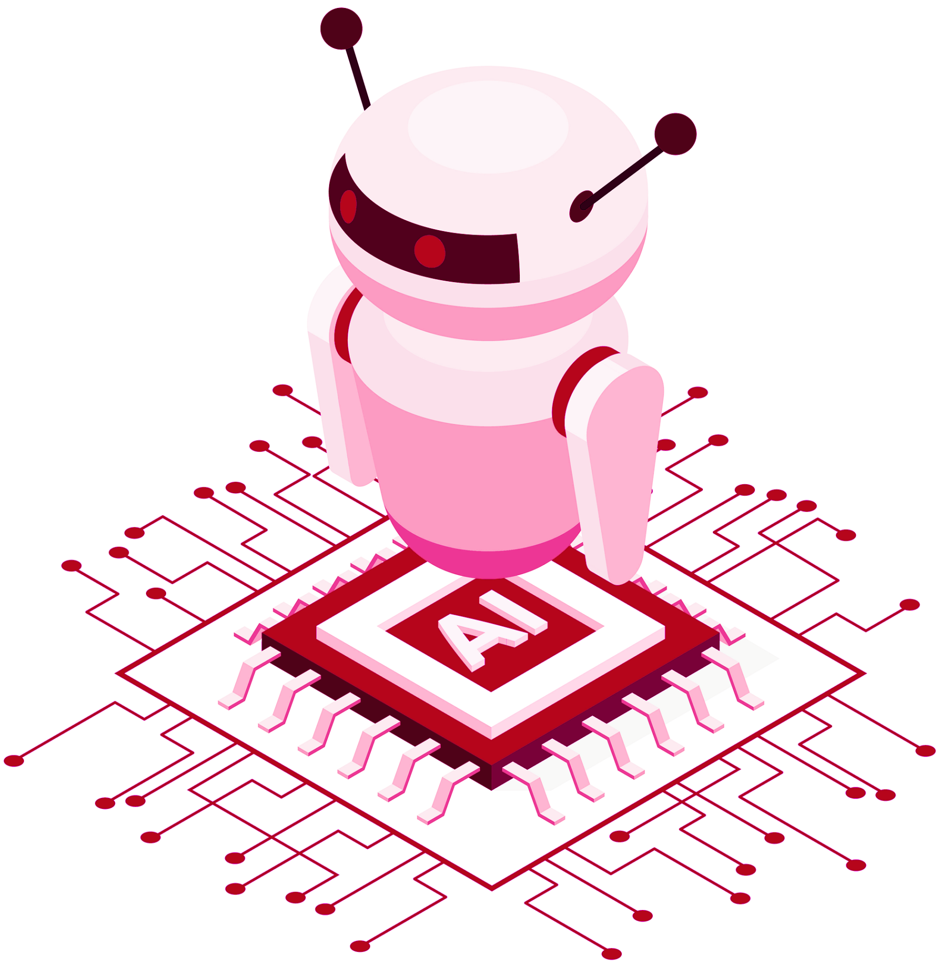

Deep Reinforcement Learning

Professor Mohammad Hossein Rohban

Solution for Homework 12:

Offline Methods

By:

[Asemaneh Nafe]

[400105285]

Spring 2025

Contents

1 Part 1 [60-points]

1. Considering the Bellman update, explain with reasoning why value estimation suffers from overestimation in the offline framework. [10-points]

$$Q(s,a) \leftarrow r(s,a) + \mathbb{E}_{a' \sim r_{new}}[Q(s',a')]$$

In offline reinforcement learning (RL), we optimize the Q-function using a fixed dataset collected by a behavior policy π_β. However, in Bellman updates, we often evaluate:

$$Q(s,a) \leftarrow r(s,a) + \mathbb{E}_{a' \sim \pi_{\text{new}}}[Q(s',a')] = y(s,a)$$

Here, the target value $y(s,a)$ involves sampling actions $a' \sim \pi_{\text{new}}$, which may differ from the actions in the dataset. The Bellman update is trained to minimize the mean squared error:

$$\min_Q \mathbb{E}_{(s,a)\sim\pi_\beta}\left[(Q(s,a) - y(s,a))^2\right]$$

This optimization assumes $y(s,a)$ is accurate, but that only holds when $\pi_\beta(a|s) \approx \pi_{\text{new}}(a|s)$ — i.e., when the learned policy and the behavior policy overlap. In practice, especially in offline RL, this overlap is often limited.

