Offline Reinforcement Learning

Sergey Levine
UC Berkeley
What makes modern machine learning work?
What about reinforcement learning?

Mnih et al. ‘13

Schulman et al. ‘14 & ‘15

Levine*, Finn*, et al. ‘16

this is done many times

enormous gulf
Can we develop **data-driven** RL methods?

Why is offline RL difficult?

How do we design offline RL algorithms?

Conservative Q-Learning

Model-based offline RL

How do we evaluate offline RL methods?
Why is offline RL difficult?

How do we design offline RL algorithms?

Conservative Q-Learning

Model-based offline RL

How do we evaluate offline RL methods?
On-policy, off-policy, and offline RL

"Classic" RL diagram:

This is a very **online** view of RL

More typical use case:

1. Offline Learning
   - off-policy data
   - expert demos
   - prior runs of RL
   - \( D = \{(s, a, s', r)\}_t \)
   - Update \( \pi_\theta, Q_\phi \)

2. Online Fine-tuning
   - \( p(s'|s, a) \)
   - \( (s, a, s', r) \)
   - Update \( \pi_\theta, Q_\phi \)

**on-policy RL**

- rollout data \( \{(s_i, a_i, s'_i, r_i)\}_t \)
- \( s, \pi_k \rightarrow a \rightarrow \pi_k + 1 \)
- rollout(s)
- data collected once with **any** policy

**off-policy RL**

- rollout data \( \{(s_i, a_i, s'_i, r_i)\}_t \)
- \( s, \pi_k \rightarrow a \rightarrow \pi_k + 1 \)
- rollout(s)
- training phase

**offline reinforcement learning**

- \( s, \pi \rightarrow a \rightarrow \pi \)
- data collected once with **any** policy
- deployment
The RL objective is very hard to optimize this with off-policy data directly.

\[
p_\theta(s_1, a_1, \ldots, s_T, a_T) = p(s_1) \prod_{t=1}^{T} \pi_\theta(a_t|s_t)p(s_{t+1}|s_t, a_t)
\]

\[
\theta^* = \arg \max_{\theta} E_{\tau \sim p_\theta(\tau)} \left[ \sum_t \gamma^t r(s_t, a_t) \right]
\]
The RL objective

\[
E_{\tau \sim p_\theta(\tau)} \left[ \sum_t \gamma^t r(s_t, a_t) \right] \approx \frac{1}{N} \sum_{i=1}^N E_{a \sim \pi_\theta(a|s_i)} \left[ Q^\pi(s_i, a) \right]
\]

sum over all states in the dataset

\[
\sum_{t'=t}^{\infty} \gamma^{t'-t} r(s_{t'}, a_{t'})
\]

if we just knew this, all would be easy
so let’s learn it!

**Aside:** recovering the policy

could optimize the above objective \(w.r.t.\ \pi_\theta\) directly

\[
\pi(a_t|s_t) = \begin{cases} 
1 & \text{if } a_t = \arg\max_{a_t} Q^\pi(s_t, a_t) \\
0 & \text{otherwise}
\end{cases}
\]

“greedy” policy

can recover with optimization (e.g., CEM)
The Q-function

\[ Q^\pi(s_t, a_t) = E\left[ \sum_{t'=t}^{\infty} \gamma^{t'-t} r(s_{t'}, a_{t'}) \right] = r(s_t, a_t) + \gamma E[r(s_{t+1}, a_{t+1})] + \gamma E[r(s_{t+1}, a_{t+2})] \ldots \]

Bellman equation:

\[ Q^\pi(s_t, a_t) = r(s_t, a_t) + \gamma E[Q^\pi(s_{t+1}, a_{t+1})] \]

let's say we have a trajectory \( s_1, a_1, s_2, a_2, \ldots, s_T, a_T \) generated by some other policy \( \pi_\beta \)
can we estimate the Bellman equation?

these come from our trajectory this is sampled from \( \pi_\theta \)

\[ Q^\pi(s_t, a_t) \approx r(s_t, a_t) + \gamma Q^\pi(s_{t+1}, a_{t+1}) \]

this is a single sample estimate of the expectation
The Q-function

\[ Q^\pi(s_t, a_t) \approx r(s_t, a_t) + \gamma Q^\pi(s_{t+1}, a_{t+1}) \]

In reality, we use a minibatch, not just one transition!

