Sharif University of Technology

Masoud Tahmasbi Fard

Student ID: 402200275

CE957: Reinforcement Learning

Assignment #4

July 12, 2024

Table of Contents

1 Offline RL

1.1 Conservative Q-Learning

In the Conservative Q-Learning (CQL) objective function, the terms $\mathbb{E}_\mu[Q]$ and $\mathbb{E}_{\hat{\pi}_\beta}[Q]$ play crucial roles:

1. $\mathbb{E}_\mu[Q]$: Expected Q-value under the current policy μ

$$\mathbb{E}_\mu[Q] = \mathbb{E}_{s\sim\mathcal{D},a\sim\mu}[Q(s,a)]$$

Where s is sampled from the state distribution $\mathcal{D}$, and a is sampled from the policy μ we're learning.

2. $\mathbb{E}_{\hat{\pi}_\beta}[Q]$: Expected Q-value under the behavior policy $\hat{\pi}_\beta$

$$\mathbb{E}_{\hat{\pi}_\beta}[Q] = \mathbb{E}_{s\sim\mathcal{D},a\sim\hat{\pi}_\beta}[Q(s,a)]$$

Where s is sampled from the state distribution $\mathcal{D}$, and a is sampled from the behavior policy $\hat{\pi}_\beta$.

The interplay between these terms creates a balanced approach:

- $\mathbb{E}_\mu[Q]$ pushes all Q-values down indiscriminately. This provides the conservative aspect, generally lowering Q-values to avoid overestimation, especially for unseen or rarely seen state-action pairs.
- $\mathbb{E}_{\hat{\pi}_\beta}[Q]$ counteracts this by pushing up the Q-values for state-action pairs (s,a) that are actually observed in the offline dataset. This prevents the algorithm from being overly pessimistic.

This balance allows CQL to:

1. Be conservative about unseen or rare state-action pairs, reducing the risk of overoptimistic extrapolation.
2. Maintain higher Q-values for state-action pairs that are well-represented in the dataset, allowing effective learning from available data.

3. Create a "soft constraint" that keeps the learned policy close to the behavior policy in the dataset, while still allowing for some improvement.

The overall effect is a Q-function that is pessimistic about unseen actions but still allows for learning and potential improvement over the behavior policy within the confines of the available data.

1.2 CQL($\mathcal{H}$)

One of the challenges in the original CQL is solving a maximization problem over μ. To address this, we can add a regularization term:

$$\mathcal{R}(\mu) = \mathbb{E}_{s\sim\mathcal{D}}[\mathcal{H}(\mu(\cdot|s))]$$

This term reduces the conservativeness of CQL by encouraging μ to distribute around high Q values instead of learning exactly the high Q points. The advantage of this approach is that it allows the maximization problem to be solved in closed form. Using Lagrange multipliers, we can rewrite the CQL objective as:

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

Focusing on the maximization over μ:

$$\begin{aligned}\max_\mu \mathbb{E}_{\mathbf{s}\sim\mathcal{D},\mathbf{a}\sim\mu(\mathbf{a}|\mathbf{s})}[Q(\mathbf{s},\mathbf{a})] + \mathcal{R}(\mu) &= \max_\mu \mathbb{E}_{\mathbf{s}\sim\mathcal{D},\mathbf{a}\sim\mu(\mathbf{a}|\mathbf{s})}[Q(\mathbf{s},\mathbf{a})] - \mathbb{E}_{\mathbf{s}\sim\mathcal{D},\mathbf{a}\sim\mu(\mathbf{a}|\mathbf{s})}[\log\mu(\mathbf{a}|\mathbf{s})] \\ &= \max_\mu \mathbb{E}_{\mathbf{a}\sim\mu(\mathbf{a}|\mathbf{s})}[Q(\mathbf{s},\mathbf{a}) - \log\mu(\mathbf{a}|\mathbf{s})]\end{aligned}$$

This maximization problem can be formulated with constraints:

$$\max_\mu \mathbb{E}_{\mathbf{a}\sim\mu(\mathbf{a}|\mathbf{s})}[Q(\mathbf{s},\mathbf{a}) - \log\mu(\mathbf{a}|\mathbf{s})] \quad \text{s.t. } \sum_{\mathbf{a}} \mu(\mathbf{a}|\mathbf{s}) = 1,\ \mu(\mathbf{a}|\mathbf{s}) \geq 0$$

Using the method of Lagrange multipliers:

$$\begin{aligned}\mathcal{L} &= \sum_{\mathbf{a}} \mu(\mathbf{a}|\mathbf{s})\left(Q(\mathbf{s},\mathbf{a}) - \log\mu(\mathbf{a}|\mathbf{s})\right) + \lambda\left(\sum_{\mathbf{a}}\mu(\mathbf{a}|\mathbf{s}) - 1\right) \\ \frac{\partial\mathcal{L}}{\partial\mu(\mathbf{a}|\mathbf{s})} &= Q(\mathbf{s},\mathbf{a}) - \log\mu(\mathbf{a}|\mathbf{s}) - 1 + \lambda = 0\end{aligned}$$

Solving this equation yields the optimal policy:

$$\mu^*(\mathbf{a}|\mathbf{s}) = \frac{1}{Z}\exp(Q(\mathbf{s},\mathbf{a})),\ Z = \sum_{\mathbf{a}} \exp(Q(\mathbf{s},\mathbf{a}))$$

Therefore:

$$\begin{aligned}\max_{\mu} \mathbb{E}_{\mathbf{a}\sim\mu(\mathbf{a}|\mathbf{s})}[Q(\mathbf{s},\mathbf{a}) - \log\mu(\mathbf{a}|\mathbf{s})] &= \mathbb{E}_{\mathbf{a}\sim\mu^*(\mathbf{a}|\mathbf{s})}[Q(\mathbf{s},\mathbf{a}) - \log\mu^*(\mathbf{a}|\mathbf{s})] \\ &= \mathbb{E}_{\mathbf{a}\sim\mu^*(\mathbf{a}|\mathbf{s})}[Q(\mathbf{s},\mathbf{a}) + \log Z - Q(\mathbf{s},\mathbf{a})] \\ &= \log Z = \log\sum_{\mathbf{a}}\exp(Q(\mathbf{s},\mathbf{a}))\end{aligned}$$

Hence, the objective function of CQL can be written as:

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

1.3 Updating Policy

We start with the objective function for Q^{k+1} in terms of Q^k, assuming $\mu^*(\mathbf{a}|\mathbf{s})$ is the optimal value of the internal maximization problem:

$$\begin{aligned}&\arg\min_{Q} \alpha\mathbb{E}_{\mathbf{s}\sim\mathcal{D},\mathbf{a}\sim\mu^*(\mathbf{a}|\mathbf{s})}[Q(\mathbf{s},\mathbf{a})] + \frac{1}{2}\mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mathcal{D}}\left[\left(Q(\mathbf{s},\mathbf{a}) - \hat{\mathcal{B}}^{\pi}\hat{Q}^k(\mathbf{s},\mathbf{a})\right)^2\right] \\ =&\arg\min_{Q} \alpha\mathbb{E}_{\mathbf{s}\sim\mathcal{D},\mathbf{a}\sim\mu^*(\mathbf{a}|\mathbf{s})}[Q(\mathbf{s},\mathbf{a})] + \frac{1}{2}\mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mathcal{D}}\left[\left(Q(\mathbf{s},\mathbf{a}) - (r(\mathbf{s},\mathbf{a}) + \mathbb{E}_{\hat{\pi}_\beta}[\hat{Q}(\mathbf{s},\mathbf{a}))]\right)^2\right]\end{aligned}$$