Overestimation from Distributional Shift:

- When the policy is updated toward $\pi_{\text{new}} = \arg\max_\pi \mathbb{E}_{a\sim\pi(a|s)}[Q(s,a)]$, it favors actions with higher estimated Q-values — even if they are out-of-distribution (OOD).
- This leads to selecting actions (or states) that have not been seen in the dataset. Since Q-values for OOD inputs are unreliable, they may be spuriously high.
- As a result, maximizing over these values is essentially maximizing over noisy or random numbers — which is likely to return an abnormally large value.
- These overestimated Q-values are then propagated backward through the Bellman equation, recursively inflating the value of earlier state-action pairs. This effect snowballs and causes widespread overestimation across the value function.

Offline RL suffers from overestimation bias due to distributional shift between the policy being optimized and the dataset. Bellman updates incorporate inaccurate Q estimates from OOD actions, and maximization in the backup equation causes recursive inflation of value estimates, severely harming learning stability.

2. One of the solutions to address the overestimation problem in the offline framework is CQL, whose objective function for computing the value is given below. Explain the role of each of the four terms in this objective function. [20-points]

$$\hat{Q}^T = \arg\min_Q \max_\mu \quad \underbrace{\alpha \mathbb{E}_{s\sim D, a\sim\mu(a|s)}[Q(s,a)]}_{\text{(1) Penalize Q-values on unseen actions}} \underbrace{-\alpha \mathbb{E}_{(s,a)\sim D}[Q(s,a)]}_{\text{(2) Encourage Q-values on dataset actions}} \underbrace{-\mathbb{E}_{s\sim D}[\mathcal{H}(\mu(\cdot|s))]}_{\text{(3) Regularize exploration policy}} \underbrace{+\mathbb{E}_{(s,a,s')\sim D}\left[(Q(s,a) - (r(s,a) + \mathbb{E}[Q(s',a')]))^2\right]}_{\text{(4) Bellman error term}}$$

Explanation of Each Term:

(a) **Term 1:** $\alpha \mathbb{E}_{s\sim D, a\sim\mu(a|s)}[Q(s,a)]$
This term encourages the learned Q-function to assign low values to actions a that are sampled from some broad distribution $\mu(a|s)$ (often uniform or learned). These actions may be out-of-distribution (OOD) relative to the dataset. Maximizing this expectation encourages the worst-case Q-value, and then minimizing it leads to conservative value estimates on unseen or less certain actions.

(b) **Term 2:** $-\alpha \mathbb{E}_{(s,a)\sim D}[Q(s,a)]$
This term counters Term 1 by encouraging the Q-function to assign higher values to actions in the dataset. It reinforces trusting actions actually observed in the data. The subtraction means we are rewarding accurate value estimates on in-distribution actions.

(c) **Term 3:** $-\mathbb{E}_{s\sim D}[\mathcal{H}(\mu(\cdot|s))]$
This term is an entropy regularizer on the policy μ. It discourages μ from becoming too stochastic, pushing it toward sharp distributions. That means μ is encouraged to focus on the most problematic (high Q) actions — a pessimistic estimation strategy.

(d) **Term 4: Bellman Error Term**

$$\mathbb{E}_{(s,a,s')\sim D}\left[(Q(s,a) - (r(s,a) + \mathbb{E}_{a'\sim\pi}[Q(s',a')]))^2\right]$$

This is the standard Bellman residual term. It ensures that the Q-function is consistent with observed rewards and the discounted next-state values. This term anchors learning to actual data dynamics.

CQL prevents overestimation in offline RL by learning conservative Q-values — penalizing high values on OOD actions (Term 1), encouraging high values only on dataset actions (Term 2), focusing μ on the most uncertain areas (Term 3), and grounding estimates through Bellman consistency (Term 4). This helps stabilize learning when out-of-distribution generalization can cause harmful value inflation.

