Constrained Policy Optimization

Joshua Achiam [1] **David Held** [1] **Aviv Tamar** [1] **Pieter Abbeel** [1 2]

Abstract

For many applications of reinforcement learning it can be more convenient to specify both a reward function and constraints, rather than trying to design behavior through the reward function. For example, systems that physically interact with or around humans should satisfy safety constraints. Recent advances in policy search algorithms (Mnih et al., 2016; Schulman et al., 2015; Lillicrap et al., 2016; Levine et al., 2016) have enabled new capabilities in high-dimensional control, but do not consider the constrained setting. We propose Constrained Policy Optimization (CPO), the first general-purpose policy search algorithm for constrained reinforcement learning with guarantees for near-constraint satisfaction at each iteration. Our method allows us to train neural network policies for high-dimensional control while making guarantees about policy behavior all throughout training. Our guarantees are based on a new theoretical result, which is of independent interest: we prove a bound relating the expected returns of two policies to an average divergence between them. We demonstrate the effectiveness of our approach on simulated robot locomotion tasks where the agent must satisfy constraints motivated by safety.

1. Introduction

Recently, deep reinforcement learning has enabled neural network policies to achieve state-of-the-art performance on many high-dimensional control tasks, including Atari games (using pixels as inputs) (Mnih et al., 2015; 2016), robot locomotion and manipulation (Schulman et al., 2015; Levine et al., 2016; Lillicrap et al., 2016), and even Go at the human grandmaster level (Silver et al., 2016).

[1]UC Berkeley [2]OpenAI. Correspondence to: Joshua Achiam <jachiam@berkeley.edu>.

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017.

In reinforcement learning (RL), agents learn to act by trial and error, gradually improving their performance at the task as learning progresses. Recent work in deep RL assumes that agents are free to explore *any behavior* during learning, so long as it leads to performance improvement. In many realistic domains, however, it may be unacceptable to give an agent complete freedom. Consider, for example, an industrial robot arm learning to assemble a new product in a factory. Some behaviors could cause it to damage itself or the plant around it—or worse, take actions that are harmful to people working nearby. In domains like this, *safe exploration* for RL agents is important (Moldovan & Abbeel, 2012; Amodei et al., 2016). A natural way to incorporate safety is via constraints.

A standard and well-studied formulation for reinforcement learning with constraints is the constrained Markov Decision Process (CMDP) framework (Altman, 1999), where agents must satisfy constraints on expectations of auxilliary costs. Although optimal policies for finite CMDPs with known models can be obtained by linear programming, methods for high-dimensional control are lacking.

Currently, policy search algorithms enjoy state-of-the-art performance on high-dimensional control tasks (Mnih et al., 2016; Duan et al., 2016). Heuristic algorithms for policy search in CMDPs have been proposed (Uchibe & Doya, 2007), and approaches based on primal-dual methods can be shown to converge to constraint-satisfying policies (Chow et al., 2015), but there is currently no approach for policy search in continuous CMDPs that guarantees every policy during learning will satisfy constraints. In this work, we propose the first such algorithm, allowing applications to constrained deep RL.

Driving our approach is a new theoretical result that bounds the difference between the rewards or costs of two different policies. This result, which is of independent interest, tightens known bounds for policy search using trust regions (Kakade & Langford, 2002; Pirotta et al., 2013; Schulman et al., 2015), and provides a tighter connection between the theory and practice of policy search for deep RL. Here, we use this result to derive a policy improvement step that guarantees both an increase in reward and satisfaction of constraints on other costs. This step forms the basis for our algorithm, *Constrained Policy Optimization* (CPO), which

computes an approximation to the theoretically-justified update.

In our experiments, we show that CPO can train neural network policies with thousands of parameters on high-dimensional simulated robot locomotion tasks to maximize rewards while successfully enforcing constraints.

2. Related Work

Safety has long been a topic of interest in RL research, and a comprehensive overview of safety in RL was given by (García & Fernández, 2015).

Safe policy search methods have been proposed in prior work. Uchibe and Doya (2007) gave a policy gradient algorithm that uses gradient projection to enforce active constraints, but this approach suffers from an inability to prevent a policy from becoming unsafe in the first place. Bou Ammar et al. (2015) propose a theoretically-motivated policy gradient method for lifelong learning with safety constraints, but their method involves an expensive inner loop optimization of a semi-definite program, making it unsuited for the deep RL setting. Their method also assumes that safety constraints are linear in policy parameters, which is limiting. Chow et al. (2015) propose a primal-dual subgradient method for risk-constrained reinforcement learning which takes policy gradient steps on an objective that trades off return with risk, while simultaneously learning the trade-off coefficients (dual variables).

Some approaches specifically focus on application to the deep RL setting. Held et al. (2017) study the problem for robotic manipulation, but the assumptions they make restrict the applicability of their methods. Lipton et al. (2017) use an 'intrinsic fear' heuristic, as opposed to constraints, to motivate agents to avoid rare but catastrophic events. Shalev-Shwartz et al. (2016) avoid the problem of enforcing constraints on parametrized policies by decomposing 'desires' from trajectory planning; the neural network policy learns desires for behavior, while the trajectory planning algorithm (which is not learned) selects final behavior and enforces safety constraints.

In contrast to prior work, our method is the first policy search algorithm for CMDPs that both 1) guarantees constraint satisfaction throughout training, and 2) works for arbitrary policy classes (including neural networks).

3. Preliminaries

A Markov decision process (MDP) is a tuple, (S, A, R, P, μ), where S is the set of states, A is the set of actions, $R : S \times A \times S \to \mathbb{R}$ is the reward function, $P : S \times A \times S \to [0, 1]$ is the transition probability function (where $P(s'|s, a)$ is the probability of transitioning to state s' given that the previous state was s and the agent took action a in s), and $\mu : S \to [0, 1]$ is the starting state distribution. A stationary policy $\pi : S \to \mathcal{P}(A)$ is a map from states to probability distributions over actions, with $\pi(a|s)$ denoting the probability of selecting action a in state s. We denote the set of all stationary policies by Π.