To find the minimum, we set the derivative to zero. We use the following property of expectations:

$$\frac{\partial}{\partial g(x)}\mathbb{E}_{X\sim f(x)}[g(x)] = \frac{\partial}{\partial g(x)}\sum_x f(x)g(x) = f(x)$$

Applying this to our objective function and setting the derivative to zero:

$$\begin{aligned}&\alpha\mu^*(\mathbf{a}|\mathbf{s}) + 2\cdot\frac{1}{2}\hat{\pi}_\beta(\mathbf{a}|\mathbf{s})\left(Q(\mathbf{s},\mathbf{a}) - (r(\mathbf{s},\mathbf{a}) + \mathbb{E}_{\hat{\pi}_\beta}[\hat{Q}(\mathbf{s},\mathbf{a}))]\right) = 0 \ \Rightarrow \\ &\alpha\mu^*(\mathbf{a}|\mathbf{s}) + \hat{\pi}_\beta(\mathbf{a}|\mathbf{s})\left(Q(\mathbf{s},\mathbf{a}) - \hat{\mathcal{B}}^{\pi}\hat{Q}^k(\mathbf{s},\mathbf{a})\right) = 0 \ \Rightarrow \quad Q(\mathbf{s},\mathbf{a}) = \hat{\mathcal{B}}^{\pi}\hat{Q}^k(\mathbf{s},\mathbf{a}) - \alpha\frac{\mu^*(\mathbf{a}|\mathbf{s})}{\hat{\pi}\beta(\mathbf{a}|\mathbf{s})}\end{aligned}$$

Therefore, we can express $\hat{Q}^{k+1}$ in terms of $\hat{Q}^k$ as follows:

$$\forall\mathbf{s},\mathbf{a}\in\mathcal{D},k,\quad \hat{Q}^{k+1}(\mathbf{s},\mathbf{a}) = \hat{\mathcal{B}}^{\pi}\hat{Q}^k(\mathbf{s},\mathbf{a}) - \alpha\frac{\mu^*(\mathbf{a}|\mathbf{s})}{\hat{\pi}_\beta(\mathbf{a}|\mathbf{s})}$$

where μ^* is defined as:

$$\mu^* = \text{argmax}_\mu \, \mathbb{E}_{\mathbf{s}\sim\mathcal{D},\mathbf{a}\sim\mu(\mathbf{a}|\mathbf{s})}[Q(\mathbf{s},\mathbf{a})] + \mathcal{R}(\mu)$$

1.4 Policy Improvement

Theorem 1. *For any $\mu(\mathbf{a}|\mathbf{s})$ with supp $\mu \subseteq$ supp $\hat{\pi}_\beta$, with probability $\geq 1-\delta$, $\hat{Q}^\pi$ (the Q-function obtained by iterating the CQL update equation) satisfies:*

$$\forall \mathbf{s} \in \mathcal{D}, \mathbf{a}, \hat{Q}^\pi(\mathbf{s},\mathbf{a}) \leq Q^\pi(\mathbf{s},\mathbf{a}) - \alpha \left[(I-\gamma P^\pi)^{-1}\frac{\mu}{\hat{\pi}_\beta}\right](\mathbf{s},\mathbf{a}) + \left[(I-\gamma P^\pi)^{-1}\frac{C_{r,T,\delta}R_{\max}}{(1-\gamma)\sqrt{|\mathcal{D}|}}\right](\mathbf{s},\mathbf{a}).$$

Thus, if α is sufficiently large, then $\hat{Q}^\pi(\mathbf{s},\mathbf{a}) \leq Q^\pi(\mathbf{s},\mathbf{a}), \forall \mathbf{s} \in \mathcal{D}, \mathbf{a}$. When $\hat{\beta}^\pi = \beta^\pi$, any $\alpha > 0$ guarantees $\hat{Q}^\pi(\mathbf{s},\mathbf{a}) \leq Q^\pi(\mathbf{s},\mathbf{a}), \forall \mathbf{s} \in \mathcal{D}, \mathbf{a} \in \mathcal{A}$.

Proof. We begin by establishing a bound on the difference between the empirical Bellman operator $\hat{\mathcal{B}}^\pi$ and the actual Bellman operator $\mathcal{B}^\pi$. Under the assumption of concentration properties for the reward function and transition dynamics, we can show that with high probability $(1-\delta)$:

$$\forall Q, \mathbf{s}, \mathbf{a} \in \mathcal{D}, \quad |\hat{\mathcal{B}}^\pi Q(\mathbf{s},\mathbf{a}) - \mathcal{B}^\pi Q(\mathbf{s},\mathbf{a})| \leq \frac{C_{r,T,\delta}R_{\max}}{(1-\gamma)\sqrt{|\mathcal{D}(\mathbf{s},\mathbf{a})|}}$$

This bound is derived from the assumption that the following relationships hold with high probability $(1-\delta)$:

$$|r - r(\mathbf{s},\mathbf{a})| \leq \frac{C_{r,\delta}}{\sqrt{|\mathcal{D}(\mathbf{s},\mathbf{a})|}}, \quad \|\hat{T}(\mathbf{s}'|\mathbf{s},\mathbf{a}) - T(\mathbf{s}'|\mathbf{s},\mathbf{a})\|_1 \leq \frac{C_{T,\delta}}{\sqrt{|\mathcal{D}(\mathbf{s},\mathbf{a})|}}$$

Under this assumption, the difference between the empirical Bellman operator and the actual Bellman operator can be bounded:

$$\begin{aligned}
\left|\left(\hat{\mathcal{B}}^\pi \hat{Q}^k\right) - \left(\mathcal{B}^\pi \hat{Q}^k\right)\right| &= \left|(r - r(\mathbf{s},\mathbf{a})) + \gamma \sum_{\mathbf{s}'} \left(\hat{T}\left(\mathbf{s}' \mid \mathbf{s},\mathbf{a}\right) - T\left(\mathbf{s}' \mid \mathbf{s},\mathbf{a}\right)\right) \mathbb{E}_{\pi(\mathbf{a}'|\mathbf{s}')}\left[\hat{Q}^k\left(\mathbf{s}',\mathbf{a}'\right)\right]\right| \\
&\leq |r - r(\mathbf{s},\mathbf{a})| + \gamma \left|\sum_{\mathbf{s}'} \left(\hat{T}\left(\mathbf{s}' \mid \mathbf{s},\mathbf{a}\right) - T\left(\mathbf{s}' \mid \mathbf{s},\mathbf{a}\right)\right) \mathbb{E}_{\pi(\mathbf{a}'|\mathbf{s}')}\left[\hat{Q}^k\left(\mathbf{s}',\mathbf{a}'\right)\right]\right| \\
&\leq \frac{C_{r,\delta} + \gamma C_{T,\delta} 2R_{\max}/(1-\gamma)}{\sqrt{|\mathcal{D}(\mathbf{s},\mathbf{a})|}}
\end{aligned}$$

