IPO: Interior-point Policy Optimization under Constraints

Yongshuai Liu, Jiaxin Ding, Xin Liu
University of California, Davis
{yshliu, jxding, xinliu}@ucdavis.edu

Abstract

In this paper, we study reinforcement learning (RL) algorithms to solve real-world decision problems with the objective of maximizing the long-term reward as well as satisfying cumulative constraints. We propose a novel first-order policy optimization method, Interior-point Policy Optimization (IPO), which augments the objective with logarithmic barrier functions, inspired by the interior-point method. Our proposed method is easy to implement with performance guarantees and can handle general types of cumulative multi-constraint settings. We conduct extensive evaluations to compare our approach with state-of-the-art baselines. Our algorithm outperforms the baseline algorithms, in terms of reward maximization and constraint satisfaction.

Introduction

Recent advances have demonstrated significant potentials of deep reinforcement learning (RL) in solving complex sequential decision and control problems, e.g., the Atari game (Mnih et al. 2015), robotics (Andrychowicz et al. 2018), Go (Silver et al. 2016), etc. In such RL problems, the objective is to maximize the discounted cumulative reward. In many other problems, in addition to maximizing the reward, a policy needs to satisfy certain constraints. For example, in a cellular network, a common objective for the network operator is to maximize the throughput or cumulative data transmitted to users. At the same time, users may have different requirements for the quality of service, such as the requirements on average latency, cumulative throughput, or the average package loss rate, which are constraints on the optimization problem (Julian et al. 2002). Consider another example of robot manipulation and control. In the task of placing an object (Pham, De Magistris, and Tachibana 2018), the reward is the measurement of how well the object is placed, while there are constraints on the motion of the robot arm, such as how much the arm can twist.

RL with constraints is usually modeled as a Constrained Markov Decision Process (CMDP)(Altman 1999), where the agent must act with respect to constraints, in addition to reward maximization. There are two types of constraints: instantaneous constraints (e.g. robot arm twist) and cumulative constraints (e.g. average latency). An instantaneous constraint is a constraint that the chosen action needs to satisfy in each step. A cumulative constraint requires that the sum of one constraint variable from the beginning to the current time step is within a certain limit. In this work, we focus on cumulative constraints, including both discounted cumulative constraints and mean valued constraints.

A common approach to solve CMDPs is the Lagrangian relaxation method (Chow et al. 2017; Tessler, Mankowitz, and Mannor 2018). The constrained optimization problem is reduced to an unconstrained one by augmenting the objective function with a sum of the constraint functions weighted by their corresponding Lagrange multipliers. Then, the Lagrange multipliers are updated in the dual problem to satisfy the constraints. Although constraints are satisfied when the policy converges, this approach is sensitive to the initialization of the Lagrange multipliers and the learning rate, and the policy obtained during training does not consistently satisfy the constraints, as discussed in (Achiam et al. 2017; Chow et al. 2019).

Constrained policy optimization (CPO) (Achiam et al. 2017) is proposed to solve CMDPs. It extends the trust-region policy optimization (TRPO) algorithm (Schulman et al. 2015) to handle the constraints. CPO monotonically improves the policy during training, demonstrating promising empirical performance, and it guarantees constraint satisfaction during the training process once the constraints are satisfied (Chow et al. 2019). However, CPO needs to calculate the second-order derivatives and thus is complicated to compute and implement. In addition, CPO does not handle mean valued constraints (Tessler, Mankowitz, and Mannor 2018), and it is difficult to employ CPO when there are multiple constraints.

In this paper, we propose a first-order optimization method, Interior-point Policy Optimization (IPO), to solve CMDPs with different types of cumulative constraints. Specifically, inspired by the interior-point method (Boyd and Vandenberghe 2004), we augment the objective function of IPO with logarithmic barrier functions as penalty functions to accommodate the constraints. Intuitively, we would like to construct functions such that 1) if a constraint is satis-

fied, the penalty added to the reward function is zero, and 2) if the constraint is violated, the penalty goes to negative infinity. The logarithmic barrier functions satisfy these requirements, are easy to implement, and also provide nice analytical properties. For policy optimization, we leverage PPO (Schulman et al. 2017), and thus inherit its trust region property. We note that other policy optimization algorithms can be integrated when needed, which increases the flexibility of the proposed methodology. Our algorithm is easy to implement and the hyperparameters are convenient to tune.

In summary, our contributions are as follows:

- We propose IPO, a first-order optimization RL algorithm under cumulative constraints. The algorithm is easy-to-implement, can handle different types and multiple constraints, with easy-to-tune hyperparameters.
- We provide the performance bound of the IPO in terms of reward functions using primal-dual analysis.
- We conduct extensive experiments to compare IPO with the Lagrangian relaxation method and CPO on continuous control tasks, such as MuJoco and grid-world in the robotics setting. IPO outperforms the state-of-art methods with higher long-term reward and lower cumulative constraint values.

Related work

Reinforcement learning with constraints is a significant and challenging topic. A comprehensive overview can be found in (Garcıa and Fernández 2015; Dulac-Arnold, Mankowitz, and Hester 2019).

