Offline Reinforcement Learning
and Offline Multi-Task RL

CS 330
Reminders

Next Monday (Nov 8th): Homework 4 (optional) is due
The recipe that has worked in other fields so far:

A lot of data

Expressive, capable models
The reinforcement learning recipe so far:

A lot of data

Expressive, capable models

Hard to achieve same level of generalization with a per-experiment active learning loop!
The Plan

Offline RL problem formulation

Offline RL solutions

Offline multi-task RL and data sharing

Offline goal-conditioned RL
The Plan

Offline RL problem formulation

Offline RL solutions

Offline multi-task RL and data sharing

Offline goal-conditioned RL
The anatomy of a reinforcement learning algorithm

compute $\hat{Q} = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$ (MC policy gradient)
fit $Q_\phi(s, a)$ (actor-critic, Q-learning)
estimate $p(s'|s, a)$ (model-based)

$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$ (policy gradient)
$\pi(s) = \arg \max Q_\phi(s, a)$ (Q-learning)
optimize $\pi_\theta(a|s)$ (model-based)
On-policy vs Off-policy

- Data comes from the current policy
- Compatible with all RL algorithms
- Can’t reuse data from previous policies

- Data comes from any policy
- Works with specific RL algorithms
- Much more sample efficient, can re-use old data
Can you think of potential applications of offline RL?
What can offline RL do?

Find best behaviors in a dataset

Generalize best behaviors to similar situations

“Stitch” together parts of good behaviors into a better behavior
Fitted Q-iteration algorithm

full fitted Q-iteration algorithm:

1. collect dataset \( \{(s_i, a_i, s'_i, r_i)\} \) using some policy

2. set \( y_i \leftarrow r(s_i, a_i) + \gamma \max_{a'_i} Q_\phi(s'_i, a'_i) \)

3. set \( \phi \leftarrow \arg \min_{\phi} \frac{1}{2} \sum_i \|Q_\phi(s_i, a_i) - y_i\|^2 \)

Algorithm hyperparameters

- dataset size \( N \), collection policy
- iterations \( K \)
- gradient steps \( S \)

Result: get a policy \( \pi(a|s) \) from \( \arg \max_a Q_\phi(s, a) \)

Important notes:

- We can reuse data from previous policies!
- an off-policy algorithm using replay buffers
- This is not a gradient descent algorithm!

Slide adapted from Sergey Levine
QT-Opt: Q-learning at scale

In-memory buffers
- off-policy \((s, a, s', r)\)
- on-policy \((s, a, s', r)\)
- labeled \((s, a, Q_T(s, a))\)

Bellman updaters
compute \(Q_T(s, a) = r + \max_{a'} Q_\theta(s', a')\)

Training jobs
\[
\min_{\theta} \|Q_\theta(s, a) - Q_T(s, a)\|^2
\]

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

Kalashnikov, et al. QT-Opt, 2018
QT-Opt: setup and results

7 robots collected 580k grasps

Unseen test objects

580k offline + 28k online —— 96%
580k offline —— 87%
The Plan

Offline RL problem formulation

Offline RL solutions

Offline multi-task RL and data sharing

Offline goal-conditioned RL
The offline RL problem

Bellman equation:  
\[ Q^*(s_t, a_t) = \mathbb{E}_{s' \sim P(\cdot|s, a)} \left[ r(s, a) + \gamma \max_{a'} Q^*(s', a') \right] \]

How well it does?  How well it thinks it does?
The offline RL problem

Bellman equation: \( Q^*(s_t, a_t) = \mathbb{E}_{s' \sim p(\cdot | s, a)} [r(s, a) + \gamma \max_{a'} Q^*(s', a')] \)

We don’t know the value of the actions we haven’t taken (counterfactuals)

Q-learning is an adversarial algorithm!

What happens in the online case?
Solutions to the offline RL problem (explicit)

\[ \pi_\phi := \arg \max_{\phi} E_{a \sim \pi_\phi(a|s)}[Q(s,a)] \quad \text{s.t.} \quad D(\pi_\phi(a|s), \pi_\beta(a|s)) \leq \varepsilon \]

We need a constraint

KL-divergence: \( D_{KL}(\pi_\theta||\pi_\beta) \)

\( \pi_\theta(a|s) > 0 \) only if \( \pi_\beta(a|s) > \varepsilon \)

Wu et al. Behavior Regularized Offline RL, 2019
Kumar et al. Stabilizing Off-Policy Q-Learning via Bootstrapping Error Reduction, 2020
Solutions to the offline RL problem (implicit)

Solve constrained optimization via duality

$$\pi_\phi := \arg\max_{\phi} E_{a \sim \pi_\phi(a|s)}[Q(s, a)] \quad \text{s.t.} \quad D(\pi_\phi(a|s), \pi_\beta(a|s)) \leq \varepsilon$$

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

Peng*, Kumar* et al. Advantage-Weighted Regression, 2019

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

Peters et al, REPS

Nair et al. Accelerating Online RL with Offline Datasets, 2020
Solutions to the offline RL problem (implicit)

Conservative Q-Learning (CQL)

