Generalized Advantage Estimation for Policy Gradients

Authors

Department of Electrical Engineering and Computer Science

Abstract

Value functions provide an elegant solution to the delayed reward problem in reinforcement learning, but it is difficult to accurately estimate and approximate them when the state space is high-dimensional. As a result, policy gradient methods that use Monte Carlo estimation are often preferred over methods that approximate the value function. We propose a method for using an approximate value function to help estimate the advantage function and obtain better policy gradient estimates, even when the value function is inaccurate. These estimators use a timescale parameter that makes an explicit tradeoff between bias and variance, and they empirically achieve faster policy improvement than Monte Carlo estimation and the actor-critic method, which can be viewed as limiting cases of these estimators. We present experimental results on a standard cart-pole benchmark task, as well as a number of highly challenging 3D locomotion tasks, where we show that our approach can learn complex gaits using neural network function approximators with over 10^4 parameters for both the policy and the value function.

1 Introduction

Value functions lie at the core of standard, well-known algorithms such as policy iteration and value iteration. The canonical versions of these algorithms represent the policy and value function with tables, which are only practical when the state space is finite and small. Approximate dynamic programming (ADP) algorithms, which represent the policy and value function with function approximators, can extend these approaches to MDPs with large or infinite state and action spaces. Unfortunately, convergence of such approximate methods requires a number of stringent constraints to be satisfied by the value function approximator [12]. In theory, and often in practice, small errors in the estimated value function can cause instability, nonconvergence and poor performance, particularly when using nonlinear function approximators. While ADP has been found to sometimes work well far outside the regime that is justified by the theory, it remains difficult to apply to high-dimensional problems, where nonlinear function approximators are needed to fit the value function with reasonable accuracy.

Policy gradient methods provide a conceptually straightforward approach to learning policies since they directly optimize the expected cost. Monte-Carlo estimators of the policy gradient [19] are unbiased and do not require a value function. However, the variance 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 estimators uses the value function to estimate the advantage of each action. These methods, called *actor-critic* methods [10], avoid the problem of high variance of policy gradient estimators, at the cost of introducing bias (due to inaccuracy of the value function). A biased estimator may cause the algorithm to diverge or converge to a suboptimal solution.

We propose a family of policy gradient estimators that interpolate between these two cases, providing explicit control over the bias-variance tradeoff. We call this estimation scheme, parameterized by $\lambda \in [0, 1]$, the *generalized advantage estimator* (GAE).

Related schemes have been proposed in the context of online actor-critic methods [9, 18], however, we provide a more general analysis based on estimating advantages. One interpretation of the GAE is that it replaces the discounted sum of costs by a discounted sum of *shaped* costs, where the learned value function is used as a potential field for cost shaping.

Our empirical results show that the GAE (with $0 < \lambda < 1$) achieves significantly faster policy improvement compared to both pure Monte Carlo estimation and the actor-critic method (which are limiting cases of our approach). We present experimental results on a standard cart-pole benchmark task, as well as 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 with over 10^4 parameters for both the policy and the value function.

2 Preliminaries

We will consider an undiscounted formulation of the policy optimization problem. The initial state s_0 is sampled from the 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 $s_{t+1} \sim P(s_{t+1} \mid s_t, a_t)$. A cost $c_t = c(s_t, a_t, s_{t+1})$ is incurred at each timestep. The goal is to minimize the expected total cost $C = \sum_{t=0}^{\infty} c_t$. This formulation encompases episodic and discounted formulations of the problem: to recover the episodic case with a fixed horizon of T timesteps, we can set $c(s_t, a_t, s_{t+1}) = 0$ for all $t \geq T$; to recover the discounted case, include t in the state and add a factor γ^t in the cost.

We will view the discount γ only as a parameter of the algorithm rather than the underlying MDP; this viewpoint has been proposed and discussed by several prior articles [8, 2, 7] in the context of average-cost problems. However, this work uses a total-cost formulation, rather than an average-cost formulation. The average-cost setting requires ergodicity and fast mixing [2], which are not satisfied by the MDPs we will consider.