The Lagrangian relaxation method is widely applied to solve the RL with constraints. Primal-Dual Optimization (PDO) (Chow et al. 2017) employs the Lagrangian relaxation method to devise policy gradient and actor-critic algorithm for risk-constrained RL. PDO is further adapted to off-line policy learning in (Liang, Que, and Modiano 2018) aiming to accelerate the learning process. They use off-line data to pre-train the Lagrange multipliers and reduce the iterations of online updating. A batch policy learning based on the Lagrangian method is proposed in (Le, Voloshin, and Yue 2019) which considers both sampling efficiency and constraint satisfaction challenges. Reward Constrained Policy Optimization (RCPO) (Tessler, Mankowitz, and Mannor 2018) is proposed as a multi-timescale actor-critic approach. They take advantage of TD-learning to update the policy and handle mean valued constraints based on the Lagrangian method.

Differing from above methods which tunes Lagrange multipliers in primal and dual space, Constrained Policy Optimization (CPO) (Achiam et al. 2017) uses new approximation methods from scratch to compute the Lagrange multiplier and enforce constraints in each iteration during the training process.

Another sort of algorithms leverage Lyapunov functions (Khalil 2002; Neely 2010) to handle constraints. In (Chow et al. 2018; 2019), safe approximation policy and value iteration algorithms are induced by Lyapunov constraints.

There are also works on adding a constrained layer to the policy network to satisfy zero-constraint violation at each time step, such as in (Dalal et al. 2018; Pham, De Magistris, and Tachibana 2018) .

To the best of our knowledge, there are no previous works using the interior-point method to solve RL problems with constraints.

Preliminaries

Markov Decision Process

A Markov Decision Process (MDP) is represented by a tuple $(S, A, R, P, \mu, \gamma)$ (Sutton and Barto 2018), where S is the set of states, A is the set of actions, $R : S \times A \times S \mapsto \mathbb{R}$ is the reward function, $P : S \times A \times S \mapsto [0, 1]$ is the transition probability function, where $P(s^{'}|s, a)$ is the transition probability from state s to state $s^{'}$ with taking action a, $\mu : S \mapsto [0, 1]$ is the initial state distribution and γ is the discount factor for future reward. A policy $\pi : S \mapsto \mathcal{P}(A)$ is a mapping from states to a probability distribution over actions and $\pi(a|s)$ is the probability of taking action a in state s. We write a policy π as π_θ to emphasize its dependence on the parameter θ (e.g., a neural network policy with parameter θ). A common goal of a MDP is to select a policy π_θ which maximizes the discounted cumulative reward. It is denoted as

$$\max_\theta \ J_R^{\pi_\theta} = \mathbb{E}_{\tau\sim\pi_\theta}[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1})] \tag{1}$$

where $\tau = (s_0, a_0, s_1, a_1...)$ denotes a trajectory, and $\tau \sim \pi_\theta$ means that the distribution over trajectories is following policy π_θ.

For a trajectory starting from state s, the value function of state s is

$$V_R^{\pi_\theta}(s) = \mathbb{E}_{\tau\sim\pi_\theta}[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1})|s_0 = s]$$

The action-value function of state s and action a is

$$Q_R^{\pi_\theta}(s, a) = \mathbb{E}_{\tau\sim\pi_\theta}[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1})|s_0 = s, a_0 = a]$$

and the advantage function is

$$A_R^{\pi_\theta}(s, a) = Q_R^{\pi_\theta}(s, a) - V_R^{\pi_\theta}(s) \tag{2}$$

Constrained Markov Decision Process

A Constrained Markov Decision Process (CMDP) extends the MDP by introducing a cost function $C : S \times A \times S \mapsto \mathbb{R}$ (similar to the reward function) for each transition tuple $(s, a, s^{'})$, several constraints $\widetilde{C_i} = f(C(s_0, a_0, s_1)), ..., C(s_n, a_n, s_{n+1}))$, and constraint limits $\epsilon_1, ..., \epsilon_m$. The constraints include discounted cumulative constraints, and mean valued constraints (Altman 1999) . The expectation over a constraint is defined as:

$$J_{C_i}^{\pi_\theta} = \mathbb{E}_{\tau\sim\pi_\theta}[\widetilde{C_i}]$$

The discounted cumulative constraint is in the form of:

$$J_{C_i}^{\pi_\theta} = \mathbb{E}_{\tau\sim\pi_\theta}[\sum_{t=0}^{\infty}\gamma^t C(s_t, a_t, s_{t+1})] \tag{3}$$

The mean valued constraint is in the form of:

$$J_{C_i}^{\pi_\theta} = \mathbb{E}_{\tau\sim\pi_\theta}[\frac{1}{T}\sum_{t=0}^{T-1} C(s_t, a_t, s_{t+1})] \tag{4}$$

where T is the total number of time steps in each trajectory.

For a CMDP, the goal is to find a policy π_θ which maximizes the discounted cumulative reward while satisfying the cumulative constraints. It is denoted as

$$\begin{aligned} &\max_\theta J_R^{\pi_\theta} \\ &s.t. \quad J_{C_i}^{\pi_\theta} \leq \epsilon_i \end{aligned} \tag{5}$$

where ϵ_i is the limit for each constraint.

Policy Gradient Methods

Policy gradient (Sutton et al. 2000) is a method for finding an optimal policy of a MDP problem. It first calculates gradient of the objective Eq. (1),

$$\bigtriangledown J_R^{\pi_\theta} = \mathbb{E}_t[\bigtriangledown_\theta log \pi_\theta(a_t|s_t) A_t]$$

where π_θ is the current policy under parameter θ and A_t is the advantage function Eq. (2) at time step t. Thereafter, θ is updated as

$$\theta = \theta + \eta \bigtriangledown J_R^{\pi_\theta},$$

where η is the learning rate.