Now, consider the CQL update equation:

$$\hat{Q}^{k+1}(\mathbf{s},\mathbf{a}) = \hat{\mathcal{B}}^{\pi}\hat{Q}^{k}(\mathbf{s},\mathbf{a}) - \alpha\frac{\mu(\mathbf{a}|\mathbf{s})}{\hat{\pi}_{\beta}(\mathbf{a}|\mathbf{s})}$$

According to the result of the last section, since, $\mu(\mathbf{a}|\mathbf{s}) > 0, \alpha > 0, \hat{\pi}_{\beta}(\mathbf{a}|\mathbf{s}) > 0$, we observe that at each iteration we underestimate the next Q -value iterate, i.e. $\hat{Q}^{k+1} \leq \hat{\mathcal{B}}^{\pi}\hat{Q}^{k}$. Using the bound on the difference between $\hat{\mathcal{B}}^{\pi}$ and $\mathcal{B}^{\pi}$, we can write:

$$\hat{Q}^{k+1}(\mathbf{s},\mathbf{a}) \leq \mathcal{B}^{\pi}\hat{Q}^{k}(\mathbf{s},\mathbf{a}) - \alpha\frac{\mu(\mathbf{a}|\mathbf{s})}{\hat{\pi}_{\beta}(\mathbf{a}|\mathbf{s})} + \frac{C_{r,T,\delta}R_{\max}}{(1-\gamma)\sqrt{|\mathcal{D}(\mathbf{s},\mathbf{a})|}}$$

At the fixed point of this update procedure, we have:

$$\hat{Q}^{\pi} \leq \mathcal{B}^{\pi}\hat{Q}^{\pi} - \alpha\frac{\mu(\mathbf{a}|\mathbf{s})}{\hat{\pi}_{\beta}(\mathbf{a}|\mathbf{s})} + \frac{C_{r,T,\delta}R_{\max}}{(1-\gamma)\sqrt{|\mathcal{D}(\mathbf{s},\mathbf{a})|}}$$

Rearranging this inequality:

$$\hat{Q}^{\pi} \leq (I - \gamma P^{\pi})^{-1}\left[R - \alpha\frac{\mu}{\hat{\pi}_{\beta}} + \frac{C_{r,T,\delta}R_{\max}}{(1-\gamma)\sqrt{|\mathcal{D}|}}\right]$$

$$\hat{Q}^{\pi}(\mathbf{s},\mathbf{a}) \leq Q^{\pi}(\mathbf{s},\mathbf{a}) - \alpha\left[(I - \gamma P^{\pi})^{-1}\frac{\mu}{\hat{\pi}_{\beta}}\right](\mathbf{s},\mathbf{a}) + \left[(I - \gamma P^{\pi})^{-1}\frac{C_{r,T,\delta}R_{\max}}{(1-\gamma)\sqrt{|\mathcal{D}|}}\right](\mathbf{s},\mathbf{a})$$

This proves the relationship stated in the theorem. When α is sufficiently large, the negative term dominates, ensuring $\hat{Q}^{\pi}(\mathbf{s},\mathbf{a}) \leq Q^{\pi}(\mathbf{s},\mathbf{a})$. In the special case where $\hat{\mathcal{B}}^{\pi} = \mathcal{B}^{\pi}$, the last term vanishes, and any positive α is sufficient to guarantee the inequality. □

2 Inverse RL

2.1 Behavioral Cloning vs Inverse RL

Inverse Reinforcement Learning (IRL) and Behavioral Cloning (BC) differ fundamentally in their approach:

- IRL learns the reward function from demonstrations, then uses it to derive a policy.
- BC learns the policy directly from the demonstration dataset.

This difference leads to several advantages for IRL:

1. **Generalization:** By learning the underlying reward function, IRL often generalizes better to new situations.
2. **Robustness:** The two-step process of IRL (reward learning, then policy optimization) typically results in more robust performance when task dynamics change.
3. **Interpretability:** The learned reward function provides insights into the expert's intentions.
4. **Long-term planning:** IRL can capture overarching goals, while BC may focus on immediate actions.
5. **Data efficiency:** IRL can often extract more information from fewer demonstrations.
6. **Handling suboptimal demonstrations:** IRL can potentially infer true objectives even from imperfect examples.

2.2 Maximum Entropy Inverse Reinforcement Learning

Maximum Entropy Inverse Reinforcement Learning (MaxEnt IRL) addresses the ambiguity problem in feature matching approaches to IRL. In these methods, the goal is to find a reward

function $R(s)$ that makes the expert's behavior optimal. However, multiple reward functions can often explain the observed behavior equally well.

MaxEnt IRL solves this by assuming the expert demonstrates the most likely trajectories, not just the optimal ones. It seeks the maximum entropy distribution over trajectories that matches feature expectations. Mathematically, it models the probability of a trajectory τ as:

$$P(\tau) \propto \exp(R(\tau)) = \exp(\sum_t R(s_t))$$

where $R(\tau)$ is the cumulative reward along the trajectory.

The method aims to find the reward function R that maximizes the likelihood of the demonstrated trajectories D:

$$R^* = \arg\max_R \prod_{\tau \in D} P(\tau|R)$$

subject to the constraint that the expected feature counts match the empirical feature counts:

$$E_P[f_i] = \hat{f}_i \quad \text{for all features } i$$

where f_i are the features and $\hat{f}_i$ are their empirical expectations from demonstrations.

This optimization problem is typically solved using gradient-based methods. The resulting reward function R^* is unique and provides the most general explanation of the demonstrated behavior.

By maximizing entropy, MaxEnt IRL chooses the reward function that makes the demonstrated behavior most likely while remaining as uncertain as possible about everything else. This approach effectively resolves the ambiguity issue by providing a principled way to select among equally fitting reward functions.

The method produces a unique solution to the IRL problem, often leading to more robust and generalizable learned policies. It also captures stochasticity in expert behavior, allowing for suboptimal demonstrations, which is reflected in the probabilistic formulation.

The proof for the exponential form of optimal trajectory probability is provided in the following.