In reinforcement learning, we aim to select a policy π which maximizes a performance measure, $J(\pi)$, which is typically taken to be the infinite horizon discounted total return, $J(\pi) \doteq \mathop{\mathrm{E}}_{\tau\sim\pi}[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1})]$. Here $\gamma \in [0, 1)$ is the discount factor, τ denotes a trajectory ($\tau = (s_0, a_0, s_1, ...)$), and $\tau \sim \pi$ is shorthand for indicating that the distribution over trajectories depends on π: $s_0 \sim \mu$, $a_t \sim \pi(\cdot|s_t)$, $s_{t+1} \sim P(\cdot|s_t, a_t)$.

Letting $R(\tau)$ denote the discounted return of a trajectory, we express the on-policy value function as $V^\pi(s) \doteq \mathrm{E}_{\tau\sim\pi}[R(\tau)|s_0 = s]$ and the on-policy action-value function as $Q^\pi(s, a) \doteq \mathrm{E}_{\tau\sim\pi}[R(\tau)|s_0 = s, a_0 = a]$. The advantage function is $A^\pi(s, a) \doteq Q^\pi(s, a) - V^\pi(s)$.

Also of interest is the discounted future state distribution, d^π, defined by $d^\pi(s) = (1-\gamma)\sum_{t=0}^{\infty} \gamma^t P(s_t = s|\pi)$. It allows us to compactly express the difference in performance between two policies π', π as

$$J(\pi') - J(\pi) = \frac{1}{1-\gamma} \mathop{\mathrm{E}}_{\substack{s\sim d^{\pi'} \\ a\sim\pi'}} [A^\pi(s, a)], \tag{1}$$

where by $a \sim \pi'$, we mean $a \sim \pi'(\cdot|s)$, with explicit notation dropped to reduce clutter. For proof of (1), see (Kakade & Langford, 2002) or Section 10 in the supplementary material.

4. Constrained Markov Decision Processes

A constrained Markov decision process (CMDP) is an MDP augmented with constraints that restrict the set of allowable policies for that MDP. Specifically, we augment the MDP with a set C of auxiliary cost functions, $C_1, ..., C_m$ (with each one a function $C_i : S \times A \times S \to \mathbb{R}$ mapping transition tuples to costs, like the usual reward), and limits $d_1, ..., d_m$. Let $J_{C_i}(\pi)$ denote the expected discounted return of policy π with respect to cost function C_i: $J_{C_i}(\pi) = \mathop{\mathrm{E}}_{\tau\sim\pi}[\sum_{t=0}^{\infty} \gamma^t C_i(s_t, a_t, s_{t+1})]$. The set of feasible stationary policies for a CMDP is then

$$\Pi_C \doteq \{\pi \in \Pi : \forall i, J_{C_i}(\pi) \leq d_i\},$$

and the reinforcement learning problem in a CMDP is

$$\pi^* = \arg\max_{\pi\in\Pi_C} J(\pi).$$

The choice of optimizing only over stationary policies is justified: it has been shown that the set of all optimal policies for a CMDP includes stationary policies, under mild

technical conditions. For a thorough review of CMDPs and CMDP theory, we refer the reader to (Altman, 1999).

We refer to J_{C_i} as a *constraint return*, or C_i-return for short. Lastly, we define on-policy value functions, action-value functions, and advantage functions for the auxiliary costs in analogy to V^π, Q^π, and A^π, with C_i replacing R: respectively, we denote these by $V^\pi_{C_i}$, $Q^\pi_{C_i}$, and $A^\pi_{C_i}$.

5. Constrained Policy Optimization

For large or continuous MDPs, solving for the exact optimal policy is intractable due to the curse of dimensionality (Sutton & Barto, 1998). Policy search algorithms approach this problem by searching for the optimal policy within a set $\Pi_\theta \subseteq \Pi$ of parametrized policies with parameters θ (for example, neural networks of a fixed architecture). In local policy search (Peters & Schaal, 2008), the policy is iteratively updated by maximizing $J(\pi)$ over a local neighborhood of the most recent iterate π_k:

$$\begin{aligned} \pi_{k+1} = \arg\max_{\pi \in \Pi_\theta} \; & J(\pi) \\ \text{s.t. } & D(\pi, \pi_k) \leq \delta, \end{aligned} \tag{2}$$

where D is some distance measure, and $\delta > 0$ is a step size. When the objective is estimated by linearizing around π_k as $J(\pi_k) + g^T(\theta - \theta_k)$, g is the policy gradient, and the standard policy gradient update is obtained by choosing $D(\pi, \pi_k) = \|\theta - \theta_k\|_2$ (Schulman et al., 2015).

In local policy search for CMDPs, we additionally require policy iterates to be feasible for the CMDP, so instead of optimizing over Π_θ, we optimize over $\Pi_\theta \cap \Pi_C$:

$$\begin{aligned} \pi_{k+1} = \arg\max_{\pi \in \Pi_\theta} \; & J(\pi) \\ \text{s.t. } & J_{C_i}(\pi) \leq d_i \quad i = 1, ..., m \\ & D(\pi, \pi_k) \leq \delta. \end{aligned} \tag{3}$$

This update is difficult to implement in practice because it requires evaluation of the constraint functions to determine whether a proposed point π is feasible. When using sampling to compute policy updates, as is typically done in high-dimensional control (Duan et al., 2016), this requires off-policy evaluation, which is known to be challenging (Jiang & Li, 2015). In this work, we take a different approach, motivated by recent methods for trust region optimization (Schulman et al., 2015).

We develop a principled approximation to (3) with a particular choice of D, where we replace the objective and constraints with *surrogate* functions. The surrogates we choose are easy to estimate from samples collected on π_k, and are good local approximations for the objective and constraints. Our theoretical analysis shows that for our choices of surrogates, we can bound our update's worst-case performance and worst-case constraint violation with values that depend on a hyperparameter of the algorithm.

To prove the performance guarantees associated with our surrogates, we first prove new bounds on the difference in returns (or constraint returns) between two arbitrary stochastic policies in terms of an average divergence between them. We then show how our bounds permit a new analysis of trust region methods in general: specifically, we prove a worst-case performance degradation at each update. We conclude by motivating, presenting, and proving gurantees on our algorithm, Constrained Policy Optimization (CPO), a trust region method for CMDPs.