Trust Region Policy Optimization (TRPO) (Schulman et al. 2015) is proposed to achieve monotonic improvement when updating policy. The objective is approximated with a surrogate function, and the step size is limited by the Kullback Leibler (KL) divergence (Kullback and Leibler 1951), shown as follows.

$$\begin{aligned} &\max_\theta \; L^{TRPO}(\theta) = \mathbb{E}_t[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} A_t] \\ &s.t. \quad \mathbb{E}_t[KL[\pi_{\theta_{old}}(a_t|s_t), \pi_\theta(a_t|s_t)]] \leq \delta. \end{aligned}$$

where δ is the step size limitation.

TRPO can be approximately solved with a conjugate gradient optimization, which is efficient.

Proximal Policy Optimization (PPO) (Schulman et al. 2017) approximates the objective by a first-order surrogate optimization problem to reduce the complexity of TRPO, defined as

$$\begin{aligned} &\max_\theta \; L^{CLIP}(\theta) \\ &\quad = \mathbb{E}_t[\min(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t)], \end{aligned} \tag{6}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$, A_t is the advantage function, $\text{clip}(\cdot)$ is the clip function and $r_t(\theta)$ is clipped between $[1-\epsilon, 1+\epsilon]$.

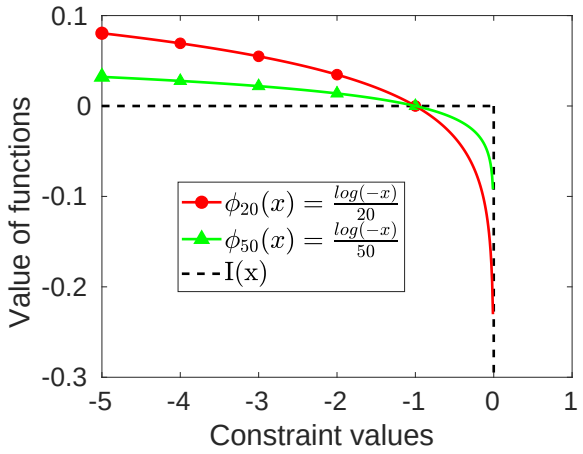


Figure 1: Value of indicator function $I(x)$ and logarithmic barrier functions $\phi(x) = \frac{log(-x)}{t}$. The dashed line is the indicator function and two solid lines are logarithmic barrier function with $t = 20$ and $t = 50$. We get better approximation with higher t comparing these two solid lines.

Interior-point Policy Optimization

Now we introduce our Interior-point Policy Optimization (IPO) to solve CMDP. We employ the clipped surrogate objective of PPO in Eq. (6) as our objective, and augment it with the logarithmic barrier function for the constraints in the interior-point method.

Our problem is defined as

$$\begin{aligned} &\max_\theta \; L^{CLIP}(\theta), \\ &\quad s.t. \quad J_{C_i}^{\pi_\theta} \leq \epsilon_i. \end{aligned} \tag{7}$$

Logarithmic Barrier Function

Now we denote $\widehat{J}_{C_i}^{\pi_\theta} = J_{C_i}^{\pi_\theta} - \epsilon_i$, to simplify the notation. Our constrained optimization problem can be reduced to an unconstrained one by augmenting the objective with indicator functions $I(\widehat{J}_{C_i}^{\pi})$, for each constraint $\widehat{J}_{C_i^\pi}$ satisfying

$$I(\widehat{J}_{C_i}^{\pi_\theta}) = \begin{cases} 0 & \widehat{J}_{C_i}^{\pi_\theta} \leq 0, \\ -\infty & \widehat{J}_{C_i}^{\pi_\theta} > 0. \end{cases}$$

It means that when the constraints are satisfied, we solve the problem as an unconstrained policy optimization problem only considering the reward; however, when any constraint is violated, we must adjust the policy to satisfy the constraint first, since the penalty is $-\infty$.

The logarithm barrier function is a differentiable approximation of the indicator function, defined as

$$\phi(\widehat{J}_{C_i}^{\pi_\theta}) = \frac{\log(-\widehat{J}_{C_i}^{\pi_\theta})}{t},$$

where $t > 0$ is a hyperparameter. The larger t is, the better the approximation is to the indicator function, as shown in Figure 1.

Algorithm 1 The procedure of IPO

Input: Initialize policy π with parameter $\theta = \theta_0$. Set the hyperparameter r for PPO clip rate and t for logarithmic barrier function

Output: The policy parameters θ

1: Initialize the computational graph structure.
2: **for** iteration k=0,1,2,... **do**
3: Sample N trajectories $\tau_1, ..., \tau_N$ including observations, actions, rewards and costs under the current policy θ_k
4: Process the trajectories to advantages, constraint values, etc
5: Update the policy parameter with first order optimizer $\theta_{k+1} = \theta_k + \alpha \bigtriangledown_\theta L^{IPO}(\theta)$ where α is learning rate based on the processed trajectories.
6: **end for**
7: **return** policy parameters $\theta = \theta_{k+1}$

Now our objective becomes

$$\max_\theta \ L^{IPO}(\theta), \tag{8}$$

where

$$L^{IPO}(\theta) = L^{CLIP}(\theta) + \sum_{i=1}^{m} \phi(\widehat{J}^{\pi_\theta}_{C_i}).$$

Thereafter, we can perform first order optimization (e.g. Adam optimizer) to update the parametric policy (e.g. neural network). The pseudo-code of IPO is shown in Algorithm 1.

