Paper Report

Shuvrajeet Das, DA24D402

Course: DA7400, Fall 2024, IITM
Paper: Constraints Penalized Q-learning for Safe Offline Reinforcement Learning
Date: 18/11/2024

---

1 Introduction

Reinforcement Learning (RL) has demonstrated remarkable success in solving complex tasks, such as gaming and robotics. However, most RL algorithms require extensive trials and errors in simulated environments to learn effective policies. In real-world scenarios, such as self-driving cars or industrial control systems, data collection is costly or risky. This motivates the need for *safe offline RL*, where the goal is to learn a policy that maximizes long-term rewards while satisfying safety constraints using only pre-collected offline data.

Safe RL is commonly modeled as a Constrained Markov Decision Process (CMDP), with constraints categorized into *hard constraints*, which prohibit any violation during a trajectory, and *soft constraints*, which require expected satisfaction over trajectories. In this work, we focus on soft constraints. While existing algorithms address safe exploration, these methods are on-policy and unsuitable for offline settings.

The primary challenge in safe offline RL lies in evaluating constraint violations accurately while maximizing rewards. This is difficult due to the distributional shift between the offline dataset and the learned policy. We propose a novel algorithm, **Constraints Penalized Q-Learning (CPQ)**, to address this issue by penalizing out-of-distribution actions and ensuring constraint satisfaction. Our method supports mixed-behavior datasets and avoids explicit policy constraints, enabling robust policy learning across various benchmark tasks.

We provide theoretical guarantees for the effectiveness of CPQ and demonstrate its superiority over existing baselines through extensive experiments on continuous control tasks.

2 Literature Survey

Reinforcement Learning (RL) has seen significant advancements in recent years, particularly in solving complex problems like gaming, robotics, and optimization tasks. However, most traditional RL methods rely on online interaction with the environment, which can be impractical in scenarios where data collection is expensive or risky. This limitation has driven research in two major areas: Safe Reinforcement Learning and Offline Reinforcement Learning.

Safe Reinforcement Learning focuses on developing algorithms that ensure policies adhere to safety constraints while maximizing cumulative rewards. This is typically modeled as a Constrained Markov Decision Process (CMDP), which imposes either hard constraints that forbid violations entirely or soft constraints that allow violations in expectation. Many algorithms in this area rely on safe exploration techniques, which assume an online learning setting where the agent interacts with the environment to evaluate and enforce constraints.

Offline Reinforcement Learning, also referred to as batch RL, addresses the challenge of learning policies from pre-collected datasets without further interaction with the environment. This paradigm is particularly useful in real-world applications, where online exploration may be infeasible due to high costs or safety concerns. A key challenge in offline RL is the distributional shift problem, where the policy learned may deviate significantly from the data distribution, leading to erroneous value estimations.

Naïve approaches that combine safe RL and offline RL often result in suboptimal solutions due to their inability to effectively balance reward maximization and constraint satisfaction.

These methods may overfit to the dataset distribution or fail to penalize unsafe actions appropriately. Advanced methods have introduced techniques like divergence constraints, cost critics, and pessimistic value estimations to tackle these challenges, but they often struggle with high-dimensional continuous control tasks or mixed-behavior datasets.

Despite these efforts, the field lacks a robust framework that effectively integrates safety constraints into offline RL while maintaining high performance across diverse environments. This gap motivates the development of novel approaches that can address these limitations systematically.

3 Methodology

The goal of this work is to develop a robust framework for safe offline reinforcement learning (RL) that maximizes long-term rewards while satisfying safety constraints using only offline data. We propose a novel algorithm, Constraints Penalized Q-Learning (CPQ), which integrates constraint satisfaction into the Q-learning framework. CPQ consists of three key components: a cost critic for evaluating safety constraints, a reward critic for maximizing rewards, and an out-of-distribution (OOD) detection mechanism for ensuring safe policy updates.

3.1 Problem Formulation

We define the problem within the framework of a Constrained Markov Decision Process (CMDP), represented as a tuple $(S, A, r, c, P, \gamma, l)$. Here:

- S and A are the state and action spaces, respectively.
- $r(s, a)$ is the reward function, and $c(s, a)$ is the cost function.
- $P(s'|s, a)$ is the transition probability, and $\gamma \in [0, 1)$ is the discount factor.
- l denotes the cumulative cost constraint threshold.