Maximizing the entropy over paths: (as uniform as possible)

$$\max_P -\sum_\tau P(\tau)\log P(\tau)$$

While matching feature counts (and being a probability distribution):

$$\sum_\tau P(\tau) f_\tau = f_{\text{dem}}$$

$$\sum_\tau P(\tau) = 1$$

Cost of a trajectory τ (linear):

$$c_\theta(\tau) = \theta^T \mathbf{f}_\tau = \sum_{s\in\tau} \theta^T \mathbf{f}_s$$

Constraint: (Match the cost of expert trajectories in expectation):

$$\int p(\tau) c_\theta(\tau) d\tau = \frac{1}{|D|} \sum_{\tau^* \in D_\tau} c_\theta(\tau^*)$$

Maximum Entropy:

$$\begin{aligned} \min \quad & -H(p(\tau)) \\ \text{s.t.} \quad & \int p(\tau) c_\theta(\tau) d\tau = \tilde{c}, \int p(\tau) d\tau = 1 \end{aligned}$$

Maximum Entropy:

$$\begin{aligned} &\min \ -H(p(\tau)) \\ &\text{s.t.} \ \int p(\tau) c_\theta(\tau) d\tau = \tilde{c}, \int p(\tau) d\tau = 1 \\ &\iff \mathcal{L}(p,\lambda) = \int p(\tau)\log(p(\tau)) d\tau + \lambda_1 \left(\int p(\tau) c_\theta(\tau) d\tau - \tilde{c} \right) \\ &+ \lambda_0 \left(\int p(\tau) d\tau - 1 \right) \\ &\frac{\partial \mathcal{L}}{\partial p} = \log p(\tau) + 1 + \lambda_1 c_\theta(\tau) + \lambda_0 \\ &\frac{\partial \mathcal{L}}{\partial p} = 0 \iff \log p(\tau) = -1 - \lambda_1 c_\theta(\tau) - \lambda_0 \\ &p(\tau) = e^{(-1-\lambda_0-\lambda_1 c_\theta(\tau))} \\ &p(\tau) \propto e^{c_\theta(\tau)} \end{aligned}$$

- Maximizing the entropy of the distribution over paths subject to the feature constraints from observed data implies that we maximize the likelihood of the observed data under the maximum entropy (exponential family) distribution (Jaynes 1957).

$$P(\tau_i \mid \theta) = \frac{1}{Z(\theta)} e^{\theta^T f_{\tau_i}} = \frac{1}{Z(\theta)} e^{\sum_{s_j \in \tau_i} \theta^T f_{s_j}}$$

$$Z(\theta, s) = \sum_{\tau_S} e^{\theta^T f_{\tau_S}}$$

2.3 Generative Adversarial Imitation Learning (GAIL)

GAIL frames inverse reinforcement learning as a zero-sum game between two players:

1. The policy agent (π) tries to minimize the objective.
2. The discriminator (D) tries to maximize the objective.

The GAIL objective function is:

$$\min_{\pi} \max_{D} \mathbb{E}_{\pi}[\log(D(s,a))] + \mathbb{E}_{\pi_E}[\log(1 - D(s,a))] - \lambda H(\pi)$$

Where π is the learned policy, π_E is the expert policy, $D(s,a)$ is the discriminator, which tries to distinguish between the learned and expert policies, $H(\pi)$ is the entropy of the policy π, and λ is a hyperparameter controlling the entropy regularization

The game proceeds as follows:

1. The discriminator D tries to maximize its ability to distinguish between state-action pairs from the expert policy (π_E) and the learned policy (π).
2. The policy π tries to minimize this difference, effectively "fooling" the discriminator by producing behavior similar to the expert.
3. The entropy term $H(\pi)$ encourages exploration and prevents premature convergence to a deterministic policy.

$D(s, a)$ outputs a probability between 0 and 1, indicating how likely the state-action pair (s, a) came from the expert policy. The policy π aims to make this probability close to 0.5, making it hard for D to distinguish.

$H(\pi)$ is the entropy of the policy, calculated as:

$$H(\pi) = -\sum_{a} \pi(a|s) \log \pi(a|s)$$

This term encourages the policy to maintain some randomness, promoting exploration and preventing overfitting.

The algorithm alternates between optimizing π and D, gradually improving the imitation of the expert policy while maintaining a balance between imitation and exploration.

2.4 MaxEnt RL vs GAIL

Main Problems of Feature Matching Methods (e.g., MaxEnt IRL):

1. **Feature Engineering:** These methods rely heavily on hand-crafted features, which can be challenging to design and may not capture all relevant aspects of the task.

2. **Scalability:** As the complexity of the environment increases, the number of features required often grows, leading to computational challenges.

3. **Expressiveness:** The reward function is limited by the chosen feature set, potentially restricting its ability to represent complex behaviors.

4. **Reward Ambiguity:** Multiple reward functions can often explain the same observed behavior, leading to ambiguity in the solution.

How GAIL Addresses These Issues:

1. **Feature Learning:** GAIL learns a discriminator that implicitly captures relevant features, eliminating the need for manual feature engineering.

2. **Scalability:** By using neural networks for both the policy and discriminator, GAIL can scale to high-dimensional state and action spaces.

3. **Expressiveness:** The discriminator can learn complex, non-linear reward structures, potentially capturing more nuanced behaviors.

4. **Direct Policy Learning:** GAIL bypasses explicit reward function estimation, instead directly learning a policy that mimics the expert behavior. This avoids the reward ambiguity problem.

5. **Robustness:** The adversarial training process can make GAIL more robust to suboptimal demonstrations and variations in expert behavior.

In essence, GAIL shifts the focus from estimating a reward function to directly matching the distribution of expert trajectories, addressing many limitations of traditional IRL approaches.

3 Model-Based Policy Optimization

3.1 Interaction with Environment

In Model-Based Reinforcement Learning (RL), challenges arise when relying only on a learned dynamical model or only interacting with the environment.

Challenges in Pure Model-Based Methods::

- **Model Bias:** Learned dynamics models often accumulate errors over time, leading to unrealistic long-term predictions.
- **Accuracy:** Learned models may not perfectly capture the dynamics of the real environment, leading to inaccurate predictions.
- **Distribution Mismatch:** Differences between the model's predictions and actual environment dynamics can cause performance degradation when using the model for decision-making.
- **Generalization:** Models trained on limited data may struggle to generalize to unseen states or situations, impacting robustness.

Interacting with Environment Challenges:

- **Sample Inefficiency:** Direct interaction with the environment can be time-consuming and costly in terms of data collection, especially in complex or high-dimensional spaces.
- **Exploration vs. Exploitation:** Balancing exploration (to gather data) and exploitation (to maximize rewards) efficiently can be challenging without prior knowledge or model guidance.
- **Safety and Cost:** In some environments, exploration through interaction can be risky or expensive, making it impractical for real-world applications.

Hybrid Models:

- **Short-Horizon Planning:** Leverage the model for short-term planning while relying on model-free methods for long-term value estimation, mitigating model bias.
- **Combining Strengths:** Hybrid models leverage the strengths of both learned models and real-world interaction. They use learned models to simulate environments and predict outcomes, allowing for efficient policy search and evaluation.
- **Adaptation:** By continuously updating the learned model with new interaction data, hybrid models can adapt to changes in the environment, improving accuracy and robustness.
- **Uncertainty-Aware Planning:** Incorporate model uncertainty to guide exploration and avoid overconfidence in inaccurate regions of the learned model.
- **Risk Mitigation:** Hybrid approaches can reduce risks associated with only relying on learned models or direct interaction, providing a more stable and effective framework for reinforcement learning tasks.