Performance Guarantee Bound

Theorem 1. *The maximum gap between the optimal value of the constrained optimization problem in Eq. (7) and the objective of IPO in Eq. (8) is bounded by $\frac{m}{t}$, where m is the number of constraints and t is the hyperparameter of logarithmic barrier function, if the optimal policy is strictly feasible.*

Proof. To be consistent with the standard optimization problem, we first convert our maximization problem to a minimization problem by obtaining its negation. The problem defined in Eq. (7) is now

$$\begin{aligned} &\min_\theta -L^{CLIP}(\theta), \\ &s.t. \quad \widehat{J}^{\pi_\theta}_{C_i} \leq 0. \end{aligned} \tag{9}$$

The objective of IPO in Eq. (8) becomes

$$\min_\theta -L^{CLIP}(\theta) - \sum_{i=1}^{m} \frac{log(-\widehat{J}^{\pi_\theta}_{C_i})}{t}. \tag{10}$$

The Lagrangian function of Eq. (9) is

$$L(\theta, \lambda_i) = -L^{CLIP}(\theta) + \sum_{i=1}^{m} \lambda_i \widehat{J}^{\pi_\theta}_{C_i}, \tag{11}$$

where $\lambda_i \geq 0$ is the Lagrange multiplier.

The dual function is

$$g(\lambda_i) = \min_\theta -L^{CLIP}(\theta) + \sum_{i=1}^{m} \lambda_i \widehat{J}^{\pi_\theta}_{C_i}. \tag{12}$$

If the problem is strictly feasible which means an optimal parameter θ^* for Eq. (10) exists and $\widehat{J}^{\pi_{\theta^*}}_{C_i} < 0$. The optimal parameter θ^* must satisfy

$$-\bigtriangledown L^{CLIP}(\theta^*) + \sum_{i=1}^{m} \frac{1}{-t \times \widehat{J}^{\pi_{\theta^*}}_{C_i}} \bigtriangledown \widehat{J}^{\pi_{\theta^*}}_{C_i} = 0. \tag{13}$$

We set

$$\lambda_i^* = -\frac{1}{t \times \widehat{J}^{\pi_{\theta^*}}_{C_i}}, \tag{14}$$

and plug λ_i^* into Eq. (13). We obtain

$$-\bigtriangledown L^{CLIP}(\theta^*) + \sum_{i=1}^{m} \lambda_i^* \bigtriangledown \widehat{J}^{\pi_{\theta^*}}_{C_i} = 0 \tag{15}$$

It means that θ^* minimizes the Lagrangian Eq. (11) under $\lambda_i = \lambda_i^*$. That is,

$$\begin{aligned} g(\lambda_i^*) &= -L^{CLIP}(\theta^*) + \sum_{i=1}^{m} \lambda_i^* \widehat{J}^{\pi_{\theta^*}}_{C_i} \\ &= -L^{CLIP}(\theta^*) - \frac{m}{t} \end{aligned} \tag{16}$$

Let p^* be the optimal value in the problem Eq. (9). By the property of duality gap, $p^* \geq g(\lambda^*)$. Therefore,

$$-L^{CLIP}(\theta^*) - p^* \leq \frac{m}{t}. \tag{17}$$

It means the gap between the optimal value of the original constrained problem with clipped surrogate function (Eq. (7)) and IPO (Eq. (8)) is bounded by $\frac{m}{t}$. □

Theorem 1 indicates that a larger t provides a better approximation of the original objective. Empirically, we notice that a larger t can lead to a higher reward and cost, and lower convergence rate. This monotonicity enables us to employ a binary search algorithm to find a t to balance the convergence rate and optimization.

Experiments

In the experiment, we demonstrate the following properties of IPO:

- IPO can handle more general types of cumulative constraints including discounted cumulative constraints and mean valued constraints. It outperforms the state-of-the-art baselines, CPO (Achiam et al. 2017) and PDO (Chow et al. 2017), on both constraints.
- IPO's hyperparameter is easy to tune, compared to PDO.
- IPO can be easily extended to handle optimizations with multiple constraints.
- IPO is robust in stochastic environments.

