

# Deterministic Policy Optimization by Combining Pathwise and Score Function Estimators for Discrete Action Spaces

Daniel Levy, Stefano Ermon

Department of Computer Science, Stanford University  
{danilevy,ermon}@cs.stanford.edu

## Abstract

Policy optimization methods have shown great promise in solving complex reinforcement and imitation learning tasks. While model-free methods are broadly applicable, they often require many samples to optimize complex policies. Model-based methods greatly improve sample-efficiency but at the cost of poor generalization, requiring a carefully handcrafted model of the system dynamics for each task. Recently, hybrid methods have been successful in trading off applicability for improved sample-complexity. However, these have been limited to continuous action spaces. In this work, we present a new hybrid method based on an approximation of the dynamics as an expectation over the next state under the current policy. This relaxation allows us to derive a novel hybrid policy gradient estimator, combining score function and pathwise derivative estimators, that is applicable to discrete action spaces. We show significant gains in sample complexity, ranging between 1.7 and  $25\times$ , when learning parameterized policies on Cart Pole, Acrobot, Mountain Car and Hand Mass. Our method is applicable to both discrete and continuous action spaces, when competing pathwise methods are limited to the latter.

## Introduction

Reinforcement and imitation learning using deep neural networks have achieved impressive results on a wide range of tasks spanning manipulation (Levine et al. 2016; Levine and Abbeel 2014), locomotion (Silver et al. 2014), games (Mnih et al. 2015; Silver et al. 2016), and autonomous driving (Ho and Ermon 2016; Li, Song, and Ermon 2017). Model-free methods search for optimal policies without explicitly modeling the system’s dynamics (Williams 1992; Schulman et al. 2015). Most model-free algorithms build an estimate of the policy gradient by sampling trajectories from the environment and perform gradient ascent. However, these methods suffer from either very high sample complexity due to the generally large variance of the estimator, or are restricted to policies with few parameters.

On the other hand, model-based reinforcement learning methods learn an explicit model of the dynamics of the system. A policy can then be optimized under this model by back-propagating the reward signal through the learned dy-

namics. While these approaches can greatly improve sample efficiency, the dynamics model needs to be carefully hand-crafted for each task. Recently, hybrid approaches, at the interface of model-free and model-based, have attempted to balance sample-efficiency with generalizability, with notable success in robotics (Levine et al. 2016).

Existing methods, however, are limited to continuous action spaces (Levine and Abbeel 2014; Heess et al. 2015). In this work, we introduce a hybrid reinforcement learning algorithm for deterministic policy optimization that is applicable to continuous *and* discrete action spaces. Starting from a class of deterministic policies, we relax the corresponding policy optimization problem to one over a carefully chosen set of stochastic policies under approximate dynamics. This relaxation allows the derivation of a novel policy gradient estimator, which combines pathwise derivative and score function estimator. This enables incorporating model-based assumptions while remaining applicable to discrete action spaces. We then perform gradient-based optimization on this larger class of policies while slowly annealing stochasticity, converging to an optimal deterministic solution. We additionally justify and bound the dynamics approximation under certain assumptions. Finally, we complement this estimator by a scalable method to estimate the dynamics, first introduced in (Levine and Abbeel 2014), and with a general extension to non-differentiable rewards, rendering it applicable to a large class of tasks. Our contributions are as follows:

- We introduce a novel deterministic policy optimization method, that leverages a model of the dynamics and can be applied to any action space, whereas existing methods are limited to continuous action spaces. We also provide theoretical guarantees for our approximation.
- We show how our estimator can be applied to a broad class of problems by extending it to discrete rewards, and utilizing a sample-efficient dynamics estimation method (Levine and Abbeel 2014).
- We show that this method successfully optimizes complex neural network deterministic policies without additional variance reduction techniques. We present experiments on tasks with discrete action spaces where model-based or hybrid methods are not applicable. On these tasks, we show significant gains in terms of sample-complexity, ranging between 1.7 and  $25\times$ .

## Related Work

Sample efficiency is a key metric for RL methods, especially when used on real-world physical systems. Model-based methods improve sample efficiency at the cost of defining and learning task-specific dynamics models, while model-free methods typically require significantly more samples but are more broadly applicable.

Model-based methods approximate dynamics using various classes of models, spanning from gaussian processes (Deisenroth and Rasmussen 2011) to neural networks (Fu, Levine, and Abbeel 2016). Assuming the reward function is known and differentiable, the policy gradient can be computed exactly by differentiating through the dynamics model, thus enabling gradient-based optimization. Heess et al. (2015) extends this idea to stochastic dynamics and policies by expressing randomness as an exogenous variable and making use of the “re-parameterization trick”. This requires differentiability of the dynamics function w.r.t. both state and action, limiting its applicability to continuous action spaces. On the other hand, model-free algorithms are broadly applicable but require significantly more samples as well as variance reduction techniques through control variates (Mnih et al. 2016; Ho, Gupta, and Ermon 2016) or trust regions (Schulman et al. 2015) to converge in reasonable time.

Recently, hybrid approaches have attempted to balance sample efficiency with applicability. Levine and Abbeel (2014) locally approximate dynamics to fit locally optimal policies, which then serve as supervision for global policies. A good dynamics model also enables generating artificial trajectories to limit strain on the simulator (Gu et al. 2016). However, these works are once again limited to continuous action spaces. Our work can also be considered a hybrid algorithm that can handle and improve sample-efficiency for discrete action spaces as well.

## Background

In this section, we first present the canonical RL formalism. We then review the score function and pathwise derivative estimators, and their respective advantages. We show applications of these with first, the REINFORCE estimator (Williams 1992), followed by a standard method for model-based policy optimization consisting of back-propagating through the dynamics equation.