5.1. Policy Performance Bounds

In this section, we present the theoretical foundation for our approach—a new bound on the difference in returns between two arbitrary policies. This result, which is of independent interest, extends the works of (Kakade & Langford, 2002), (Pirotta et al., 2013), and (Schulman et al., 2015), providing tighter bounds. As we show later, it also relates the theoretical bounds for trust region policy improvement with the actual trust region algorithms that have been demonstrated to be successful in practice (Duan et al., 2016). In the context of constrained policy search, we later use our results to propose policy updates that both improve the expected return and satisfy constraints.

The following theorem connects the difference in returns (or constraint returns) between two arbitrary policies to an average divergence between them.

Theorem 1. *For any function $f : S \to \mathbb{R}$ and any policies π' and π, define $\delta_f(s, a, s') \doteq R(s, a, s') + \gamma f(s') - f(s)$,*

$$\epsilon^{\pi'}_f \doteq \max_s \left| \mathrm{E}_{a \sim \pi', s' \sim P}[\delta_f(s, a, s')] \right|,$$

$$L_{\pi,f}(\pi') \doteq \mathop{\mathrm{E}}_{\substack{s \sim d^\pi \\ a \sim \pi \\ s' \sim P}} \left[\left(\frac{\pi'(a|s)}{\pi(a|s)} - 1 \right) \delta_f(s, a, s') \right], \text{ and}$$

$$D^\pm_{\pi,f}(\pi') \doteq \frac{L_{\pi,f}(\pi')}{1-\gamma} \pm \frac{2\gamma\epsilon^{\pi'}_f}{(1-\gamma)^2} \mathop{\mathrm{E}}_{s \sim d^\pi} \left[D_{TV}(\pi' || \pi)[s] \right],$$

where $D_{TV}(\pi' || \pi)[s] = (1/2) \sum_a |\pi'(a|s) - \pi(a|s)|$ is the total variational divergence between action distributions at s. The following bounds hold:

$$D^+_{\pi,f}(\pi') \geq J(\pi') - J(\pi) \geq D^-_{\pi,f}(\pi'). \tag{4}$$

Furthermore, the bounds are tight (when $\pi' = \pi$, all three expressions are identically zero).

Before proceeding, we connect this result to prior work. By bounding the expectation $\mathrm{E}_{s \sim d^\pi}\left[D_{TV}(\pi' || \pi)[s]\right]$ with $\max_s D_{TV}(\pi' || \pi)[s]$, picking $f = V^\pi$, and bounding $\epsilon^{\pi'}_{V^\pi}$

to get a second factor of $\max_s D_{TV}(\pi'||\pi)[s]$, we recover (up to assumption-dependent factors) the bounds given by Pirotta et al. (2013) as Corollary 3.6, and by Schulman et al. (2015) as Theorem 1a.

The choice of $f = V^\pi$ allows a useful form of the lower bound, so we give it as a corollary.

Corollary 1. *For any policies π', π, with $\epsilon^{\pi'} \doteq \max_s |\mathrm{E}_{a\sim\pi'}[A^\pi(s,a)]|$, the following bound holds:*

$$J(\pi') - J(\pi) \geq \frac{1}{1-\gamma} \mathop{\mathrm{E}}_{\substack{s\sim d^\pi \\ a\sim\pi'}} \left[A^\pi(s,a) - \frac{2\gamma\epsilon^{\pi'}}{1-\gamma} D_{TV}(\pi'||\pi)[s] \right]. \quad (5)$$

The bound (5) should be compared with equation (1). The term $(1-\gamma)^{-1}\mathrm{E}_{s\sim d^\pi, a\sim\pi'}[A^\pi(s,a)]$ in (5) is an approximation to $J(\pi') - J(\pi)$, using the state distribution d^π instead of $d^{\pi'}$, which is known to equal $J(\pi') - J(\pi)$ to first order in the parameters of π' on a neighborhood around π (Kakade & Langford, 2002). The bound can therefore be viewed as describing the worst-case approximation error, and it justifies using the approximation as a *surrogate* for $J(\pi') - J(\pi)$.

Equivalent expressions for the auxiliary costs, based on the upper bound, also follow immediately; we will later use them to make guarantees for the safety of CPO.

Corollary 2. *For any policies π', π, and any cost function C_i, with $\epsilon^{\pi'}_{C_i} \doteq \max_s |\mathrm{E}_{a\sim\pi'}[A^\pi_{C_i}(s,a)]|$, the following bound holds:*

$$J_{C_i}(\pi') - J_{C_i}(\pi) \leq \frac{1}{1-\gamma} \mathop{\mathrm{E}}_{\substack{s\sim d^\pi \\ a\sim\pi'}} \left[A^\pi_{C_i}(s,a) + \frac{2\gamma\epsilon^{\pi'}_{C_i}}{1-\gamma} D_{TV}(\pi'||\pi)[s] \right]. \quad (6)$$

The bounds we have given so far are in terms of the TV-divergence between policies, but trust region methods constrain the KL-divergence between policies, so bounds that connect performance to the KL-divergence are desirable. We make the connection through Pinsker's inequality (Csiszar & Körner, 1981): for arbitrary distributions p, q, the TV-divergence and KL-divergence are related by $D_{TV}(p||q) \leq \sqrt{D_{KL}(p||q)/2}$. Combining this with Jensen's inequality, we obtain

$$\mathop{\mathrm{E}}_{s\sim d^\pi}[D_{TV}(\pi'||\pi)[s]] \leq \mathop{\mathrm{E}}_{s\sim d^\pi}\left[\sqrt{\frac{1}{2} D_{KL}(\pi'||\pi)[s]}\right] \leq \sqrt{\frac{1}{2} \mathop{\mathrm{E}}_{s\sim d^\pi}[D_{KL}(\pi'||\pi)[s]]} \quad (7)$$

From (7) we immediately obtain the following.

Corollary 3. *In bounds (4), (5), and (6), make the substitution*

$$\mathop{\mathrm{E}}_{s\sim d^\pi}[D_{TV}(\pi'||\pi)[s]] \to \sqrt{\frac{1}{2} \mathop{\mathrm{E}}_{s\sim d^\pi}[D_{KL}(\pi'||\pi)[s]]}.$$

The resulting bounds hold.

