Offline RL for Real-Robot Pre-Training and Fine-Tuning

Aviral Kumar
Offline Reinforcement Learning

Standard Online RL

Offshore RL

Collect a one-time dataset

No unsafe or costly exploration

Potential to bring generalization benefits of supervised learning
What this Picture Actually Looks Like

Several key properties

- Multi-task data, no reward
- Humans or “other agents”
- Directly from raw visual observations

How can we apply offline RL in the presence of all of this?
Learning from Diverse Robot Datasets

Pre-trained representation

Imitation learning

Ebert et al. 2021
Young et al. 2021
and many more...

Pre-training on broad data

Fine-tuning on limited, task-specific data

Can we instead use offline RL for both pre-training & fine-tuning?
Pre-Training for Robots Using Offline RL

1. Pre-train via offline RL

10 domains
100 tasks
12k demos

2. Continue fine-tuning with offline RL

Bridge data: Boosting Generalization of Robotic Skills. RSS 2022.
Ingredient 1: Conservative Q-Learning

The issue in offline RL is erroneous Q-values at out-of-distribution (OOD) actions

\[ Q(s, a) \leftarrow r(s, a) + \gamma \mathbb{E}_{a' \sim \pi(\theta)}(Q(s', a')) \]

Control this value somehow!

\[ Q_{CQL}(s, a) := \arg \min_{Q} \max_{\mu} \left( \mathbb{E}_{s \sim D} \mathbb{E}_{a \sim \mu(s)}[Q(s, a)] - \mathbb{E}_{s, a \sim D}[Q(s, a)] \right) + \frac{1}{2\alpha} \mathbb{E}_{s, a, s' \sim D} \left[ (Q(s, a) - y(s, a))^2 \right] \]

Maximize the data Q-values

Minimize OOD Q-values

“Bake the pessimism” into the Q-function

Ingredient 2: Architecture

Modified ResNet with group normalization

Output ResNet feature maps

Learned position embeddings

Fully-connected Layers

Task ID

Action vector duplicated

$Q_\phi(s, a)$
**Ingredient 3: Rewards & Checkpoint Selection**

**Rewards**

Sparse, binary rewards, +1 at the end of the trajectory

**Important:** the binary values matter (-1, +10)

**Checkpoint selection**

**Worst case:** impossible!

But can use the knowledge that data is “expert”
Summary of Ingredients in PTR

**Ingredient 1:** An Offline RL Algorithm (CQL)

**Ingredient 2:** A high-capacity architecture (ResNet + group normalization + action duplication + learned spatial embeddings)

**Ingredient 3:** Reward functions + checkpoint selection heuristic
Now some Empirical Results....
Task: Solving A Task in A New Domain

1. Pre-Train on Bridge Data, 12 doors 800 demonstrations

2. Fine-Tune on Target Domain Data: 1 door, 10 demonstrations
Results: Solving A Task in A New Domain

Method: Imitation (Best prior method)

Task: Open Door

Method: PTR (Ours)

Task: Open Door
Task: Solving New Tasks in New Domains

10 target demonstrations
Results: Solving New Tasks in New Domains

- Best Prior method
  - Task: Put sweet potato on plate
    - Task: Put knife in pot
    - Task: Put sushi in pot

- PTR (Ours)
  - Task: Put sweet potato on plate
  - Task: Put knife in pot
  - Task: Put sushi in pot
Some Quantitative Results

<table>
<thead>
<tr>
<th>Task</th>
<th>PTR (Ours)</th>
<th>BC (fine.)</th>
<th>Autoreg. BC</th>
<th>BeT</th>
<th>Joint training</th>
<th>Target data only</th>
<th>Pre-train. rep. + BC finetune</th>
</tr>
</thead>
<tbody>
<tr>
<td>Take croissant from metal bowl</td>
<td>7/10</td>
<td>3/10</td>
<td>5/10</td>
<td>1/10</td>
<td>4/10</td>
<td>0/10</td>
<td>1/10</td>
</tr>
<tr>
<td>Put sweet potato on plate</td>
<td>7/20</td>
<td>1/20</td>
<td>1/20</td>
<td>0/20</td>
<td>0/20</td>
<td>0/20</td>
<td>0/20</td>
</tr>
<tr>
<td>Place knife in pot</td>
<td>4/10</td>
<td>2/10</td>
<td>2/10</td>
<td>0/10</td>
<td>1/10</td>
<td>3/10</td>
<td>0/10</td>
</tr>
<tr>
<td>Put cucumber in pot</td>
<td>5/10</td>
<td>0/10</td>
<td>1/10</td>
<td>0/10</td>
<td>2/10</td>
<td>1/10</td>
<td>0/10</td>
</tr>
</tbody>
</table>

Joint training vs pre-training??

**Takeaway:** Offline RL learns useful representations + better fine-tuning
Scaling Curve And Analysis

Why would RL enable better performance... ....when the data is collected via human teleoperation?

Preview: Value-functions can learn what’s critical!

Better performance with larger networks!

Qualitative Comparison of BC (finetune) and PTR

Task: Take Croissant from Metal Bowl
BC (finetune) Failure: grasps bowl instead of croissant when croissant is not underneath
PTR Success: grasps croissant and puts by sink

Task: Put Cucumber in Bowl
BC (finetune) Failure: executes an imprecise grasp, and fails to locate the pot accurately
PTR Success: Places Cucumber in Pot
Analysis of Why PTR Outperforms Imitation

With near-expert data

Just need to learn to roughly go right

Agent can only move within this region

High reward, \( r(s) = 1 \)

Single action succeeds ("critical")

\[ \text{SubOpt}(\pi_{RL}) \leq \text{SubOpt}(\pi_{BC}) \]

if volume of non-critical states in a trajectory is large

**Test:** Run weighted BC, where weights come from the learned Q-function!

<table>
<thead>
<tr>
<th>Task</th>
<th>BC (finetune)</th>
<th>PTR (Ours)</th>
<th>Advantage-weighted BC (finetune)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Put cucumber in pot</td>
<td>0/10</td>
<td>5/10</td>
<td>5/10</td>
</tr>
<tr>
<td>Take croissant from metal bowl</td>
<td>3/10</td>
<td>7/10</td>
<td>6/10</td>
</tr>
</tbody>
</table>

Takeaways and Future Directions

- Offline RL can be good for both representation learning and control, even with human demonstration data.

Future Directions:
- Extend to use videos and multi-robot data on more dexterous tasks.
- **Goal specification:** language? goals? reward learning?
- **Workflows:** “How should a practitioner tune this approach on their problem”

Thank You!

Code: https://github.com/Asap7772/PTR