**Notations and definitions** Let  $\mathcal{X}$  and  $\mathcal{A}$  denote the state and action spaces, respectively.  $f$  refers to a deterministic dynamics function i.e.  $f : \mathcal{X} \times \mathcal{A} \rightarrow \mathcal{X}$ . A deterministic policy is a function  $\pi : \mathcal{X} \rightarrow \mathcal{A}$ , and a stochastic policy is a conditional distribution over actions given state, denoted  $\pi(a|x)$ . For clarity, when considering parameterized policies, stochastic ones will be functions of  $\phi \in \Phi$ , and deterministic ones functions of  $\theta \in \Theta$ . Throughout this work, dynamics will be considered deterministic.

We consider the standard RL setting where an agent interacts with an environment and obtains rewards for its actions. The agent’s goal is to maximize its expected cumulative reward. Formally, there exists an initial distribution  $p_0$

over  $\mathcal{X}$  and a collection of reward functions  $(r_t)_{t \leq T}$  with  $r_t : \mathcal{X} \rightarrow \mathbf{R}$ .  $x_0$  is sampled from  $p_0$ , and at each step, the agent is presented with  $x_t$  and chooses an action  $a_t$  according to a policy  $\pi$ .  $x_{t+1}$  is then computed as  $f(x_t, a_t)$ . In the finite horizon setting, the episode ends when  $t = T$ . The agent is then provided with the cumulative reward,  $r(x_0, \dots, x_T) = \sum_{t \leq T} r_t(x_t)$ . The goal of the agent is to find a policy  $\pi$  that maximizes the expected cumulative reward  $J(\pi) = \mathbf{E}_{x_0, a_0, \dots, a_{T-1}} \left[ \sum_{t \leq T} r_t(x_t) \right]$ .

## Score Function Estimator and Pathwise Derivative

We now review two approaches for estimating the gradient of an expectation. Let  $p(x; \eta)$  be a probability distribution over a measurable set  $\mathcal{X}$  and  $\varphi : \mathcal{X} \rightarrow \mathbf{R}$ . We are interested in obtaining an estimator of the following quantity:  $\nabla_{\eta} \mathbf{E}_{x \sim p(x; \eta)} [\varphi(x)]$ .

**Score Function Estimator** The score function estimator relies on the ‘log-trick’. It relies on the following identity (given appropriate regularity assumptions on  $p$  and  $\varphi$ ):

$$\nabla_{\eta} \mathbf{E}_{x \sim p(x; \eta)} [\varphi(x)] = \mathbf{E}_{x \sim p(x; \eta)} [\varphi(x) \nabla_{\eta} \log p(x; \eta)] \quad (1)$$

This last quantity can then be estimated using Monte-Carlo sampling. This estimator is very general and can be applied even if  $x$  is a discrete random variable, as long as  $\log p(x; \eta)$  is differentiable w.r.t.  $\eta$  for all  $x$ . However, it suffers from high-variance (Glasserman 2013).

**Pathwise Derivative** The pathwise derivative estimator depends on  $p(x; \eta)$  being re-parameterizable, i.e., there exists a function  $g$  and a distribution  $p'$  (independent of  $\eta$ ) such that sampling  $x \sim p$  is equivalent to sampling  $\epsilon \sim p'$  and computing  $x = g(\eta, \epsilon)$ . Given that observation,  $\nabla_{\eta} \mathbf{E}_{x \sim p(x; \eta)} [\varphi(x)] = \nabla_{\eta} \mathbf{E}_{\epsilon \sim p'(\epsilon)} [\varphi(g(\eta, \epsilon))] = \mathbf{E}_{\epsilon \sim p'(\epsilon)} [\nabla_{\eta} \varphi(g(\eta, \epsilon))]$ . This quantity can once again be estimated using Monte-Carlo sampling, but is conversely lower variance (Glasserman 2013). This requires  $\varphi(g(\eta, \epsilon))$  to be a differentiable function of  $\eta$  for all  $\epsilon$ .

We override  $J(\pi_{\theta})$  (resp.  $J(\pi_{\phi})$ ) as  $J(\theta)$  (resp.  $J(\phi)$ ), and aim at maximizing this objective function using gradient ascent.

## REINFORCE

Using the score function estimator, we can derive the REINFORCE rule (Williams 1992), which is applicable without assumptions on the dynamics or action space. With  $\pi_{\phi}$  a stochastic policy, we want to maximize  $J(\phi) = \mathbf{E}_{x_0, a_0, \dots, a_T} \left[ \sum_{t \leq T} r_t(x_t) \right]$  where  $a_t \sim \pi_{\phi}(\cdot | x_t)$ . Then the REINFORCE rule is  $\nabla_{\phi} J(\phi) = \mathbf{E}_{x_0, a_0, \dots, a_T} \left[ \sum_{t \leq T} \left( \sum_{t' \geq t} r_{t'}(x_{t'}) \right) \nabla_{\phi} \log \pi_{\phi}(a_t | x_t) \right]$ . We can estimate this quantity using Monte-Carlo sampling. This only requires differentiability of  $\pi_{\phi}$  w.r.t.  $\phi$  and does not assume knowledge of  $f$  and  $r$ . This estimator is however not applicable to deterministic policies.

## A Method for Model-based Policy Optimization

Let  $\pi_\theta$  be a deterministic policy, differentiable w.r.t. both  $x$  and  $\theta$ . Assuming we have knowledge of  $f$  and  $r$ , we can directly differentiate  $J$ :  $\nabla_\theta J = \mathbf{E}_{x_0} \left[ \sum_{t \leq T} \nabla_x r_t(x_t) \cdot \nabla_\theta x_t \right]$ . The first term,  $\nabla_x r_t$ , is easily computed given knowledge of the reward functions. The second term, given knowledge of the dynamics, can be computed by recursively differentiating  $x_{t+1} = f(x_t, \pi_\theta(x_t))$ , i.e.  $\nabla_\theta x_0 = 0$  and:

$$\nabla_\theta x_{t+1} = \nabla_x f \cdot \nabla_\theta x_t + \nabla_a f \cdot \nabla_\theta \pi_\theta(x_t) \quad (2)$$

This method can be extended to stochastic dynamics and policies by using a pathwise derivative estimator i.e. by reparameterizing the noise (Heess et al. 2015).