Policy gradient methods estimate the gradient of the expected cost

$$\mathbf{g} = \nabla_\theta \mathbb{E}\left[\sum_{t=0}^{\infty} c_t\right]. \tag{1}$$

There are several different related expressions for the policy gradient, which have the form

$$\mathbf{g} = \mathbb{E}\left[\sum_{t=0}^{\infty} \psi_t \nabla_\theta \log \pi_\theta(a_t \mid s_t)\right], \tag{2}$$

where ψ_t may be one of the following.

1. $\sum_{s=0}^{\infty} c_s$: total cost of the trajectory.
2. $\sum_{l=0}^{\infty} c_{t+l}$: tost following action a_t.
3. $\sum_{l=0}^{\infty} c_{t+l} - b(s_t)$: 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. $\sum_{l=0}^{\infty}(c_t + V^\pi(s_{t+1}) - V^\pi(s_t))$: actor-critic formula.

These formulas can be derived from each other through straightforward manipulation of nested expectations and by using the fact that $\mathbb{E}_{a_t}\left[\nabla_\theta \log \pi_\theta(a_t \mid s_t)\right] = 0$. The latter formulas use the definitions

$$V^\pi(s_t) = \mathbb{E}_{\substack{s_{t+1:\infty}, \\ a_{t:\infty}}}\left[\sum_{l=0}^{\infty} c_{t+l}\right] \tag{3}$$

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

$$A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t). \quad \text{(Advantage function)} \tag{5}$$

Note: the subscript of $\mathbb{E}$ enumerates the variables being integrated over. In general, the formula based on total cost (Item 1) has the highest variance, and the the formula based on A^π (Item 5) has the lowest (but requires that advantage function is known, which isn't the case in practice).

For the rest of this article, we will not be estimating the policy gradient, but rather an approximation to it, which is formed by introducing a discount factor γ. The consequences of this approximation are discussed further in Section 4. First, let us define the discounted value functions

$$V^{\pi,\gamma}(s_t) = \mathbb{E}_{\substack{s_{t+1:\infty}, \\ a_{t:\infty}}} \left[\sum_{l=0}^{\infty} \gamma^l c_{t+l} \right] \tag{6}$$

$$Q^{\pi,\gamma}(s_t, a_t) = \mathbb{E}_{\substack{s_{t+1:\infty}, \\ a_{t+1:\infty}}} \left[\sum_{l=0}^{\infty} \gamma^l c_{t+l} \right] \tag{7}$$

$$A^{\pi,\gamma} = Q^{\pi,\gamma}(s, a) - V^{\pi,\gamma}(s). \tag{8}$$

where $\hat{Q}_t = c_t + \gamma c_{t+1} + \gamma^2 c_{t+2} + \dots$.

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

$$\mathbf{g}^\gamma = \sum_{t=0}^{\infty} \mathbb{E}_{s_t} \left[\mathbb{E}_{a_t \mid s_t} \left[A^{\pi,\gamma}(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t \mid s_t) \right] \right]. \tag{9}$$

The following formula is equivalent:

$$\mathbf{g}^\gamma = \mathbb{E}_{\substack{s_{0:\infty}, \\ a_{0:\infty}}} \left[\sum_{t=0}^{\infty} \hat{Q}_t^\gamma \nabla_\theta \log \pi_\theta(a_t \mid s_t) \right], \tag{10}$$

where $\hat{Q}_t = c_t + \gamma c_{t+1} + \gamma^2 c_{t+2} + \dots$.

Henceforth, we will omit the superscript γ on V^π, Q^π to reduce clutter.

If we know the advantage function A^π, we can form an estimator based on Equation (9). But in general, A^π is difficult to estimate. On the other hand, we can construct an estimator based on Equation (10) that uses a single sampled trajectory obtained by applying the current policy π. Below we will show how the estimator in Equation (10) can be decomposed into several terms, including the estimator from Equation (9) plus zero-mean noise terms from the confounding effects of past and future actions.

