Offline Reinforcement Learning: Representations, Algorithms, and Applications

Sergey Levine
UC Berkeley
If ultimately ML is always about making a decision, why don’t we treat every machine learning problem like a reinforcement learning problem?

Typical supervised learning problems have assumptions that make them “easy”:

- independent datapoints
- outputs don’t influence future inputs
- ground truth labels are provided at training-time

Decision-making problems often don’t satisfy these assumptions:

- current actions influence future observations
- goal is to maximize some utility (reward)
- optimal actions are not provided

Example: decisions made by a traffic prediction system might affect the route that people take, which changes traffic

These are not just issues for control: in many cases, real-world deployment of ML has these same feedback issues
So why aren’t we all using RL?

Reinforcement learning is two different things:

1. Framework for learning-based **decision making**

   - $f_\theta(s) = y$ [object label]
   - $\pi_\theta(s) = a$ [decision]

   almost all real-world learning problems look like this

2. Active, online learning algorithms for control

   - almost all real-world learning problems make it very difficult to do this

   this is done many times
Making RL look more like supervised learning

on-policy RL

offline reinforcement learning

\[ \pi_\theta(s) = a \]
Offline RL challenges & methods

Workflows for offline RL

Offline RL and representations

Offline RL without *explicit* pessimism?
Offline RL challenges & methods

Workflows for offline RL

Offline RL and representations

Offline RL without explicit pessimism?
Off-policy RL

This talk focuses entirely on approximate dynamic programming methods, but there are other methods too!

RL objective: \( \max_{\pi} \sum_{t=1}^{T} E_{s_t, a_t \sim \pi}[r(s_t, a_t)] \)

Q-function: \( Q^\pi(s_t, a_t) = \sum_{t'=t}^{T} E_{s_{t'}, a_{t'} \sim \pi}[r(s_{t'}, a_{t'})|s_t, a_t] \)

\( \pi(a|s) = 1 \) if \( a = \arg \max_a Q^\pi(s, a) \)

\( Q^*(s, a) = r(s, a) + \max_{a'} Q^*(s', a') \)

enforce this equation at all states!

minimize \( \sum_i (Q(s_i, a_i) - [r(s_i, a_i) + \max_{a'_i} Q(s'_i, a'_i)])^2 \)

minimize \( \sum_i (Q(s_i, a_i) - y_i)^2 \)
Why offline RL suffers from distributional shift

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

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

\[
y(s, a)
\]

what is the objective?

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

behavior policy

target value

expect good accuracy when \(\pi_{\beta}(a|s) = \pi_{\text{new}}(a|s)\)

even worse: \(\pi_{\text{new}} = \arg \max_{\pi} \mathbb{E}_{a \sim \pi(a|s)}[Q(s, a)]\)

how often does that happen?

Kumar, Fu, Tucker, Levine. Stabilizing Off-Policy Q-Learning via Bootstrapping Error Reduction. NeurIPS ‘19
Training the Q-function to avoid OOD errors

There are many other ways to address OOD actions, but this is the one I’m going to focus on (mostly)

how well it does how well it thinks it does (Q-values)

\[
\hat{Q}^\pi = \arg \min_Q \max_{\mu} \alpha E_{s \sim D, a \sim \mu(s)} [Q(s, a)] \\
\text{regular objective} \quad \left\{ + E_{(s, a, s') \sim D} \left[ (Q(s, a) - (r(s, a) + E_\pi[Q(s', a')])^2 \right] \right. \\
\text{term to push down big Q-values}
\]

can show that \( \hat{Q}^\pi \leq Q^\pi \) for large enough \( \alpha \)

true Q-function
Learning with Q-function lower bounds

Conservative Q-learning (CQL)