The objective is to maximize the cumulative reward while satisfying the constraint:

$$\max_{\pi} \; R(\pi) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right], \tag{1}$$

$$\text{subject to} \quad C(\pi) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t c(s_t, a_t)\right] \leq l. \tag{2}$$

A naive approach to solve safe offline RL is combining techniques from safe RL and offline RL. For example, we could train an additional cost critic Q-network, along with a divergence constraint to prevent large distributional shift from π_β. Formally, we update both the reward and cost critic Q-networks by the empirical Bellman evaluation operator T^π for $(s, a, s', r, c) \sim B$:

$$Q_r(s, a) = r + \gamma \mathbb{E}_{a' \sim \pi(\cdot|s')}\left[Q_r(s', a')\right]$$
$$Q_c(s, a) = c + \gamma \mathbb{E}_{a' \sim \pi(\cdot|s')}\left[Q_c(s', a')\right]$$

Then the policy can be derived by solving the following optimization problem:

$$\pi_\theta := \max_{\pi \in \Delta^{|S|}} \mathbb{E}_{s \sim B, a \sim \pi(\cdot|s)}\left[Q_r(s, a)\right]$$

subject to:

$$\mathbb{E}_{s\sim B, a\sim\pi(\cdot|s)}\left[Q_c(s,a)\right] \leq l \quad \text{(Constraint 1)}$$
$$D(\pi, \pi_\beta) \leq \xi \quad \text{(Constraint 2)}$$

where D can be any off-the-shelf divergence metric (e.g., KL divergence or MMD distance) and ξ is an approximately chosen small value. We can convert the constrained optimization problem to an unconstrained form by using the Lagrangian relaxation procedure and solve it via dual gradient descent. However, we argue that in this approach, Constraint 1 and Constraint 2 may not be satisfied simultaneously. Suppose B consists of transitions from both safe and unsafe policies. When the policy π satisfies Constraint 2, it will match the density of the behavior policy distribution. When the behavior policy distribution contains some unsafe actions, the resulting policy may violate Constraint 1. One may consider subtracting transitions of the safe policy from B to construct a new "safe dataset" and only use it for training.

3.2 Constraints Penalized Q-Learning

CPQ addresses the challenges of safe offline RL by incorporating safety and distributional considerations into the learning process. The algorithm is designed as follows:

3.2.1 Cost Critic for Safety Constraints

The cost critic, $Q_c(s,a)$, is trained to estimate the cumulative cost of state-action pairs:

$$Q_c(s,a) = c(s,a) + \gamma \mathbb{E}_{s'\sim P}\left[\max_{a'} Q_c(s',a')\right]. \tag{3}$$

To ensure safety, out-of-distribution (OOD) actions are penalized with a higher cost. This is achieved by modifying the Bellman update as:

$$Q_c(s,a) = c(s,a) + \gamma \mathbb{E}_{s'\sim P}\left[\max_{a'} \left(Q_c(s',a') + \alpha \cdot \mathbb{I}_{\text{OOD}}(a')\right)\right], \tag{4}$$

where α is a penalty weight, and $\mathbb{I}_{\text{OOD}}(a')$ is an indicator function for OOD actions.

3.2.2 Reward Critic for Maximizing Rewards

The reward critic, $Q_r(s,a)$, is trained to estimate the cumulative reward using the penalized Bellman operator:

$$Q_r(s,a) = r(s,a) + \gamma \mathbb{E}_{s'\sim P}\left[\max_{a'} \mathbb{I}_{\text{safe}}(s',a') Q_r(s',a')\right], \tag{5}$$

where $\mathbb{I}_{\text{safe}}(s',a')$ ensures that only safe state-action pairs contribute to the update.

3.2.3 Out-of-Distribution Detection

A Conditional Variational Autoencoder (CVAE) is pre-trained on the offline dataset to model the behavior policy. The CVAE detects OOD actions using the latent space:

$$\mathcal{L}_{\text{CVAE}} = \mathbb{E}_{z\sim q_\phi(z|s,a)}\left[\log p_\theta(a|s,z)\right] - \beta \cdot D_{\text{KL}}\left[q_\phi(z|s,a) \| p(z)\right], \tag{6}$$

where q_ϕ and p_θ are the encoder and decoder, respectively, $p(z)$ is a Gaussian prior, and β is a scaling factor.