This method is applicable for a deterministic and differentiable policy w.r.t. both  $x$  and  $\theta$ , differentiable dynamics w.r.t. both variables and differentiable reward function. This implies that  $\mathcal{A}$  must be continuous. This model-based method aims at utilizing the dynamics and differentiating the dynamics equation.

### Relaxing the Policy Optimization Problem

Deterministic policies are optimal in a non-game theoretic setting. Our objective, in this work, is to perform hybrid policy optimization for deterministic policies for both continuous and discrete action spaces. In order to accomplish this, we present a relaxation of the RL optimization problem. This relaxation allows us to derive, in the subsequent section, a policy gradient estimator that differentiates approximate dynamics and leverages model-based assumptions for any action spaces. Contrary to traditional hybrid methods, *this does not assume differentiability of the dynamics function w.r.t. the action variable, thus elegantly handling discrete action spaces*. As in the previous section, we place ourselves in the finite-horizon setting. We assume terminal rewards and convex state space  $\mathcal{X}$ .

### Relaxing the dynamics constraint

In this section, we describe our relaxation. Starting from a class of deterministic policies  $D(\Theta)$  parameterized by  $\theta \in \Theta$ , we can construct a class of stochastic policies  $S(\Phi, \Lambda) \supset D(\Theta)$  parameterized by  $(\phi \in \Phi, \lambda \in \Lambda)$ , that can be chosen as close as desired to  $D(\Theta)$  by adjusting  $\lambda$ . On this extended class, we can derive a low-variance gradient estimator. We first explain the relaxation, then detail how to construct  $S(\Phi, \Lambda)$  from  $D(\Theta)$ , and finally, provide guarantees on the approximation.

Formally, the RL problem with deterministic policy and dynamics can be written as:

$$\begin{aligned} & \underset{\pi_\theta \in D(\Theta)}{\text{maximize}} && \mathbf{E}_{x_0 \sim p_0} [r(x_T)] \\ & \text{subject to} && x_{t+1} = f(x_t, \pi_\theta(x_t)) \end{aligned} \quad (3)$$

Given that  $\pi_\theta$  is deterministic, the constraint can be equivalently rewritten as:

$$x_{t+1} = \mathbf{E}_{a \sim \pi_\theta(\cdot|x_t)} [f(x_t, a)] \quad (4)$$

Having made this observation, we proceed to relaxing the optimization from over  $D(\Theta)$  to  $S(\Phi, \Lambda)$ , with the constraint now being over approximate and in particular differentiable dynamics. The relaxed optimization program is therefore:

$$\begin{aligned} & \underset{\pi_{\phi, \lambda} \in S(\Phi, \Lambda)}{\text{maximize}} && \mathbf{E}_{x_0 \sim p_0} [r(x_T)] \\ & \text{subject to} && x_{t+1} = \mathbf{E}_{a \sim \pi_{\phi, \lambda}(\cdot|x_t)} [f(x_t, a)] \end{aligned} \quad (5)$$

This relaxation casts the optimization to be over stochastic policies, which allows us to derive a policy gradient in Theorem 3, but under different, approximated dynamics. We later describe how to project the solution in  $S(\Phi, \Lambda)$  back to an element on  $D(\Theta)$ .

### Construction of $S(\Phi, \Lambda)$ from $D(\Theta)$

Here we show how to construct stochastic policies from deterministic ones while providing a parameterized ‘knob’ to control randomness and closeness to a deterministic policy.

**Discrete action spaces** The natural parameterization is as a softmax model. However, this requires careful parameterization, in order to ensure that all policies of  $D(\Theta)$  are included. We use the deterministic policy as a prior of which we can control the strength. Formally, we choose a class of parameterized functions  $\{g_\psi : \mathcal{X} \rightarrow \mathbf{R}^{|\mathcal{A}|}, \psi \in \Psi\}$ . For  $\theta, \psi$  and  $\lambda \geq 0$ , we can define the following stochastic policy  $\pi_{\theta, \psi, \lambda}$  s.t.  $\pi_{\theta, \psi, \lambda}(a|x) \propto \exp \left[ g_\psi(x)_a + \frac{\mathbf{1}_{a=\pi_\theta(x)}}{\lambda} \right]$ .

We have therefore defined  $S(\Phi, \Lambda)$  where  $\Phi \triangleq \Theta \times \Psi$  and  $\Lambda \triangleq [0, \infty)$ . We easily verify that  $D(\Theta) \subset S(\Phi, \Lambda)$ , as, for any  $\theta \in \Theta$ , we can choose an arbitrary  $\psi \in \Psi$  and define  $\phi = (\theta, \psi) \in \Phi$ , we then have  $\pi_\theta = \lim_{\lambda \rightarrow 0} \pi_{\phi, \lambda}$ .

**Continuous action spaces** In the continuous setting, a very simple parameterization is by adding Gaussian noise, of which we can control variance. Formally, given  $D(\Theta)$ , let  $\Phi \triangleq \Theta, \Lambda \triangleq [0, \infty)$  and  $S(\Phi, \Lambda) \triangleq \{x \mapsto \mathcal{N}(\pi_\theta(x), \lambda^2 I), \theta \in \Theta, \lambda \geq 0\}$ . The surjection can be derived by setting  $\lambda$  to 0. More complicated stochastic parameterizations can be easily derived as long as the density remains tractable.