Conservative Q-learning (CQL) push down big Q values

\[
\hat{Q}_{\text{CQL}}^\pi := \min_Q \max_{\mu} \mathbb{E}_{a \sim \mu(s|a)}[Q(s, a)] \\
+ \frac{1}{2\alpha} \mathbb{E}_{s, a, s' \sim D} \left[ (Q(s, a) - (r(s, a) + \gamma \mathbb{E}_{a \sim \pi_\phi(s'|a)}[Q(s', a')]))^2 \right]
\]

Conservative Q-learning (CQL) push up data samples

\[
\hat{Q}_{\text{CQL}}^\pi := \min_Q \max_{\mu} \mathbb{E}_{a \sim \mu(s|a)}[Q(s, a)] - \mathbb{E}_{a \sim D(a|s)}[Q(s, a)] \\
+ \frac{1}{2\alpha} \mathbb{E}_{s, a, s' \sim D} \left[ (Q(s, a) - (r(s, a) + \gamma \mathbb{E}_{a \sim \pi_\phi(s'|a)}[Q(s', a')]))^2 \right]
\]

Kumar et al. Conservative Q-Learning for Offline RL, 2020
Solutions to the offline RL problem (implicit)

Conservative Q-Learning (CQL)

\[
\hat{Q}_{\text{CQL}}^\pi := \min_Q \max_\mu \mathbb{E}_{a \sim \mu(a|s)}[Q(s, a)] - \mathbb{E}_{a \sim D(a|s)}[Q(s, a)]
\]
\[
+ \frac{1}{2\alpha} \mathbb{E}_{s, a, s' \sim D} \left[ (Q(s, a) - (r(s, a) + \gamma \mathbb{E}_{a' \sim \pi(s|a)}[\hat{Q}(s', a')]))^2 \right]
\]

Kumar et al. Conservative Q-Learning for Offline RL, 2020
Conservative Q-Learning (CQL)

1. Update $\hat{Q}^\pi$ w.r.t. $\mathcal{L}_{\text{CQL}}(\hat{Q}^\pi)$ using $\mathcal{D}$
2. Update policy $\pi$

\[
\hat{Q}^\pi_{\text{CQL}} := \min_Q \max_{\mu} \mathbb{E}_{a \sim \mu(a|s)}[Q(s, a)] - \mathbb{E}_{a \sim \mathcal{D}(a|s)}[Q(s, a)] + \frac{1}{2\alpha} \mathbb{E}_{s, a, s' \sim \mathcal{D}}[(Q(s, a) - (r(s, a) + \gamma \mathbb{E}_{a \sim \pi(s|a)}[Q(s', a')]))^2]
\]

<table>
<thead>
<tr>
<th>Domain</th>
<th>Task Name</th>
<th>BC</th>
<th>SAC</th>
<th>BEAR</th>
<th>BRAC-p</th>
<th>BRAC-γ</th>
<th>CQL(ℋ)</th>
<th>CQL(δ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AntMaze</td>
<td>antmaze-umaze</td>
<td>65.0</td>
<td>0.0</td>
<td>73.0</td>
<td>50.0</td>
<td>70.0</td>
<td>74.0</td>
<td>73.5</td>
</tr>
<tr>
<td></td>
<td>antmaze-umaze-diverse</td>
<td>55.0</td>
<td>0.0</td>
<td>61.0</td>
<td>40.0</td>
<td>70.0</td>
<td>84.0</td>
<td>61.0</td>
</tr>
<tr>
<td></td>
<td>antmaze-medium-play</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>61.2</td>
<td>4.6</td>
</tr>
<tr>
<td></td>
<td>antmaze-medium-diverse</td>
<td>0.0</td>
<td>0.0</td>
<td>8.0</td>
<td>0.0</td>
<td>0.0</td>
<td>53.7</td>
<td>5.1</td>
</tr>
<tr>
<td></td>
<td>antmaze-large-play</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>15.8</td>
<td>3.2</td>
</tr>
<tr>
<td></td>
<td>antmaze-large-diverse</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>14.9</td>
<td>2.3</td>
</tr>
<tr>
<td>Adroit</td>
<td>pen-human</td>
<td>34.4</td>
<td>6.3</td>
<td>-1.0</td>
<td>8.1</td>
<td>0.6</td>
<td>37.5</td>
<td>55.8</td>
</tr>
<tr>
<td></td>
<td>hammer-human</td>
<td>1.5</td>
<td>0.5</td>
<td>0.3</td>
<td>0.3</td>
<td>0.2</td>
<td>4.4</td>
<td>2.1</td>
</tr>
<tr>
<td></td>
<td>door-human</td>
<td>0.5</td>
<td>3.9</td>
<td>-0.3</td>
<td>-0.3</td>
<td>-0.3</td>
<td>9.9</td>
<td>9.1</td>
</tr>
<tr>
<td></td>
<td>relocate-human</td>
<td>0.0</td>
<td>0.0</td>
<td>-0.3</td>
<td>-0.3</td>
<td>-0.3</td>
<td>0.20</td>
<td>0.35</td>
</tr>
<tr>
<td></td>
<td>pen-cloned</td>
<td>56.9</td>
<td>23.5</td>
<td>26.5</td>
<td>1.6</td>
<td>-2.5</td>
<td>39.2</td>
<td>40.3</td>
</tr>
<tr>
<td></td>
<td>hammer-cloned</td>
<td>0.8</td>
<td>0.2</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
<td>2.1</td>
<td>5.7</td>
</tr>
<tr>
<td></td>
<td>door-cloned</td>
<td>-0.1</td>
<td>0.0</td>
<td>-0.1</td>
<td>-0.1</td>
<td>-0.1</td>
<td>0.4</td>
<td>3.5</td>
</tr>
<tr>
<td></td>
<td>relocate-cloned</td>
<td>-0.1</td>
<td>-0.2</td>
<td>-0.3</td>
<td>-0.3</td>
<td>-0.3</td>
<td>-0.1</td>
<td>-0.1</td>
</tr>
<tr>
<td>Kitchen</td>
<td>kitchen-complete</td>
<td>33.8</td>
<td>15.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>43.8</td>
<td>31.3</td>
</tr>
<tr>
<td></td>
<td>kitchen-partial</td>
<td>33.8</td>
<td>0.0</td>
<td>13.1</td>
<td>0.0</td>
<td>0.0</td>
<td>49.8</td>
<td>50.1</td>
</tr>
<tr>
<td></td>
<td>kitchen-undirected</td>
<td>47.5</td>
<td>2.5</td>
<td>47.2</td>
<td>0.0</td>
<td>0.0</td>
<td>51.0</td>
<td>52.4</td>
</tr>
</tbody>
</table>

