Offline Reinforcement Learning

Sanjiban Choudhury
The story thus far ...

- Decision-making
- Perception
- Models of humans

☐ Practical Robot Learning

Today-> ☐ Offline RL
☐ Sim-to-Real
Today’s class

- What is offline RL? Why do we need it for robots?

- Paradigm 1: Offline RL via Pessimism
  - Problem with Q-learning
  - Pessimism to the rescue

- Paradigm 2: RL via Supervised Learning
  - Return-conditioned Supervised Learning
  - Problem in Stochastic MDPs
Why do we need offline RL for robots?
Robots today still only work in CLOSED world
Generalize to variations of the OPEN world?
Why can’t we do RL with robots in the real world?
Machine learning’s answer!

Credit: Sergey Levine “Offline RL lecture"
Efforts underway to scale up robotics data!

1M trajectories, 22 robots, 21 different institutions

Open-X Embodiment Dataset
Hope: Data grows logarithmically with tasks

On the quest for shared priors
w/ machine learning

Interact with the physical world to learn bottom-up commonsense
i.e. "how the world works"

Credit: Andy Zeng
Reality: Data grows linearly with tasks

On the quest for shared priors w/ machine learning

Interact with the physical world to learn bottom-up commonsense

i.e. "how the world works"

Credit: Andy Zeng
But for today, let’s pretend we can collect a ton of data that “covers” tasks we care about
How can we learn optimal from large data collected by any policy?
Can we develop **data-driven** RL methods?

**Goal:** Offline Reinforcement Learning

*Credit: Sergey Levine “Offline RL lecture”*
Different paradigms of RL

on-policy RL

Credit: Sergey Levine “Offline RL lecture”
Different paradigms of RL

on-policy RL

off-policy RL

Credit: Sergey Levine “Offline RL lecture”
Different paradigms of RL

offline reinforcement learning

Credit: Sergey Levine “Offline RL lecture”
Offline RL enables robots to learn:
from pre-collected datasets
without real-time interaction,
enabling safer training
and leveraging diverse experiences.
Today’s class

☑️ What is offline RL? Why do we need it for robots?

☐ Paradigm 1: Offline RL via Pessimism
  ☐ Problem with Q-learning
  ☐ Pessimism to the rescue

☐ Paradigm 2: RL via Supervised Learning
  ☐ Return-conditioned Supervised Learning
  ☐ Problem in Stochastic MDPs
Let’s begin with a simple “offline” RL algorithm
We have already covered a fundamental algorithm in class that can learn from offline data.

What is it?
For every $\left(s_t, a_t, r_t, s_{t+1}\right)$

$$Q^*(s_t, a_t) = Q^*(s_t, a_t) + \alpha (r(s_t, a_t) + \gamma \max_{a'} Q^*(s_{t+1}, a') - Q^*(s_t, a_t))$$
Fitted Q-Iteration

Training is a regression problem

\[ \ell(\theta) = \sum_{i=1}^{N} (Q_\theta(s_i, a_i) - \text{target})^2 \]

Fitted Q-iteration

Given \( \{s_i, a_i, r_i, s'_i\}_{i=1}^{N} \)

Init \( Q_\theta(s, a) \leftarrow 0 \)

while not converged do
  \[ D \leftarrow \emptyset \]
  for \( i \in 1, \ldots, N \) do
    input \( \leftarrow \{s_i, a_i\} \)
    target \( \leftarrow r_i + \gamma \max_{a'} Q_\theta(s'_i, a') \)
    \( D \leftarrow D \cup \{\text{input, target}\} \)
  end for
  \( Q_\theta \leftarrow \text{Train}(D) \)
return \( Q_\theta \)
Activity!
Consider the following MDP
Let's say I collected some data from the MDP.
What policy would Q-learning pick?

Assume we are in the tabular case.

Initialize Q values with 0’s.

For every \((s_t, a_t, r_t, s_{t+1})\)

\[
Q^*(s_t, a_t) = Q^*(s_t, a_t) + \alpha (r(s_t, a_t) + \gamma \max_{a'} Q^*(s_{t+1}, a') - Q^*(s_t, a_t))
\]
Think-Pair-Share!

Think (30 sec): What policy would Q-learning pick in the tabular setting? Why? Ideas to fix it?