**Rounding** We assume that there is a surjection from  $S(\Phi, \Lambda)$  to  $D(\Theta)$ , s.t. any stochastic policy can be made deterministic by setting  $\lambda$  to a certain value. For the above examples, the mapping consists of setting  $\lambda$  to 0.

Similar in spirit to simulated annealing for optimization (Kirkpatrick et al. 1983), we optimize over  $\phi$ , while slowly annealing  $\lambda$  to converge to an optimal solution in  $D(\Theta)$ .

### Theoretical Guarantees

Having presented the relaxation, we now provide theoretical justifications, to show, first, that given conditions on the stochastic policy, a trajectory computed with approximate dynamics under a stochastic policy is close to the trajectory computed with the true dynamics under a deterministic policy. We additionally present connections in the case where our dynamics are discretization of a continuous-time system.

**Bounding the deviation from real dynamics** In this paragraph, we assume that the action space is continuous. Given the terminal reward setting, the amount of approximation can be defined as the divergence between  $(x_t)_{t \leq T}$ , the trajectory from following a deterministic policy  $\pi_\theta$ , and  $(\tilde{x}_t)_{t \leq T}$ , the trajectory corresponding to the approximate dynamics with a policy  $\pi_{\phi, \lambda}$ . This will allow us to relate the optimal value of the relaxed program with the optimal value of the initial optimization problem.

**Theorem 1** (Approximation Bound). *Let  $\pi_\theta$  and  $\pi_{\phi, \lambda}$  be a deterministic and stochastic policy, respectively, s.t.  $\forall x \in \mathcal{X}, \mathbf{E}_{a \sim \pi_{\phi, \lambda}(\cdot|x)} a = \pi_\theta$ . Let us suppose that  $f$  is  $\rho$ -lipschitz and  $\pi_\theta$  is  $\rho'$ -lipschitz. We further assume that  $\sup_x \text{tr} [\text{Var}_{\pi_{\phi, \lambda}(\cdot|x)} a] \leq M$  and that  $\alpha \triangleq \rho(\rho' + 1) < 1$ . We have the following guarantee:*

$$\mathbf{E}_{x_0} \|x_T - \tilde{x}_T\| \leq \frac{1}{1 - \alpha} \sqrt{M} \quad (6)$$

Furthermore, if  $[\pi_\phi(\cdot|x)]_{x \in \mathcal{X}}$  are distributions of fixed variance  $\lambda^2 I$ , the approximation converges towards 0 when  $\lambda \rightarrow 0$ .

*Proof.* See Appendix.  $\square$

We know that solving the relaxed optimization problem will provide an upper-bound on the expected terminal reward from a policy in  $D(\Theta)$ . Given a  $\rho''$ -lipschitz reward function, this bound shows that the optimal value of the true optimization program is within  $\frac{\rho''}{1 - \alpha} \sqrt{M}$  of the optimal value of the relaxed one.

**Equivalence in continuous-time systems** The relaxation has strong theoretical guarantees when the dynamics are a discretization of a continuous-time system. With analogous notations as before, let us consider a continuous-time dynamical system:  $x(0) = x_0, \dot{x} = f(x(t), a(t))$ .

A  $\delta$ -discretization of this system can be written as  $x_{t+1} = x_t + \delta f(x_t, a_t)$ . We can thus write the dynamics of the relaxed problem:  $x_{t+1} = x_t + \delta \mathbf{E}_{a \sim \pi_{\phi, \lambda}(a|x_t)} [f(x_t, a)]$ .

Intuitively, when the discretization step  $\delta$  tends to 0, the policy converges to a deterministic one. In the limit, our approximation is the true dynamics for continuous time systems. Proposition 2 formalizes this idea.

**Proposition 2.** *Let  $(X_t^\delta)_{t \leq T}$  be a trajectory obtained from the  $\delta$ -discretized relaxed system, following a stochastic policy. Let  $x(t)$  be the continuous time trajectory. Then, with probability 1:*

$$X_t^\delta \rightarrow x(t) \quad (7)$$

*Proof.* See (Munos 2006).  $\square$

## Relaxed Policy Gradient

The relaxation presented in the previous section allows us to differentiate through the dynamics for any action space. In this section, we derive a novel deterministic policy gradient estimator, Relaxed Policy Gradient (RPG), that combines pathwise derivative and score function estimators, and is applicable to all action spaces. We then show how to apply our

estimator, using a sample-efficient and scalable method to estimate the dynamics, first presented in (Levine and Abbeel 2014). We conclude by extending the estimator to discrete reward functions, effectively making our algorithm applicable to a wide variety of problems. For simplicity, we omit  $\lambda$  (the stochasticity) from our derivations and consider it fixed.

## Estimator

We place ourselves in the relaxation setting. Letting  $\pi_\phi$  be a parameterized stochastic policy, we define  $J(\phi) = \mathbf{E}_{x_0, a_0, \dots, a_{T-1}} [r(x_T)]$ . Our objective is to find  $\phi^* = \arg \max_\phi J(\phi)$ . To that aim, we wish to perform gradient ascent on  $J$ .

**Theorem 3** (Relaxed Policy Gradient). *Given a trajectory  $(x_0, a_1, \dots, x_T)$ , sampled by following a stochastic policy  $\pi_\phi$ , we define the following quantity  $\hat{g} = \nabla_x r(x_T) \cdot \nabla_\phi x_T$  where  $\nabla_\phi x_T$  can be computed with the recursion defined by  $\nabla_\phi x_0 = 0$  and:*

$$\begin{aligned} \nabla_\phi x_{t+1} &= \nabla_x f(x_t, a_t) \cdot \nabla_\phi x_t \\ &+ f(x_t, a_t) \nabla_\phi \log \pi_\phi(a_t|x_t) \\ &+ f(x_t, a_t) \nabla_x \log \pi_\phi(a_t|x_t) \cdot \nabla_\phi x_t \end{aligned} \quad (8)$$

$\hat{g}$  is an unbiased estimator of the policy gradient of the relaxed problem, defined in Equation 5.

