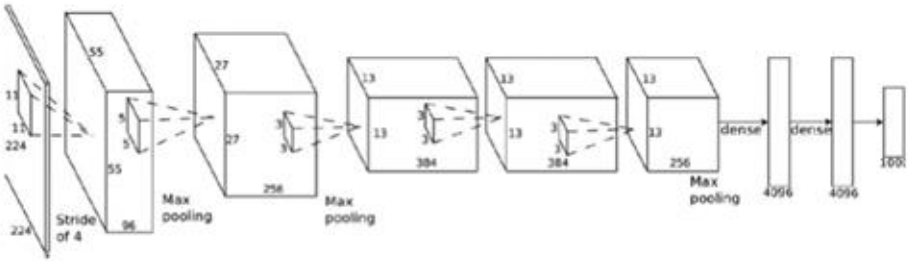
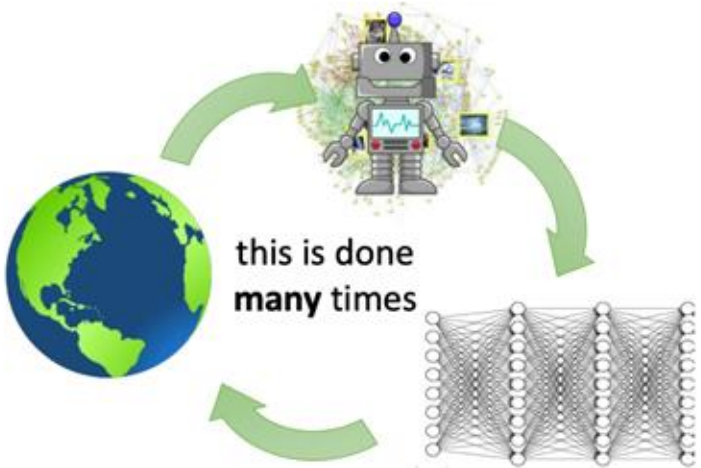
Offline Reinforcement Learning

Hanpu Shen
UC Irvine

What makes modern machine learning work?

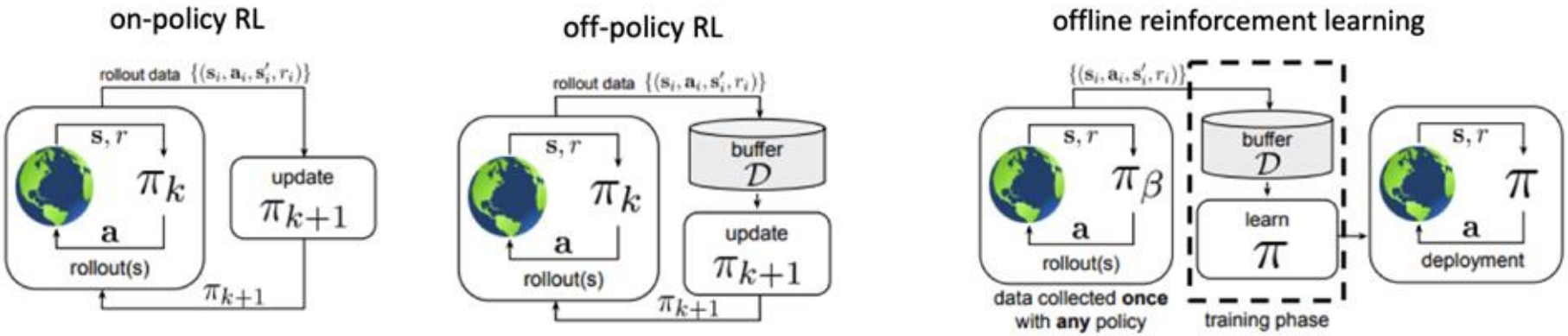


Why should we care Offline RL?



What does offline RL mean?

On policy — Off policy — Offline



$$(s,a) \sim P_{\pi_k} \quad (s,a) \sim \text{mix}(P_{\pi_0}, \cdots, P_{\pi_k}) \qquad (s,a) \sim P_{\pi_\beta}$$

Offline RL problems: improve over the behavior policy

$$\mathcal{D} = \{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)\}$$

$$\mathbf{s} \sim d^{\pi_\beta}(\mathbf{s})$$
$$\mathbf{a} \sim \pi_\beta(\mathbf{a}|\mathbf{s})$$
$$\mathbf{s}' \sim p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$$
$$r \leftarrow r(\mathbf{s}, \mathbf{a})$$

Offline RL objective:

Given $\mathcal{D}$, learn the best possible policy π_θ.