3. Rewrite the optimization problem from part 3 as a minimization-only problem. [20-points]

We start from the general CQL objective with a regularizer $\mathcal{R}(\mu)$:

$$\min_Q \max_\mu \; \alpha \left(\mathbb{E}_{s\sim\mathcal{D}, a\sim\mu(a|s)}\left[Q(s,a)\right] - \mathbb{E}_{s\sim\mathcal{D}, a\sim\hat{\pi}_\beta(a|s)}\left[Q(s,a)\right]\right) + \frac{1}{2}\mathbb{E}_{s,a,s'\sim\mathcal{D}}\left[\left(Q(s,a) - \hat{\mathcal{B}}^{\pi_k}\hat{Q}^k(s,a)\right)^2\right] + \mathcal{R}($$

We choose $\mathcal{R}(\mu)$ to be **negative entropy regularization**, i.e.,

$$\mathcal{R}(\mu) = \mathbb{E}_{s\sim\mathcal{D}}\left[\mathcal{H}(\mu(\cdot|s))\right] = -\mathbb{E}_{s\sim\mathcal{D},a\sim\mu(a|s)}\left[\log\mu(a|s)\right]$$

Step 1: Isolate Inner Maximization Over μ

For fixed S, define the inner maximization problem:

$$\max_{\mu(\cdot|s)} \ \alpha\mathbb{E}_{a\sim\mu(a|s)}\left[Q(s,a)\right] - \mathbb{E}_{a\sim\mu(a|s)}\left[\log\mu(a|s)\right]$$

This is a **constrained optimization problem** where $\mu(a|s)$ is a probability distribution over a:

- $\sum_a \mu(a|s) = 1$
- $\mu(a|s) \geq 0$

We solve this using the method of **Lagrange multipliers**.

Define the Lagrangian:

$$\mathcal{L}(\mu,\lambda) = \alpha\sum_a \mu(a|s)Q(s,a) - \sum_a \mu(a|s)\log\mu(a|s) + \lambda\left(1 - \sum_a \mu(a|s)\right)$$

Step 2: Take Derivative and Set to Zero

Take derivative w.r.t. $\mu(a|s)$:

$$\frac{\partial\mathcal{L}}{\partial\mu(a|s)} = \alpha Q(s,a) - (1 + \log\mu(a|s)) - \lambda = 0$$

Rearranging:

$$\log\mu(a|s) = \alpha Q(s,a) - 1 - \lambda$$

Exponentiate both sides:

$$\mu(a|s) = \exp\left(\alpha Q(s,a) - 1 - \lambda\right) = C\cdot\exp\left(\alpha Q(s,a)\right)$$

Where $C = \exp(-\lambda - 1)$ is a normalizing constant.

To compute C, normalize:

$$\sum_a \mu(a|s) = 1 \Rightarrow C\sum_a \exp\left(\alpha Q(s,a)\right) = 1 \Rightarrow C = \frac{1}{\sum_a \exp(\alpha Q(s,a))}$$

Therefore, the optimal $\mu^*(a|s)$ is:

$$\mu^*(a|s) = \frac{\exp(\alpha Q(s,a))}{\sum_{a'}\exp(\alpha Q(s,a'))}$$

Step 3: Plug in the optimal $\mu(a|s)$ and simplify the objective

Recap: Inner Maximization Solution

From the previous step, we showed that the optimal $\mu(a|s)$ is:

$$\mu(a|s) = \frac{\exp(\alpha Q(s,a))}{Z(s)} \quad \text{where } Z(s) = \sum_{a'} \exp(\alpha Q(s,a'))$$

This is a **Boltzmann policy** over Q-values with temperature $1/\alpha$. We now **plug this back** into the inner maximization term:

$$\max_{\mu} \ \alpha \mathbb{E}_{a\sim\mu(a|s)}[Q(s,a)] - \mathbb{E}_{a\sim\mu(a|s)}[\log \mu(a|s)]$$

Let's evaluate this expression using the optimal $\mu(a|s)$:

$$\begin{aligned}
&\alpha \mathbb{E}_{a\sim\mu(a|s)}[Q(s,a)] - \mathbb{E}_{a\sim\mu(a|s)}[\log \mu(a|s)] \\
&= \sum_a \mu(a|s)\left[\alpha Q(s,a) - \log \mu(a|s)\right] \\
&= \sum_a \mu(a|s)\left[\alpha Q(s,a) - (\alpha Q(s,a) - \log Z(s))\right] \quad \text{(from earlier: } \log \mu(a|s) = \alpha Q(s,a) - \log Z(s)) \\
&= \sum_a \mu(a|s) \log Z(s) = \log Z(s) \\
&= \log \sum_a \exp(\alpha Q(s,a))
\end{aligned}$$