*Proof.* See Appendix.  $\square$

The presented RPG estimator is effectively a combination of pathwise derivative and score function estimators. Intuitively, the pathwise derivative cannot be used for discrete action spaces as it requires differentiability w.r.t. the action. To this end, we differentiate pathwise through  $x$  and handle the discrete portion with a score function estimator.

**Benefits of pathwise derivative** Gradient estimates through pathwise derivatives are considered lower variance (Glasserman 2013). In the context of RL, intuitively, REINFORCE suffers from high variance as it requires many samples to properly establish credit assignment, increasing or decreasing probabilities of actions seen during a trajectory based on the reward. Conversely, as RPG estimates the dependency of the state on  $\phi$ , the gradient can directly adjust  $\phi$  to steer the trajectory to high reward regions. This is possible because the estimator utilizes a model of the dynamics. While REINFORCE assigns credit indifferently to actions, RPG adjusts the entire trajectory.

However, when examining our expression for RPG, the computation requires the gradient of the dynamics w.r.t. the state, unlike REINFORCE. In the next section, we present a scalable and sample-efficient dynamics fitting method, further detailed in (Levine et al. 2016), to estimate this term.

## Scalable estimation of the dynamics

The Relaxed Policy Gradient estimator can be computed exclusively from sampled trajectories if provided with the state-gradient of the dynamics  $\nabla_x f$ . However, this is not an information available to the agent and thus needs to be estimated from samples. We review a method, first presented

in (Levine and Abbeel 2014), that provides estimation of the state-gradient of the dynamics, incorporating information from multiple sampled trajectories. Although the dynamics are deterministic, we utilize a stochastic estimation to account for modeling errors.

Formally, we have  $m$  sampled trajectories  $\{\tau_i\}_{i \leq m}$  of the form  $\{x_t^i, a_t^i, x_{t+1}^i\}_{i \leq m, t \leq T}$  from the dynamical system. Our goal is not only to accurately model the dynamics of the system, but also to estimate correctly the state-gradient. This prevents us from simply fitting a parametric function  $f_\phi$  to the samples. Indeed, this approach would provide no guarantee that  $\nabla_x f_\phi$  is a good approximation of  $\nabla_x f$ . This is especially important as the variance of the estimator is dependent on the quality of our approximation of  $\nabla_x f$  and not the approximation of  $f$ .

In order to fit a dynamics model, we choose to locally model the dynamics of the discretized stochastic process as  $X_{t+1} \sim \mathcal{N}(A_t X_t + B_t a_t + c_t, F_t)$ , parameterized by  $(\{A_t, B_t, C_t\})_{t \leq T}$ . We choose this approach as it does not model global dynamics but instead a good time-varying local model around the latest sampled trajectories, which corresponds to the term we want to estimate. Under that model, the term we are interested in estimating is then  $A_t$ .

While this approach is simple, it requires a large number of samples to be accurate. To greatly increase sample-efficiency, we assume a prior over sampled trajectories. We use the GMM approach of (Levine and Abbeel 2014) which corresponds to softly piecewise linear dynamics. At each iteration, we update our prior using the current trajectories and fit our model. This allows us to utilize both past trajectories as well as nearby time steps that are included in the prior. Another key advantage of this stochastic estimation is that it elegantly handles discontinuous dynamics models. Indeed, by averaging the dynamics locally around the discontinuities, it effectively smooths out the discontinuities of the underlying deterministic dynamics. We refer to (Levine et al. 2016) for more detailed derivations.

### Extension to discrete rewards

Prior to this work, estimators incorporating models of the dynamics such as (Heess et al. 2015; Levine and Abbeel 2014) were constrained to continuously differentiable rewards. We present an extension of this type of estimators to a class of non-continuous rewards. To do so, we make assumptions on the form of the reward and approximate it by a smooth function.

We assume that  $\mathcal{X} = \mathbf{R}^n$  and that the reward can be written as a sum of indicator functions, i.e.:

$$r(x) = \sum_{i=1}^k \lambda_i \mathbf{1}_{x \in K_i} \quad (9)$$

where  $(K_i)_{i \leq k}$  are compact subsets of  $\mathbf{R}^n$ . This assumption covers a large collection of tasks where the goal is to end up in a compact subspace of  $\mathcal{X}$ . For each  $K_i$ , we are going to approximate  $\mathbf{1}_{x \in K_i}$  by an arbitrarily close smooth function.

**Proposition 4** (Smooth approximation of an indicator function). *Let  $K$  be a compact of  $\mathbf{R}^n$ . For any neighborhood  $\Omega$  of  $K$ , there exists,  $\psi : \mathbf{R}^n \rightarrow \mathbf{R}$ , smooth, s.t.  $\mathbf{1}_K \leq \psi \leq \mathbf{1}_\Omega$ .*

*Proof.* See Appendix.  $\square$

We can now approximate each  $(\mathbf{1}_{K_i})_{i \leq k}$  by a smooth function  $(\psi_i)_{i \leq k}$ . Given this surrogate reward  $\tilde{r} \triangleq \sum_{i=1}^k \lambda_i \psi_i$ , we can apply our estimator. In practice, however, discrete reward functions are often defined as  $r(s) = \lambda \mathbf{1}_{h(s) \geq 0}$ , where  $h$  is a function from  $\mathcal{X}$  to  $\mathbf{R}$ . We approximate the reward function by  $\tilde{r}_\alpha(s) = \lambda \sigma(\alpha h(s))$  ( $\sigma$  being the sigmoid function). If  $h$  is  $\mathcal{C}^1$ , then  $\tilde{r}$  is too and  $\tilde{r}_\alpha$  pointwise converges to  $r$ . We present in the Appendix an example of the approximate reward functions for the Mountain Car task.