- Best on support?

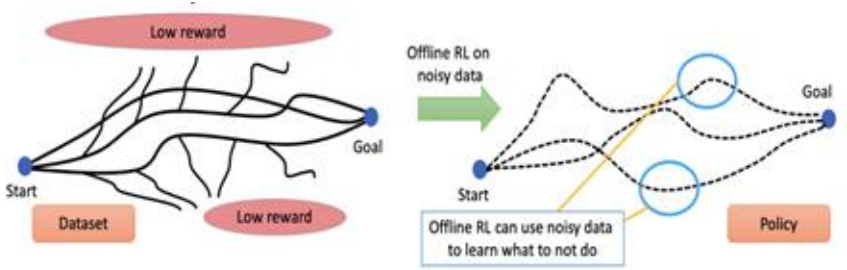
  $$\pi_\theta \in \mathrm{Supp}(\pi_\beta)$$

- How about generalization?

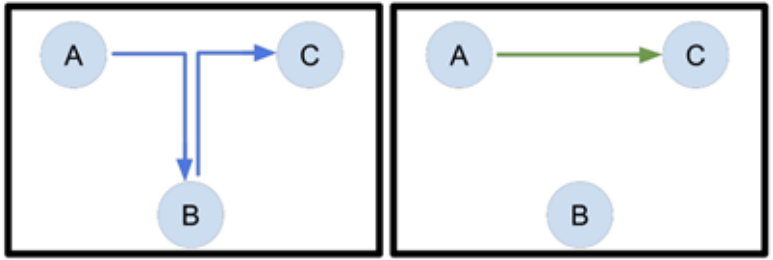
  Still an open question. This usually classified into meta RL

How is this even possible?

Learn from noise data: find the “good stuff” in the dataset mixed by good and bad behaviors

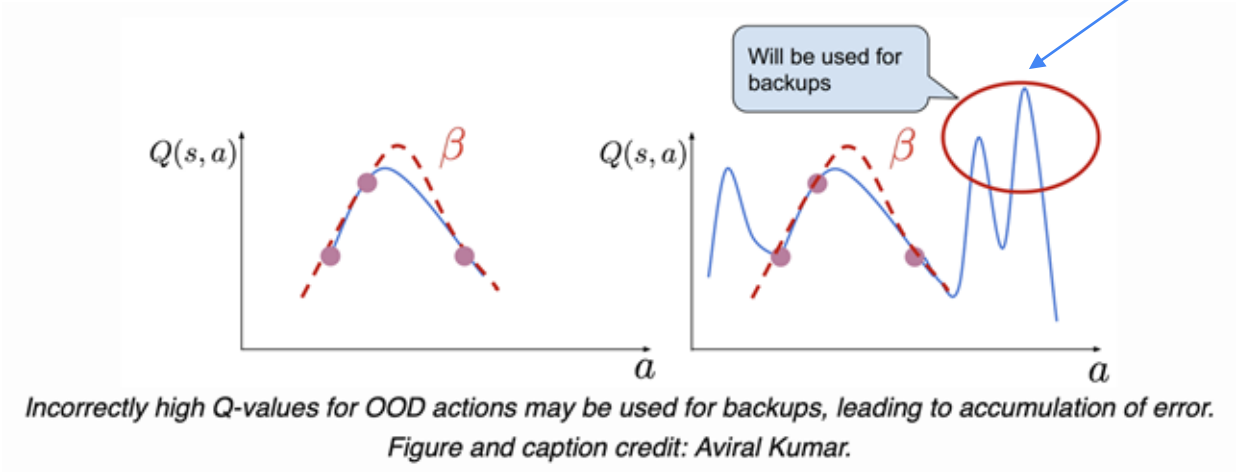


Stitching: parts of good behaviors can be recombined to form better policy



Why not just use Off-policy RL algorithm?

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s', a'))$$



Incorrectly high Q-values for OOD actions may be used for backups, leading to accumulation of error.

Figure and caption credit: Aviral Kumar.