5.2. Trust Region Methods

Trust region algorithms for reinforcement learning (Schulman et al., 2015; 2016) have policy updates of the form

$$\pi_{k+1} = \arg\max_{\pi\in\Pi_\theta} \mathop{\mathrm{E}}_{\substack{s\sim d^{\pi_k} \\ a\sim\pi}} [A^{\pi_k}(s,a)] \quad \text{s.t. } \bar{D}_{KL}(\pi||\pi_k) \leq \delta, \quad (8)$$

where $\bar{D}_{KL}(\pi||\pi_k) = \mathrm{E}_{s\sim\pi_k}[D_{KL}(\pi||\pi_k)[s]]$, and $\delta > 0$ is the step size. The set $\{\pi_\theta \in \Pi_\theta : \bar{D}_{KL}(\pi||\pi_k) \leq \delta\}$ is called the *trust region*.

The primary motivation for this update is that it is an approximation to optimizing the lower bound on policy performance given in (5), which would guarantee monotonic performance improvements. This is important for optimizing neural network policies, which are known to suffer from performance collapse after bad updates (Duan et al., 2016). Despite the approximation, trust region steps usually give monotonic improvements (Schulman et al., 2015; Duan et al., 2016) and have shown state-of-the-art performance in the deep RL setting (Duan et al., 2016; Gu et al., 2017), making the approach appealing for developing policy search methods for CMDPs.

Until now, the particular choice of trust region for (8) was heuristically motivated; with (5) and Corollary 3, we are able to show that it is principled and comes with a worst-case performance degradation guarantee that depends on δ.

Proposition 1 (Trust Region Update Performance)**.** *Suppose π_k, π_{k+1} are related by (8), and that $\pi_k \in \Pi_\theta$. A lower bound on the policy performance difference between π_k and π_{k+1} is*

$$J(\pi_{k+1}) - J(\pi_k) \geq \frac{-\sqrt{2\delta}\gamma\epsilon^{\pi_{k+1}}}{(1-\gamma)^2}, \quad (9)$$

where $\epsilon^{\pi_{k+1}} = \max_s \left|\mathrm{E}_{a\sim\pi_{k+1}}[A^{\pi_k}(s,a)]\right|$.

Proof. π_k is a feasible point of (8) with objective value 0, so $\mathrm{E}_{s\sim d^{\pi_k}, a\sim\pi_{k+1}}[A^{\pi_k}(s,a)] \geq 0$. The rest follows by (5) and Corollary 3, noting that (8) bounds the average KL-divergence by δ. □

This result is useful for two reasons: 1) it is of independent interest, as it helps tighten the connection between theory and practice for deep RL, and 2) the choice to develop CPO as a trust region method means that CPO inherits this performance guarantee.

5.3. Trust Region Optimization for Constrained MDPs

Constrained policy optimization (CPO), which we present and justify in this section, is a policy search algorithm for CMDPs with updates that approximately solve (3) with a particular choice of D. First, we describe a policy search update for CMDPs that alleviates the issue of off-policy evaluation, and comes with guarantees of monotonic performance improvement and constraint satisfaction. Then, because the theoretically guaranteed update will take too-small steps in practice, we propose CPO as a practical approximation based on trust region methods.

By corollaries 1, 2, and 3, for appropriate coefficients α_k, β_k^i the update

$$\pi_{k+1} = \arg\max_{\pi\in\Pi_\theta} \mathop{\mathrm{E}}_{\substack{s\sim d^{\pi_k}\\ a\sim\pi}} [A^{\pi_k}(s,a)] - \alpha_k\sqrt{\bar{D}_{KL}(\pi||\pi_k)}$$
$$\text{s.t. } J_{C_i}(\pi_k) + \mathop{\mathrm{E}}_{\substack{s\sim d^{\pi_k}\\ a\sim\pi}} \left[\frac{A_{C_i}^{\pi_k}(s,a)}{1-\gamma}\right] + \beta_k^i\sqrt{\bar{D}_{KL}(\pi||\pi_k)} \leq d_i$$

is guaranteed to produce policies with monotonically non-decreasing returns that satisfy the original constraints. (Observe that the constraint here is on an upper bound for $J_{C_i}(\pi)$ by (6).) The off-policy evaluation issue is alleviated, because both the objective and constraints involve expectations over state distributions d^{π_k}, which we presume to have samples from. Because the bounds are tight, the problem is always feasible (as long as π_0 is feasible). However, the penalties on policy divergence are quite steep for discount factors close to 1, so steps taken with this update might be small.

Inspired by trust region methods, we propose CPO, which uses a trust region instead of penalties on policy divergence to enable larger step sizes:

$$\begin{aligned}\pi_{k+1} = &\arg\max_{\pi\in\Pi_\theta} \mathop{\mathrm{E}}_{\substack{s\sim d^{\pi_k}\\ a\sim\pi}} [A^{\pi_k}(s,a)] \\ &\text{s.t. } J_{C_i}(\pi_k) + \frac{1}{1-\gamma}\mathop{\mathrm{E}}_{\substack{s\sim d^{\pi_k}\\ a\sim\pi}} \left[A_{C_i}^{\pi_k}(s,a)\right] \leq d_i \quad \forall i \\ &\bar{D}_{KL}(\pi||\pi_k) \leq \delta.\end{aligned} \tag{10}$$

Because this is a trust region method, it inherits the performance guarantee of Proposition 1. Furthermore, by corollaries 2 and 3, we have a performance guarantee for approximate satisfaction of constraints:

Proposition 2 (CPO Update Worst-Case Constraint Violation)**.** *Suppose π_k, π_{k+1} are related by (10), and that Π_θ in (10) is any set of policies with $\pi_k \in \Pi_\theta$. An upper bound on the C_i-return of π_{k+1} is*

$$J_{C_i}(\pi_{k+1}) \leq d_i + \frac{\sqrt{2\delta}\gamma\epsilon_{C_i}^{\pi_{k+1}}}{(1-\gamma)^2},$$

where $\epsilon_{C_i}^{\pi_{k+1}} = \max_s \left|\mathrm{E}_{a\sim\pi_{k+1}}\left[A_{C_i}^{\pi_k}(s,a)\right]\right|$.