3 Policy gradient estimators using a value function

This section will show how to use the value function to construct estimators of the advantage function, which can then be used for policy gradient estimation. Define $\delta_t^V = c_t + \gamma V(s_{t+1}) - V(s_t)$, where V denotes an approximation to the policy's value function V^π. $\hat{Q}$ can be rewritten using a telescoping sum:

$$\begin{aligned} \hat{Q}_t &= \sum_{l=0}^{\infty} \gamma^l c_{t+l} \\ &= V(s_t) + \sum_{l=0}^{\infty} \gamma^l \delta_{t+l}^V \end{aligned} \tag{11}$$

$$= V(s_t) + \delta_t^V + \sum_{l=1}^{\infty} \gamma^l \delta_{t+l}^V. \tag{12}$$

For any V, the leftmost term in Equation (12) gives no contribution to the gradient:

$$\begin{aligned} \mathbb{E}_{a_t \sim \pi_\theta(a_t \mid s_t)} \left[\nabla_\theta \log \pi_\theta(a_t \mid s_t) V(s_t) \right] &= \mathbb{E}_{a_t \sim \pi_\theta(a_t \mid s_t)} \left[\nabla_\theta \log \pi_\theta(a_t \mid s_t) \right] V(s_t) \\ &= 0. \end{aligned} \tag{13}$$

For $V = V^\pi$, the rightmost term in Equation (12) also gives no contribution:

$$\begin{aligned} \mathbb{E}_{\substack{s_{t:\infty}, \\ a_{t:\infty}}} \left[\nabla_\theta \log \pi_\theta(a_t \mid s_t) \cdot \delta_{t'}^{V^\pi} \right] &= \mathbb{E}_{\substack{s_{t:\infty}, \\ a_{t:\infty}}} \left[\nabla_\theta \log \pi_\theta(a_t \mid s_t) \cdot \mathbb{E}_{s_{t'}, a_{t'}} \left[\delta_{t'}^{V^\pi} \right] \right] \\ &= 0. \end{aligned} \tag{14}$$

The last equation follows because $\mathbb{E}_{s_{t'}, a_{t'}} \left[\delta_{t'}^{V^\pi} \right] = \mathbb{E}_{s_{t'}, a_{t'}} \left[c_{t'} + \gamma V^\pi(s_{t'+1}) - V^\pi(s_{t'}) \right] = 0$.

Hence, if $V = V^\pi$, Equation (12) can be interpreted as follows: the leftmost term encodes the contribution of predecessors of a_t to $\hat{Q}_t$, the rightmost term encodes the contribution of successors of a_t to $\hat{Q}_t$, and the middle term encodes the contribution of a_t itself. The policy gradient estimator in Equation (9) is based on using just the middle term in Equation (12), i.e., subtracting out the zero-mean contributions from the past and future.

The first way to incorporate an approximate value function V into policy gradient estimation is as a baseline, where we replace $\hat{Q}_t$ by $\hat{Q}_t - V(s_t)$ in Equation (10). An accurate approximation $V \approx V^\pi$ will reduce the variance of the policy gradient estimator, and the baseline will not introduce any bias even if $V(s_t) \neq V^\pi(s_t)$ [6]. However, as we show later, greater variance reduction can be achieved by incorporating the value function at future time steps.

A second way we can incorporate the value function is to truncate the sum of costs after some finite number of timesteps, using the value function as a terminal cost. Consider the sum $\hat{Q}_t - V(s_t) = \sum_{l=0}^{\infty} \gamma^l \delta_{t+l}^V$. All of the terms $\nabla_\theta \log \pi_\theta(a_t \mid s_t) \delta_{t+l}^V$ with $l > 0$ have expectation zero for $V = V^\pi$. Hence we can construct a policy gradient estimator that drops these terms (corresponding to long-term dependencies of actions), which introduces no bias for $V = V^\pi$. The limiting case of this procedure, where the sum is truncated immediately, corresponds to the actor-critic method, which uses $c_t + \gamma V(s_{t+1}) - V(s_t)$ as the advantage estimator.