Bellman backup on the OOD actions can result in overestimated Q values. This is the main challenge for offline RL – distribution shift.

Distribution shift (cont.)

Considering ERM problem: $\theta \leftarrow \arg\min_{\theta} E_{\mathbf{x}\sim p(\mathbf{x}), y\sim p(y|\mathbf{x})}\left[(f_\theta(\mathbf{x}) - y)^2\right]$

In supervised learning, usually we are not worried – neural nets generalize well!

In RL setting the objective becomes: $\min_{Q} E_{(\mathbf{s},\mathbf{a})\sim\pi_\beta(\mathbf{s},\mathbf{a})}\left[(Q(\mathbf{s},\mathbf{a}) - y(\mathbf{s},\mathbf{a}))^2\right]$

behavior policy

target value

$r(s,a) + E_{a'\sim\pi_{\text{new}}} Q(s',a')$

The target value is evaluating the new policy but only have access to the behavior policy data. Would be not accurate when $D(\pi_{\text{new}} \| \pi_\beta)$ is large.

Summary of existing Offline RL

Offline RL algorithm are predominantly designed around the **principle of pessimism** to overcome the OOD issue. For the Model-free methods

- **Explicit Policy Constraints:** $D(\pi_{\text{new}} \| \pi_\beta)$ is small. Like CQL (2019), TD3+BC (2021), ReBRAC (2023). And DICE type of algorithm which transfer the RL problem into a linear programming problem (Algaedice 2019, OptiDICE 2021, ODICE 2024, Diffusion-DICE 2024).
- **Implicit Value Constraints:** Avoid OOD query $supp(\pi_{\text{new}}) \subseteq supp(\pi_\beta)$. Like IQL (2021), EQL (2023), IVR (2023), IQDL (2023)

ReBRAC: Explicit policy constraint method

The most straightforward idea is to bring the OOD penalization into the learning objective. And ReBRAC is inspired by this idea and scale up the NN architecture. They considered deterministic policy.

- Algorithm: Bring the MSE penalization to the actor-critic algorithm

$$\pi = \underset{\pi}{\operatorname{argmax}}\, \mathbb{E}_{(s,a)\sim D}\left[Q_\theta(s,\pi(s)) - \beta_1 \cdot (\pi(s)-a)^2\right]$$

$$\theta = \underset{\theta}{\operatorname{argmin}}\, \mathbb{E}_{\substack{(s,a,r,s',\hat{a}')\sim D \\ a'\sim\pi(s')}}\left[(Q_\theta(s,a) - (r+\gamma(Q_{\bar{\theta}}(s',a') - \beta_2 \cdot (a'-\hat{a}')^2)))^2\right]$$

In implementation, LayerNorm and the actor penalty is considered as the most critical component.

DICE: Explicit policy constraint method

Instead of directly constraint on the policy divergence, Distribution Correction Estimation (DICE) constraint on the (s,a) occupancy measure $d_\pi(s,a)$. More specifically, the following regularized RL problem:

$$\max_\pi E_{(s,a)\sim d_\pi}[r(s,a)] - \alpha D_f(d_\pi(s,a)\|d_D(s,a))$$

$$s.t.\ d_\pi(s,a) = (1-\gamma)d_0(s)\pi(a|s) + \gamma \sum_{s',a'} \pi(a|s)p(s|s',a')d_\pi(s',a')$$

Where the regularizer encourages the agent to the data support $d_D(s,a)$. This problem has regularization free dual expression due to the Fenchel duality.

DICE (cont.)

By applying the duality and convex conjugate, we can derive the final actor-critic style learning objective:

$$\min_V \alpha E_{(s,a)\sim D}\left[f^*([r(s,a)+\gamma E_p V(s')) - V(s)]/\alpha)\right] + (1-\gamma)E_{s\sim d_0}[V(s)]$$

Where $f^*(x) := \sup_y x^T y - f(y)$ is the convex conjugate of f.

With the policy extraction rule by minimize the $D_f(d_\pi \| d_{\pi^*|D})$:

$$\max_\pi [\omega(s,a)\log\pi(a|s)]$$

Where $\omega(s,a) = \max(0, (f')^{-1}(A(s,a)))$ is the ratio between the optimal policy occupancy and the data support.