Rough sketch:

1. Load \( s_t, a_t, s_{t+1} \) from buffer
2. Get \( a_{t+1} \sim \pi_\theta(a_{t+1}|s_{t+1}) \)
3. Compute target value \( y = r(s_t, a_t) + \gamma Q^\pi(s_{t+1}, a_{t+1}) \)
4. Take gradient step on \( \mathcal{E} = (Q^\pi(s_t, a_t) - y)^2 \)

\[ \pi(a_t|s_t) = \begin{cases} 1 & \text{if } a_t = \arg \max_{a_t} Q^\pi(s_t, a_t) \\ 0 & \text{otherwise} \end{cases} \]
Off-policy RL summary

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

---

don’t need on-policy data for this!

off-policy Q-learning:

1. collect dataset \( \{(s_i, a_i, s'_i, r_i)\} \) using some policy, add it to \( \mathcal{B} \)

2. sample a batch \( (s_i, a_i, s'_i, r_i) \) from \( \mathcal{B} \)

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

---

more typical use case:

See, e.g.
Riedmiller, Neural Fitted Q-Iteration ‘05
Ernst et al., Tree-Based Batch Mode RL ‘05

---

\( \pi(a|s) \) (with exploration)
An instantiation of this idea...

Kalashnikov, Irpan, Pastor, Ibarz, Herzong, Jang, Quillen, Holly, Kalakrishnan, Vanhoucke, Levine. QT-Opt: Scalable Deep Reinforcement Learning of Vision-Based Robotic Manipulation Skills
Does it work?

<table>
<thead>
<tr>
<th>Method</th>
<th>Dataset</th>
<th>Success</th>
<th>Failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Offline QT-Opt</td>
<td>580k offline</td>
<td>87%</td>
<td>13%</td>
</tr>
<tr>
<td>Finetuned QT-Opt</td>
<td>580k offline + 28k online</td>
<td>96%</td>
<td>4%</td>
</tr>
</tbody>
</table>

Even worse...

**Figure 2:** Baseline comparison for the example task. Neither QT-Opt [3], nor AWAC [2] can solve the task.

Training on random(ized) offline data different from training data, but somewhat in-distribution.

Training on demo data.
What’s the problem?

Hypothesis 1: Overfitting

Hypothesis 2: Training data is not good

Usually not the case: behavioral cloning of best data does better!
Distribution shift in a nutshell

Example empirical risk minimization (ERM) problem:

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

given some $x^*$, is $f_\theta(x^*)$ correct?

$$E_{x \sim p(x), y \sim p(y|x)} \left[ (f_\theta(x) - y)^2 \right] \text{ is low}$$

$$E_{x \sim \tilde{p}(x), y \sim p(y|x)} \left[ (f_\theta(x) - y)^2 \right] \text{ is not, for general } \tilde{p}(x) \neq p(x)$$

what if $x^* \sim p(x)$? not necessarily...

usually we are not worried – neural nets generalize well!

what if we pick $x^* \leftarrow \arg \max_x f_\theta(x)$?
Where do we suffer from distribution shift?

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

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

\[ y(s, a) \]

what is the objective?

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

behavior policy
target value

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

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

(what if we pick \( x^* \leftarrow \arg \max_{x} f_{\theta}(x) \)?)

Kumar, Fu, Tucker, Levine. Stabilizing Off-Policy Q-Learning via Bootstrapping Error Reduction. NeurIPS ‘19
Why is offline RL difficult?

How do we design offline RL algorithms?