Pair: Find a partner

Share (45 sec): Partners exchange ideas
The Problem with Q-learning

Fundamental problem: counterfactual queries

Q-learning can be incorrectly optimistic about actions it has not seen in the data.
Today’s class

☑ What is offline RL? Why do we need it for robots? (Enables safer training, leverages diverse experience)

☐ Paradigm 1: Offline RL via Pessimism
  ☑ Problem with Q-learning (Incorrectly optimistic about unseen actions)
  ☐ Pessimism to the rescue

☐ Paradigm 2: RL via Supervised Learning
  ☐ Return-conditioned Supervised Learning
  ☐ Problem in Stochastic MDPs
Pessimism
Pessimism as a policy constraint

Don’t deviate too much from the data collecting policy
Pessimism as a policy constraint

Don’t deviate too much from the data collecting policy

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

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

Typical Q-learning

Credit: Sergey Levine “Offline RL lecture”
Pessimism as a policy constraint

Don’t deviate too much from the data collecting policy

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

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

Typical Q-learning

Add a constraint on policy

Credit: Sergey Levine “Offline RL lecture”
TD3+BC: Most simple and effective offline RL!

A Minimalist Approach to Offline Reinforcement Learning

Scott Fujimoto\textsuperscript{1,2} \quad Shixiang Shane Gu\textsuperscript{2}
\textsuperscript{1}Mila, McGill University
\textsuperscript{2}Google Research, Brain Team
scott.fujimoto@mail.mcgill.ca

\[
\pi = \arg\max_{\pi} \mathbb{E}_{(s,a) \sim D} [Q(s, \pi(s))].
\]

\[
\pi = \arg\max_{\pi} \mathbb{E}_{(s,a) \sim D} \left[ \lambda Q(s, \pi(s)) - (\pi(s) - a)^2 \right],
\]
<table>
<thead>
<tr>
<th>Medium</th>
<th>HalfCheetah</th>
<th>BRAC-p</th>
<th>AWAC</th>
<th>CQL</th>
<th>Fisher-BRC</th>
<th>TD3+BC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>2.0 ±0.1</td>
<td>23.5</td>
<td>2.2</td>
<td>21.7 ±0.9</td>
<td>32.2 ±2.2</td>
<td>10.2 ±1.3</td>
</tr>
<tr>
<td>Hopper</td>
<td>9.5 ±0.1</td>
<td>11.1</td>
<td>9.6</td>
<td>10.7 ±0.1</td>
<td>11.4 ±0.2</td>
<td>11.0 ±0.1</td>
</tr>
<tr>
<td>Walker2d</td>
<td>1.2 ±0.2</td>
<td>0.8</td>
<td>5.1</td>
<td>2.7 ±1.2</td>
<td>0.6 ±0.6</td>
<td>1.4 ±1.6</td>
</tr>
<tr>
<td>Medium</td>
<td>36.6 ±0.6</td>
<td>44.0</td>
<td>37.4</td>
<td>37.2 ±0.3</td>
<td>41.3 ±0.5</td>
<td>42.8 ±0.3</td>
</tr>
<tr>
<td>Hopper</td>
<td>30.0 ±0.5</td>
<td>31.2</td>
<td>72.0</td>
<td>44.2 ±10.8</td>
<td>99.4 ±0.4</td>
<td>99.5 ±1.0</td>
</tr>
<tr>
<td>Walker2d</td>
<td>11.4 ±6.3</td>
<td>72.7</td>
<td>30.1</td>
<td>57.5 ±8.3</td>
<td>79.5 ±1.0</td>
<td>79.7 ±1.8</td>
</tr>
<tr>
<td>Medium</td>
<td>34.7 ±1.8</td>
<td>45.6</td>
<td>-</td>
<td>41.9 ±1.1</td>
<td>43.3 ±0.9</td>
<td>43.3 ±0.5</td>
</tr>
<tr>
<td>Hopper</td>
<td>19.7 ±5.9</td>
<td>0.7</td>
<td>-</td>
<td>28.6 ±0.9</td>
<td>35.6 ±2.5</td>
<td>31.4 ±3.0</td>
</tr>
<tr>
<td>Walker2d</td>
<td>8.3 ±1.5</td>
<td>-0.3</td>
<td>-</td>
<td>15.8 ±2.6</td>
<td>42.6 ±7.0</td>
<td>25.2 ±5.1</td>
</tr>
<tr>
<td>Medium</td>
<td>67.6 ±13.2</td>
<td>43.8</td>
<td>36.8</td>
<td>27.1 ±3.9</td>
<td>96.1 ±9.5</td>
<td>97.9 ±4.4</td>
</tr>
<tr>
<td>Hopper</td>
<td>89.6 ±27.6</td>
<td>1.1</td>
<td>80.9</td>
<td>111.4 ±1.2</td>
<td>90.6 ±43.3</td>
<td>112.2 ±0.2</td>
</tr>
<tr>
<td>Walker2d</td>
<td>12.0 ±5.8</td>
<td>-0.3</td>
<td>42.7</td>
<td>68.1 ±13.1</td>
<td>103.6 ±4.6</td>
<td>101.1 ±9.3</td>
</tr>
<tr>
<td>Expert</td>
<td>105.2 ±1.7</td>
<td>3.8</td>
<td>78.5</td>
<td>82.4 ±7.4</td>
<td>106.8 ±3.0</td>
<td>105.7 ±1.9</td>
</tr>
<tr>
<td>Hopper</td>
<td>111.5 ±1.3</td>
<td>6.6</td>
<td>85.2</td>
<td>111.2 ±2.1</td>
<td>112.3 ±0.2</td>
<td>112.2 ±0.2</td>
</tr>
<tr>
<td>Walker2d</td>
<td>56.0 ±24.9</td>
<td>-0.2</td>
<td>57.0</td>
<td>103.8 ±7.6</td>
<td>79.9 ±32.4</td>
<td>105.7 ±2.7</td>
</tr>
<tr>
<td>Total</td>
<td>595.3 ±91.5</td>
<td>284.1</td>
<td>-</td>
<td>764.3 ±61.5</td>
<td>974.6 ±108.3</td>
<td>979.3 ±33.4</td>
</tr>
</tbody>
</table>
Works on real self-driving problems!

