

1. Introduction

Constrained Reinforcement Learning (CRL) characterized by a constraint Markov decision process $\mathcal{M} := (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathbf{r}, \{\mathbf{c}\}_{i=1}^m, \gamma, \mu)$ aims to optimize a policy π from

$$\operatorname{argmax}_{\pi} J_r(\pi), \text{ s.t. } J_c(\pi) \leq d, \quad (1a)$$

where $J_r(\pi) := \mathbb{E}_\tau[\sum_{t=0}^{\infty} \gamma^t r(s_t)]$ is the expected discounted returns, and $J_c(\pi)$ is the expected discounted costs. Constrained Proximal Policy Optimization (CPPO) conceptualized such CRL problem as a probabilistic inference problem and proposed a first-order Expectation-Maximization (EM) framework to solve this CRL problem [1]¹.

Conceptualizing CRL as a Probabilistic Inference Problem. Given the feasible posterior distribution $q(\mathbf{a}|\mathbf{s})$ under current policy π and the policy distribution $p_\pi(\mathbf{a}|\mathbf{s})$, the evidence lower bound for $\mathbb{P}(\pi_\theta = \pi_\theta^*)$ can be derived from

$$\log \mathbb{P}_{\pi_\theta}(\pi_\theta = \pi_\theta^*) \geq \mathbb{E}_{\mathbf{s} \sim \pi^s, \mathbf{a} \sim \pi} \left[\frac{q(\mathbf{a}|\mathbf{s})}{p_\pi(\mathbf{a}|\mathbf{s})} A(\mathbf{s}, \mathbf{a}) \right] - \alpha D_{KL}(q||\pi_\theta) + \alpha \log p(\theta). \quad (2a)$$

$$\text{s.t. } J_c(\pi) + \frac{1}{1-\gamma} \mathbb{E}_{\mathbf{s} \sim \pi^s, \mathbf{a} \sim \pi} \left[\frac{q(\mathbf{a}|\mathbf{s})}{p_\pi(\mathbf{a}|\mathbf{s})} A_c(\mathbf{s}, \mathbf{a}) \right] \leq d. \quad (2b)$$

where $p(\theta)$ is the prior distribution of policy parameters. Such conversion ensures the first-order nature of the CRL problem, thus improving computational efficiency.

Simplifying Using Relationship between Probability Ratio and KL Divergence. Using importance sampling, the KL-divergence should satisfy:

$$D_{KL}(q||\pi_\theta) = \mathbb{E}_{\mathbf{s} \sim \pi^s, \mathbf{a} \sim \pi} [v \log v] \leq \operatorname{Var}(\bar{\mathbf{v}}) \quad (3)$$

where $\mathbf{v} := \frac{q(\mathbf{a}|\mathbf{s})}{p_\pi(\mathbf{a}|\mathbf{s})}$ and $\bar{\mathbf{v}} = \mathbf{v} - 1$, with an expectation of 0, i.e., $\mathbb{E}[\bar{\mathbf{v}}] = 0$. Accordingly, Eq.(2) is equivalently written as

$$\max_{\bar{\mathbf{v}}} \bar{\mathbf{v}} A \quad \text{s.t. } \bar{\mathbf{v}} A_c \leq N d', \quad \|\bar{\mathbf{v}}\|_2 \leq 2N\delta' \quad (4a)$$

$$\mathbb{E}(\bar{\mathbf{v}}) = 0, \quad \bar{\mathbf{v}} > -1 \text{ element-wise} \quad (4b)$$

Such conversion enables a geometric solution for Eq.(2) as $\bar{\mathbf{v}} = 2N\delta'(\cos\theta'\tilde{A}_c + \sin\theta'\tilde{A})$.

Proposing Iterative Heuristic Algorithm with A Recover Update. The algorithm iteratively projects the probability ratio v onto a feasible region defined by reward and cost advantages, and adjusts search directions to satisfy element-wise constraint $\bar{\mathbf{v}} > -1$. When constraint violations occur, a recovery update is triggered where the gradient is modified to prioritize returning to the feasible region before pursuing reward improvement, thus ensuring stable constraint satisfaction.

2. Modification Description

Inspired by Dyna [2], we integrate a dynamics model into CPPO. The training process involves three interactions: agent training with real rollouts, dynamics model updating using real rollouts, and agent training with imaginary rollouts by the dynamics model.

Agent Training Using Real Rollouts. A process follows the update style of naive CPPO, where the agent is trained with real rollouts collected from the environment.