Now we are ready to describe estimators that drop off the terms with large delay l, to reduce variance. Define the truncated advantage estimator $\hat{A}_t^{V,k} := \sum_{l=0}^{k-1} \gamma^l \delta_{t+l}^V = -V_t + c_t + \gamma c_{t+1} + \cdots + \gamma^{k-1} c_{t+k-1} + \gamma^k V_{t+k}$. $\hat{A}_t^{V,k}$ is an unbiased estimator (of the advantage function) when $V = V^\pi$. We can take a linear combination of these truncated estimators $\hat{A}^k$ with different horizons k, giving us an estimator that averages the value function at multiple states. We will focus on the exponentially-weighted average, defined as

$$\hat{A}_t^\lambda = (1 - \lambda) \sum_{k=0}^{\infty} \lambda^k \hat{A}_t^k, \tag{15}$$

where $\lambda \in (0, 1)$. Rearrangement of this sum gives

$$\hat{A}_t^\lambda = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}^V. \tag{16}$$

Hence, $\hat{A}_t^\lambda$ can be interpreted as a discounted sum of Bellman error terms δ_t^V, with discount $\gamma\lambda$. Note that TD(λ) for value function estimation involves the same sum. The resulting policy gradient estimator, which

we call the generalized advantage estimator, is given by

$$\mathbf{g}^{\gamma}_{\text{GAE}} = \mathbb{E}_{\substack{s_{0:\infty}, \\ a_{0:\infty}}} \left[\sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta^V_{t+l} \right]. \tag{17}$$

Since each truncated estimator $\hat{A}^{V,k}_t$ is unbiased for $V = V^\pi$, it follows that the estimator from Equation (17) is also unbiased for $V = V^\pi$.

Taking $\lambda = 1$ and expanding the telescoping sum, we obtain the baselined Monte-Carlo policy gradient estimator:

$$\mathbf{g}^{\gamma} = \mathbb{E}_{\substack{s_{0:\infty}, \\ a_{0:\infty}}} \left[\sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \hat{A}^{\lambda=1}_t \right] \tag{18}$$

$$= \mathbb{E}_{\substack{s_{0:\infty}, \\ a_{0:\infty}}} \left[\sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t \mid s_t) (\hat{Q}_t - V(s_t)) \right] \tag{19}$$

Taking $\lambda \to 0$, all of the δ_{t+l} terms in Equation (16) drop except the first one, with $l = 0$, yielding

$$\mathbf{g}^{\gamma} = \mathbb{E}_{\substack{s_{0:\infty}, \\ a_{0:\infty}}} \left[\sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \hat{A}^{\lambda=0}_t \right] \tag{20}$$

$$= \mathbb{E}_{\substack{s_{0:\infty}, \\ a_{0:\infty}}} \left[\sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \delta_t \right], \tag{21}$$

which uses the "one-step" advantage estimator $\hat{A}^{\lambda=0}_t = \delta_t = c_t + \gamma v_{t+1} - v_t$, analogously to actor-critic methods [10]. Thus these well-known estimators are boundary cases of GAE.

At this point it is worth clarifying the differing roles of γ and λ, both of which introduce bias-variance tradeoffs. Taking $\gamma < 1$ introduces bias, regardless of the value function's accuracy. Also, crucially, γ determines the scale of the value function $V^{\pi,\gamma}$. On the other hand, $\lambda < 1$ introduces bias only when the value function is inaccurate. In our experiments that follow, the best value of λ (empirically) is as low as 0.9, whereas the best γ was approximately 0.99, and $\gamma = 0.9$ would lead to a disastrous amount of bias.