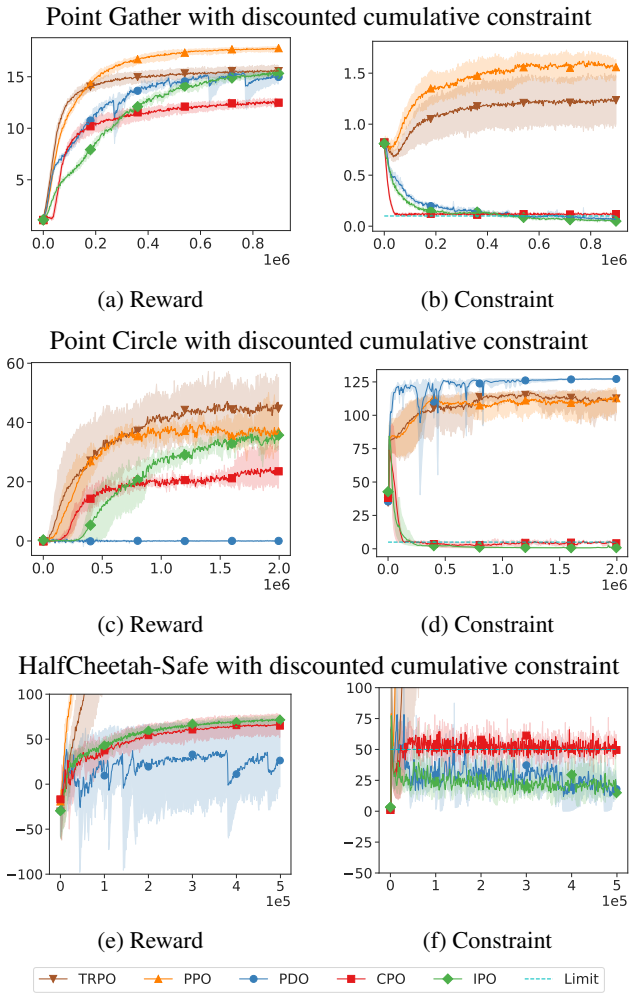


Figure 2: Average performance of TRPO, PPO, PDO, CPO and IPO under Point Gather, Point Circle and HalfCheetah-Safe with discounted cumulative constraints. The x-axis is the number of trajectories. The dashed lines are constrained limits for different tasks which is 0.1 for Point Gather, 5 for Point Circle and 50 for HalfCheetah-Safe.

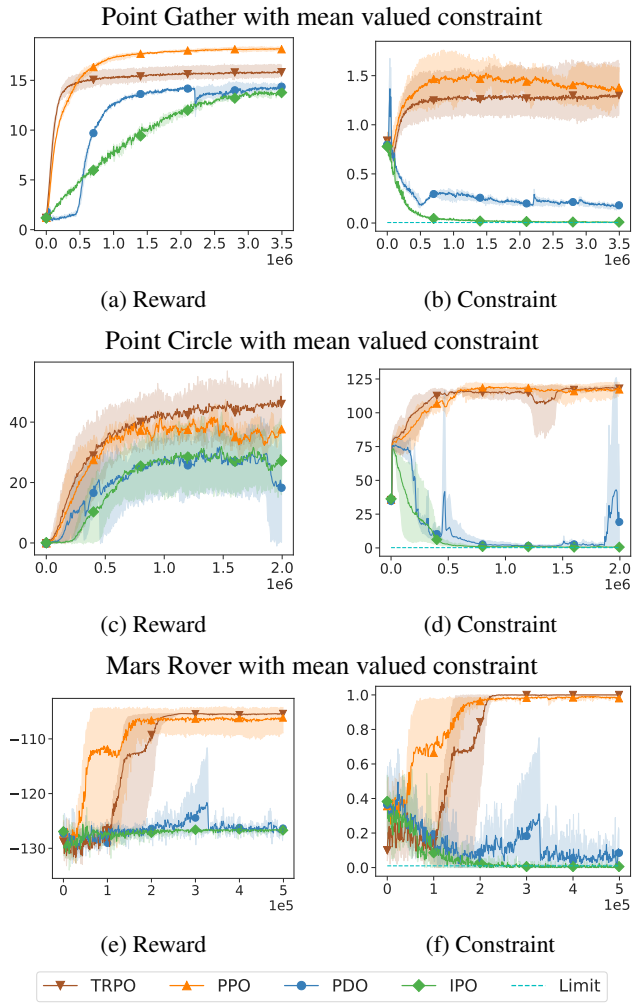


Figure 3: Average performance of TRPO, PPO, PDO and IPO under Point Gather, Point Circle and Mars Rover with mean valued constraints. The dashed lines are constrained limits for different tasks which is 0.005 for Point Gather, 0.2 for Point Circle and 0.01 for Mars Rover.

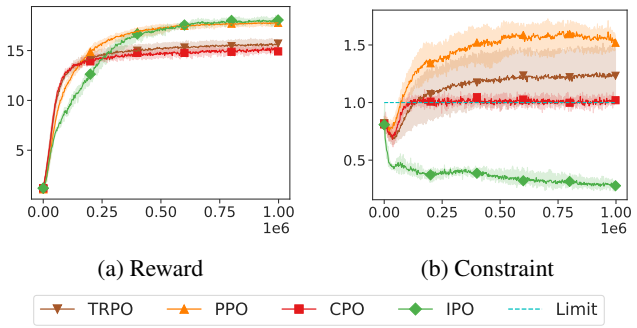
We conduct experiments and compare IPO with CPO and PDO in various scenarios: three tasks in the Mujoco simulator (Point-Gather, Point-Circle (Achiam et al. 2017), HalfCheetah-Safe (Chow et al. 2019)) and a grid-world task (Mars-Rover) inspired by (Chow et al. 2015).

To be fair, the baseline algorithms inherit all advantages in IPO. For example, PDO is implemented with the same PPO clipped surrogate objective. Additionally, we demonstrate the performance of PPO and TRPO for reference, although these algorithms do not take constraints into consideration. We run each experiment ten times with different random seeds and show the average performance.

