of this paper. First, the state-value function has a lower-dimensional input and is thus easier to learn than a state-action value function. Second, the method of this paper allows us to smoothly interpolate between the high-bias estimator ($\lambda = 0$) and the low-bias estimator ($\lambda = 1$). On the other hand, using a parameterized Q-function only allows us to use a high-bias estimator. We have found that the bias is prohibitively large when using a one-step estimate of the returns, i.e., the $\lambda = 0$ estimator, $\hat{A}_t = \delta_t^V = r_t + \gamma V(s_{t+1}) - V(s_t)$. We expect that similar difficulty would be encountered when using an advantage estimator involving a parameterized Q-function, $\hat{A}_t = Q(s, a) - V(s)$. There is an interesting space of possible algorithms that would use a parameterized Q-function and attempt to reduce bias, however, an exploration of these possibilities is beyond the scope of this work.

B PROOFS

Proof of Proposition 1: First we can split the expectation into terms involving Q and b,

$$\begin{aligned}
&\mathbb{E}_{s_{0:\infty}, a_{0:\infty}} \left[\nabla_\theta \log \pi_\theta(a_t \mid s_t)(Q_t(s_{0:\infty}, a_{0:\infty}) - b_t(s_{0:t}, a_{0:t-1}))\right] \\
&\quad = \mathbb{E}_{s_{0:\infty}, a_{0:\infty}} \left[\nabla_\theta \log \pi_\theta(a_t \mid s_t)(Q_t(s_{0:\infty}, a_{0:\infty}))\right] \\
&\quad\quad - \mathbb{E}_{s_{0:\infty}, a_{0:\infty}} \left[\nabla_\theta \log \pi_\theta(a_t \mid s_t)(b_t(s_{0:t}, a_{0:t-1}))\right]
\end{aligned} \tag{33}$$

We'll consider the terms with Q and b in turn.

$$\begin{aligned}
&\mathbb{E}_{s_{0:\infty},a_{0:\infty}}\left[\nabla_\theta \log \pi_\theta(a_t \mid s_t) Q_t(s_{0:\infty}, a_{0:\infty})\right] \\
&\quad = \mathbb{E}_{s_{0:t},a_{0:t}}\left[\mathbb{E}_{s_{t+1:\infty},a_{t+1:\infty}}\left[\nabla_\theta \log \pi_\theta(a_t \mid s_t) Q_t(s_{0:\infty}, a_{0:\infty})\right]\right] \\
&\quad = \mathbb{E}_{s_{0:t},a_{0:t}}\left[\nabla_\theta \log \pi_\theta(a_t \mid s_t) \mathbb{E}_{s_{t+1:\infty},a_{t+1:\infty}}\left[Q_t(s_{0:\infty}, a_{0:\infty})\right]\right] \\
&\quad = \mathbb{E}_{s_{0:t},a_{0:t-1}}\left[\nabla_\theta \log \pi_\theta(a_t \mid s_t) A^\pi(s_t, a_t)\right]
\end{aligned}$$

Next,

$$\begin{aligned}
&\mathbb{E}_{s_{0:\infty},a_{0:\infty}}\left[\nabla_\theta \log \pi_\theta(a_t \mid s_t) b_t(s_{0:t}, a_{0:t-1})\right] \\
&\quad = \mathbb{E}_{s_{0:t},a_{0:t-1}}\left[\mathbb{E}_{s_{t+1:\infty},a_{t:\infty}}\left[\nabla_\theta \log \pi_\theta(a_t \mid s_t) b_t(s_{0:t}, a_{0:t-1})\right]\right] \\
&\quad = \mathbb{E}_{s_{0:t},a_{0:t-1}}\left[\mathbb{E}_{s_{t+1:\infty},a_{t:\infty}}\left[\nabla_\theta \log \pi_\theta(a_t \mid s_t)\right] b_t(s_{0:t}, a_{0:t-1})\right] \\
&\quad = \mathbb{E}_{s_{0:t},a_{0:t-1}}\left[0 \cdot b_t(s_{0:t}, a_{0:t-1})\right] \\
&\quad = 0.
\end{aligned}$$

REFERENCES

Barto, Andrew G, Sutton, Richard S, and Anderson, Charles W. Neuronlike adaptive elements that can solve difficult learning control problems. *Systems, Man and Cybernetics, IEEE Transactions on*, (5):834–846, 1983.