6. Practical Implementation

In this section, we show how to implement an approximation to the update (10) that can be efficiently computed, even when optimizing policies with thousands of parameters. To address the issue of approximation and sampling errors that arise in practice, as well as the potential violations described by Proposition 2, we also propose to tighten the constraints by constraining upper bounds of the auxilliary costs, instead of the auxilliary costs themselves.

6.1. Approximately Solving the CPO Update

For policies with high-dimensional parameter spaces like neural networks, (10) can be impractical to solve directly because of the computational cost. However, for small step sizes δ, the objective and cost constraints are well-approximated by linearizing around π_k, and the KL-divergence constraint is well-approximated by second order expansion (at $\pi_k = \pi$, the KL-divergence and its gradient are both zero). Denoting the gradient of the objective as g, the gradient of constraint i as b_i, the Hessian of the KL-divergence as H, and defining $c_i \doteq J_{C_i}(\pi_k) - d_i$, the approximation to (10) is:

$$\begin{aligned}\theta_{k+1} = \arg\max_\theta \quad & g^T(\theta-\theta_k) \\ \text{s.t.} \quad & c_i + b_i^T(\theta-\theta_k) \leq 0 \quad i = 1,...,m \\ & \frac{1}{2}(\theta-\theta_k)^T H(\theta-\theta_k) \leq \delta.\end{aligned} \tag{11}$$

Because the Fisher information matrix (FIM) H is always positive semi-definite (and we will assume it to be positive-definite in what follows), this optimization problem is convex and, when feasible, can be solved efficiently using duality. (We reserve the case where it is not feasible for the next subsection.) With $B \doteq [b_1,...,b_m]$ and $c \doteq [c_1,...,c_m]^T$, a dual to (11) can be expressed as

$$\max_{\substack{\lambda\geq 0\\ \nu\succeq 0}} \frac{-1}{2\lambda}\left(g^TH^{-1}g - 2r^T\nu + \nu^TS\nu\right) + \nu^Tc - \frac{\lambda\delta}{2}, \tag{12}$$

where $r \doteq g^TH^{-1}B$, $S \doteq B^TH^{-1}B$. This is a convex program in $m+1$ variables; when the number of constraints is small by comparison to the dimension of θ, this is much easier to solve than (11). If λ^*, ν^* are a solution to the dual, the solution to the primal is

$$\theta^* = \theta_k + \frac{1}{\lambda^*}H^{-1}\left(g - B\nu^*\right). \tag{13}$$

Our algorithm solves the dual for λ^*, ν^* and uses it to propose the policy update (13). For the special case where there is only one constraint, we give an analytical solution in the supplementary material (Theorem 2) which removes the need for an inner-loop optimization. Our experiments

Algorithm 1 Constrained Policy Optimization

Input: Initial policy $\pi_0 \in \Pi_\theta$ tolerance α
for $k = 0, 1, 2, ...$ **do**
  Sample a set of trajectories $\mathcal{D} = \{\tau\} \sim \pi_k = \pi(\theta_k)$
  Form sample estimates $\hat{g}, \hat{b}, \hat{H}, \hat{c}$ with $\mathcal{D}$
  if approximate CPO is feasible **then**
    Solve dual problem (12) for λ_k^*, ν_k^*
    Compute policy proposal θ^* with (13)
  else
    Compute recovery policy proposal θ^* with (14)
  end if
  Obtain θ_{k+1} by backtracking linesearch to enforce satisfaction of sample estimates of constraints in (10)
end for

have only a single constraint, and make use of the analytical solution.

Because of approximation error, the proposed update may not satisfy the constraints in (10); a backtracking line search is used to ensure surrogate constraint satisfaction. Also, for high-dimensional policies, it is impractically expensive to invert the FIM. This poses a challenge for computing $H^{-1}g$ and $H^{-1}b_i$, which appear in the dual. Like (Schulman et al., 2015), we approximately compute them using the conjugate gradient method.

6.2. Feasibility

Due to approximation errors, CPO may take a bad step and produce an infeasible iterate π_k. Sometimes (11) will still be feasible and CPO can automatically recover from its bad step, but for the infeasible case, a recovery method is necessary. In our experiments, where we only have one constraint, we recover by proposing an update to purely decrease the constraint value:

$$\theta^* = \theta_k - \sqrt{\frac{2\delta}{b^T H^{-1} b}} H^{-1} b. \tag{14}$$

As before, this is followed by a line search. This approach is principled in that it uses the limiting search direction as the intersection of the trust region and the constraint region shrinks to zero. We give the pseudocode for our algorithm (for the single-constraint case) as Algorithm 1, and have made our code implementation available online.[1]

6.3. Tightening Constraints via Cost Shaping

Because of the various approximations between (3) and our practical algorithm, it is important to build a factor of safety into the algorithm to minimize the chance of constraint violations. To this end, we choose to constrain upper bounds on the original constraints, C_i^+, instead of the original constraints themselves. We do this by cost shaping:

$$C_i^+(s, a, s') = C_i(s, a, s') + \Delta_i(s, a, s'), \tag{15}$$

where $\Delta_i : S \times A \times S \to \mathbb{R}_+$ correlates in some useful way with C_i.

In our experiments, where we have only one constraint, we partition states into *safe states* and *unsafe states*, and the agent suffers a safety cost of 1 for being in an unsafe state. We choose Δ to be the probability of entering an unsafe state within a fixed time horizon, according to a learned model that is updated at each iteration. This choice confers the additional benefit of smoothing out sparse constraints.

7. Connections to Prior Work

Our method has similar policy updates to primal-dual methods like those proposed by Chow et al. (2015), but crucially, we differ in computing the dual variables (the Lagrange multipliers for the constraints). In primal-dual optimization (PDO), dual variables are stateful and learned concurrently with the primal variables (Boyd et al., 2003). In a PDO algorithm for solving (3), dual variables would be updated according to

$$\nu_{k+1} = \left(\nu_k + \alpha_k \left(J_C(\pi_k) - d\right)\right)_+, \tag{16}$$

where α_k is a learning rate. In this approach, intermediary policies are not guaranteed to satisfy constraints—only the policy at convergence is. By contrast, CPO computes new dual variables from scratch at each update to exactly enforce constraints.