4 Interpretation as Cost Shaping

Cost shaping refers to the following transformation of the cost function of an MDP: let $\Phi : \mathcal{S} \to \mathbb{R}$ be an arbitrary scalar-valued function on state space, and define the transformed cost function $\tilde{c}$ by

$$\tilde{c}(s, a, s') = c(s, a, s') + \gamma\Phi(s') - \Phi(s). \tag{22}$$

As pointed out by Ng et al. [11], this shaping transformation does not alter the optimal policy. In fact, it leaves the advantage function A^π unchanged for any policy π. This follows from the observation that if we consider any two trajectories starting from the same state, the difference in their discounted costs is unchanged by the shaping transformation. Note that if Φ happens to be the value function V^π (in the original MDP), then the new MDP has the interesting property that V^π is zero at every state. Now it becomes clear that the terms δ^V_t in the policy gradient estimators of Section 2 can be interpreted as reshaped costs, using the approximate value function $\Phi = V$ for shaping.

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

$$\chi(l; s, a) = \mathbb{E}\left[c_t \mid s_0 = s, a_0 = a\right] - \mathbb{E}\left[c_t \mid s_0 = s\right]. \tag{23}$$

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 costs are correspond to nonzero values of the response function for $l \gg 0$.

Given the shaped cost function $\tilde{c}$ using $\Phi = V^\pi$, we have $\mathbb{E}\left[\tilde{c}_{t+l} \mid s_t, a_t\right] = \mathbb{E}\left[\tilde{c}_{t+l} \mid s_t\right] = 0$ for $l > 0$, i.e., the response function is only nonzero at $l = 0$. Therefore, this shaping transformation turns a temporally extended response into an immediate response.

Given that V^π completely reduces the temporal spread of the response function, we would hope that a good approximation V would partially reduce it. This observation suggests an interpretation of Equation (17): take the original Monte-Carlo estimator for the policy gradient in Equation (10), reshape the costs using V to shrink the temporal extent of the response function, and then introduce a "steeper" discount $\gamma\lambda$ to cut off the terms corresponding to long temporal delays.

Having defined response functions, 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 (9) has a sum of terms of the form

$$A^{\pi,\gamma}(s_t,a_t)\nabla_\theta \log \pi_\theta(a_t \mid s_t) = \sum_{l=0}^{\infty} \gamma^l \chi(l;s,a)\nabla_\theta \log \pi_\theta(a_t \mid s_t). \tag{24}$$

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. Intuitively, we expect $\chi(l;s,a) \to 0$ as $l \to \infty$ for most MDPs—that is, the effect of an action on costs is "forgotten" after enough timesteps.

5 Value Function Estimation

A variety of different methods can be used to estimate the value function (see, e.g., [3]). When using a nonlinear function approximator to represent the value function, the simplest approach is solve a nonlinear regression problem:

$$\underset{\phi}{\text{minimize}} \sum_{n=1}^{N} \|V_\phi(s_n) - \hat{V}_n\|^2, \tag{25}$$

where $\hat{V}_t = \sum_{l=0}^{\infty} \gamma^l c_{t+l}$ is the discounted sum of costs, 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 [14]. [1]

For the experiments in this work, we used a trust region method to optimize the value function in each iteration of a policy iteration 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:

$$\begin{aligned} &\underset{\phi}{\text{minimize}} \sum_{n=1}^{N} \|V_\phi(s_n) - \hat{V}_n\|^2 \\ &\text{subject to} \quad \frac{1}{N}\sum_{n=1}^{N} \frac{\|V_\phi(s_n) - V_{\phi_{\text{old}}}(s_n)\|^2}{2\sigma^2} \le \epsilon. \end{aligned} \tag{26}$$

[1] Another natural choice is to compute target values with an estimator based on the TD(λ) backup [3, 14], 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 (25).

We compute an approximate solution to the trust region problem using the conjugate gradient algorithm [20].

6 Experiments

We designed a set of experiments to investigate following questions:

1. What is the empirical effect of varying $\lambda \in [0, 1]$ when using generalized advantage estimation in a policy gradient algorithm?
2. Can generalized advantage estimation be used along with a policy gradient algorithm to robustly optimize large and complex policies?