Given these arbitrarily close surrogate rewards, we can now prove a continuity result, i.e. the sequence of optimal policies under the surrogate rewards converges to the optimal policy.

**Proposition 5** (Optimal policies under surrogate reward functions). *Let  $r$  be a reward function defined as  $r = \mathbf{1}_K$  and  $\pi^*$  the optimal policy under this reward. Let us define  $V^\pi(x)$  to be the value of state  $x$  under the policy  $\pi$ . Then, there exists a sequence of smooth reward functions  $(r_n)_{n \in \mathbf{N}}$  s.t. if  $\pi_n$  is optimal under the reward  $r_n$ ,  $\forall x \in \mathcal{X}$ ,  $\lim_{n \rightarrow \infty} V^{\pi_n}(x) = V^{\pi^*}$ .*

*Proof.* See Appendix.  $\square$

## Practical Algorithm

In Algorithm 1, we present a practical algorithm to perform deterministic policy optimization based on the previously defined RPG. In summary, given  $D(\Theta)$ , we construct our extended class of stochastic policies  $S(\Phi, \Lambda)$ . We then perform gradient-based optimization over  $\phi$ , i.e. stochastic policies, while slowly annealing  $\lambda$ , converging to a policy in  $D(\Theta)$ . We easily extend the estimator to the case of non-terminal rewards as:

$$\nabla_\phi J(\phi, \lambda) = \sum_{t \leq T} \nabla_x r(x_t) \cdot \nabla_\phi x_t \quad (10)$$

## Experiments

We empirically evaluate our algorithm to investigate the following questions: 1) How much does RPG improve sample complexity? 2) Can RPG learn an optimal policy for the true reward using a smooth surrogate reward? 3) How effective is our approximation of the dynamics?

In an attempt to evaluate the benefits of the relaxation compared to other estimators as fairly as possible, we do not use additional techniques, such as trust regions (Schulman et al. 2015). Both our method and the compared ones can incorporate these improvements, but we leave study of these more sophisticated estimators for future work.

### Classical Control Tasks

We apply our Relaxed Policy Gradient (RPG) to a set of classical control tasks with *discrete* action spaces: Cart Pole (Barto, Sutton, and Anderson 1983), Acrobot (Sutton 1996), Mountain Car (Sutton 1996) and Hand Mass (Munos

---

**Algorithm 1** Relaxed Policy Gradient

---

**Inputs:** Environment giving  $x_{t+1}, \nabla_x r(x_t)$  given  $x_t, a_t$ , deterministic class of policies  $D(\Theta)$ , stochastic class of policies  $S(\Phi, \Lambda)$ , number of training episodes  $N_{\text{ep}}$ , learning rate schedule  $(\alpha_N)_{N \leq N_{\text{ep}}}$ , annealing schedule  $(\gamma_N)_{N \leq N_{\text{ep}}}$ , initial parameters  $\phi_0 \in \Phi, \lambda_0 \in \Lambda$ .

Initialize  $\phi \leftarrow \phi_0$  and  $\lambda \leftarrow \lambda_0$ .

**for**  $N = 1$  to  $N_{\text{ep}}$  **do**

  Sample  $\{\tau_i\}_{i \leq m}$  with policy  $\pi_{\phi, \lambda}$

  Update GMM prior over dynamics

  Fit dynamics  $(A_t, B_t, C_t)_{t \leq T}$

**for**  $i = 1$  to  $m$  **do**

$\hat{g}_i \leftarrow 0$

$\nabla_{\phi} x_0 \leftarrow 0$

**for**  $t = 0$  to  $T - 1$  **do**

$\hat{g}_i \leftarrow \hat{g}_i + \nabla_x r(x_t) \cdot \nabla_{\phi} x_{t-1}$

$\hat{f}_t = x_{t+1} - x_t$

$\nabla_{\phi} x_{t+1} \leftarrow A_t \cdot \nabla_{\phi} x_t + \hat{f}_t \nabla_{\phi} \log \pi_{\phi, \lambda}(a_t | x_t)$   
       $\quad + \hat{f}_t \nabla_x \log \pi_{\phi, \lambda}(a_t | x_t) \cdot \nabla_{\phi} x_t$

**end for**

$\hat{g}_i \leftarrow \hat{g}_i + \nabla_x R(x_T) \cdot \nabla_{\phi} x_T$

**end for**

$\phi \leftarrow \phi + \alpha_N \hat{g}$

$\lambda \leftarrow \gamma_N \lambda$

**end for**

**Return:**  $\pi_{\phi, \lambda=0} \in D(\Theta)$

---

2006). For the first three tasks, we used the OpenAI Gym (Brockman et al. 2016) implementation and followed (Munos 2006) for the implementation of Hand Mass. A diagram of our tasks are presented in Figure 1.

**Baselines** We compare our methods against two different baselines: a black-box derivative free optimization algorithm called Cross-Entropy Method (CEM) (Szita and Lörincz 2006) and the Actor-Critic algorithm (A3C) (Mnih et al. 2016). CEM is an evolutionary-type algorithm that samples a different set of parameters at each step, simulates using these parameters and keeps only the top 2%, using those to re-fit the distribution over parameters. CEM is known to

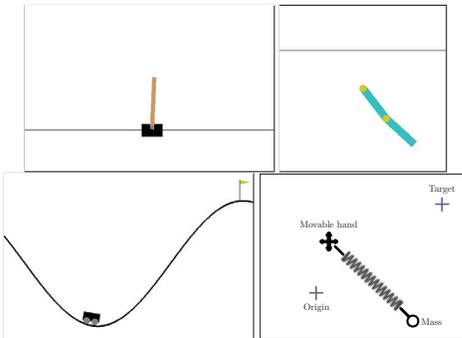


Figure 1: *Top row:* Cart Pole, Acrobot. *Bottom row:* Mountain Car, Hand Mass.

work well (Duan et al. 2016), but lacks sample efficiency as the algorithm does not exploit any structure. A3C is a variant of REINFORCE (Williams 1992) that learns a value function to reduce the variance of the estimator. For each task and algorithm, we evaluate for 5 distinct random seeds.