A better bound: always pushes Q-values down

\[
\hat{Q}^\pi = \arg \min_Q \max_{\mu} \alpha E_{s \sim D, a \sim \mu(a|s)}[Q(s, a)] - \alpha E_{(s,a) \sim D}[Q(s, a)] \\
+ E_{(s,a,s') \sim D} \left[ (Q(s, a) - (r(s, a) + E_{\pi}[Q(s', a')]))^2 \right]
\]

no longer guaranteed that \( \hat{Q}^\pi(s, a) \leq Q^\pi(s, a) \) for all \((s, a)\)

but guaranteed that \( E_{\pi(a|s)}[\hat{Q}^\pi(s, a)] \leq E_{\pi(a|s)}[Q^\pi(s, a)] \) for all \(s \in D\)

Kumar, Zhou, Tucker, Levine. *Conservative Q-Learning for Offline Reinforcement Learning.* ‘20
Offline RL challenges & methods

Workflows for offline RL

Offline RL and representations

Offline RL without explicit pessimism?
The hyperparameter problem

Standard formulation:
off-policy evaluation + model selection
+ very widely studied
- introduces its own hyperparameters
- generally a very hard problem

Supervised learning: train/val split
Offline RL: ???

Key observation: to tune hyperparameters, we don’t need to evaluate any policy, only the policies produced by our specific offline RL method!

Can we leverage properties of a specific offline RL method (e.g., CQL) to develop a workflow that allows selecting hyperparameters without off-policy evaluation?
“Overfitting” vs. “underfitting”

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Supervised Learning</th>
<th>Conservative Offline RL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test error</td>
<td>Loss $\mathcal{L}$ evaluated on test data, $\mathcal{D}_{test}$</td>
<td>Performance of policy, $J(\pi)$</td>
</tr>
<tr>
<td>Train error</td>
<td>Loss $\mathcal{L}$ evaluated on train data, $\mathcal{D}_{train}$</td>
<td>Objective in Equations 2, 1</td>
</tr>
</tbody>
</table>

Overfitting:
- $\mathcal{L}(\mathcal{D}_{train})$ low, $\mathcal{L}(\mathcal{D}_{val})$ high, $\mathcal{D}_{val}$ is a validation set drawn i.i.d. as $\mathcal{D}_{train}$
- High value of train error $\mathcal{L}(\mathcal{D}_{train})$

Underfitting:
- Low value of train error $\mathcal{L}(\mathcal{D}_{train})$

---

$$\min_\theta \alpha \left( \mathbb{E}_{s \sim \mathcal{D}, a \sim \pi(\cdot|s)} [Q_\theta(s, a)] - \mathbb{E}_{s \sim \mathcal{D}} [Q_\theta(s, a)] \right) + \frac{1}{2} \mathbb{E}_{s,a,s',r \sim \mathcal{D}} \left( (Q_\theta(s, a) - B^\pi Q(s, a))^2 \right)$$

(conservative Q-learning)

$$\pi^* := \arg \max_\pi J_\mathcal{D}(\pi) - \alpha D(\pi, \pi_\beta)$$

(abstract model of a conservative offline RL method)

---

Handling “Overfitting”

Why?

If overfitting, these become very low

\[
\hat{Q}^\pi = \arg \min_Q \max_\mu \alpha E_{s \sim D, a \sim \mu(a|s)} [Q(s, a)] - \alpha E_{(s, a) \sim D} [Q(s, a)] \\
+ E_{(s, a, s') \sim D} [(Q(s, a) - (r(s, a) + E_\pi [Q(s', a')]))^2]
\]

Therefore this becomes very low! So this becomes very low!

So we get low \( E_{(s, a) \sim D} [Q(s, a)] \).

If dataset Q-values drop, that means we have too much capacity!

We can fix this by reducing capacity or increasing regularization.

Handling “Overfitting”

If dataset Q-values drop, that means we have too much capacity!

We can fix this by reducing capacity or increasing regularization.

