$\lambda \in [0,1]$, 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 $(s_0, a_0, s_1, a_1, \ldots)$ 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

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

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