6.1 Policy Optimization Algorithm

For the experiments that follow, we performed the policy updates using trust region policy optimization [13]. This method repeatedly computes policy updates that minimize $g^T(\theta - \theta_{old})$ subject to a constraint on the KL divergence between the old and new policy, yielding a step in the direction $\Delta \propto -A^{-1}g$, where A is the average Fisher information matrix.

6.2 Experimental Setup

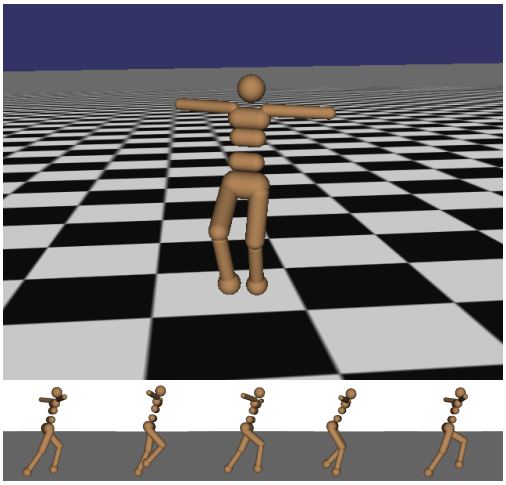

Figure 1: Top: robot model used for 3D bipedal locomotion. Bottom: a sequence of frames from the learned locomotion gait. Since the cost function was not concerned with the arms, the policy chose to use them to help with lateral balancing. The policy produces a naturalistic gait, which may be viewed on the project website: `https://sites.google.com/site/gaepapersupp/`

We evaluated our approach on the classic cart-pole balancing problem, as well as two challenging 3D locomotion tasks: humanoid walking, and non-humanoid walking for a model inspired by the "TARS" robot from the recent film Interstellar (2014). For all tasks, the value function was represented by a neural network. The 3D humanoid robot model is shown in Figure 1 and the TARS robot model is shown in Figure 2.

We used the same neural network architecture for the the humanoid and TARS locomotion tasks: a standard feedforward network with two hidden layers, each with 50 rectified linear (ReLU) units, given by $f(x) = \max(x, 0)$. The final output layer had linear activation. Moreover, we used the same architecture for the policy and value function, up until the final outputs of the networks, which have different dimensionality. (Note, however, that no parameters were shared between the policy and value function networks.)

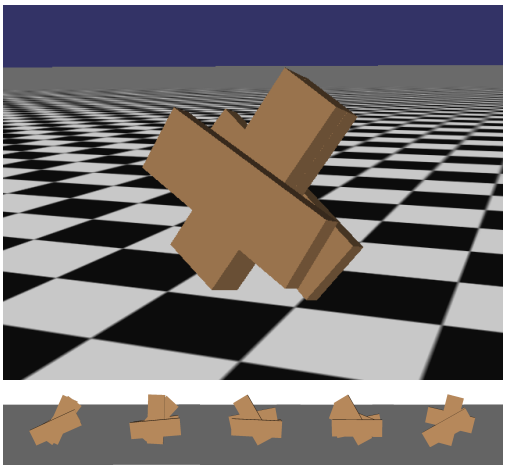

Figure 2: Simulated robot model inspired by the TARS character from the movie Interstellar.

Details of the cart-pole system are discussed by Barto et al. [1]. The locomotion tasks used the MuJoCo physics simulator [16]. The humanoid model has 33 state dimensions and 10 actuated degrees of freedom, while the TARS model has 21 state dimensions and 4 dimensional actions. Both locomotion tasks require the policy to deal with underactuation and contact discontinuities, as well as the high dimensionality of the policy. The initial state for both locomotion tasks consisted of a uniform distribution centered on a reference configuration, which required the policy to learn to deal with perturbations. Episodes were terminated when the height of the root link crossed a falling threshold, and the following terms were included in the cost function:

- Linear reward for forward progress
- Quadratic costs on torques
- Quadratic costs on foot impacts, to discourage hopping gaits
- A constant reward for being in a non-terminal state

This cost function is minimally informative about the task, with only the impact cost encoding prior information about the gait (namely, that a gentle rolling gait is preferred over hopping).