3.2 MBPO Learning Process

Combining Environment Interactions and Dynamical Model Learning

MBPO integrates real environment interactions with a learned dynamical model in the following way:

1. **Data Collection:** The agent interacts with the real environment, collecting state-action-reward-next state tuples (s_t, a_t, r_t, s_{t+1}).
2. **Model Learning:** A dynamics model $\hat{p}(s_{t+1}|s_t, a_t)$ is learned from the collected data.
3. **Trajectory Generation:** MBPO generates synthetic trajectories by:
    1. Starting from a real state s_0 sampled from the replay buffer.
    2. Using the learned model to predict next states for k steps.
    3. The generated trajectory is of the form $(s_0, a_0, r_0, \hat{s}_1, a_1, \hat{r}_1, ..., \hat{s}_k)$.

4. **Policy Optimization:** Both real and model-generated data are used to train the policy using an off-policy RL algorithm (e.g., Soft Actor-Critic).

4. **Iterative Improvement:** Steps 1-4 are repeated, continuously improving both the model and the policy.

Trade-off in Rollout Length

The choice of rollout length k presents a crucial trade-off:

- **Large Rollout Length:**
  - *Advantage:* Allows for long-term planning and potentially better policy optimization.
  - *Disadvantage:* Increases the risk of compounding model errors, leading to unrealistic trajectories.
- **Short Rollout Length:**
  - *Advantage:* Minimizes the impact of model errors, keeping generated data more realistic.
  - *Disadvantage:* Limits the ability to learn from long-term consequences of actions.

MBPO addresses this trade-off by:

1. Using relatively short rollouts (often $k \leq 5$) to balance between leveraging the model and avoiding compounding errors.

2. Gradually increasing the rollout length as the model becomes more accurate.

3. Employing uncertainty estimation techniques to detect when model predictions become unreliable.

4. Combining model-generated data with real data to ground the policy in actual environment dynamics.

5. Incorporating Model Predictive Control (MPC): MBPO can use MPC as a complementary approach to address the rollout length trade-off:
    - MPC uses the learned model to plan actions over a short horizon at each time step.
    - It recomputes the plan at every step, using the most recent state information.
    - This allows for longer-term planning while mitigating the impact of model errors.
    - The MPC planner can be used to generate high-quality actions for policy training or as a fallback when the learned policy is uncertain.

This approach allows MBPO to benefit from the sample efficiency of model-based methods while maintaining the asymptotic performance guarantees of model-free algorithms.

3.3 MBPO and Offline RL

MBPO, as originally designed, is not suitable for offline RL due to several key reasons:

1. **Policy Evaluation in Real Environment:**
    - MBPO evaluates and fine-tunes policies in the actual environment.
    - Offline RL cannot perform such evaluations or refinements.
2. **Handling Distribution Shift:**
    - MBPO can correct for distribution shifts by gathering new data.
    - In offline RL, the learned policy may suggest actions outside the dataset distribution, leading to unreliable model predictions.
3. **Reliance on Online Data Collection:**
    - MBPO iteratively collects new data from the environment to train and refine its dynamics model.
    - In offline RL, the agent cannot interact with the environment to gather new data.

4. **Iterative Model Improvement:**
    - MBPO continuously improves its dynamics model with new real-world data.
    - In offline RL, the model can only be trained on the static dataset, potentially leading to overfitting.
5. **Bootstrapping Issues:**
    - MBPO's model rollouts might generate state-action pairs significantly different from the offline dataset.
    - This can lead to compounding errors and unrealistic trajectories in offline settings.

To adapt model-based methods like MBPO for offline RL, significant modifications are necessary:

- Implementing conservative model-based rollouts to stay within the data distribution.
- Incorporating uncertainty estimation to avoid using unreliable model predictions.
- Constraining the policy to actions seen in the dataset.
- Using techniques like importance sampling to correct for distribution mismatch.

3.4 Uncertainty Estimation the Model-based Offline Reinforcement Learning(MOReL)

Ensemble of Neural Networks

This is a popular method for estimating model uncertainty in MOReL:

1. **Approach:** Train multiple (e.g., 5-10) neural networks with different random initializations on the same offline dataset.
2. **Prediction:** For a given state-action pair (s, a), each network in the ensemble produces a prediction of the next state and reward.

3. **Uncertainty Measure:** The disagreement among these predictions is used as a measure of uncertainty.
    - For continuous states: Use the variance of predictions across the ensemble.
    - For discrete states: Use metrics like entropy of the predicted state distribution.
4. **Advantages:**
    - Captures both aleatoric (inherent in the data) and epistemic (due to model limitations) uncertainty.
    - Relatively simple to implement and computationally efficient during inference.
    - Doesn't require modifying the base model architecture.
5. **Usage in MOReL:**
    - High uncertainty indicates state-action pairs poorly represented in the offline dataset.
    - These high-uncertainty regions are treated as terminal states in the constructed MDP, discouraging the policy from entering them.

The ensemble method provides a practical way to estimate uncertainty in complex, high-dimensional environments, making it suitable for use in offline RL algorithms like MOReL.

3.5 Conservative Offline Model-Based Policy Optimization (COMBO)

COMBO integrates conservative Q-learning with model-based RL techniques to address the challenges of offline reinforcement learning.

Key Components

1. **Learned Dynamics Model:**
    - Similar to Dyna, COMBO learns a model of the environment's dynamics from the offline dataset.

- This model is used to generate synthetic data for policy improvement.

2. **Conservative Q-Learning:**

   - Adopts the conservative nature of CQL to address overestimation bias in offline RL.
   - Penalizes Q-values for state-action pairs not well-represented in the dataset.

3. **Model-Based Data Augmentation:**

   - Uses the learned model to generate additional training data.
   - Applies the conservative principle to both real and simulated data.

How COMBO Works

1. **Model Learning:** Train a dynamics model using the offline dataset.
2. **Data Generation:** Use the model to generate synthetic transitions.
3. **Conservative Q-Learning:** Apply CQL to both real and simulated data:

$$\min_Q \underbrace{\alpha \mathbb{E}_{s\sim\mathcal{D}}[\log \sum_a \exp(Q(s,a))]}_{\text{CQL regularization}} - \underbrace{\mathbb{E}_{(s,a)\sim\mathcal{D}}[Q(s,a)]}_{\text{Q-value fitting}}$$

   where $\mathcal{D}$ includes both real and simulated data.

4. **Policy Improvement:** Update the policy to maximize the conservative Q-function.

Benefits from CQL and Dyna

- **From CQL:**
  - Mitigates overestimation bias in Q-values.
  - Encourages conservatism, preventing the policy from exploiting unreliable parts of the Q-function.
- **From Dyna:**

  - Improves sample efficiency by leveraging a learned model.
  - Allows for better generalization beyond the offline dataset.

