Offline RL

TLDR

Why Offline RL?

1. **Suboptimal Data**: Learn from datasets containing both good and bad behaviors, useful in fields like robotics where expert demonstrations are impractical.
2. **Generalization**: Good behavior in one context suggests good behavior in another.
3. **Stitching**: Combine parts of good behaviors to form optimal paths (e.g., combine paths A→B and B→C to form A→C).

Distribution Shift:

- Off-policy methods like Q-learning fail due to Q-value divergence over time caused by distribution shift.
- Unlike supervised learning, RL samples are not i.i.d. and are chosen to maximize a function, leading to trouble.

Importance Sampling:

- Naive importance sampling fails due to high variance and unknown behavior policies.
- **Policy-Constrained Methods**:
    - BCQ: Uses a generative model and perturbation model to constrain actions to those likely from the behavior policy.
    - BEAR: Similar to BCQ, addresses uncertainty by restricting policy to certain areas.

Conservative Methods:

- Estimate lower bounds for Q-values in areas of high uncertainty.
- CQL: *Minimizes* Q-values on **unseen actions** while *maximizing* expected Q-values on **the dataset.**

Why Offline RL?

1. **Suboptimal data**: the "good stuff" in a dataset full of good and bad behaviors. It would be infeasible to provide expert demonstrations in fields like robotics. So we should be able to learn from suboptimal data instead.
2. Generalization: good behavior in one place may suggest good behavior in another place
3. "Stitching": parts of good behaviors can be recombined
   Example: you observe paths from A to B and B to C separately, but you can stitch together the corresponding halves to form the shortest path from A to C.

Distribution Shift

Applying off-policy methods (q-learning) won't work. Q-values diverge over time.
The problem is the distribution shift.
Recall true risk minimization: $x \sim p(x)$ and $y \sim p(y|x)$, and minimize the expected error.
Empirical risk minimization involves samples and in supervised learning we avoid significant distribution shift due to generalization of neural networks.
However, in (offline) RL settings, x are not sampled i.i.d and are actually chosen to **maximize** $f_\theta(x)$: $x^* \leftarrow argmax_x f_\theta(x)$. No matter how good is our neural net, we get into trouble.
Remember in our $Q-$ learning framework, we have to compute target values:

$$Q(s,a) \leftarrow r(s,a) + max_{a'} Q(s',a')$$

or writing it in terms of expectations:

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

We can expect good accuracy if we just use the reference policy. But, our goal is to find a better policy than our reference policy. Therefore:

$$\pi_{new} = argmax_\pi E_{a' \sim \pi(a|s)[Q(s,a)]}$$

Maximizing the value function (which is trained on a distribution) with a different distribution
Action distribution shift arises already during **training time** as inaccurate action values are used as bootstrapped targets.
The **maximizing** action is bootstrapped which drives the policy into out of distribution actions with large positive extrapolation error. The importance ration $\rho(s,a)$ becomes worse.
However, in online RL settings, these overestimated $Q-$ values are going to be corrected.
State distribution arises during **test time**.

Importance Sampling

Naive Importance Sampling doesn't work here. Define importance ratio and reweigh the actions in the policy evaluation objective. This doesn't fix the problems:

- high variance if importance ratio are large
- importance ratio is large if π and π_b differ strongly.
- in many cases we don't know π_β
  Two solutions:

Policy-constrained methods

Batch-constrained Q-learning (BCQ), similar to PPO/TRPO policy constraint (KL divergence thingy).
The idea is to run normal Q-learning, but in the maximization step, instead of considering the max over all possible actions, we want to only consider actions a' such that (s',a') actually

appeared in the batch of the data. Or, in more realistic cases, eliminate actions which are *unlikely* to be selected by the behavior policy π_b (the policy that generated the static data).