6.3 Experimental Results

We trained policies using different settings of λ for each task, with $\gamma = 0.99$. The results on the cart-pole are averaged across 21 experiments with different random initializations, while the other results are averaged over 3 experiments, due to the increased computational cost of training complex locomotion controllers.

The results for the cart-pole, shown in Figure 3, indicate that the algorithm learns fastest with $\lambda \in [0.92, 0.98]$. Learning is significantly faster when using a value function, and slightly faster than $\lambda = 1$, which corresponds to only using the value function as a baseline.

The results for the locomotion tasks, shown in Figure 4 and Figure 5, indicate that the GAE also achieves fast learning on these much larger domains. The best values for λ are again intermediate values in the range $\lambda \in [0.92, 0.98]$. The best policies on the humanoid locomotion task correspond to fast, naturalistic running gaits. Snapshots of these gaits are shown in Figure 1, and videos are provided on the project website: `https://sites.google.com/site/gaepapersupp/`. Note that, unlike prior work on learning locomotion policies [15, 5, 17], we use minimal prior information and general-purpose neural network policies, which do not encode basic concepts like balance and footsteps.

The non-humanoid TARS robot was able to learn a fast hopping gait that uses the inner blocks for propulsion and the outer blocks for stability. Unlike with the humanoid model, the non-humanoid model

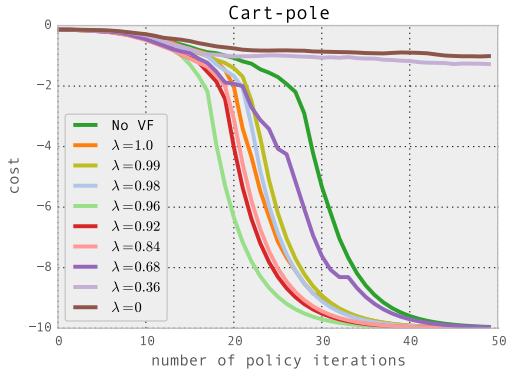

Figure 3: Learning curves for the cart-pole task, using generalized advantage estimation with varying values of λ. The results are averaged across 21 runs of the algorithm. Note that the fastest policy improvement is obtain by intermediate values of λ in the range $[0.92, 0.98]$.

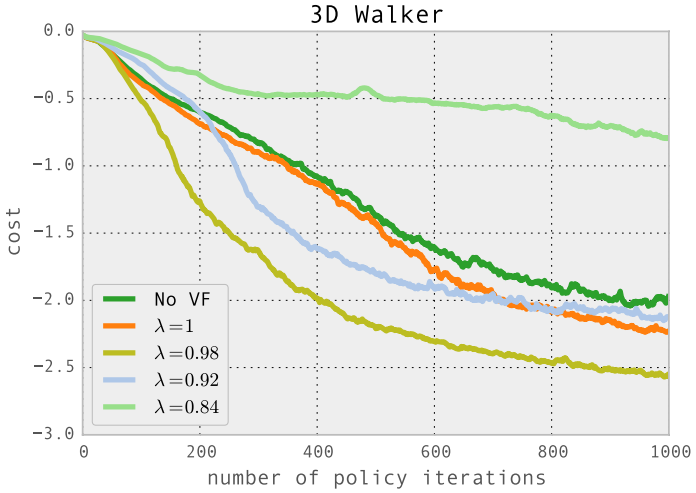

Figure 4: Learning curves for 3D bipedal locomotion, averaged across three runs of the algorithm. Setting $\lambda = 0.98$ achieved the fastest learning, and produced the best final policy.

offers little guidance about how to design an effective gait, and the ability of our method to synthesize an effective gait with minimal prior information showcases one of the key strengths of reinforcement learning: the ability to automatically learn effective control policies even on complex, unfamiliar systems. Unlike in the humanoid locomotion domain, $\lambda = 1$ performs just as well as the lower values of λ, i.e., GAE provides no advantage over the classic baselined estimator.