- **Synergistic Benefits:**
  - The conservatism of CQL helps prevent the policy from exploiting errors in the learned model.
  - Model-based augmentation provides more diverse data for learning, potentially reducing the need for extreme conservatism.

3.6 Conservative Methods vs Uncertainty-Based Methods

Conservative Methods

Key Idea: Introduce a pessimistic bias to prevent overestimation and constrain the policy to the data distribution.

- Penalize Q-values for state-action pairs not well-represented in the dataset.
- Often use a regularization term to push down Q-values globally.
- Focus on learning a lower bound on the true Q-function.

Advantages:

- Simple to implement and integrate with existing Q-learning algorithms.
- Effective at preventing extrapolation errors.
- Often provide theoretical guarantees on policy performance.

Limitations:

- May be overly conservative, limiting performance in well-explored regions.
- Tuning the level of conservatism can be challenging.

Uncertainty-Based Methods

Key Idea: Explicitly estimate uncertainty in value or model predictions and use it to guide decision-making.

- Use techniques like ensembles or Bayesian neural networks to estimate uncertainty.
- Often construct uncertainty-penalized MDPs.
- May treat high-uncertainty states as terminal or add uncertainty penalties to rewards.

Advantages:

- Can provide more nuanced estimates of reliability for different state-action pairs.
- Potentially less conservative in well-explored regions of the state space.
- Uncertainty estimates can be used for safe exploration in some settings.

Limitations:

- Accurate uncertainty estimation can be challenging, especially in high-dimensional spaces.
- May be computationally more expensive, especially for ensemble methods.
- Performance heavily depends on the quality of uncertainty estimates.

Comparison

- **Granularity:** Uncertainty-based methods often provide more fine-grained assessments of data reliability, while conservative methods tend to apply a more uniform pessimism.
- **Computational Cost:** Conservative methods are typically less computationally intensive than uncertainty-based methods.
- **Theoretical Guarantees:** Conservative methods often come with stronger theoretical guarantees, while the guarantees for uncertainty-based methods depend on the accuracy of uncertainty estimation.

- **Flexibility:** Uncertainty-based methods can potentially adapt better to datasets with varying levels of coverage in different regions of the state space.
- **Interpretability:** Uncertainty estimates can provide more interpretable measures of model confidence, which can be valuable in some applications.

In practice, some methods (like COMBO) combine elements of both approaches to leverage their respective strengths.

4 Bandit Learning

4.1 UCB Upper Bound

Proof of Markov Inequality

Theorem 2 (Markov Inequality)**.** *For a non-negative random variable X and $a > 0$,*

$$P(X \geq a) \leq \frac{E[X]}{a}$$

Proof. Let I be the indicator function of the event $\{X \geq a\}$. Then:

$$I = \begin{cases} 1 & \text{if } X \geq a \\ 0 & \text{otherwise} \end{cases}$$

Observe that $aI \leq X$ for all outcomes. Therefore:

$$E[aI] \leq E[X]$$

The left side can be simplified:

$$aE[I] \leq E[X]$$

$$aP(X \geq a) \leq E[X]$$

Dividing both sides by a:

$$P(X \geq a) \leq \frac{E[X]}{a}$$

□

Proof of Chernoff Bound

The Chernoff bound is an application of the Markov inequality to the moment generating function.

Theorem 3 (Chernoff Bound)**.** *For a random variable X and $t > 0$,*

$$P(X \geq a) \leq \frac{E[e^{tX}]}{e^{ta}}$$

Proof. Apply the Markov inequality to e^{tX}:

$$P(e^{tX} \geq e^{ta}) \leq \frac{E[e^{tX}]}{e^{ta}}$$

Since $t > 0$, the events $\{X \geq a\}$ and $\{e^{tX} \geq e^{ta}\}$ are equivalent. Therefore:

$$P(X \geq a) \leq \frac{E[e^{tX}]}{e^{ta}}$$

□

Proof of Hoeffding's Bound

Hoeffding's bound is a special case of the Chernoff bound for the sum of independent, bounded random variables.

Theorem 4 (Hoeffding's Bound)**.** *Let $X_1, ..., X_n$ be independent random variables with $a_i \leq X_i \leq b_i$ for all i. Let $S_n = \sum_{i=1}^{n} X_i$. Then for any $t > 0$:*

$$P(S_n - E[S_n] \geq t) \leq \exp\left(-\frac{2t^2}{\sum_{i=1}^{n}(b_i - a_i)^2}\right)$$

Proof. We'll use the Chernoff bound and then bound the moment generating function.

For any $s > 0$:

$$P(S_n - E[S_n] \geq t) = P(e^{s(S_n - E[S_n])} \geq e^{st}) \leq \frac{E[e^{s(S_n - E[S_n])}]}{e^{st}}$$

Now, we need to bound $E[e^{s(S_n - E[S_n])}]$. Using the independence of X_i:

$$E[e^{s(S_n - E[S_n])}] = \prod_{i=1}^{n} E[e^{s(X_i - E[X_i])}]$$

For each term, we can use Hoeffding's lemma:

$$E[e^{s(X_i - E[X_i])}] \leq \exp\left(\frac{s^2(b_i - a_i)^2}{8}\right)$$

Proof of Hoeffding's lemma

Theorem 5 (Hoeffding's lemma)**.** *Let X be any real-valued random variable such that $a \leq X \leq b$ almost surely, i.e., with probability one. Then, for all $\lambda \in \mathbb{R}$,*

$$\mathbb{E}\left[e^{\lambda X}\right] \leq \exp\left(\lambda \mathbb{E}[X] + \frac{\lambda^2 (b-a)^2}{8}\right),$$

or equivalently,

$$\mathbb{E}\left[e^{\lambda (X - \mathbb{E}[X])}\right] \leq \exp\left(\frac{\lambda^2 (b-a)^2}{8}\right).$$

Proof. Without loss of generality, by replacing X by $X - \mathbb{E}[X]$, we can assume $\mathbb{E}[X] = 0$, so that $a \leq 0 \leq b$. Since $e^{\lambda x}$ is a convex function of x, we have that for all $x \in [a, b]$,

$$e^{\lambda x} \leq \frac{b-x}{b-a} e^{\lambda a} + \frac{x-a}{b-a} e^{\lambda b}$$

So,

$$\begin{aligned}
\mathbb{E}\left[e^{\lambda X}\right] &\leq \frac{b - \mathbb{E}[X]}{b-a} e^{\lambda a} + \frac{\mathbb{E}[X] - a}{b-a} e^{\lambda b} \\
&= \frac{b}{b-a} e^{\lambda a} + \frac{-a}{b-a} e^{\lambda b} \\
&= e^{L(\lambda (b-a))}
\end{aligned}$$

where $L(h) = \frac{ha}{b-a} + \ln\left(1 + \frac{a - e^h a}{b-a}\right)$. By computing derivatives, we find

$$L(0) = L'(0) = 0 \text{ and } L''(h) = -\frac{abe^h}{\left(b - ae^h\right)^2}.$$