Baxter, Jonathan and Bartlett, Peter L. Reinforcement learning in POMDPs via direct gradient ascent. In *ICML*, pp. 41–48, 2000.

Bertsekas, Dimitri P. *Dynamic programming and optimal control*, volume 2. Athena Scientific, 2012.

Bhatnagar, Shalabh, Precup, Doina, Silver, David, Sutton, Richard S, Maei, Hamid R, and Szepesvári, Csaba. Convergent temporal-difference learning with arbitrary smooth function approximation. In *Advances in Neural Information Processing Systems*, pp. 1204–1212, 2009.

Greensmith, Evan, Bartlett, Peter L, and Baxter, Jonathan. Variance reduction techniques for gradient estimates in reinforcement learning. *The Journal of Machine Learning Research*, 5:1471–1530, 2004.

Hafner, Roland and Riedmiller, Martin. Reinforcement learning in feedback control. *Machine learning*, 84 (1-2):137–169, 2011.

Heess, Nicolas, Wayne, Greg, Silver, David, Lillicrap, Timothy, Tassa, Yuval, and Erez, Tom. Learning continuous control policies by stochastic value gradients. *arXiv preprint arXiv:1510.09142*, 2015.

Hull, Clark. Principles of behavior. 1943.

Kakade, Sham. A natural policy gradient. In *NIPS*, volume 14, pp. 1531–1538, 2001a.

Kakade, Sham. Optimizing average reward using discounted rewards. In *Computational Learning Theory*, pp. 605–615. Springer, 2001b.

Kimura, Hajime and Kobayashi, Shigenobu. An analysis of actor/critic algorithms using eligibility traces: Reinforcement learning with imperfect value function. In *ICML*, pp. 278–286, 1998.

Konda, Vijay R and Tsitsiklis, John N. On actor-critic algorithms. *SIAM journal on Control and Optimization*, 42(4):1143–1166, 2003.

Lillicrap, Timothy P, Hunt, Jonathan J, Pritzel, Alexander, Heess, Nicolas, Erez, Tom, Tassa, Yuval, Silver, David, and Wierstra, Daan. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Marbach, Peter and Tsitsiklis, John N. Approximate gradient methods in policy-space optimization of markov reward processes. *Discrete Event Dynamic Systems*, 13(1-2):111–148, 2003.

Minsky, Marvin. Steps toward artificial intelligence. *Proceedings of the IRE*, 49(1):8–30, 1961.

Ng, Andrew Y, Harada, Daishi, and Russell, Stuart. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, volume 99, pp. 278–287, 1999.

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

Schulman, John, Levine, Sergey, Moritz, Philipp, Jordan, Michael I, and Abbeel, Pieter. Trust region policy optimization. *arXiv preprint arXiv:1502.05477*, 2015.

Sutton, Richard S and Barto, Andrew G. *Introduction to reinforcement learning*. MIT Press, 1998.

Sutton, Richard S, McAllester, David A, Singh, Satinder P, and Mansour, Yishay. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*, volume 99, pp. 1057–1063. Citeseer, 1999.

Thomas, Philip. Bias in natural actor-critic algorithms. In *Proceedings of The 31st International Conference on Machine Learning*, pp. 441–448, 2014.

Todorov, Emanuel, Erez, Tom, and Tassa, Yuval. 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.

Wawrzyński, Paweł. Real-time reinforcement learning by sequential actor–critics and experience replay. *Neural Networks*, 22(10):1484–1497, 2009.

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

Wright, Stephen J and Nocedal, Jorge. *Numerical optimization*. Springer New York, 1999.