Imitation Is Not Enough: Robustifying Imitation with Reinforcement Learning for Challenging Driving Scenarios

Yiren Lu\textsuperscript{1}, Justin Fu\textsuperscript{1}, George Tucker\textsuperscript{2}, Xinlei Pan\textsuperscript{1}, Eli Bronstein\textsuperscript{1}, Rebecca Roelofs\textsuperscript{2}, Benjamin Sapp\textsuperscript{1}, Brandyn White\textsuperscript{1}, Aleksandra Faust\textsuperscript{2}, Shimon Whiteson\textsuperscript{1}, Dragomir Anguelov\textsuperscript{1}, Sergey Levine\textsuperscript{2,3}
Works on real self-driving problems!

**Imitation Is Not Enough: Robustifying Imitation with Reinforcement Learning for Challenging Driving Scenarios**

Yiren Lu$^1$, Justin Fu$^1$, George Tucker$^2$, Xinlei Pan$^1$, Eli Bronstein$^1$, Rebecca Roelofs$^2$, Benjamin Sapp$^1$, Brandyn White$^1$, Aleksandra Faust$^2$, Shimon Whiteson$^1$, Dragomir Anguelov$^1$, Sergey Levine$^{2,3}$

[Image: Two side-by-side diagrams showing a comparison of MGAII (left) and BC-SAC (ours) in top 5 case scenarios.]
Many more sophisticated offline RL methods

Conservative Q-Learning for Offline Reinforcement Learning

Aviral Kumar¹, Aurick Zhou¹, George Tucker², Sergey Levine¹,²
¹UC Berkeley, ²Google Research, Brain Team
aviralk@berkeley.edu

Adversarially Trained Actor Critic for Offline Reinforcement Learning

Ching-An Cheng¹, Tengyang Xie², Nan Jiang², Alekh Agarwal³

Instead of constraining policy, compute pessimistic Q values

Optimize the best worst case performance
Today’s class

☑ What is offline RL? Why do we need it for robots?  
(Enables safer training, leverages diverse experience)

☐ Paradigm 1: Offline RL via Pessimism
  ☑ Problem with Q-learning  
    (Incorrectly optimistic about unseen actions)
  ☑ Pessimism to the rescue  
    (Constrain policy to not deviate from data)

☐ Paradigm 2: RL via Supervised Learning
  ☐ Return-conditioned Supervised Learning
  ☐ Problem in Stochastic MDPs
Reinforcement Learning is Hard ...
Many horror stories of RL!

Nightmares of Policy Optimization
Need many tricks to make Q-learning work in practice!

Double Q Learning
Prioritized Replay
Dueling Networks
Multi-step Learning
Distributional RL
Noisy Nets
Can we just go back to good old supervised learning?
Supervised Learning success stories
What if I did supervised learning (BC) here?
What if I did supervised learning (BC) only on the top % rollouts?