From the inequality of arithmetic and geometric means we thus see that $L''(h) \leq \frac{1}{4}$ for all h, and thus, from Taylor's theorem, there is some $0 \leq \theta \leq 1$ such that

$$L(h) = L(0) + hL'(0) + \frac{1}{2} h^2 L''(h\theta) \leq \frac{1}{8} h^2.$$

Thus, $\mathbb{E}\left[e^{\lambda X}\right] \leq e^{\frac{1}{8}\lambda^2 (b-a)^2}$. □

Applying Hoeffding's lemma to all terms:

$$E[e^{s(S_n - E[S_n])}] \leq \exp\left(\frac{s^2 \sum_{i=1}^n (b_i - a_i)^2}{8}\right)$$

Substituting back into the Chernoff bound:

$$P(S_n - E[S_n] \geq t) \leq \exp\left(\frac{s^2 \sum_{i=1}^n (b_i - a_i)^2}{8} - st\right)$$

To get the tightest bound, we minimize the right-hand side with respect to s. The optimal s is:

$$s = \frac{4t}{\sum_{i=1}^{n}(b_i - a_i)^2}$$

Substituting this back:

$$P(S_n - E[S_n] \geq t) \leq \exp\left(-\frac{2t^2}{\sum_{i=1}^{n}(b_i - a_i)^2}\right)$$

□

The Upper Confidence Bound (UCB) algorithm uses Hoeffding's bound to determine an upper bound on the expected value of reward for each action.

Let $\hat{\mu}_i(1), \hat{\mu}_i(2), ..., \hat{\mu}_i(T_i(t))$ be the rewards observed for action i up to time t, where $T_i(t)$ is the number of times action i has been chosen.

Let $\bar{\hat{\mu}}_i(t) = \frac{1}{T_i(t)} \sum_{s=1}^{T_i(t)} \hat{\mu}_i(s)$ be the sample mean reward for action i.

We want to find an upper bound $U_i(t)$ such that $P(\mu_i > U_i(t)) \leq \delta$, where μ_i is the true expected reward of action i and δ is a small probability.

Applying the Hoeffding's bound:

$$P(\bar{\hat{\mu}}_i(t) - \mu_i \geq \epsilon) \leq \exp\left(-\frac{2T_i(t)\epsilon^2}{(b-a)^2}\right)$$

where $[a, b]$ is the range of the rewards.

Setting this equal to δ and solving for ϵ:

$$\epsilon = \sqrt{\frac{(b-a)^2 \ln(1/\delta)}{2T_i(t)}}$$

Therefore, with probability at least $1 - \delta$:

$$\mu_i \leq \bar{\hat{\mu}}_i(t) + \sqrt{\frac{(b-a)^2 \ln(1/\delta)}{2T_i(t)}}$$

This gives us the UCB formula:

$$U_i(t) = \bar{\hat{\mu}}_i(t) + \sqrt{\frac{(b-a)^2 \ln(1/\delta)}{2T_i(t)}}$$

In practice, UCB often uses $\delta = 1/t$, leading to the common UCB1 formula:

$$UCB1 = \bar{\hat{\mu}}_i(t) + \sqrt{\frac{2 \ln t}{n_i(t)}}$$

where we assume rewards are in [-1,1], so $(b-a)^2 = 4$.

This formula provides an upper bound on the expected reward of each action, balancing exploitation (the sample mean $\bar{\hat{\mu}}_i(t)$) with exploration (the confidence term $\sqrt{\frac{2 \ln t}{T_i(t)}}$).

4.2 Part b & c

We want to prove that for any horizon n, if $\delta = 1/n^2$, then:

$$R_n \leq 3 \sum_{i=1}^{k} \Delta_i + \sum_{i:\Delta_i > 0} \frac{16 \log(n)}{\Delta_i}.$$

Before the proof we need a little more notation. Let $(X_{ti})_{t \in [n], i \in [k]}$ be a collection of independent random variables with the law of X_{ti} equal to P_i. Then define $\hat{\mu}_{is} = \frac{1}{s} \sum_{u=1}^{s} X_{ui}$ to be the empirical mean based on the first s samples. Assuming that the reward in round t is

$$X_t = X_{T_{A_t}(t) A_t}.$$

Then we define $\hat{\mu}_i(t) = \hat{\mu}_{iT_i(t)}$ to be the empirical mean of the ith arm after round t. This proof relies on the basic regret decomposition identity,

$$R_n = \sum_{i=1}^{k} \Delta_i \mathbb{E}[T_i(n)]. \tag{1}$$

We will show that $\mathbb{E}[T_i(n)]$ is not too large for suboptimal arms i. The key observation is that after the initial period where the algorithm chooses each action once, action i can only be chosen if its index is higher than that of an optimal arm. This can only happen if at least one of the following is true:

1. The index of action i is larger than the true mean of a specific optimal arm.

2. The index of a specific optimal arm is smaller than its true mean.

Since with reasonably high probability the index of any arm is an upper bound on its mean, we don't expect the index of the optimal arm to be below its mean. Furthermore, if the suboptimal arm i is played sufficiently often, then its exploration bonus becomes small and simultaneously the empirical estimate of its mean converges to the true value, putting an upper bound on the expected total number of times when its index stays above the mean of the optimal arm.

Without loss of generality, we assume the first arm is optimal so that $R_1 = R_{max} \quad (\mu_1 = \mu^*)$. As noted above,

$$R_n = \sum_{i=1}^{k} \Delta_i \mathbb{E}[T_i(n)].$$

We will bound $\mathbb{E}[T_i(n)]$ for each suboptimal arm i. We make use of a relatively standard idea, which is to decouple the randomness from the behaviour of the UCB algorithm. Let G_i be the 'good' event defined by

$$G_i = \{\mu_1 < \min_{t \in [n]} \mathrm{UCB}_1(t, \delta)\} \cap \{\hat{\mu}_{iu_i} + \sqrt{\frac{2}{u_i} \log(\frac{1}{\delta})} < \mu_1\},$$

where $u_i \in [n]$ is a constant to be chosen later. So G_i is the event when μ_1 is never underestimated by the upper confidence bound of the first arm, while at the same time the upper confidence bound for the mean of arm i after u_i observations are taken from this arm is below the pay-off of the optimal arm. We will show two things:

1. If G_i occurs, then arm i will be played at most u_i times: $T_i(n) \leq u_i$.
2. The complement event G_i^c occurs with low probability (governed in some way yet to be discovered by u_i).