Scenario Description

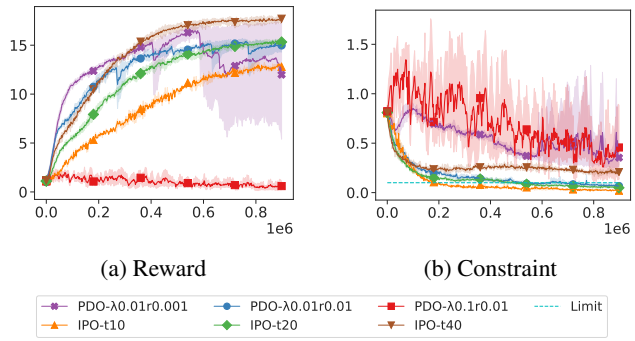
We first describe our experiment scenarios. The objective in all following scenarios is to maximize the reward (the higher the better) while satisfying the constraints (the lower the better).

- Point-Gather: A Point agent moves in a fixed square region where there are randomly located apples (2 apples) and bombs (8 bombs). The agent is rewarded for collecting apples and there is a constraint on the number of bombs collected;
- Point-Circle: A Point agent moves in a circular region with radius r. It's rewarded for running counter-clockwise along the circle, but is restricted to stay within a safe region, smaller than the circular region;
- HalfCheetah-Safe: The HalfCheetah agent (2-legged simulated robot) is rewarded for running with a limited speed, which is less than 1, for stability and safety;
- Grid-world: A rover travels in a fixed square region. It starts from the top left corner and its destination is the



(a) Reward (b) Constraint

Figure 4: Average performance of TRPO, PPO, CPO and IPO under constraint limit 1.



(a) Reward (b) Constraint

Figure 5: Average performance of PDO and IPO with different hyperparameters.

top right corner. The rover gets a negative reward for each step it moves. There are fixed holes in the region. If the rover falls into a hole, the trip terminates. The constraint is on the possibility of the rover falling into a hole.

Discounted Cumulative Constraints

First, we demonstrate results on optimization with discounted cumulative constraints (Eq. (3)) on three Mujoco tasks (Point-Gather, Point-Circle, and HalfCheetah-Safe) to compare IPO with CPO and PDO.

In Figure 2, we can see our overall performance is better than CPO and PDO under all the three Mujoco tasks. IPO achieves higher discounted cumulative reward with lower discounted cumulative cost than CPO. CPO converges faster than IPO, but we notice that CPO always stops improving when the constraint is satisfied. On the contrary, IPO continues to search for a better policy even if the constraint is satisfied. Hence, it converges to a better reward and lower cost.

For PDO, we try to find the satisfactory initial Lagrange multiplier and learning rate with grid search, which is time-consuming. From Figure 2a and 2b, we can see that PDO can converge to a policy as good as IPO, however, the variance of the performance during training is high. The performance of PDO in Figure 2c and 2d is worst of all. It indicates that the Lagrange multiplier 0.01 and learning rate 0.01 is a bad initialization. In the HalfCheetah-Safe task, Figure 2e and 2f, PDO achieves a policy whose constraint value is lower than the limit, but the reward is the lowest as well. PDO is sensitive to the initialization of the Lagrange multiplier and learning rate. We will also demonstrate the impact of hyperparameters in the following experiment.

The PPO and TRPO consider the optimization without constraints. They achieve higher rewards as well as violating the constraints more, compared to IPO, CPO and PDO.

Mean Valued Constraints

Our algorithm can not only support discounted cumulative constraints but also mean valued constraints (Eq. (4)).

We conduct experiments on optimization with the mean valued constrains on two Mujoco tasks (Point-Gather, Point-Circle) and the grid-world (Mars-Rover). Because the CPO does not support mean valued constraints. We only compare IPO with PDO.

Figure 3 shows that IPO can consistently converge to a policy with high discounted cumulative reward and satisfy the mean valued constrains on all tasks. PDO, however, sometimes converges to a policy violating the constraints (Figure 3b) and has a higher variance during training (Figure 3d and Figure 3f).

Constraint Effects

In this experiment, we would like to analyze the effects when changing the constraints. We loosen the constraint in Point Gather with a larger threshold, to be 1, which means that the Point agent can collect at most one bomb on average in each play. Figure 4 demonstrates that both IPO and CPO can obtain almost the same discounted cumulative reward as their corresponding unconstrained method, PPO and TRPO. It indicates that such a constraint is so loose that the performance of the constrained optimization is equivalent to the unconstrained one. We observe that CPO still increases its cost to reach the constraint 1, which is even worse than the randomly initialized policy (around 0.8 at the first iteration). The experiment reflects that CPO always makes efforts to push its cost to the constraint threshold. On the contrary, IPO keeps decreasing its cost after the constraint is satisfied. Statistically, the average number of bombs collected for CPO is around 1 and the number for IPO is around 0.25. We also attach the video visualizing the actions of policies learned with IPO and CPO playing a Point Gather task with a fixed configuration of 2 apples and 8 bombs, in the supplemental materials.

Hyperparameter Tuning

Compared to PDO, our hyperparameter t is easier to tune. We conduct experiments in Point Gather. As shown in Figure 5, the performance of PDO is sensitive to the initialization of the Lagrange multiplier λ from 0.01 to 0.1. Figure 5 also demonstrates that PDO is affected by the learning rate which changes from 0.01 to 0.001. The smaller learning rate slows down the policy convergence pace.