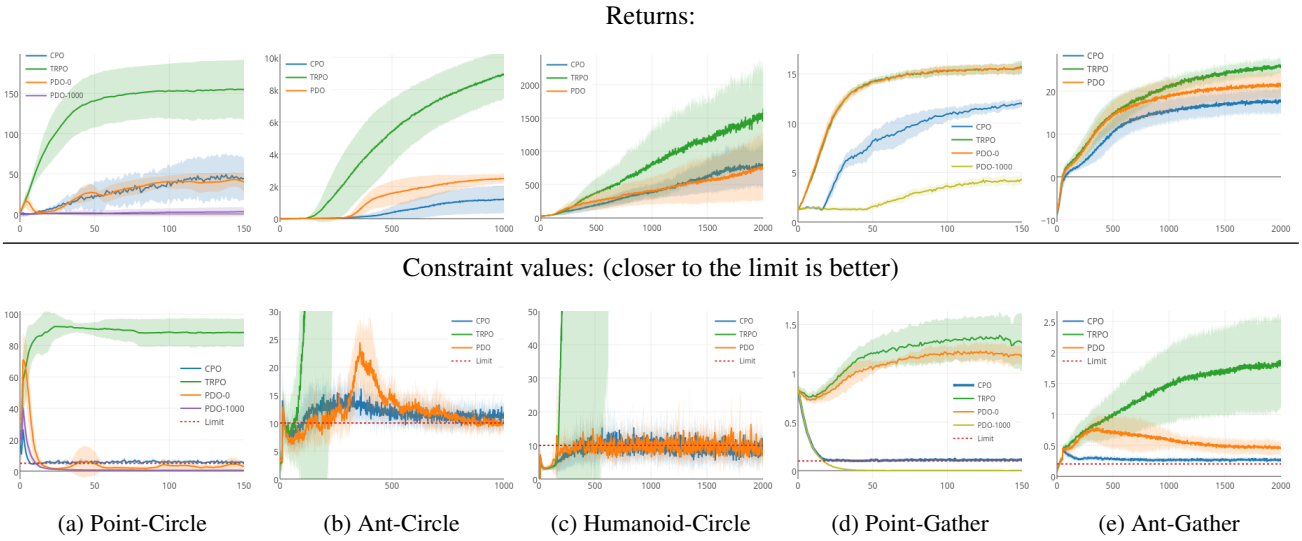
8. Experiments

In our experiments, we aim to answer the following:

- Does CPO succeed at enforcing behavioral constraints when training neural network policies with thousands of parameters?
- How does CPO compare with a baseline that uses primal-dual optimization? Does CPO behave better with respect to constraints?
- How much does it help to constrain a cost upper bound (15), instead of directly constraining the cost?
- What benefits are conferred by using constraints instead of fixed penalties?

We designed experiments that are easy to interpret and motivated by safety. We consider two tasks, and train multiple different agents (robots) for each task:

[1] https://github.com/jachiam/cpo



(a) Point-Circle (b) Ant-Circle (c) Humanoid-Circle (d) Point-Gather (e) Ant-Gather

Figure 1. Average performance for CPO, PDO, and TRPO over several seeds (5 in the Point environments, 10 in all others); the x-axis is training iteration. CPO drives the constraint function almost directly to the limit in all experiments, while PDO frequently suffers from over- or under-correction. TRPO is included to verify that optimal unconstrained behaviors are infeasible for the constrained problem.

- **Circle**: The agent is rewarded for running in a wide circle, but is constrained to stay within a safe region smaller than the radius of the target circle.
- **Gather**: The agent is rewarded for collecting green apples, and constrained to avoid red bombs.

For the Circle task, the exact geometry is illustrated in Figure 5 in the supplementary material. Note that there are no physical walls: the agent only interacts with boundaries through the constraint costs. The reward and constraint cost functions are described in supplementary material (Section 10.3.1). In each of these tasks, we have only one constraint; we refer to it as C and its upper bound from (15) as C^+.

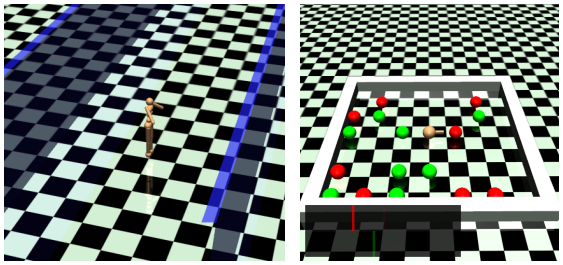
We experiment with three different agents: a point-mass ($S \subseteq \mathbb{R}^9, A \subseteq \mathbb{R}^2$), a quadruped robot (called an 'ant') ($S \subseteq \mathbb{R}^{32}, A \subseteq \mathbb{R}^8$), and a simple humanoid ($S \subseteq \mathbb{R}^{102}, A \subseteq \mathbb{R}^{10}$). We train all agent-task combinations except for Humanoid-Gather.

For all experiments, we use neural network policies with two hidden layers of size $(64, 32)$. Our experiments are implemented in rllab (Duan et al., 2016).

8.1. Evaluating CPO and Comparison Analysis

Learning curves for CPO and PDO are compiled in Figure 1. Note that our constraint value graphs show C^+ return, instead of the C return (except for in Point-Gather, where we did not use cost shaping due to that environment's short time horizon), because this is what the algorithm actually constrains in these experiments.

For our comparison, we implement PDO with (16) as the update rule for the dual variables, using a constant learning rate α; details are available in supplementary material (Section 10.3.3). We emphasize that in order for the comparison to be fair, we give PDO every advantage that is given to CPO, including equivalent trust region policy updates. To benchmark the environments, we also include TRPO (trust region policy optimization) (Schulman et al., 2015), a state-of-the-art *unconstrained* reinforcement learning algorithm. The TRPO experiments show that optimal unconstrained behaviors for these environments are constraint-violating.



(a) Humanoid-Circle (b) Point-Gather

Figure 2. The Humanoid-Circle and Point-Gather environments. In Humanoid-Circle, the safe area is between the blue panels.

We find that CPO is successful at approximately enforcing constraints in all environments. In the simpler environments (Point-Circle and Point-Gather), CPO tracks the constraint return almost *exactly* to the limit value.