Handling “Underfitting”

**Underfitting**
- Policy return
- TD error/ CQL regularizer
- Optimal return

**Why?**

\[
\hat{Q}^\pi = \arg \min_Q \max_{\mu} \alpha E_{s \sim D, a \sim \mu(a|s)} [Q(s, a)] - \alpha E_{(s,a) \sim D} [Q(s, a)] + E_{(s,a,s') \sim D} \left[ (Q(s, a) - (r(s, a) + E^\pi_{s', a'}[Q(s', a')]))^2 \right]
\]

if underfitting, this is too big (so we get overestimation)

or this is too big

**Metric 4.2 (Underfitting).** Compute the values of the training TD error, \(L_{TD}(\theta)\) and CQL regularizer, \(R(\theta)\) for the current run and another identical run with increased model capacity. If the training errors reduce with increasing model capacity, the original run was underfitting.

**Figure 8:** Average Q-value and TD error on Sawyer tasks as model capacity increases. Q-values increase over training with lower capacity ruling out overfitting and increasing model capacity leads to a reduction in TD error indicating the presence of underfitting.

Does it work?

<table>
<thead>
<tr>
<th>Method</th>
<th>Epoch 50</th>
<th>Epoch 75</th>
<th>Epoch 100</th>
<th>Epoch 200</th>
</tr>
</thead>
<tbody>
<tr>
<td>CQL</td>
<td>7/9</td>
<td>4/9</td>
<td>4/9</td>
<td>2/9</td>
</tr>
<tr>
<td>CQL + VIB</td>
<td>3/9</td>
<td>8/9</td>
<td>7/9</td>
<td>7/9</td>
</tr>
</tbody>
</table>

Questions, open problems, opportunities

➢ We have a “workflow” that allows tuning (some) hyperparameters, but doesn’t require OPE
➢ It appears to work in practice, because we can get our robots to work
➢ It’s easier than OPE, because it leverages properties of the corresponding algorithm
➢ It’s rather heuristic
➢ It’s not guaranteed to work every time
➢ Can we devise more formally justified, general, and effective workflows?
Offline RL challenges & methods

Workflows for offline RL

Offline RL and representations

Offline RL without *explicit* pessimism?
The fact that it’s a neural network matters

**Conclusion:** if we back up **out-of-sample** actions (even if they are not out of distribution!) we get this strange “high dot product” (feature alignment) problem.

The longer we train, the worse it gets.

That’s a **big problem**, because with deep learning, we want to **train for a long time with lots of data!**

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

*Q(s',a')* where \((s',a') \in D\)

*Q(s',a')* where \(s' \in D, a' \sim \pi_{\beta}(a'|s')\)

“SARSA”

“TD”

Kumar, Agrawal, Tucker, Ma, Levine. *DR3: Value-Based Deep Reinforcement Learning Requires Explicit Regularization*. ‘21
What’s going on?

Implicit regularization:

\[ \theta_{k+1} = \theta_k - \eta \nabla_\theta L(\theta) + \eta \varepsilon_k, \quad \varepsilon_k \sim \mathcal{N}(0, M) \]

Implicit regularization in reinforcement learning:

**Main result:** if we follow the TD pseudo-gradient

\[ \theta_{k+1} = \theta_k - \eta \left( \sum_{i} \nabla_\theta Q(s_i, a_i) \left( Q_\theta(s_i, a_i) - (r_i + \gamma Q_\theta(s'_i, a'_i)) \right) \right) + \eta \varepsilon_k, \quad \varepsilon_k \sim \mathcal{N}(0, M) \]

\[ R_{\text{TD}}(\theta) = \eta \sum_{i=1}^{\left| \mathcal{D} \right|} \nabla_\theta Q(s_i, a_i)^\top \Sigma_M \nabla_\theta Q(s_i, a_i) - \eta \gamma \sum_{i=1}^{\left| \mathcal{D} \right|} \text{trace} \left( \Sigma_M \nabla_\theta Q(s_i, a_i) \left[ [\nabla_\theta Q(s'_i, a'_i)^\top] \right] \right) \]