$\pi(a \mid s)$
An embarrassingly simply algorithm: BC%

1. Collect offline dataset using whatever behavior policy

2. Get the top % trajectories based on returns

3. Do BC on just that!
Does this even work ?!? 

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Environment</th>
<th>10%BC</th>
<th>25%BC</th>
<th>40%BC</th>
<th>100%BC</th>
<th>CQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Medium</td>
<td>HalfCheetah</td>
<td>42.9</td>
<td>43.0</td>
<td>43.1</td>
<td>43.1</td>
<td>44.4</td>
</tr>
<tr>
<td>Medium</td>
<td>Hopper</td>
<td>65.9</td>
<td>65.2</td>
<td>65.3</td>
<td>63.9</td>
<td>58.0</td>
</tr>
<tr>
<td>Medium</td>
<td>Walker</td>
<td>78.8</td>
<td>80.9</td>
<td>78.8</td>
<td>77.3</td>
<td>79.2</td>
</tr>
<tr>
<td>Medium</td>
<td>Reacher</td>
<td>51.0</td>
<td>48.9</td>
<td>58.2</td>
<td>58.4</td>
<td>26.0</td>
</tr>
<tr>
<td>Medium-Replay</td>
<td>HalfCheetah</td>
<td>40.8</td>
<td>40.9</td>
<td>41.1</td>
<td>4.3</td>
<td>46.2</td>
</tr>
<tr>
<td>Medium-Replay</td>
<td>Hopper</td>
<td>70.6</td>
<td>58.6</td>
<td>31.0</td>
<td>27.6</td>
<td>48.6</td>
</tr>
<tr>
<td>Medium-Replay</td>
<td>Walker</td>
<td>70.4</td>
<td>67.8</td>
<td>67.2</td>
<td>36.9</td>
<td>26.7</td>
</tr>
<tr>
<td>Medium-Replay</td>
<td>Reacher</td>
<td>33.1</td>
<td>16.2</td>
<td>10.7</td>
<td>5.4</td>
<td>19.0</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td></td>
<td><strong>56.7</strong></td>
<td><strong>52.7</strong></td>
<td><strong>49.4</strong></td>
<td><strong>39.5</strong></td>
<td><strong>43.5</strong></td>
</tr>
</tbody>
</table>
An embarrassingly simply algorithm: BC%

1. Collect offline dataset using whatever behavior policy
   2. Get the top % trajectories based on returns
   3. Do BC on just that!

Challenge with BC%:
What happens as I vary % from small to high values?
Can we have a more principled approach?
Idea: Train a policy *conditioned* on the returns

\[ \pi(a \mid s, R) \]
RvS: What is Essential for Offline RL via Supervised Learning?

Scott Emmons\textsuperscript{1}, Benjamin Eysenbach\textsuperscript{2}, Ilya Kostrikov\textsuperscript{1}, Sergey Levine\textsuperscript{1}
\textsuperscript{1}UC Berkeley, \textsuperscript{2}Carnegie Mellon University

emmons@berkeley.edu
The Idea

(a) replay buffer
The Idea

(b) training dataset
The Idea

(c) network architecture
The Algorithm

\[
\max_{\theta} \sum_{\tau \in \mathcal{D}} \sum_{1 \leq t \leq |\tau|} \mathbb{E}_{\omega \sim f(\omega | \tau_{t:H})} \left[ \log \pi_{\theta}(a_t | s_t, \omega) \right].
\]

Algorithm 1 RvS-Learning

1: **Input**: Dataset of trajectories, \( \mathcal{D} = \{\tau\} \)
2: Initialize policy \( \pi_{\theta}(a | s, \omega) \).
3: **while** not converged **do**
4: Randomly sample trajectories: \( \tau \sim \mathcal{D} \).
5: Sample time index for each trajectory, \( t \sim [1, H] \), and sample a corresponding outcome: \( \omega \sim f(\omega | \tau_{t:H}) \).
6: Compute loss: \( \mathcal{L}(\theta) \leftarrow \sum_{(s_t,a_t,\omega)} \log \pi_{\theta}(a_t | s_t, \omega) \)
7: Update policy parameters: \( \theta \leftarrow \theta + \eta \nabla_{\theta} \mathcal{L}(\theta) \)
8: **end while**
9: **return** Conditional policy \( \pi_{\theta}(a | s, \omega) \)
What are some choices for “outcomes”? 

Option 1: What is the future state the agent ended up at?

RvS-G (Goal conditioned)

Option 2: What is the total return that the agent got?