7 Discussion

In problems with large or continuous state spaces, nonlinear function approximators are required to estimate the value function with any degree of accuracy. However, such approximators typically offer few useful guarantees about the accuracy of the value function, and can often result in poor performance or divergence of ADP algorithms. In this paper, we describe a generalized advantage estimator for policy gradient methods, which allows us to incorporate approximate value function estimators into the policy optimization while explicitly controlling the tradeoff between bias and variance. This results in methods that combine the

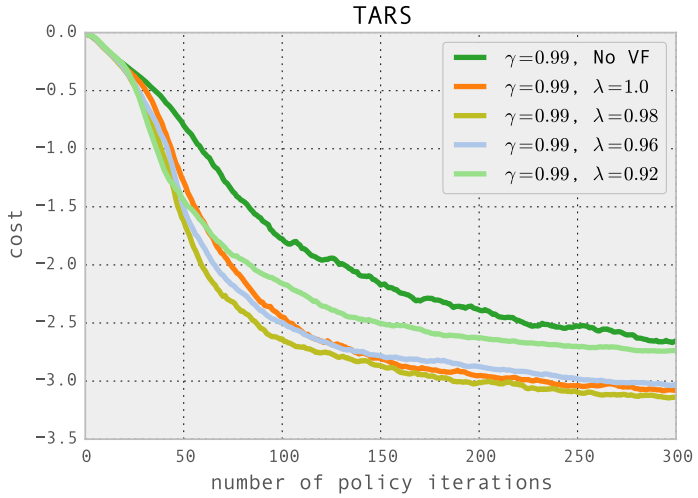

Figure 5: Learning curves for TARS robot, averaged across three runs of the algorithm. Setting $\lambda = 0.98$ again achieved the fastest learning, though the final policy was comparable to the one obtained with $\lambda = 1.0$.

reliability and scalability of policy gradient methods with the efficiency of value function estimation.

The generalized advantage estimator interpolates between two well-known estimators. Taking $\lambda = 1$, we obtain the baselined Monte Carlo estimator $\hat{A}_t = -V_t + \sum_{t=0}^{\infty} \gamma^l c_{t+l}$. Taking $\lambda = 0$, we obtain the one-step advantage estimator that has previously been proposed in the context of actor-critic methods: $\hat{A}_t = c_t + \gamma V_{t+1} - V_t$.

Our main experimental validation of generalized advantage estimation is in the domain of simulated robotic locomotion, which is challenging due to the high dimensionality of the state and action space and the complexity of the function approximators for the policy and value function, which we represent with large neural networks. In these domains, the $\lambda = 0$ estimator has excessively high bias, and the $\lambda = 1$ estimator has excessively high variance. As shown in our experiments, choosing an intermediate value of λ in the range $[0.9, 0.99]$ results in faster learning and, in the case of the bipedal humanoid gait, arrives at a better solution, enabling the algorithm to learn effective locomotion policies from scratch.

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 [4].

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.

References

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

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

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

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

[5] T. Geng, B. Porr, and F. Wörgötter. Fast biped walking with a reflexive controller and realtime policy searching. In *Advances in Neural Information Processing Systems (NIPS)*, 2006.

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

[7] Sham Kakade. Optimizing average reward using discounted rewards. In *Computational Learning Theory*, pages 605–615. Springer, 2001.

[8] Hajime Kimura. Reinforcement learning by stochastic hill climbing on discounted reward. In *Proceedings of the 12th International Conference on Machine Learning*, pages 295–303, 1995.

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

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

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

[12] Bruno Scherrer. Approximate policy iteration schemes: A comparison. *arXiv preprint arXiv:1405.2878*, 2014.

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

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

[15] R. Tedrake, T. Zhang, and H. Seung. Stochastic policy gradient reinforcement learning on a simple 3d biped. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2004.

[16] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 5026–5033. IEEE, 2012.

[17] Kevin Wampler and Zoran Popović. Optimal gait and form for animal locomotion. In *ACM Transactions on Graphics (TOG)*, volume 28, page 60. ACM, 2009.

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

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

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