Conservative Q-Learning

Model-based offline RL

How do we evaluate offline RL methods?
How do prior methods address this?

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

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

This solves distribution shift, right?

No more erroneous values?

**Issue 1:** Estimating the behavior policy is difficult

**Issue 2:** This might be too conservative
   (we’ll come back to this)

“policy constraint” method

**very** old idea (but it had no single name?)

Todorov et al. [passive dynamics in linearly-solvable MDPs]

Kappen et al. [KL-divergence control, etc.]

trust regions, covariant policy gradients,
natural policy gradients, etc.

used in some form in recent papers:

Fox et al. ‘15 (“Taming the Noise…”)

Fujimoto et al. ‘18 (“Off Policy…”)

Jaques et al. ‘19 (“Way Off Policy…”)

Kumar et al. ‘19 (“Stabilizing…”)

Wu et al. ‘19 (“Behavior Regularized…”)

When is estimating the behavior policy hard?

**Issue 1:** Estimating the behavior policy is difficult

- **Easy case:** all data comes from the same Markovian policy
  - This is not very common or realistic
- **Hard case:** data comes from many different policies
  - Very common in reality (e.g., some demo data from humans, some scripted data)
  - Very common during online finetuning
Avoiding behavior policies with implicit constraints

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

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

straightforward to show via duality

approximate via weighted max likelihood!

$$\pi_{\text{new}}(a|s) = \arg\max_{\pi} E_{(s,a) \sim \pi_\beta} \left[ \log \pi(a|s) \frac{1}{Z(s)} \exp\left( \frac{1}{\lambda} A^{\pi_{\text{old}}}(s,a) \right) \right]$$

See also:
Peters et al. (REPS)
Rawlik et al. (“psi-learning”)
...many follow-ups

Peng*, Kumar*, Levine. **Advantage-Weighted Regression.** ‘19

Nair, Dalal, Gupta, Levine. **Accelerating Online Reinforcement Learning with Offline Datasets.** ‘20
Why is offline RL difficult?

How do we design offline RL algorithms?

Conservative Q-Learning

Model-based offline RL

How do we evaluate offline RL methods?
What about those Q-value errors?

how well it does

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

$$\hat Q^\pi = \arg \min_Q \max_{\mu} \alpha E_{s \sim D, a \sim \mu(s)} [Q(s, a)]$$  

term to push down big Q-values

regular objective  

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

can show that $\hat Q^\pi \leq Q^\pi$ for large enough $\alpha$

true Q-function
A better bound: always pushes Q-values down push up on (s, a) samples in data

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

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

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

Kumar, Zhou, Tucker, Levine. Conservative Q-Learning for Offline Reinforcement Learning. ‘20
Why is offline RL difficult?

How do we design offline RL algorithms?

Conservative Q-Learning

Model-based offline RL

How do we evaluate offline RL methods?
How does **model-based** RL work?

The model answers "what if" questions... so the model’s predictions are invalid if these states are OOD.

What goes wrong when we can’t collect more data?

The model answers "what if" questions.
MOPO: Model-Based Offline Policy Optimization

solution: “punish” the policy for exploiting

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

...and then use any existing model-based RL algorithm

Yu*, Thomas*, Yu, Ermon, Zou, Levine, Finn, Ma. MOPO: Model-Based Offline Policy Optimization. ‘20

See also: Kidambi et al., MOREL: Model-Based Offline Reinforcement Learning. ’20 (concurrent)
MOPO: Theoretical Analysis

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

we can represent the value function

model error is bounded (above) by \( u(s, a) \)

Theorem 4.4. Under Assumption 4.2 and 4.3, the learned policy \( \hat{\pi} \) in MOPO (Algorithm 1) satisfies

true return of policy trained under model

\[ \eta_M(\hat{\pi}) \geq \sup_{\pi} \{ \eta_M(\pi) - 2\lambda \epsilon_u(\pi) \} \]  