RvS-R (Return conditioned)
A very popular idea

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. **Decision transformer: Reinforcement learning via sequence modeling**

Felipe Codevilla, Matthias Muller, Antonio Lopez, Vladlen Koltun, and Alexey Dosovitskiy. **End-to-end driving via conditional imitation learning**

Yiming Ding, Carlos Florensa, Pieter Abbeel, and Mariano Phielipp. **Goal-conditioned imitation learning.**

Michael Janner, Qiyang Li, and Sergey Levine. **Offline reinforcement learning as one big sequence modeling problem**

Aviral Kumar, Xue Bin Peng, and Sergey Levine. **Reward-conditioned policies**

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. **Advantage-weighted regression: Simple and scalable off-policy reinforcement learning**

Rupesh Kumar Srivastava, Pranav Shyam, Filipe Mutz, Wojciech Jaskowski, and Jurgen Schmidhuber. **Training agents using upside-down reinforcement learning**
Many popular algorithm, e.g. Decision Transformer

\[ \hat{R}_{t-1} \]
Today’s class

☑️ What is offline RL? Why do we need it for robots?  
(Enables safer training, leverages diverse experience)

☐ Paradigm 1: Offline RL via Pessimism
  ☑️ Problem with Q-learning  (Incorrectly optimistic about unseen actions)
  ☑️ Pessimism to the rescue  (Constrain policy to not deviate from data)

☐ Paradigm 2: RL via Supervised Learning
  ☑️ Return-conditioned Supervised Learning
  ☐ Problem in Stochastic MDPs
Activity!
Consider the following MDP

\[ S_0 \]

- \( a_0 \):
  - 50% chance of going to state 1 with reward \( r = -5 \)
  - 50% chance of going to state 2 with reward \( r = -15 \)

- \( a_1 \):
  - 50% chance of going to state 1 with reward \( r = 1 \)
  - 50% chance of going to state 2 with reward \( r = -6 \)

- \( a_2 \):
  - 50% chance of going to state 1 with reward \( r = 1 \)
  - 50% chance of going to state 2 with reward \( r = -6 \)
Consider the following MDP

What is the optimal action? What will Decision Transformer play?
Think-Pair-Share!

Think (30 sec): What is the optimal action? What would decision transformers play?

Pair: Find a partner

Share (45 sec):
Partners exchange ideas

- $s_0$ to $a_0$: 50% chance, $r = -5$; 50% chance, $r = -15$
- $s_0$ to $a_1$: 50% chance, $r = 1$
- $s_0$ to $a_2$: 50% chance, $r = 1$; 50% chance, $r = -6$
\[ \hat{R} = \sum_{t=0}^{T-1} r_t \]

\[ \hat{R}_0 = \sum_{t=0}^{T-1} r_t \]

\[ \hat{R}_1 = \sum_{t=1}^{T-1} r_t \]
Introducing Decision Transformers on Hugging Face 😊
Test Time

Start at initial state $s_0$

Specify the desired target return $R_0$

$a_0 = \text{Transformer}(R_0, s_0)$

Execute action, observe reward and next state $(r_0, s_1)$

Decrement the target return $R_1 = R_0 - r_0$

$a_1 = \text{Transformer}(R_0, s_0, a_0, R_1, s_1)$
Seems to work!
Seems to work!

<table>
<thead>
<tr>
<th>Game</th>
<th>DT (Ours)</th>
<th>CQL</th>
<th>QR-DQN</th>
<th>REM</th>
<th>BC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breakout</td>
<td>267.5 ± 97.5</td>
<td>211.1</td>
<td>17.1</td>
<td>8.9</td>
<td>138.9 ± 61.7</td>
</tr>
<tr>
<td>Qbert</td>
<td>15.4 ± 11.4</td>
<td>104.2</td>
<td>0.0</td>
<td>0.0</td>
<td>17.3 ± 14.7</td>
</tr>
<tr>
<td>Pong</td>
<td>106.1 ± 8.1</td>
<td>111.9</td>
<td>18.0</td>
<td>0.5</td>
<td>85.2 ± 20.0</td>
</tr>
<tr>
<td>Seaquest</td>
<td>2.5 ± 0.4</td>
<td>1.7</td>
<td>0.4</td>
<td>0.7</td>
<td>2.1 ± 0.3</td>
</tr>
</tbody>
</table>

Atari
Today’s class

☑ What is offline RL? Why do we need it for robots? (Enables safer training, leverages diverse experience)

☐ Paradigm 1: Offline RL via Pessimism
  ☑ Problem with Q-learning (Incorrectly optimistic about unseen actions)
  ☑ Pessimism to the rescue (Constrain policy to not deviate from data)

☐ Paradigm 2: RL via Supervised Learning
  ☐ Return-conditioned Supervised Learning
  ☐ Problem in Stochastic MDPs