Dynamics Model Updating Using Real Rollouts. The dynamics model $\mathcal{F}_\phi(\hat{\mathbf{s}}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ is trained to approximate the real environment dynamics $\mathcal{P}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ by minimizing

$$\mathcal{L}(\phi) = \sum_{k=1}^K w_k \mathbb{E} [\|\mathcal{F}_\phi(\mathbf{s}_t, \mathbf{a}_{t:t+k-1}) - \mathbf{s}_{t+k}\|_2]. \quad (5)$$

¹Code available at: <https://github.com/520yrn/cppo>

Algorithm 1 Dyna-CPPO Training Framework

Require: Initial policy π_θ , dynamics model \mathcal{F}_ϕ , replay buffer $\mathcal{D}_{\text{real}}$

- 1: **while** not converged **do**
- 2: Collect real rollouts $\{(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1})\}$ from the environment
- 3: Update policy π_θ and value network using Eq.(4) \triangleright Agent Training Using Real Rollouts
- 4: Store transitions in $\mathcal{D}_{\text{real}}$
- 5: Sample mini-batches $\{(\mathbf{s}_t, \mathbf{a}_{t:t+k-1}, \mathbf{s}_{t+k})\}$ from $\mathcal{D}_{\text{real}}$
- 6: Update ϕ by minimizing Eq.(5) \triangleright Dynamics Model Updating Using Real Rollouts
- 7: Initialize \mathbf{s}_t from the current state distribution.
- 8: **for** $t = 0$ to $H - 1$ **do**
- 9: Generate imaginary rollouts $\{(\hat{\mathbf{s}}_t, \mathbf{a}_t, \hat{r}_t, \hat{\mathbf{s}}_{t+1})\}$ using \mathcal{F}_ϕ
- 10: Store $(\hat{\mathbf{s}}_t, \mathbf{a}_t, \hat{r}_t, \hat{\mathbf{s}}_{t+1})$ in \mathcal{D}_{img} \triangleright truncate rollout after horizon H
- 11: **end for**
- 12: Update π_θ and value network on \mathcal{D}_{img} using Eq.(4) \triangleright same update rule as real rollouts
- 13: **end while**

Note that considering multi-step dynamics encourages the model to learn consistent long-horizon transitions rather than optimizing only one-step predictions.

Agent Training with Imaginary Rollouts. Starting from a current state distribution, imaginary trajectories are generated by rolling out the learned dynamics model \mathcal{F}_ϕ for multiple steps. At each step, an action is sampled from the current policy, and the predicted state is used as the next input for subsequent transitions. To reduce model bias accumulation, the imaginary rollouts are truncated to a fixed horizon H . The agent is then trained on these imaginary rollouts, performing policy and value updates in the same way as in real environments.

3. Motivation

Safe Exploration. In naive CPPO, cost constraints are enforced in a reactive manner, only triggered when the agent enters an infeasible region. Such post-hoc correction is costly, especially in real-world applications where safety violations are strictly prohibited, e.g., preventing collisions in autonomous driving. Incorporating a dynamic model enables proactive and safe exploration within CPPO.

Sample Efficiency. Model-free reinforcement learning inherently suffers from low sample efficiency, as it requires extensive interactions with the environment to converge to an optimal policy. By integrating a Dyna-style framework, CPPO can leverage model-based rollouts to improve sample efficiency and explore the environment more effectively.

4. Feasibility

The theoretical feasibility is supported by the Model-Based Value Estimation (MBVE) which demonstrates that the mean-squared error of the estimated value function admits an upper bound that depends on the model error ϵ , the Lipschitz constants of the reward and value functions, and the rollout horizon H . Detailed proof is presented in Ref.[3], Theorem 3.1.

From a practical perspective, the dynamics model can be trained offline once sufficient real rollouts are collected, decoupling model learning from online interaction. This makes the approach computationally efficient and compatible with standard PPO-style updates, as the learned model can generate short-horizon imaginary rollouts to enhance policy and value learning without additional environment sampling.

References

- [1] Xuan C, Zhang F, Yin F, et al. *Constrained proximal policy optimization* [J]. arXiv preprint arXiv:2305.14216, 2023.
- [2] Sutton R S. Dyna, *an integrated architecture for learning, planning, and reacting* [J]. ACM SIGART Bulletin, 1991, 2(4): 160-163.
- [3] Feinberg V, Wan A, Stoica I, et al. *Model-based value estimation for efficient model-free reinforcement learning* [J]. arXiv preprint arXiv:1803.00101, 2018.