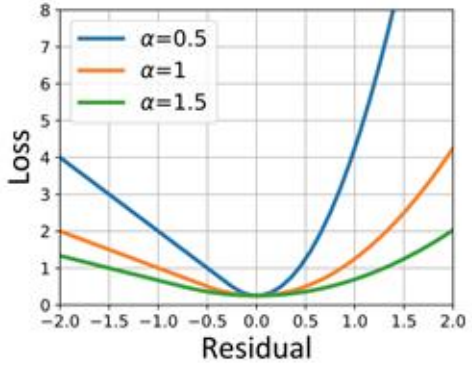
DICE: practical algorithm

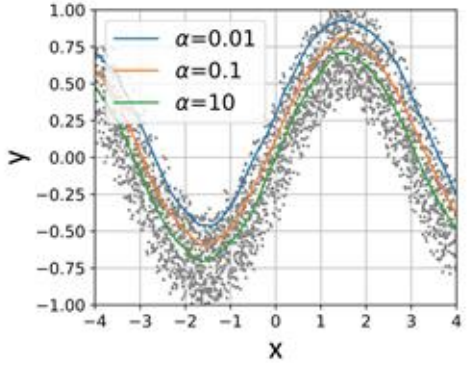
We are now ready to instance the DICE by choosing Pearson Chi square divergence. So the final objectives look is

$$\min_V E_{(s,a,s')\sim D}[\max(0, 1 + \frac{1}{2\alpha}(r(s,a) + \gamma V(s') - V(s)))^2] + \frac{E_{s\sim D}[V(s)]}{\alpha}$$

$$\max_\pi E_{(s,a)\sim D}[\max(0, A(s,a)) \log \pi(a|s)]$$

$\alpha \to 0$ finds the max in-support value of action





Implicit Q-Learning (IQL): Implicate Value method

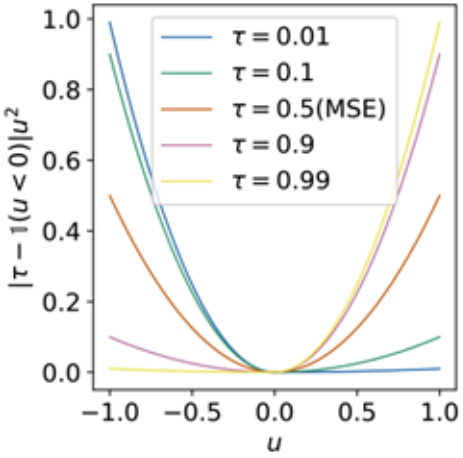
IQL is probably the most well known offline RL algorithm. The idea is to focusing on find the actions that supported by the behavior policy with in-sample inquiry.

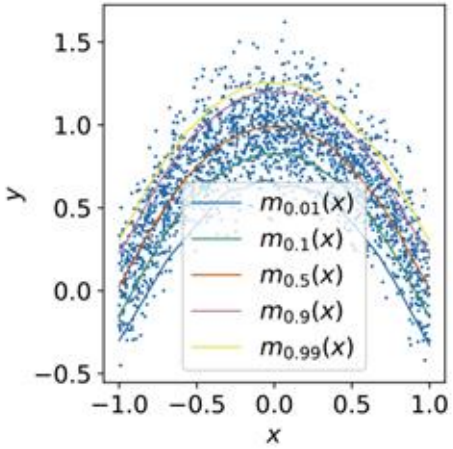
$$\max_{a' \; s.t. \; \mu(a'|s')>0} Q_{\bar{\theta}}(s', a') = \arg\min_{V_\psi} \mathbb{E}_{(s,a)\sim D}\left[\max(0, Q_{\bar{\theta}}(s,a) - V_\psi(s))^2\right]$$

However, directly optimizing this objective is unstable and prone to overestimation due to “lucky” samples. IQL introduced a smart trick using expectile regression to softly approximate the in-sample maximization.

$$\mathcal{L}_V(\psi) = \mathbb{E}_{(s,a)\sim D}\left[L_2^\tau(Q_{\bar{\theta}}(s,a) - V_\psi(s))\right] \quad L_2^\tau(u) = |\tau - \mathbb{I}(u<0)|u^2$$

Implicit Q-Learning (IQL): Implicate Value method





$$\mathcal{L}_V(\psi) = \mathbb{E}_{(s,a)\sim D}\left[L_2^\tau(Q_{\bar{\theta}}(s,a) - V_\psi(s))\right] \qquad L_2^\tau(u) = |\tau - \mathbb{I}(u < 0)|u^2$$

IQL: What about policy extraction?