- Q-Learning:

$$\min_{\theta}(r + \gamma \max_{a' \in A(s')} Q_\theta(s', a') - Q_\theta(s, a))^2$$

- Let us define

$$A_\epsilon^{BCQ}(s) = \left\{ a \in A(s) \colon \frac{\hat{\pi}_\beta(a|s)}{\max_a \hat{\pi}_\beta(a|s)} \geq \epsilon \right\},$$

  where $\epsilon \in [0,1]$ is the threshold parameter and $\hat{\pi}_\beta$ an estimate for the behaviour policy

- The **constrained target** is $y = r + \gamma \cdot \max_{a' \in A_\epsilon^{BCQ}(s')} Q\,(s', a')$
  - $\epsilon = 1 \rightarrow$ behavioural cloning
  - $\epsilon = 0 \rightarrow$ Q-Learning

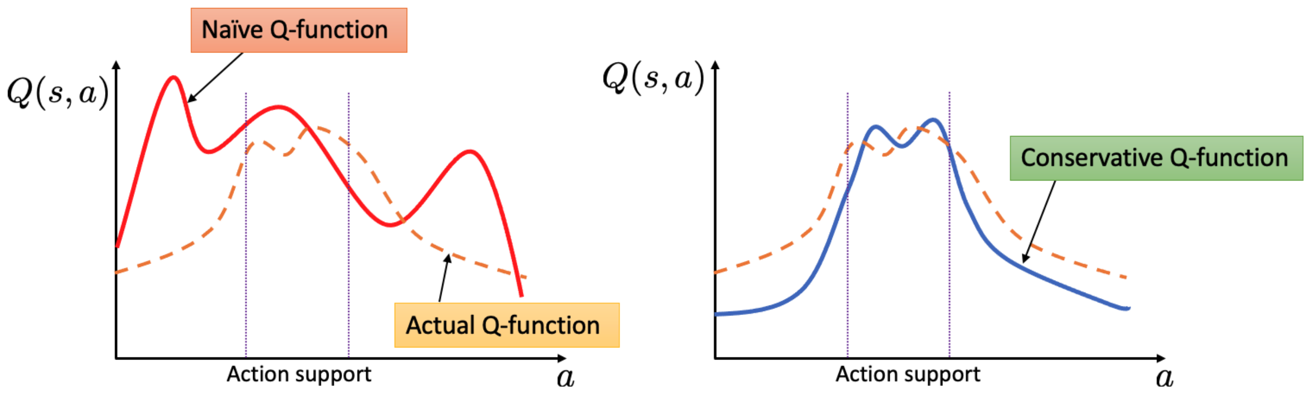
- The learned policy is $\pi(s) = \arg\max_{a \in A_\epsilon^{BCQ}(s')} Q(s, a)$

BCQ trains a generative model - conditional VAE - to generate actions that are likely to be from the batch, and **perturbation model** that further perturbs the action (to improve Q-value estimates in a neighborhood)
Perturbation parameter: $\Phi = 0 \rightarrow$ BC, $\Phi = 1 \rightarrow$ Q-learning. Trained with DDPG.
At **test time** rollouts, BCQ sample N actions via the generator, perturb each, and pick the action with the highest estimated Q-value.
Soft Clipped Double Q-learning fights overestimation in continuous action spaces.
Hyperparameter λ penalizes **uncertain** regions.
Implementation is similar to DQN. It can learn good policy even from imperfect data
BEAR is an extension. A_ϵ is different
BEAR and BCQ address uncertainty by restricting the policy to certain areas where we think we are certain.

Conservative Methods(create lower bound for Q-values in areas of high uncertainty)

Instead of action constraint, be conservative in the estimation of state-action values ($Q(s, a)$) that are not in the dataset.

A "safe" strategy when faced with distributional shift is to be **conservative**: assign a low value for the outcome of unseen states



To obtain such lower-bound on the actual Q-value function, CQL trains the Q-function using a sum of two objectives - standard TD error and a regularizer that minimizes Q-values on unseen actions with overestimated values while simultaneously maximizing the expected Q-value on the dataset:

$$\hat{Q}^{\pi}_{\text{CQL}} \leftarrow \arg\min_{Q} \max_{\mu(a|s)} \underbrace{\left(\mathbb{E}_{s\sim\text{data},a\sim\mu}[Q(s,a)] - \mathbb{E}_{s,a\sim\text{data}}[Q(s,a)]\right)}_{\text{CQL regularizer}} + \frac{1}{2\alpha} \underbrace{\mathbb{E}_{s,a,s'\sim\text{data}}\left[\left(r(s,a) + \gamma\mathbb{E}_{\pi}[\bar{Q}(s',a')] - Q(s,a)\right)^2\right]}_{\text{standard TD error}}$$

Conservative control:
Iterate between **policy improvement** and conservative **policy evaluation**.