3.3 Algorithm Workflow

The CPQ algorithm follows these steps:

1. Pre-train the CVAE to detect OOD actions based on the latent distribution.
2. Update the cost critic Q_c to penalize OOD actions and estimate cumulative costs.
3. Update the reward critic Q_r to maximize rewards while considering only safe actions.
4. Train the policy π_θ by maximizing the safe, penalized Q-values:

$$\pi_\theta = \arg\max_{\pi} \mathbb{E}_{s\sim B, a\sim\pi}\left[\mathbb{I}_{\text{safe}}(s,a)Q_r(s,a)\right]. \tag{7}$$

Algorithm 1 Constraints Penalized Q-Learning (CPQ)

```
Require: $\mathcal{B}$, constraint limit $l$, threshold $d$.
Initialize: encoder $E_{\omega_1}$ and decoder $D_{\omega_2}$.
0: // VAE Training
0: for $t = 0, 1, \ldots, M$ do
0:     Sample mini-batch of state-action pairs $(s, a) \sim \mathcal{B}$.
0:     Update encoder and decoder by Eq. (??).
0: end for
0: // Policy Training
0: Initialize reward critic ensemble $\{Q_{r_i}(s, a|\phi_{r_i})\}_{i=1}^2$ and cost critic $Q_c(s, a|\phi_c)$, actor $\pi_\theta$, La-
   grange multiplier $\alpha$, target networks $\{Q'_{r_i}\}_{i=1}^2$ and $Q'_c$, with $\phi'_{r_i} \leftarrow \phi_{r_i}$ and $\phi'_c \leftarrow \phi_c$.
0: for $t = 0, 1, \ldots, N$ do
0:     Sample mini-batch of transitions $(s, a, r, c, s') \sim \mathcal{B}$.
0:     Sample $n$ actions $\{a_i \sim \pi_\theta(a|s)\}_{i=1}^n$, get latent mean and std $\{\mu_i, \sigma_i\} = E_{\omega_1}(s, a_i)$, and
   extract $m$ $(m \geq 0)$ actions $\{a_j \mid D_{\mathrm{KL}}(\mathcal{N}(\mu_j, \sigma_j) \| \mathcal{N}(0, 1)) \geq d\}_{j=1}^m$ from them.
0:     Let $\nu(s) = \frac{1}{m}\sum_j Q_c(s, a_j)$ if $m > 0$, otherwise 0.
0:     Update cost critic by Eq. (??) and reward critics by Eq. (??).
0:     Update actor by Eq. (??) using the policy gradient.
0:     Update target cost critic: $\phi'_c \leftarrow \tau\phi_c + (1-\tau)\phi'_c$.
0:     Update target reward critics: $\phi'_{r_i} \leftarrow \tau\phi_{r_i} + (1-\tau)\phi'_{r_i}$.
0: end for=0
```

3.4 Theoretical Guarantees

CPQ provides theoretical guarantees that the learned policy satisfies the safety constraint:

$$\mathbb{E}_\pi\left[C(\pi)\right] \leq l, \tag{8}$$

and achieves near-optimal reward performance under mild assumptions on the data distribution and penalty parameters.

4 Experiment Setup

To evaluate the performance of the proposed Constraints Penalized Q-Learning (CPQ) algorithm, we conduct experiments on continuous control tasks from the Mujoco simulator. These tasks are designed to emulate real-world scenarios where safety constraints are critical. The experimental setup is detailed below.

4.1 Tasks and Environments

We use three benchmark environments: `Hopper-v2`, `HalfCheetah-v2`, and `Walker2d-v2`. These environments simulate robotic systems where the agent controls multiple joints by applying torques. The goal in these tasks is to maximize forward movement while adhering to safety constraints, defined as cumulative torque limits to prolong motor life.

4.2 Dataset Generation

For each environment, we generate offline datasets by mixing trajectories collected from two types of policies:

- **Safe Policy:** A policy trained to strictly satisfy safety constraints but with low rewards.
- **Unsafe Policy:** A policy trained to maximize rewards, often violating safety constraints.

Each dataset consists of 50% transitions from the safe policy and 50% from the unsafe policy, capturing a wide range of behaviors. The total dataset size for each environment is 2×10^6 samples.