By contrast, although PDO usually converges to constraint-satisfying policies in the end, it is not consistently constraint-satisfying throughout training (as expected). For example, see the spike in constraint value that it experi-

ences in Ant-Circle. Additionally, PDO is sensitive to the initialization of the dual variable. By default, we initialize $\nu_0 = 0$, which exploits no prior knowledge about the environment and makes sense when the initial policies are feasible. However, it may seem appealing to set ν_0 high, which would make PDO more conservative with respect to the constraint; PDO could then decrease ν as necessary after the fact. In the Point environments, we experiment with $\nu_0 = 1000$ and show that although this does assure constraint satisfaction, it also can substantially harm performance with respect to return. Furthermore, we argue that this is not adequate in general: after the dual variable decreases, the agent could learn a new behavior that increases the correct dual variable more quickly than PDO can attain it (as happens in Ant-Circle for PDO; observe that performance is approximately constraint-satisfying until the agent learns how to run at around iteration 350).

We find that CPO generally outperforms PDO on enforcing constraints, without compromising performance with respect to return. CPO quickly stabilizes the constraint return around to the limit value, while PDO is not consistently able to enforce constraints all throughout training.

8.2. Ablation on Cost Shaping

In Figure 3, we compare performance of CPO with and without cost shaping in the constraint. Our metric for comparison is the C return, the 'true' constraint. The cost shaping does help, almost completely accounting for CPO's inherent approximation errors. However, CPO is nearly constraint-satisfying even without cost shaping.

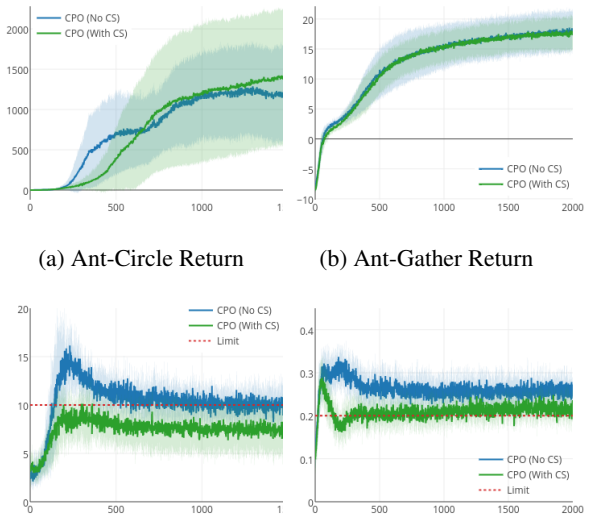
8.3. Constraint vs. Fixed Penalty

In Figure 4, we compare CPO to a fixed penalty method, where policies are learned using TRPO with rewards $R(s, a, s') - \nu C^+(s, a, s')$ for $\nu \in \{1, 5, 50\}$.

We find that fixed penalty methods can be highly sensitive to the choice of penalty coefficient: in Ant-Circle, a penalty coefficient of 1 results in reward-maximizing policies that accumulate massive constraint costs, while a coefficient of 5 (less than an order of magnitude difference) results in cost-minimizing policies that never learn how to acquire any rewards. In contrast, CPO automatically picks penalty coefficients to attain the desired trade-off between reward and constraint cost.

9. Discussion

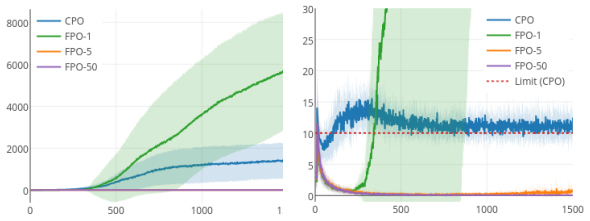
In this article, we showed that a particular optimization problem results in policy updates that are guaranteed to both improve return and satisfy constraints. This enabled the development of CPO, our policy search algorithm for CMDPs, which approximates the theoretically-guaranteed



(a) Ant-Circle Return (b) Ant-Gather Return

(c) Ant-Circle C Return (d) Ant-Gather C Return

Figure 3. Using cost shaping (CS) in the constraint while optimizing generally improves the agent's adherence to the true constraint on C return.



(a) Ant-Circle Return (b) Ant-Circle C^+-Return

Figure 4. Comparison between CPO and FPO (fixed penalty optimization) for various values of fixed penalty.

algorithm in a principled way. We demonstrated that CPO can train neural network policies with thousands of parameters on high-dimensional constrained control tasks, simultaneously maximizing reward and approximately satisfying constraints. Our work represents a step towards applying reinforcement learning in the real world, where constraints on agent behavior are sometimes necessary for the sake of safety.

Acknowledgements

The authors would like to acknowledge Peter Chen, who independently and concurrently derived an equivalent policy improvement bound.

Joshua Achiam is supported by TRUST (Team for Research in Ubiquitous Secure Technology) which receives support from NSF (award number CCF-0424422). This

project also received support from Berkeley Deep Drive and from Siemens.

References

Altman, Eitan. Constrained Markov Decision Processes. pp. 260, 1999. ISSN 01676377. doi: 10.1016/0167-6377(96)00003-X.

Amodei, Dario, Olah, Chris, Steinhardt, Jacob, Christiano, Paul, Schulman, John, and Mané, Dan. Concrete Problems in AI Safety. *arXiv*, 2016. URL http://arxiv.org/abs/1606.06565.

Bou Ammar, Haitham, Tutunov, Rasul, and Eaton, Eric. Safe Policy Search for Lifelong Reinforcement Learning with Sublinear Regret. *International Conference on Machine Learning*, 37:19, 2015. URL http://arxiv.org/abs/1505.0579.

Boyd, Stephen, Xiao, Lin, and Mutapcic, Almir. Subgradient methods. *Lecture Notes of Stanford EE392*, 2003. URL http://xxpt.ynjgy.com/resource/data/20100601/U/stanford201001010/02-subgrad{_}method{_}notes.pdf.

Chow, Yinlam, Ghavamzadeh, Mohammad, Janson, Lucas, and Pavone, Marco. Risk-Constrained Reinforcement Learning with Percentile Risk Criteria. *Journal of Machine Learning Research*, 1(xxxx):1–49, 2015.