## Results and Analysis

We present the learning curves for all tasks in Figure 2. Even when training is done with surrogate rewards, in both instances we report the actual reward. We do not show CEM learning curves as these are not comparable due to the nature of the algorithm; each CEM episode is equivalent to 20 RPG or A3C episodes, we instead report final performance in Tables 1 and 2. For all tasks, the policy is parameterized by a 2-layer neural network with  $\tanh$  non-linearities and 64 hidden units, with a softmax layer outputting a distribution over actions. In practice, we optimize over stochastic policies with a fixed  $\lambda$ . In our experiments, it converged to a near deterministic policy since the optimization is well-conditioned enough in that case. The policies are trained using Adam (Kingma and Ba 2014).

We evaluate differently depending on the task: since Cartpole and Acrobot have a fixed reward threshold at which they are considered solved, for these we present the number of training samples needed to reach that performance (in Table 1). In contrast, for Mountain Car and Hand Mass, we report (in Table 2) the reward achieved with a fixed number of samples. Both of these metrics are meant to evaluate sample-efficiency.

**Sample Complexity** As shown in Tables 1 and 2, our algorithm clearly outperforms A3C across all tasks, requiring between 1.7 and 3 times less samples to solve the tasks and showing significantly better performance for the same number of samples. As shown in Table 2, RPG performs better than CEM in Hand Mass and within 20% for Mountain Car, *despite using 20 times less samples*. We also note that CEM is particularly well suited for Acrobot, as it is a derivative-free method that can explore the space of parameters very fast and find near-optimal parameters quickly when the optimal policies are fairly simple, explaining its impressive sample-complexity on this specific task. We additionally point out that full plots of performance against number of samples are reported for all tasks in Figure 2, and the numbers in Tables 1 and 2 can all be extracted from there.

**Robustness, Variance and Training Stability** Overall, our method was very robust to the choice of hyper-parameters such as learning rate or architecture. Indeed, our policy was trained on all tasks with the same learning rate  $\alpha = 10^{-2}$ , whereas different learning rates were cross-validated for A3C. When examining the training curves, our estimator demonstrates significantly less variance than A3C and a more stable training process. In tasks where the challenges are exploratory (Acrobot or Mountain Car), RPG’s exploration process is guided by its model of the dynamics while A3C’s is completely undirected, often leading to total failure. The same phenomenon can be observed on control

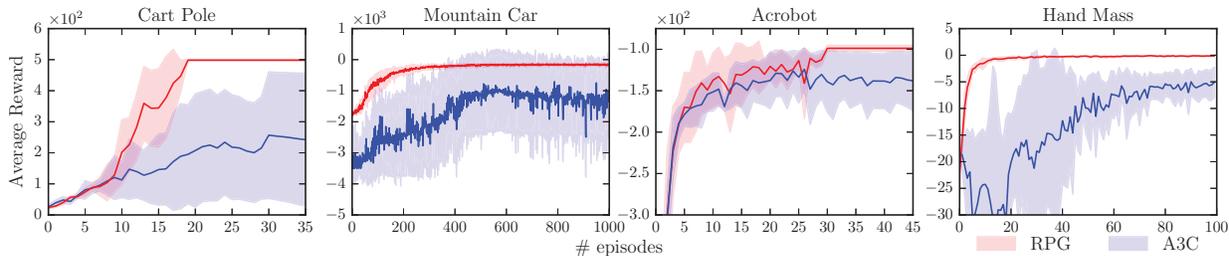


Figure 2: Mean rewards over 5 random seeds for classical control tasks. Performance of RPG is shown against A3C.

Method	Samples until solve
Cart Pole (Threshold = 495)	
CEM	470
A3C	279
RPG	91
Acrobot (Threshold = -105)	
CEM	180
A3C	541
RPG	311

Table 1: Average numbers of samples until the task is solved for the Cart Pole and Acrobot tasks for RPG, A3C and CEM.

Method	Samples	Performance
Hand Mass (150 episodes)		
CEM	50	$-0.086 \pm 0.01$
A3C	2	$-2.42 \pm 2.19$
RPG	2	$-0.026 \pm 0.01$
Mountain Car (1000 episodes)		
CEM	200	$-110.3 \pm 1.2$
A3C	10	$-176 \pm 38.6$
RPG	10	$-131.3 \pm 5.3$

Table 2: Average mean rewards for the Hand Mass and Mountain Car tasks for RPG, A3C and CEM.

tasks (Hand Mass or Cart Pole), where A3C favors bad actions seen in high reward trajectories, leading to unstable control.

**Approximate Reward** On all tasks except Hand Mass, our estimator was trained using approximate smoothed rewards. The performances reported in Figure 2 and Table 2 show that this did not impair training at all. Regarding the baselines, it is interesting to note that since CEM does not leverage the structure of the RL problem, it is natural to train with the real rewards. For A3C, we experimented with the smoothed rewards and obtained comparable numbers.

## Limitations

In this section, we explore limitations of our method. To leverage the dynamics of the RL problem, we trade-off some flexibility in the class of problems that model-free RL can tackle for better sample complexity. This can also be seen as an instance of the bias/variance trade-off.