Tuning the initial Lagrange multiplier and learning rate takes a lot of efforts in PDO. On the contrary, the reward

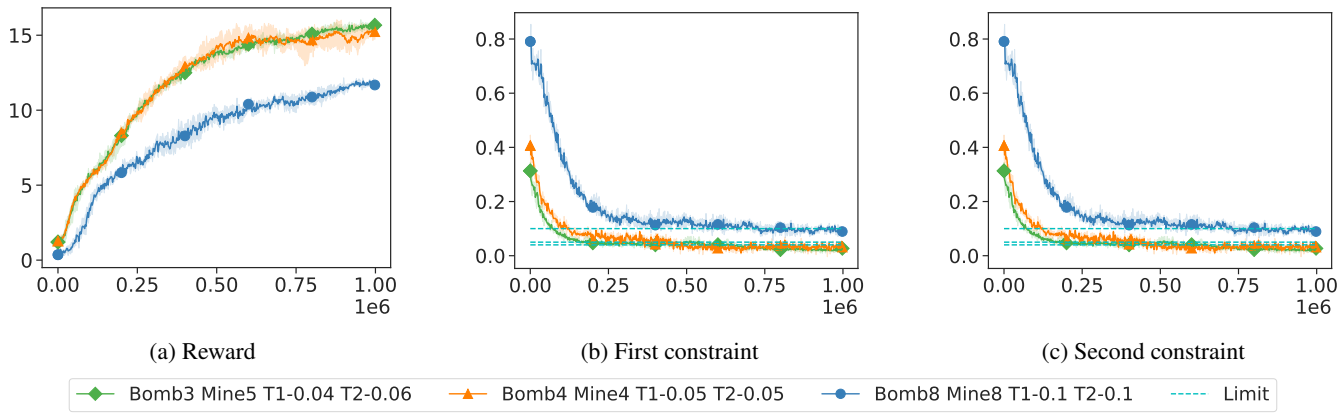


Figure 6: Average performance of IPO under multi-constraints. T1 and T2 correspond to the the limits in (b) and (c) separately. The dash lines are limits for different task settings

and cost of IPO are positively correlated with the hyperparameter t, which enables us to employ a binary search for a feasible hyperparameter conveniently. As shown in Figure 5, we achive higher reward and cost with larger t. Theoretically, we start from a value N big enough, and it takes us at most $O(\log(N))$ to find a feasible t maximize the reward and reduce the cost to satisfy the constraints. In practice, we usually initialize t to be the maximal value of discounted cumulative reward and it can be found in a few iterations. Besides, it's fixed for each task even in different settings (e.g. different constraint limits).

Multiple Constraints

IPO can be conveniently extended to the optimization with multiple constraints by adding a logarithm barrier function for each constraint, which is much easier to implement than CPO. In this section, we conduct three experiments in Point Gather, each with two constraints. To extend the task with multiple constraints, we add another type of balls, mine balls, in the task. The cost of mine balls is the same as bomb balls, which is 1. Now our goal is to maximize the number of apple balls collected, with constraints on the number of bomb balls and mine balls collected. Below we describe our settings, where the values in the parentheses are the constraints of the maximum expected numbers of corresponding balls collected in one play. Our settings are

1. two apples, three bomb balls (0.04), five mine balls (0.06);
2. two apples, four bomb balls (0.05), four mine balls (0.05);
3. two apples, eight bomb balls (0.1), eight mine balls (0.1);

From Figure 6, we can see IPO can converge to a policy satisfying both constraints in all three settings while achieving a high reward.

Stochastic Environment Effects

All the tasks mentioned above have deterministic precise feedback and control. However, in real-world scenarios, there is always uncertainty from the environment. In this section, we do an extra experiment showing that our method

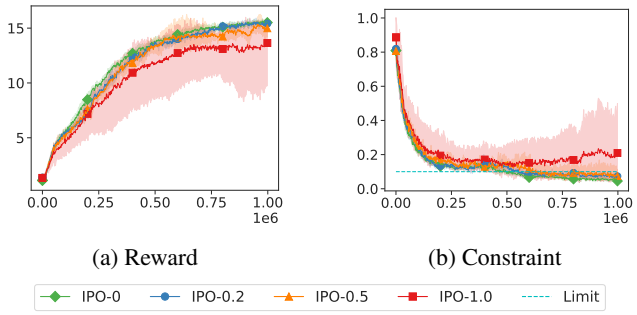


Figure 7: Average performance of IPO under different noise scale. IPO-0 means no noise is added.

is robust to a stochastic environment, where the action is affected by random noise. This setting is inspired by robotics manipulation with uncertainty.

The agent's actions is represented as a vector of velocities and heading directions ranging from -1 to 1. In the experiment, we add random noises following normal distributions with mean value 0 and variances $0.2, 0.5, 1.0$ to agent actions. By comparing the performance in Figure 7, we can see that IPO can still converge to a satisfied policy even when the scale factor is 0.5.

Conclusion

In this paper, we propose a first-order policy optimization method, Interior-point Point Optimization (IPO), to solve constrained reinforcement learning problems. Compared to the state-of-the-art methods, IPO achieves better performance and handles more general types of multiple cumulative constraints. In practice, IPO is easy to implement and the hyperparameters are easy to tune. We will further provide thorough theoretical guarantees for our algorithm, and apply more optimization techniques to handle RL with constraints.

References

Achiam, J.; Held, D.; Tamar, A.; and Abbeel, P. 2017. Constrained policy optimization. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 22–31. JMLR. org.