- make gradient inner products small (good)
- make gradient inner products big (uh oh!)

balances out if \((s', a') \in \mathcal{D}\)

runaway maximization if \((s', a') \notin \mathcal{D}\)

Blanc et al. (2020); Damian et al. (2021) if labels corrupted with \(\mathcal{N}(0, 1)\) noise

\[ M = \sum_{i=1}^{\left| \mathcal{D} \right|} \nabla_\theta f_\theta(x_i) \nabla_\theta f_\theta(x_i)^\top \]

\[ R(\theta) = \eta \sum_{i}^{\left| \mathcal{D} \right|} \| \nabla_\theta f_\theta(x_i) \|^2 \]

when overparameterized, solution is stable only if

\[ \nabla_\theta R(\theta^*) = 0 \]

this is a good thing!
Can we correct this problem?

\[
R_{TD}(\theta) = \eta \sum_{i=1}^{|D|} \nabla Q_{\theta}(s_i, a_i)^{\top} \Sigma_M^{*} \nabla Q_{\theta}(s_i, a_i) - \eta \gamma \sum_{i=1}^{|D|} \text{trace} \left( \Sigma_M^{*} \nabla Q_{\theta}(s_i, a_i) \left[ [\nabla Q_{\theta}(s'_i, a'_i)^{\top}] \right] \right)
\]

what if we add explicit regularization to balance out the second term?

should be something like \( E_D[\nabla Q_{\theta}(s_i, a_i) \cdot \nabla Q_{\theta}(s'_i, a'_i)] \)

works, but expensive

\[
\begin{align*}
\text{s} &\rightarrow \phi(s, a) \cdot w = Q(s, a) \\
\text{a} &\rightarrow \\
\end{align*}
\]

simple hack: at last layer, \( \nabla_w Q_{\theta}(s, a) = \phi(s, a) \)

\[
E_D[\nabla_w Q_{\theta}(s_i, a_i) \cdot \nabla_w Q_{\theta}(s'_i, a'_i)] = E_D[\phi(s_i, a_i) \cdot \phi(s'_i, a'_i)]
\]

cheap & easy

Kumar, Agrawal, Tucker, Ma, Levine. **DR3: Value-Based Deep Reinforcement Learning Requires Explicit Regularization.** ‘21
Figure 3: Performance of DR3 + COG on two manipulation tasks using only 5% and 25% of the data used by Singh et al. (2020) to make these more challenging. COG + DR3 outperforms COG in training and attains higher average and final performance.

Table 1: IQM normalized average performance (training stability) across 17 games, with 95% CIs in parenthesis, after 6.5M gradient steps for the 1% setting and 12.5M gradient steps for the 5%, 10% settings. Individual performances reported in Tables F.4-F.10. DR3 improves the stability over both CQL and REM.

<table>
<thead>
<tr>
<th></th>
<th>CQL</th>
<th>CQL + DR3</th>
<th>REM</th>
<th>REM + DR3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1%</td>
<td>43.7 (39.6, 48.6)</td>
<td>56.9 (52.5, 61.2)</td>
<td>4.0 (3.3, 4.8)</td>
<td>16.5 (14.5, 18.6)</td>
</tr>
<tr>
<td>5%</td>
<td>78.1 (74.5, 82.4)</td>
<td>105.7 (101.9, 110.9)</td>
<td>25.9 (23.4, 28.8)</td>
<td>60.2 (55.8, 65.1)</td>
</tr>
<tr>
<td>10%</td>
<td>59.3 (56.4, 61.9)</td>
<td>65.8 (63.3, 68.3)</td>
<td>53.3 (51.4, 55.3)</td>
<td>73.8 (69.3, 78)</td>
</tr>
</tbody>
</table>