4.3 Baseline Algorithms

We compare CPQ with the following baseline algorithms:

- **CBPL:** An extension of Constrained Batch Policy Learning to continuous control.
- **BCQ-Lagrangian:** A combination of Batch-Constrained Q-learning and a Lagrangian approach for constraint enforcement.
- **BEAR-Lagrangian:** An offline RL approach incorporating divergence penalties and safety constraints.
- **BC-Safe:** A behavior cloning baseline trained solely on safe policy data to evaluate imitation-based approaches.

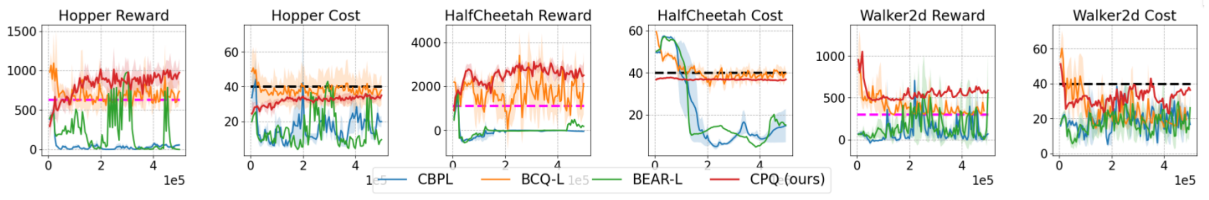

Figure 1: We evaluate CPQ and different baselines according to the experiments of Section 6.1. The shaded area represents one standard deviation around the mean. The dashed magenta line measures the performance of BC-Safe. The constraint threshold l is indicated by the dashed black line. It can be seen that CPQ is robust to learn from different scenarios, outperforms other baselines while still satisfying safe constraints.

4.4 Evaluation Protocol

Each agent is trained for 0.5 million steps. During training, the policies are evaluated every 5000 iterations on 10 independent episodes. The average reward and constraint violation are reported to assess both performance and safety.

4.5 Safety Constraints

The safety constraints are defined as the discounted cumulative torque applied by the agent across all joints. The per-step cost is the torque applied at each time step, and the cumulative cost must not exceed a predefined threshold. This constraint ensures the agent adheres to safe operational limits during evaluation.

4.6 Hyperparameter Tuning

To ensure fairness, we tune the hyperparameters for all algorithms using grid search. For CPQ, key parameters such as the OOD detection threshold and penalty weights are optimized to achieve robust performance.

4.7 Hardware and Implementation

All experiments are implemented in Python using the Mujoco simulator and TensorFlow. Training and evaluation are performed on a workstation with an NVIDIA RTX 3090 GPU and an Intel i9 CPU.

4.8 Metrics

We evaluate the algorithms based on:

- **Reward Performance:** Average cumulative rewards achieved across evaluation episodes.
- **Safety Compliance:** Average cumulative costs to assess adherence to safety constraints.
- **Stability:** Variability in performance across different training runs.

This experimental setup ensures a rigorous comparison of CPQ against state-of-the-art baseline methods while highlighting its ability to balance reward maximization and safety constraints.

5 Conclusion

In this work, we proposed Constraints Penalized Q-Learning (CPQ), a novel algorithm for safe offline reinforcement learning. CPQ addresses the challenges of balancing reward maximization and safety constraint satisfaction by introducing a cost critic to penalize out-of-distribution actions and a reward critic tailored to focus on safe, in-distribution actions. By leveraging out-of-distribution detection with a pre-trained conditional variational autoencoder (CVAE), CPQ effectively identifies and avoids unsafe actions, even when working with mixed-behavior datasets.

Through extensive theoretical analysis, we demonstrated that CPQ provides guarantees for constraint satisfaction while achieving near-optimal policy performance. Empirical evaluations on benchmark Mujoco environments showed that CPQ outperforms state-of-the-art baselines in terms of both reward and safety compliance. The results highlight CPQ's robustness in handling diverse tasks and its ability to learn high-reward policies under safety-critical conditions.

Future work includes extending CPQ to handle hard constraints, improving out-of-distribution detection techniques for enhanced performance, and applying CPQ to real-world domains such as autonomous driving and industrial control systems. We believe this work represents a significant step toward safe and reliable reinforcement learning in practical applications.