Unlike DICE formulation, IQL doesn't have the direct solving for the optimal policy. Instead, it extract the decoupled policy implied by the value function.

They considered using Advantage Weighted Regression (AWR), which can be viewed as

$$\pi_{\text{new}}(\mathbf{a}|\mathbf{s}) = \arg\max_{\pi} E_{\mathbf{a}\sim\pi(\mathbf{a}|\mathbf{s})}[Q(\mathbf{s},\mathbf{a})] \text{ s.t. } D_{\text{KL}}(\pi\|\pi_\beta) \leq \epsilon$$

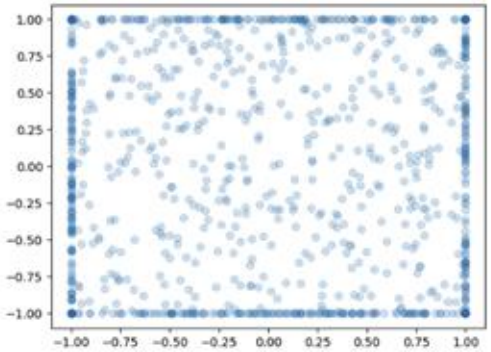
$$\pi^\star(\mathbf{a}|\mathbf{s}) = \frac{1}{Z(\mathbf{s})}\pi_\beta(\mathbf{a}|\mathbf{s})\exp\left(\frac{1}{\lambda}A^\pi(\mathbf{s},\mathbf{a})\right)$$

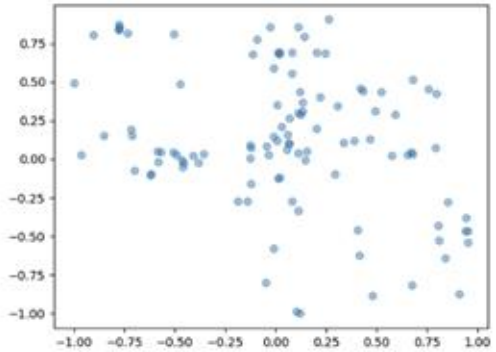
straightforward to show via duality

IQL: Can we try different policy extraction?

downside of AWR is that we assign probability mass to each action in the dataset. Since we usually use a Gaussian policy, this leads to bad behavior.

As an example, let's look at maze-2d. Here the actions are (x,y) velocities to move an agent to a goal. It's not complicated; and intuitively the best actions should be along the edge of the unit sphere. But, the policy doesn't behave that way.





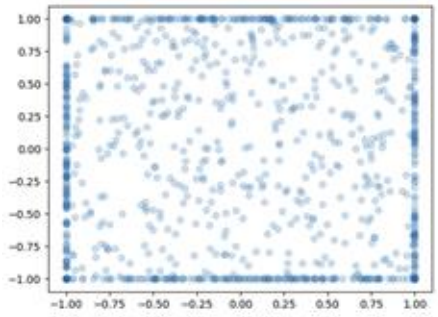
AWR

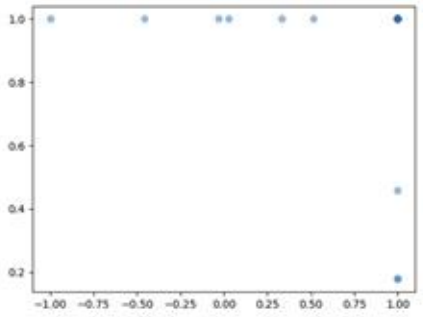
IQL: Can we try different policy extraction?

downside of AWR is that we assign probability mass to each action in the dataset. Since we usually use a Gaussian policy, this leads to bad behavior.

Let's try to fix this: DDPG-style extraction optimizes *through the Q-function* to find the best action at each state.