Figure 4: Normalized performance across 17 Atari games for REM + DR3 (top), CQL + DR3 (bottom). x-axis represents gradient steps; no new data is collected. While naïve REM suffers from a degradation in performance with more training, REM + DR3 not only remains generally stable with more training, but also attains higher final performance. CQL + DR3 attains higher performance than CQL. We report IQM with 95% stratified bootstrap CIs (Agarwal et al., 2021).
Conclusions & takeaways

- Offline RL with deep networks (i.e., with representation learning) is fundamentally different from “shallow” RL
- It’s also fundamentally different from supervised learning!
- The “usual tricks” that work so well in supervised learning might not lead to great performance in RL directly
- Analyzing the effect of RL training on representations in deep nets is important!
Offline RL challenges & methods

Workflows for offline RL

Offline RL and representations

Offline RL without *explicit* pessimism?
Can we just avoid all OOD actions in the Q update?

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

\[
V(s') \leftarrow \text{just another neural network}
\]

\[
V \leftarrow \arg \min_{V} \frac{1}{N} \sum_{i=1}^{N} \ell(V(s_i), Q(s_i, a_i))
\]

e.g., MSE loss \((V(s_i) - Q(s_i, a_i))^2\) this action comes from \(\pi_{\beta}\)

\[
\ell_2^\tau(x) = \begin{cases} 
(1 - \tau)x^2 & \text{if } x > 0 \\
\tau x^2 & \text{else}
\end{cases}
\]

\[
\text{expectile: } \ell_2^\tau(x)
\]

\[
V(s) \leftarrow \max_{a \in \Omega(s)} Q(s, a)
\]

\[
\Omega(s) = \{ a : \pi_{\beta}(a|s) \geq \epsilon \}
\]

if we use \(\ell_2^\tau\) for big \(\tau\)

Kostrikov, Nair, Levine. Offline Reinforcement Learning with Implicit Q-Learning. ‘21
Implicit Q-learning (IQL)

Q-learning with *implicit* policy improvement

\[
Q(s, a) \leftarrow r(s, a) + V(s') \\
V \leftarrow \arg \min_{V} \frac{1}{N} \sum_{i=1}^{N} \ell_2^\tau(V(s_i), Q(s_i, a_i))
\]

\[
V(s) \leftarrow \max_{a \in \Omega(s)} Q(s, a)
\]

\[
\Omega(s) = \{a : \pi_\beta(a|s) \geq \epsilon\}
\]

if we use $\ell_2^\tau$ for big $\tau$

\[
Q(s, a) \leftarrow r(s, a) + \max_{a' \in \Omega(s')} Q(s', a')
\]

“implicit” policy

\[
\pi_{\text{new}}(a|s) = \delta(a = \arg \max_{a \in \Omega(s)} Q(s, a))
\]

Now we can do value function updates without ever risking out-of-distribution actions!
## Results

Kostrikov, Nair, Levine. *Offline Reinforcement Learning with Implicit Q-Learning*. ‘21