So the entire inner maximization simplifies to:

$$\max_{\mu} \left[\alpha \mathbb{E}_{a\sim\mu(a|s)} Q(s,a) - \mathbb{E}_{a\sim\mu(a|s)} \log \mu(a|s)\right] = \log \sum_a \exp(\alpha Q(s,a))$$

Step 4: Return to the Full CQL Objective

We now substitute this back into the **original min-max problem**, and since we've solved the inner maximization, we get a **pure minimization** objective:

$$\begin{aligned}
\text{CQL}(\mathcal{H}) = \min_Q \quad & \mathbb{E}_{s\sim\mathcal{D}}\left[\log \sum_a \exp(\alpha Q(s,a)) - \alpha \mathbb{E}_{a\sim\hat{\pi}_\beta(a|s)}[Q(s,a)]\right] \\
& + \frac{1}{2}\mathbb{E}_{(s,a,s')\sim\mathcal{D}}\left[\left(Q(s,a) - \hat{\mathcal{B}}^{\pi_k}\hat{Q}^k(s,a)\right)^2\right]
\end{aligned}$$

This is the **final minimization-only form** of the CQL objective, where the conservative penalty is now expressed in terms of a **log-sum-exp** difference (i.e., a soft maximum over all actions minus the expected value of in-dataset actions).

Intuition Behind the Final Objective

- $\log \sum_a \exp(\alpha Q(s,a))$: This term acts like a soft maximum over all possible actions — encourages conservatism by penalizing overestimated values for unseen actions.
- $-\mathbb{E}_{a\sim\hat{\pi}_\beta} Q(s,a)$: Encourages accurate value estimation on actions seen in the dataset.
- **Bellman Error**: Enforces consistency with the dynamics and reward of the MDP.
- **No more max over** μ: It's been analytically solved using entropy regularization!

4. To apply this method in model-based reinforcement learning, what changes are needed in the objective function? Rewrite the new objective function. [10-points]

In offline model-based reinforcement learning, one major challenge is the overestimation of value or reward in regions of the state-action space that are not well covered by the dataset. To address this, model-based methods incorporate uncertainty-aware penalties into the objective function.

MOPO (Model-Based Offline Policy Optimization) proposes modifying the reward function by subtracting an uncertainty penalty, encouraging the agent to avoid regions with high model uncertainty. The modified reward is:

$$\tilde{r}(s,a) = r(s,a) - \lambda u(s,a)$$

where:

- $r(s,a)$ is the original reward,
- $u(s,a)$ is an uncertainty estimate for the transition (s,a),
- λ is a hyperparameter controlling the strength of the penalty.

This uncertainty-penalized reward can then be used with any standard model-based RL algorithm to learn a policy that avoids unreliable parts of the model.

COMBO (Conservative Offline Model-Based Policy Optimization) further integrates the principles of Conservative Q-Learning (CQL) into the model-based setting. It constructs a conservative Q-function by minimizing Q-values on model-generated state-action pairs. The modified Q-function update objective is:

$$\hat{Q}^{k+1} \leftarrow \arg\min_Q \beta \left(\mathbb{E}_{s,a\sim\rho(s,a)}[Q(s,a)] - \mathbb{E}_{s,a\sim\mathcal{D}}[Q(s,a)]\right) + \frac{1}{2}\mathbb{E}_{s,a,s'\sim d_f}\left[\left(Q(s,a) - \hat{\mathcal{B}}^\pi \hat{Q}^k(s,a)\right)^2\right]$$

where:

- $\rho(s,a)$ represents the distribution over state-action pairs from the model,
- $\mathcal{D}$ is the dataset distribution,
- d_f is the forward model rollout distribution,
- $\hat{\mathcal{B}}^\pi$ is the Bellman backup operator,
- β controls the degree of conservatism.

Intuition: By reducing the Q-values of out-of-distribution (OOD) state-action pairs generated by the model, COMBO ensures that the policy avoids exploiting model errors and focuses on actions that are safer and more reliable according to the data.

Thus, the key change in the objective function when applying model-based methods to offline RL is to penalize uncertainty or apply conservative estimation to ensure robustness and prevent overfitting to unreliable model-generated data.

References

[1] Cover image designed by freepik