In particular, for all \( \delta \geq \delta_{\text{min}} \),

some implications:

\[ \eta_M(\hat{\pi}) \geq \eta_M(\pi^B) - 2\lambda \epsilon_u(\pi^B) \]

➢ improves over behavior policy

\[ \eta_M(\hat{\pi}) \geq \eta_M(\pi^*) - 2\lambda \epsilon_u(\pi^*) \]

➢ quantifies "optimality gap" in terms of model error

\[ \epsilon_u(\pi) := \mathbb{E}_{(s, a) \sim \rho_{\pi}^M} [u(s, a)] \]

\[ \eta_M(\hat{\pi}) \geq \eta_M(\pi^\delta) - 2\lambda \delta \]

\[ \pi^\delta := \arg \max_{\pi : \epsilon_u(\pi) \leq \delta} \eta_M(\pi) \]
COMBO: Conservative Model-Based RL

**Basic idea:** just like CQL minimizes Q-value of policy actions, we can minimize Q-value of model state-action tuples from the model

\[
\hat{Q}^{k+1} \leftarrow \arg \min_Q \beta \left( \mathbb{E}_{s,a \sim \rho(s,a)} [Q(s,a)] - \mathbb{E}_{s,a \sim \mathcal{D}} [Q(s,a)] \right) + \frac{1}{2} \mathbb{E}_{s,a,s' \sim d_f} \left[ \left( Q(s,a) - \hat{\mathbb{E}}^n Q^k(s,a) \right)^2 \right]. \tag{4}
\]

**Intuition:** if the model produces something that looks clearly different from real data, it’s easy for the Q-function to make it look bad

<table>
<thead>
<tr>
<th>Dataset type</th>
<th>Environment</th>
<th>BC</th>
<th>COMBO (ours)</th>
<th>MOPO</th>
<th>CQL</th>
<th>SAC-off</th>
<th>BEAR</th>
<th>BRAC-p</th>
<th>BRAC-v</th>
</tr>
</thead>
<tbody>
<tr>
<td>random</td>
<td>halfcheetah</td>
<td>2.1</td>
<td>38.8</td>
<td>35.4</td>
<td>35.4</td>
<td>30.5</td>
<td>25.1</td>
<td>24.1</td>
<td>31.2</td>
</tr>
<tr>
<td>random</td>
<td>hopper</td>
<td>1.6</td>
<td>17.9</td>
<td>11.7</td>
<td>10.8</td>
<td>11.3</td>
<td>11.4</td>
<td>11.0</td>
<td>12.2</td>
</tr>
<tr>
<td>random</td>
<td>walker2d</td>
<td>9.8</td>
<td>13.6</td>
<td>7.0</td>
<td>4.1</td>
<td>7.3</td>
<td>0.2</td>
<td>1.9</td>
<td></td>
</tr>
<tr>
<td>medium</td>
<td>halfcheetah</td>
<td>36.1</td>
<td>42.3</td>
<td>44.4</td>
<td>-4.3</td>
<td>41.7</td>
<td>43.8</td>
<td>46.3</td>
<td></td>
</tr>
<tr>
<td>medium</td>
<td>hopper</td>
<td>29.0</td>
<td>94.9</td>
<td>28.0</td>
<td>86.6</td>
<td>0.8</td>
<td>52.1</td>
<td>32.7</td>
<td>31.1</td>
</tr>
<tr>
<td>medium</td>
<td>walker2d</td>
<td>6.6</td>
<td>75.5</td>
<td>17.8</td>
<td>74.5</td>
<td>0.9</td>
<td>59.1</td>
<td>77.5</td>
<td>81.1</td>
</tr>
<tr>
<td>medium-replay</td>
<td>halfcheetah</td>
<td>38.4</td>
<td>55.1</td>
<td>53.1</td>
<td>46.2</td>
<td>-2.4</td>
<td>38.6</td>
<td>45.4</td>
<td>47.7</td>
</tr>
<tr>
<td>medium-replay</td>
<td>hopper</td>
<td>11.8</td>
<td>73.1</td>
<td>67.5</td>
<td>48.6</td>
<td>3.5</td>
<td>33.7</td>
<td>0.6</td>
<td>0.6</td>
</tr>
<tr>
<td>medium-replay</td>
<td>walker2d</td>
<td>11.3</td>
<td>56.0</td>
<td>39.0</td>
<td>32.6</td>
<td>1.9</td>
<td>19.2</td>
<td>-0.3</td>
<td>0.9</td>
</tr>
<tr>
<td>med-exp</td>
<td>halfcheetah</td>
<td>35.8</td>
<td>90.0</td>
<td>63.3</td>
<td>62.4</td>
<td>1.8</td>
<td>53.4</td>
<td>44.2</td>
<td>41.9</td>
</tr>
<tr>
<td>med-exp</td>
<td>hopper</td>
<td>111.1</td>
<td>111.1</td>
<td>23.7</td>
<td>111.0</td>
<td>1.6</td>
<td>96.3</td>
<td>1.9</td>
<td>0.8</td>
</tr>
<tr>
<td>med-exp</td>
<td>walker2d</td>
<td>6.4</td>
<td>96.1</td>
<td>44.6</td>
<td>98.7</td>
<td>-0.1</td>
<td>40.1</td>
<td>76.9</td>
<td>81.6</td>
</tr>
</tbody>
</table>