<table>
<thead>
<tr>
<th>Dataset</th>
<th>BC</th>
<th>10%BC</th>
<th>DT</th>
<th>AWAC</th>
<th>OneStep RL</th>
<th>TD3+BC</th>
<th>COL</th>
<th>IQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>halfcheetah-medium-v2</td>
<td>42.6</td>
<td>42.5</td>
<td>42.6</td>
<td>43.5</td>
<td>48.4</td>
<td>48.3</td>
<td>44.0</td>
<td>47.4</td>
</tr>
<tr>
<td>hopper-medium-v2</td>
<td>52.9</td>
<td>56.9</td>
<td>67.6</td>
<td>57.0</td>
<td>59.6</td>
<td>59.3</td>
<td>58.5</td>
<td>66.3</td>
</tr>
<tr>
<td>walker2d-medium-v2</td>
<td>75.3</td>
<td>70.0</td>
<td>74.0</td>
<td>72.4</td>
<td>81.8</td>
<td>83.7</td>
<td>72.5</td>
<td>78.3</td>
</tr>
<tr>
<td>halfcheetah-medium-replay-v2</td>
<td>36.6</td>
<td>40.6</td>
<td>36.6</td>
<td>40.5</td>
<td>38.1</td>
<td>44.6</td>
<td>45.5</td>
<td>44.2</td>
</tr>
<tr>
<td>hopper-medium-replay-v2</td>
<td>18.1</td>
<td>75.9</td>
<td>82.7</td>
<td>73.2</td>
<td>95.7</td>
<td>69.0</td>
<td>95.0</td>
<td>94.7</td>
</tr>
<tr>
<td>walker2d-medium-replay-v2</td>
<td>26.0</td>
<td>62.5</td>
<td>66.0</td>
<td>37.2</td>
<td>49.5</td>
<td>81.8</td>
<td>77.2</td>
<td>73.9</td>
</tr>
<tr>
<td>halfcheetah-medium-expert-v2</td>
<td>55.2</td>
<td>92.9</td>
<td>86.8</td>
<td>42.8</td>
<td>93.4</td>
<td>90.7</td>
<td>91.6</td>
<td>86.7</td>
</tr>
<tr>
<td>hopper-medium-expert-v2</td>
<td>52.5</td>
<td>110.9</td>
<td>107.6</td>
<td>55.8</td>
<td>103.3</td>
<td>98.0</td>
<td>105.4</td>
<td>91.5</td>
</tr>
<tr>
<td>walker2d-medium-expert-v2</td>
<td>107.5</td>
<td>109</td>
<td>108.1</td>
<td>74.5</td>
<td>113</td>
<td>110.1</td>
<td>108.8</td>
<td>109.6</td>
</tr>
<tr>
<td>locomotion-v2 total</td>
<td>466.7</td>
<td>666.2</td>
<td>672.6</td>
<td>450.7</td>
<td>684.6</td>
<td>677.4</td>
<td>698.5</td>
<td>692.4</td>
</tr>
<tr>
<td>antmaze-umaze-v0</td>
<td>54.6</td>
<td>62.8</td>
<td>59.2</td>
<td>56.7</td>
<td>64.3</td>
<td>78.6</td>
<td>74.0</td>
<td>87.5</td>
</tr>
<tr>
<td>antmaze-umaze-diverse-v0</td>
<td>45.6</td>
<td>50.2</td>
<td>53.0</td>
<td>49.3</td>
<td>60.7</td>
<td>71.4</td>
<td>84.0</td>
<td>62.2</td>
</tr>
<tr>
<td>antmaze-medium-play-v0</td>
<td>0.0</td>
<td>5.4</td>
<td>0.0</td>
<td>0.0</td>
<td>0.3</td>
<td>10.6</td>
<td>61.2</td>
<td>71.2</td>
</tr>
<tr>
<td>antmaze-medium-diverse-v0</td>
<td>0.0</td>
<td>9.8</td>
<td>0.0</td>
<td>0.7</td>
<td>0.0</td>
<td>3.0</td>
<td>53.7</td>
<td>70.0</td>
</tr>
<tr>
<td>antmaze-large-play-v0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.2</td>
<td>15.8</td>
<td>39.6</td>
<td>47.5</td>
</tr>
<tr>
<td>antmaze-large-diverse-v0</td>
<td>0.0</td>
<td>6.0</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>14.9</td>
<td>47.5</td>
<td>47.5</td>
</tr>
<tr>
<td>antmaze-v0 total</td>
<td>100.2</td>
<td>134.2</td>
<td>112.2</td>
<td>107.7</td>
<td>125.3</td>
<td>163.8</td>
<td>303.6</td>
<td>378.0</td>
</tr>
<tr>
<td>total</td>
<td>566.9</td>
<td>800.4</td>
<td>784.8</td>
<td>558.4</td>
<td>809.9</td>
<td>841.2</td>
<td>1002.1</td>
<td>1070.4</td>
</tr>
<tr>
<td>kitchen-v0 total</td>
<td>154.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>144.6</td>
<td>159.8</td>
<td></td>
</tr>
<tr>
<td>adroit-v0 total</td>
<td>104.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>93.6</td>
<td>118.1</td>
<td></td>
</tr>
<tr>
<td>total+kitchen+adroit</td>
<td>825.9</td>
<td>-</td>
<td>960m</td>
<td>20m</td>
<td>20m ≈ 20m</td>
<td>1240.3</td>
<td>1348.3</td>
<td></td>
</tr>
<tr>
<td>runtime</td>
<td>10m</td>
<td>10m</td>
<td>960m</td>
<td>20m</td>
<td>20m</td>
<td>80m</td>
<td>20m</td>
<td></td>
</tr>
</tbody>
</table>