**Limitations on the reward function** While we presented a general way to extend this estimator for rewards in the discrete domain, such function approximations can be difficult to construct for high-dimensional state-spaces such as Atari games. Indeed, one would have to fit the indicator function of a very low dimensional manifold - corresponding to the set of images encoding the state of a given game score - living in a high-dimensional space (order of  $\mathbf{R}^{200 \times 200 \times 3}$ ).

**Limitations on the type of tasks** While we show results on classical control tasks, our estimator is broadly applicable to all tasks where the dynamics can be estimated reasonably well. This has been shown to be possible on a number of locomotion and robotics tasks (Levine and Abbeel 2014; Heess et al. 2015). However, our work is not directly applicable to raw pixel input tasks such as the Atari domain.

**Computational overhead** Compared to REINFORCE, our model presents some computational overhead as it requires evaluating  $(\nabla_{\phi} x_t)_{t \leq T}$  as well as fitting  $T$  dynamics matrices. In practice, this is minor compared to other training operations such as sampling trajectories or computing necessary gradients. In our experiments, computing the  $(\nabla_{\phi} x_t)_{t \leq T}$  amounts to less than 1% of overhead while dynamics estimation constitutes about 17%.

## Discussion

In this work, we presented a method to find an optimal deterministic policy for any action space and in particular discrete actions. This method relies on a relaxation of the optimization problem to a carefully chosen, larger class of stochastic policies. On this class of policies, we can derive a novel, low-variance, policy gradient estimator, Relaxed Policy Gradient (RPG), by differentiating approximate dynamics. We then perform gradient-based optimization while slowly annealing the stochasticity of our policy, converging to a deterministic solution. We showed how this method can be

successfully applied to a collection of RL tasks, presenting significant gains in sample-efficiency and training stability over existing methods. Furthermore, we introduced a way to apply this algorithm to non-continuous reward functions by learning a policy under a smooth surrogate reward function, for which we provided a construction method. It is also important to note that our method is easily amenable to problems with stochastic dynamics, assuming one can reparameterize the noise.

This work also opens the way to promising future extensions of the estimator. For example, one could easily incorporate imaginary rollouts (Gu et al. 2016) with the estimated dynamics model or trust regions (Schulman et al. 2015). Finally, this work can also be extended more broadly to gradient estimation for any discrete random variables.

**Acknowledgments** The authors thank Aditya Grover, Jiaming Song and Steve Mussmann for useful discussions, as well as the anonymous reviewers for insightful comments and suggestions. This research was funded by Intel Corporation, FLI, TRI and NSF grants #1651565, #1522054, #1733686.

## References

- Barto, A. G.; Sutton, R. S.; and Anderson, C. W. 1983. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics* (5):834–846.
- Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. Openai gym. *arXiv preprint arXiv:1606.01540*.
- Deisenroth, M., and Rasmussen, C. E. 2011. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, 465–472.
- Duan, Y.; Chen, X.; Houthoofd, R.; Schulman, J.; and Abbeel, P. 2016. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning*, 1329–1338.
- Fu, J.; Levine, S.; and Abbeel, P. 2016. One-shot learning of manipulation skills with online dynamics adaptation and neural network priors. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, 4019–4026. IEEE.
- Glasserman, P. 2013. *Monte Carlo methods in financial engineering*, volume 53. Springer Science & Business Media.
- Gu, S.; Lillicrap, T.; Sutskever, I.; and Levine, S. 2016. Continuous deep q-learning with model-based acceleration. *arXiv preprint arXiv:1603.00748*.
- Heess, N.; Wayne, G.; Silver, D.; Lillicrap, T.; Erez, T.; and Tassa, Y. 2015. Learning continuous control policies by stochastic value gradients. In *Advances in Neural Information Processing Systems*, 2944–2952.
- Ho, J., and Ermon, S. 2016. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, 4565–4573.
- Ho, J.; Gupta, J. K.; and Ermon, S. 2016. Model-free imitation learning with policy optimization. In *Proceedings of the 33rd International Conference on Machine Learning*.
- Kingma, D., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- Kirkpatrick, S.; Gelatt, C. D.; Vecchi, M. P.; et al. 1983. Optimization by simulated annealing. *science* 220(4598):671–680.
- Levine, S., and Abbeel, P. 2014. Learning neural network policies with guided policy search under unknown dynamics. In *Advances in Neural Information Processing Systems*, 1071–1079.
- Levine, S.; Finn, C.; Darrell, T.; and Abbeel, P. 2016. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research* 17(39):1–40.
- Li, Y.; Song, J.; and Ermon, S. 2017. Infogail: Interpretable imitation learning from visual demonstrations. *Advances in Neural Information Processing Systems*.
- Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529–533.
- Mnih, V.; Badia, A. P.; Mirza, M.; Graves, A.; Lillicrap, T. P.; Harley, T.; Silver, D.; and Kavukcuoglu, K. 2016. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*.
- Munos, R. 2006. Policy gradient in continuous time. *Journal of Machine Learning Research* 7(May):771–791.
- Schulman, J.; Levine, S.; Moritz, P.; Jordan, M. I.; and Abbeel, P. 2015. Trust region policy optimization. In *International Conference on Machine Learning (ICML 2015)*.
- Silver, D.; Lever, G.; Heess, N.; Degris, T.; Wierstra, D.; and Riedmiller, M. 2014. Deterministic policy gradient algorithms. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, 387–395.
- Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. 2016. Mastering the game of go with deep neural networks and tree search. *Nature* 529(7587):484–489.
- Sutton, R. S. 1996. Generalization in reinforcement learning: Successful examples using sparse coarse coding. *Advances in neural information processing systems* 1038–1044.
- Szita, I., and Lörincz, A. 2006. Learning tetris using the noisy cross-entropy method. *Neural computation* 18(12):2936–2941.
- Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8(3-4):229–256.