Csiszar, I and Körner, J. Information Theory: Coding Theorems for Discrete Memoryless Systems. *Book*, 244:452, 1981. ISSN 0895-4801. doi: 10.2307/2529636. URL http://www.getcited.org/pub/102082957.

Duan, Yan, Chen, Xi, Schulman, John, and Abbeel, Pieter. Benchmarking Deep Reinforcement Learning for Continuous Control. *The 33rd International Conference on Machine Learning (ICML 2016) (2016)*, 48:14, 2016. URL http://arxiv.org/abs/1604.06778.

García, Javier and Fernández, Fernando. A Comprehensive Survey on Safe Reinforcement Learning. *Journal of Machine Learning Research*, 16:1437–1480, 2015. ISSN 15337928.

Gu, Shixiang, Lillicrap, Timothy, Ghahramani, Zoubin, Turner, Richard E., and Levine, Sergey. Q-Prop: Sample-Efficient Policy Gradient with An Off-Policy Critic. In *International Conference on Learning Representations*, 2017. URL http://arxiv.org/abs/1611.02247.

Held, David, Mccarthy, Zoe, Zhang, Michael, Shentu, Fred, and Abbeel, Pieter. Probabilistically Safe Policy Transfer. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

Jiang, Nan and Li, Lihong. Doubly Robust Off-policy Value Evaluation for Reinforcement Learning. *International Conference on Machine Learning*, 2015. URL http://arxiv.org/abs/1511.03722.

Kakade, Sham and Langford, John. Approximately Optimal Approximate Reinforcement Learning. *Proceedings of the 19th International Conference on Machine Learning*, pp. 267–274, 2002. URL http://www.cs.cmu.edu/afs/cs/Web/People/jcl/papers/aoarl/Final.pdf.

Levine, Sergey, Finn, Chelsea, Darrell, Trevor, and Abbeel, Pieter. End-to-End Training of Deep Visuomotor Policies. *Journal of Machine Learning Research*, 17:1–40, 2016. ISSN 15337928. doi: 10.1007/s13398-014-0173-7.2.

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. In *International Conference on Learning Representations*, 2016. ISBN 2200000006. doi: 10.1561/2200000006.

Lipton, Zachary C., Gao, Jianfeng, Li, Lihong, Chen, Jianshu, and Deng, Li. Combating Deep Reinforcement Learning's Sisyphean Curse with Intrinsic Fear. In *arXiv*, 2017. ISBN 2004012439. URL http://arxiv.org/abs/1611.01211.

Mnih, Volodymyr, Kavukcuoglu, Koray, Silver, David, Rusu, Andrei a, Veness, Joel, Bellemare, Marc G, Graves, Alex, Riedmiller, Martin, Fidjeland, Andreas K, Ostrovski, Georg, Petersen, Stig, Beattie, Charles, Sadik, Amir, Antonoglou, Ioannis, King, Helen, Kumaran, Dharshan, Wierstra, Daan, Legg, Shane, and Hassabis, Demis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. ISSN 0028-0836. doi: 10.1038/nature14236. URL http://dx.doi.org/10.1038/nature14236.

Mnih, Volodymyr, Badia, Adrià Puigdomènech, Mirza, Mehdi, Graves, Alex, Lillicrap, Timothy P., Harley, Tim, Silver, David, and Kavukcuoglu, Koray. Asynchronous Methods for Deep Reinforcement Learning. pp. 1–28, 2016. URL http://arxiv.org/abs/1602.01783.

Moldovan, Teodor Mihai and Abbeel, Pieter. Safe Exploration in Markov Decision Processes. *Proceedings of the 29th International Conference on Machine Learning*, 2012. URL http://arxiv.org/abs/1205.4810.

Ng, Andrew Y., Harada, Daishi, and Russell, Stuart. Policy invariance under reward transformations : Theory and application to reward shaping. *Sixteenth International Conference on Machine Learning*, 3:278–287, 1999. doi: 10.1.1.48.345.

Peters, Jan and Schaal, Stefan. Reinforcement learning of motor skills with policy gradients. *Neural Networks*, 21 (4):682–697, 2008. ISSN 08936080. doi: 10.1016/j.neunet.2008.02.003.

Pirotta, Matteo, Restelli, Marcello, and Calandriello, Daniele. Safe Policy Iteration. *Proceedings of the 30th International Conference on Machine Learning*, 28, 2013.

Schulman, John, Moritz, Philipp, Jordan, Michael, and Abbeel, Pieter. Trust Region Policy Optimization. *International Conference on Machine Learning*, 2015.

Schulman, John, Moritz, Philipp, Levine, Sergey, Jordan, Michael, and Abbeel, Pieter. High-Dimensional Continuous Control Using Generalized Advantage Estimation. *arXiv*, 2016.

Shalev-Shwartz, Shai, Shammah, Shaked, and Shashua, Amnon. Safe, Multi-Agent, Reinforcement Learning for Autonomous Driving. *arXiv*, 2016. URL http://arxiv.org/abs/1610.03295.

Silver, David, Huang, Aja, Maddison, Chris J., Guez, Arthur, Sifre, Laurent, van den Driessche, George, Schrittwieser, Julian, Antonoglou, Ioannis, Panneershelvam, Veda, Lanctot, Marc, Dieleman, Sander, Grewe, Dominik, Nham, John, Kalchbrenner, Nal, Sutskever, Ilya, Lillicrap, Timothy, Leach, Madeleine, Kavukcuoglu, Koray, Graepel, Thore, and Hassabis, Demis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587): 484–489, 2016. ISSN 0028-0836. doi: 10.1038/nature16961. URL http://dx.doi.org/10.1038/nature16961.

Sutton, Richard S and Barto, Andrew G. Introduction to Reinforcement Learning. *Learning*, 4(1996):1–5, 1998. ISSN 10743529. doi: 10.1.1.32.7692. URL http://dl.acm.org/citation.cfm?id=551283.

Uchibe, Eiji and Doya, Kenji. Constrained reinforcement learning from intrinsic and extrinsic rewards. *2007 IEEE 6th International Conference on Development and Learning, ICDL*, (February):163–168, 2007. doi: 10.1109/DEVLRN.2007.4354030.