Because $T_i(n) \leq n$ no matter what, this will mean that

$$\mathbb{E}[T_i(n)] = \mathbb{E}[\mathbb{I}\{G_i\}T_i(n)] + \mathbb{E}[\mathbb{I}\{G_i^c\}T_i(n)] \leq u_i + \mathbb{P}(G_i^c)n. \tag{2}$$

The next step is to show that $T_i(n) \leq u_i$ on G_i and that $\mathbb{P}(G_i^c)$ is small. Let us first assume that G_i holds and show that $T_i(n) \leq u_i$, which we do by contradiction. Suppose that $T_i(n) > u_i$. Then arm i was played more than u_i times over the n rounds, and so there must exist a round $t \in [n]$ where $T_i(t-1) = u_i$ and $A_t = i$. Using the definition of G_i,

$$\begin{aligned}
\mathrm{UCB}_i(t-1,\delta) &= \hat{\mu}_i(t-1) + \sqrt{\frac{2\log(1/\delta)}{T_i(t-1)}} && \text{(definition of } \mathrm{UCB}_i(t-1,\delta)) \\
&= \hat{\mu}_{iu_i} + \sqrt{\frac{2\log(1/\delta)}{u_i}} && \text{(since } T_i(t-1) = u_i) \\
&< \mu_1 && \text{(definition of } G_i) \\
&< \mathrm{UCB}_1(t-1,\delta). && \text{(definition of } G_i)
\end{aligned}$$

Hence $A_t = \arg\max_j \mathrm{UCB}_j(t-1,\delta) \neq i$, which is a contradiction. Therefore if G_i occurs, then $T_i(n) \leq u_i$. Let us now turn to upper bounding $\mathbb{P}(G_i^c)$. By its definition,

$$G_i^c = \{\mu_1 \geq \min_{t\in[n]} \mathrm{UCB}_1(t,\delta)\} \cup \{\hat{\mu}_{iu_i} + \sqrt{\frac{2\log(1/\delta)}{u_i}} \geq \mu_1\}. \tag{3}$$

The first of these sets is decomposed using the definition of $\mathrm{UCB}_1(t,\delta)$,

$$\begin{aligned}
\{\mu_1 \geq \min_{t\in[n]} \mathrm{UCB}_1(t,\delta)\} &\subset \{\mu_1 \geq \min_{s\in[n]} \hat{\mu}_{1s} + \sqrt{\frac{2\log(1/\delta)}{s}}\} \\
&= \bigcup_{s\in[n]} \{\mu_1 \geq \hat{\mu}_{1s} + \sqrt{\frac{2\log(1/\delta)}{s}}\}.
\end{aligned}$$

Then using a union bound and the concentration bound for sums of independent subgaussian random variables, we obtain:

$$\mathbb{P}(\mu_1 \geq \min_{t\in[n]} \mathrm{UCB}_1(t,\delta)) \leq \mathbb{P}\left(\bigcup_{s\in[n]} \{\mu_1 \geq \hat{\mu}_{1s} + \sqrt{\frac{2\log(1/\delta)}{s}}\}\right) \tag{4}$$

$$\leq \sum_{s=1}^{n} \mathbb{P}\left(\mu_1 \geq \hat{\mu}_{1s} + \sqrt{\frac{2\log(1/\delta)}{s}}\right) \leq n\delta. \tag{5}$$

The next step is to bound the probability of the second set in Eq. 3. Assume that u_i is chosen large enough that

$$\Delta_i - \sqrt{\frac{2\log(1/\delta)}{u_i}} \geq c\Delta_i \tag{6}$$

for some $c \in (0,1)$ to be chosen later. Then, since $\mu_1 = \mu_i + \Delta_i$,

$$\begin{aligned}
\mathbb{P}\left(\hat{\mu}_{iu_i} + \sqrt{\frac{2\log(1/\delta)}{u_i}} \geq \mu_1\right) &= \mathbb{P}\left(\hat{\mu}_{iu_i} - \mu_i \geq \Delta_i - \sqrt{\frac{2\log(1/\delta)}{u_i}}\right) \\
&\leq \mathbb{P}(\hat{\mu}_{iu_i} - \mu_i \geq c\Delta_i) \leq \exp\left(-\frac{u_i c^2 \Delta_i^2}{2}\right).
\end{aligned}$$

Taking this together with 5 and 6, we have

$$\mathbb{P}(G_i^c) \leq n\delta + \exp\left(-\frac{u_i c^2 \Delta_i^2}{2}\right).$$

When substituted into Eq. 2 we obtain

$$\mathbb{E}[T_i(n)] \leq u_i + n\left(n\delta + \exp\left(-\frac{u_i c^2 \Delta_i^2}{2}\right)\right). \tag{7}$$

It remains to choose $u_i \in [n]$ satisfying Eq. 6. A natural choice is the smallest integer for which Eq. 6 holds, which is

$$u_i = \left\lceil \frac{2\log(1/\delta)}{(1-c)^2 \Delta_i^2} \right\rceil.$$

This choice of u_i can be larger than n, but in this case Eq. 7 holds trivially since $T_i(n) \leq n$. Then, using the assumption that $\delta = 1/n^2$ and this choice of u_i leads via 7 to

$$\mathbb{E}[T_i(n)] \leq u_i + 1 + n^{1-2c^2/(1-c)^2} = \left\lceil \frac{2\log(n^2)}{(1-c)^2 \Delta_i^2} \right\rceil + 1 + n^{1-2c^2/(1-c)^2}.$$

All that remains is to choose $c \in (0, 1)$. The second term will contribute a polynomial dependence on n unless $2c^2/(1-c)^2 \geq 1$. However, if c is chosen too close to 1, then the first term blows up. Somewhat arbitrarily we choose $c = 1/2$, which leads to

$$\mathbb{E}[T_i(n)] \leq 3 + \frac{16\log(n)}{\Delta_i^2}.$$

The result follows by substituting the above display in Eq. 1.

4.3 Part d

If $\delta = 1/n^2$, then the regret of UCB, is bounded by

$$R_n \leq 8\sqrt{nk\log(n)} + 3\sum_{i=1}^{k} \Delta_i.$$

Let $\Delta > 0$ be some value to be tuned subsequently, and recall from the proof of previous part that for each suboptimal arm i, we can bound

$$\mathbb{E}[T_i(n)] \leq 3 + \frac{16\log(n)}{\Delta_i^2}.$$

Therefore, using the basic regret decomposition again, we have

$$\begin{aligned} R_n &= \sum_{i=1}^{k} \Delta_i \mathbb{E}[T_i(n)] = \sum_{i:\Delta_i<\Delta} \Delta_i \mathbb{E}[T_i(n)] + \sum_{i:\Delta_i\geq\Delta} \Delta_i \mathbb{E}[T_i(n)] \\ &\leq n\Delta + \sum_{i:\Delta_i\geq\Delta} \left(3\Delta_i + \frac{16\log(n)}{\Delta_i}\right) \\ &\leq n\Delta + \frac{16k\log(n)}{\Delta} + 3\sum_i \Delta_i \\ &\leq 8\sqrt{nk\log(n)} + 3\sum_{i=1}^{k} \Delta_i, \end{aligned}$$

where the first inequality follows because $\sum_{i:\Delta_i<\Delta} T_i(n) \leq n$ and the last line by choosing $\Delta = \sqrt{16k\log(n)/n}$.