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





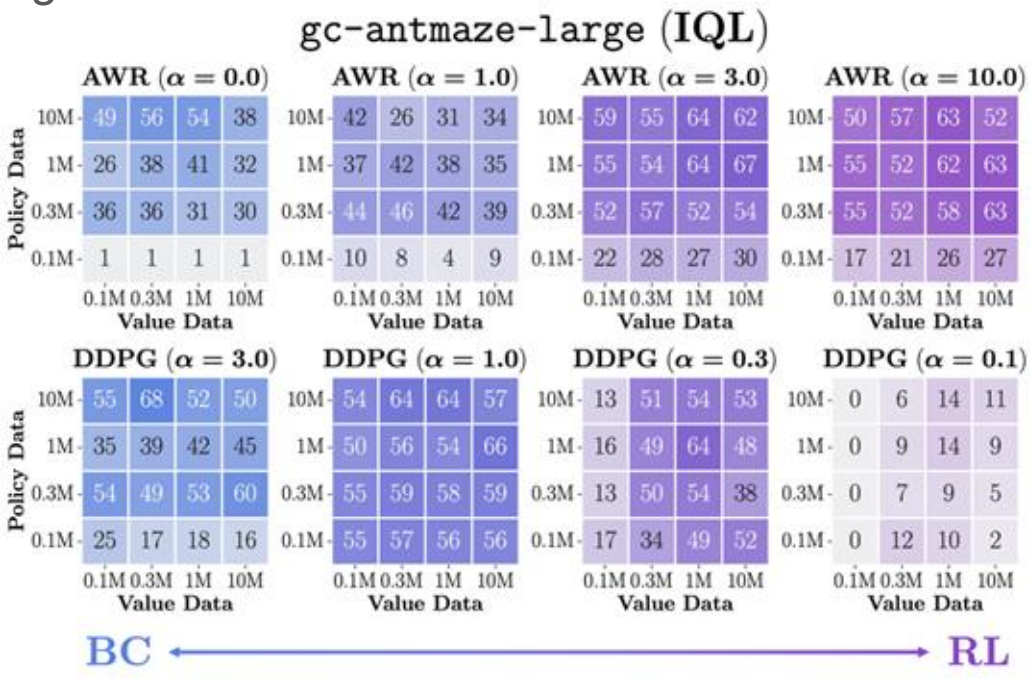
DDPG

Question: Is value learning really the bottleneck?

In the past, there is this belief that value learning is the main bottleneck for offline RL. And policy extraction has been less explored.
But we can be wrong…

Mode covering

Mode seeking



Q chunking: value learning for long horizon tasks

This work brings the idea of action chunking, which is developed for imitation learning to have more coherent actions, into Q learning framework.

$$Q(s_t, a_t) \leftarrow r_t + \gamma Q(s_{t+1}, a_{t+1}) \qquad \text{(standard 1-step TD)}$$

$$Q(s_t, a_t) \leftarrow \underbrace{\sum_{t'=t}^{t+h-1} \left[\gamma^{t'-t} r_{t'}\right]}_{\text{biased}} + \gamma^h Q(s_{t+h}, a_{t+h}), \qquad (n\text{-step return}, n = h)$$

Value: $$Q(s_t, \boldsymbol{a}_{t:t+h}) \leftarrow \underbrace{\sum_{t'=t}^{t+h-1} \left[\gamma^{t'-t} r_{t'}\right]}_{\text{unbiased}} + \gamma^h Q(s_{t+h}, \boldsymbol{a}_{t+h:t+2h}). \qquad \textbf{(Q-chunking)}$$

Policy $$L(\psi) = -\mathbb{E}_{s_t \sim \mathcal{D}, \boldsymbol{a}_{t:t+h} \sim \pi_\psi(\cdot|s_t)} \left[Q_\theta(s_t, \boldsymbol{a}_{t:t+h})\right], \text{s.t. } D(\pi_\psi(\boldsymbol{a}_{t:t+h}|s_t), \pi_\beta(\boldsymbol{a}_{t:t+h}|s_t)) \leq \varepsilon$$

Main takeaway message

For researcher in Offline RL:

- **Value learning**: suffer from long horizon problem. TD-style of objective is bias accumulative, which stops offline RL from scaling.
- **Policy learning:** What is the best way to *extract* a policy? How can policy *generalizes* well on test-time states?

For practitioners:

- **Choice of algorithm:** Try ReBRAC and IQL+DDPG first. They have been tested extensively.
- **Implementation:** Just like any other RL algorithms, performance is sensitive to implementation. Sometimes, good reward design is more important than algorithm choice.