Altman, E. 1999. *Constrained Markov decision processes*, volume 7. CRC Press.

Andrychowicz, M.; Baker, B.; Chociej, M.; Jozefowicz, R.; McGrew, B.; Pachocki, J.; Petron, A.; Plappert, M.; Powell, G.; Ray, A.; et al. 2018. Learning dexterous in-hand manipulation. *arXiv preprint arXiv:1808.00177*.

Boyd, S., and Vandenberghe, L. 2004. *Convex optimization*. Cambridge university press.

Chow, Y.; Tamar, A.; Mannor, S.; and Pavone, M. 2015. Risk-sensitive and robust decision-making: a cvar optimization approach. In *Advances in Neural Information Processing Systems*, 1522–1530.

Chow, Y.; Ghavamzadeh, M.; Janson, L.; and Pavone, M. 2017. Risk-constrained reinforcement learning with percentile risk criteria. *The Journal of Machine Learning Research* 18(1):6070–6120.

Chow, Y.; Nachum, O.; Duenez-Guzman, E.; and Ghavamzadeh, M. 2018. A lyapunov-based approach to safe reinforcement learning. In *Advances in Neural Information Processing Systems*, 8092–8101.

Chow, Y.; Nachum, O.; Faust, A.; Ghavamzadeh, M.; and Duenez-Guzman, E. 2019. Lyapunov-based safe policy optimization for continuous control. *arXiv preprint arXiv:1901.10031*.

Dalal, G.; Dvijotham, K.; Vecerik, M.; Hester, T.; Paduraru, C.; and Tassa, Y. 2018. Safe exploration in continuous action spaces. *arXiv preprint arXiv:1801.08757*.

Dulac-Arnold, G.; Mankowitz, D.; and Hester, T. 2019. Challenges of real-world reinforcement learning. *arXiv preprint arXiv:1904.12901*.

Garcıa, J., and Fernández, F. 2015. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research* 16(1):1437–1480.

Julian, D.; Chiang, M.; O'Neill, D.; and Boyd, S. 2002. Qos and fairness constrained convex optimization of resource allocation for wireless cellular and ad hoc networks. In *Proceedings. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 2, 477–486. IEEE.

Khalil, H. K. 2002. Nonlinear systems. *Upper Saddle River*.

Kullback, S., and Leibler, R. A. 1951. On information and sufficiency. *The annals of mathematical statistics* 22(1):79–86.

Le, H. M.; Voloshin, C.; and Yue, Y. 2019. Batch policy learning under constraints. *arXiv preprint arXiv:1903.08738*.

Liang, Q.; Que, F.; and Modiano, E. 2018. Accelerated primal-dual policy optimization for safe reinforcement learning. *arXiv preprint arXiv:1802.06480*.

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.

Neely, M. J. 2010. Stochastic network optimization with application to communication and queueing systems. *Synthesis Lectures on Communication Networks* 3(1):1–211.

Pham, T.-H.; De Magistris, G.; and Tachibana, R. 2018. Optlayer-practical constrained optimization for deep reinforcement learning in the real world. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 6236–6243. IEEE.

Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M.; and Moritz, P. 2015. Trust region policy optimization. In *International conference on machine learning*, 1889–1897.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

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.

Sutton, R. S., and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

Sutton, R. S.; McAllester, D. A.; Singh, S. P.; and Mansour, Y. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, 1057–1063.

Tessler, C.; Mankowitz, D. J.; and Mannor, S. 2018. Reward constrained policy optimization. *arXiv preprint arXiv:1805.11074*.

Appendix

Visualization

To better demonstrate the policy learned by IPO and CPO, we visualize them in the Point Gather task and show in the attached video.

In the task, a Point agent moves in a square region. There are green balls (apples) and red balls (bombs) in the region. The agent is rewarded by collecting green balls and constrained to collect red balls. We designed a special case of the task by fixing the position of 2 green balls and 8 red balls as shown in Figure 8 where R represents the start location of the Point agent, A_i is the location of green balls and B_i is the location for red balls. We set the constraint limit to be 1, which means that the Point agent can collect at most one red ball on average in each play.

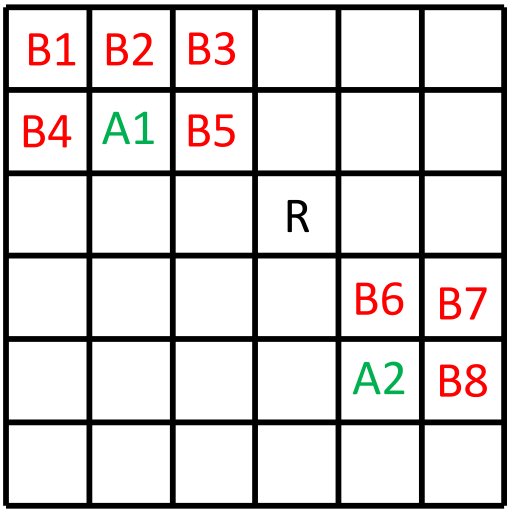


Figure 8: Point Gather task sketch map

We can observe from thousands of trajectories that, CPO will collect the red ball B_6 almost 100%. Sometimes, it collects the red balls B_2 and B_4 as well. It collects at least 1 red ball in each trajectory. On the contrary, IPO can bypass the red ball B_6 almost all the time and not collect any other red ball. For both methods, they all collect 2 green balls in each trajectory.