Chen et al. Decision Transformers

behavioral cloning

best trajectories

behavioral cloning

recent (2021) offline RL methods

most methods get similar results to good BC implementations

significant improvement from methods that properly handle compositionality
**Finetuning Comparisons**

finetunes well, but low offline performance hampers final results
great offline performance, too conservative for finetuning
generally best for finetuning

<table>
<thead>
<tr>
<th>Dataset</th>
<th>AWAC</th>
<th>CQL</th>
<th>IQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>antmaze-umaze-v0</td>
<td>56.7</td>
<td>70.1</td>
<td>86.7</td>
</tr>
<tr>
<td>antmaze-umaze-diverse-v0</td>
<td>49.3</td>
<td>31.1</td>
<td>75.0</td>
</tr>
<tr>
<td>antmaze-medium-play-v0</td>
<td>0.0</td>
<td>23.0</td>
<td>68.3</td>
</tr>
<tr>
<td>antmaze-medium-diverse-v0</td>
<td>0.7</td>
<td>23.0</td>
<td>25.5</td>
</tr>
<tr>
<td>antmaze-large-play-v0</td>
<td>0.0</td>
<td>1.0</td>
<td>42.6</td>
</tr>
<tr>
<td>antmaze-large-diverse-v0</td>
<td>1.0</td>
<td>1.0</td>
<td>60.7</td>
</tr>
<tr>
<td>antmaze-v0 total</td>
<td>107.7</td>
<td>151.5</td>
<td>370.1</td>
</tr>
<tr>
<td>pen-binary-v0</td>
<td>44.6</td>
<td>31.2</td>
<td>37.4</td>
</tr>
<tr>
<td>door-binary-v0</td>
<td>1.3</td>
<td>0.2</td>
<td>0.7</td>
</tr>
<tr>
<td>relocate-binary-v0</td>
<td>0.8</td>
<td>0.1</td>
<td>0.0</td>
</tr>
<tr>
<td>hand-v0 total</td>
<td>46.7</td>
<td>31.5</td>
<td>38.1</td>
</tr>
<tr>
<td>total</td>
<td>154.4</td>
<td>211.4</td>
<td>408.2</td>
</tr>
</tbody>
</table>

**Option 1:** avoid ever evaluating actions that are not in the dataset
**Option 2:** train the Q-function so that OOD actions never have high values

- CQL has fewer hyperparameters, cleaner workflows with offline tuning
- CQL has better theoretical guarantees
- IQL performance is slightly better
- IQL finetuning is much better
- We still don’t know which principles are going to be more effective in the long run

---

Kostrikov, Nair, Levine. *Offline Reinforcement Learning with Implicit Q-Learning*. ‘21
Offline RL challenges & methods

Workflows for offline RL

Offline RL and representations

Offline RL without *explicit* pessimism?
RAIL
Robotic AI & Learning Lab

website: http://rail.eecs.berkeley.edu
source code: http://rail.eecs.berkeley.edu/code.html