Yu, Kumar, Rafailov, Rajeswaran, Levine, Finn. **COMBO: Conservative Offline Model-Based Policy Optimization.** 2021.
Why is offline RL difficult?

How do we design offline RL algorithms?

Conservative Q-Learning

Model-based offline RL

How do we evaluate offline RL methods?
How do we evaluate offline RL methods?

> maybe just train a reference policy with RL?

**typical protocol in prior work:**

1. train $\pi_\beta$ with **online** RL
2. either collect data throughout training **OR**
2. collect data from final policy $\pi_\beta$

**this is a really bad idea**

- If you already have a good policy, why bother with offline RL?
- In the real world, data might come from non-Markovian “policies”
  - Human users
  - Hand-engineered policies
- Must use data that is **representative of real-world settings** and **leaves lots of room for improvement**
- Offline RL **must learn policies that are much better than the behavior policy!**

**without testing these properties, we cannot trust that our algorithms are good!**
D4RL: Datasets for Data-Driven Deep RL

What are some important principles to keep in mind?

**Data from non-RL policies**, including data from humans

**Stitching**: data where dynamic programming can find much better solutions

**Realistic tasks**
How does CQL compare?

CQL seems to work quite well across many tasks!

And we seem to know why it works!

But there is still plenty of room for improvement...

“1%” dataset from Agarwal et al.

Kumar, Zhou, Tucker, Levine. **Conservative Q-Learning for Offline Reinforcement Learning.** ‘20
Which offline RL method should I use?

CQL-like methods

- Seems to get best results on external benchmarks (e.g., D4RL)
- From my experience, harder to use with online finetuning (too conservative)
- Modifies the critic

AWR-like methods

- Seems to get best results on external benchmarks when finetuning
- Seems to be much worse than CQL on benchmarks (e.g., D4RL) in fully offline mode
- Modifies the actor

These are purely empirical observations, and they might change with better implementations!

Seems to imply we can combine to get the best of both worlds

We have not been successful at this so far
Summary and takeaways

- Offline RL algorithms can be built out of Q-Learning methods.
- But this can fail if there is narrow coverage (often the case in IL+RL).
- Offline RL is difficult because of distributional shift.
- Solutions typically mitigate this in some way.
- AWR & AWAC: implicit constraint formed by using a weighted imitation learning objective (weighted using the critic!).
- CQL: conservative critic objective that directly avoids overestimation.
- Model-based offline RL: similar principle, avoid overestimating by penalizing value far from data.