Kumar et al. Conservative Q-Learning for Offline RL, 2020
The Plan

Offline RL problem formulation

Offline RL solutions

Offline multi-task RL and data sharing

Offline goal-conditioned RL
Multi-task RL algorithms

Policy: $\pi_\theta(a|\bar{s}) \rightarrow \pi_\theta(a|\bar{s}, z_i)$
Q-function: $Q_\phi(\bar{s}, a) \rightarrow Q_\phi(\bar{s}, a, z_i)$

What is different about reinforcement learning?

The data distribution is controlled by the agent!

Should we share data in addition to sharing weights?
Example of multi-task Q-learning applied to robotics: MT-Opt

80% avg improvement over baselines across all the ablation tasks (4x improvement over single-task)

~4x avg improvement for tasks with little data

Fine-tunes to a new task (to 92% success) in 1 day

Kalashnikov et al. MT-Opt. CoRL ‘21
Can we share data across distinct tasks?

$$\mathcal{D} = \{(s_i, a_i, s'_{i}, r_i)\}$$

$$s \sim d^\pi(\mathbf{s})$$

$$a \sim \pi(\mathbf{a}|\mathbf{s})$$ — unknown!

$$s' \sim p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$$

- Sharing data generally helps
- It can hurt performance in some cases
- Can we characterize why it hurts performance?

Sharing data exacerbates distribution shift

<table>
<thead>
<tr>
<th>Dataset types / Tasks</th>
<th>Dataset Size</th>
<th>Avg Return No Sharing</th>
<th>Avg Return Sharing All</th>
<th>$D_{KL}(\pi, \pi_\beta)$ No Sharing</th>
<th>$D_{KL}(\pi, \pi_\beta)$ Sharing All</th>
</tr>
</thead>
<tbody>
<tr>
<td>medium-replay / run forward</td>
<td>109900</td>
<td>998.9</td>
<td>966.2</td>
<td>3.70</td>
<td>10.39</td>
</tr>
<tr>
<td>medium-replay / run backward</td>
<td>109980</td>
<td>1298.6</td>
<td>1147.5</td>
<td>4.55</td>
<td>12.70</td>
</tr>
<tr>
<td>medium-replay / jump</td>
<td>109511</td>
<td>1603.1</td>
<td>1224.7</td>
<td>3.57</td>
<td>15.89</td>
</tr>
<tr>
<td>average task performance</td>
<td>N/A</td>
<td>1300.2</td>
<td>1112.8</td>
<td>3.94</td>
<td>12.99</td>
</tr>
<tr>
<td>medium / run forward</td>
<td>27646</td>
<td>297.4</td>
<td>848.7</td>
<td>6.53</td>
<td>11.78</td>
</tr>
<tr>
<td>medium / run backward</td>
<td>31298</td>
<td>207.5</td>
<td>600.4</td>
<td>4.44</td>
<td>10.13</td>
</tr>
<tr>
<td>medium / jump</td>
<td>100000</td>
<td>351.1</td>
<td>776.1</td>
<td>5.57</td>
<td>21.27</td>
</tr>
<tr>
<td>average task performance</td>
<td>N/A</td>
<td>285.3</td>
<td>747.7</td>
<td>5.51</td>
<td>14.39</td>
</tr>
<tr>
<td>medium-replay / run forward</td>
<td>109900</td>
<td>590.1</td>
<td>701.4</td>
<td>1.49</td>
<td>7.76</td>
</tr>
<tr>
<td>medium-replay / run backward</td>
<td>31298</td>
<td>614.7</td>
<td>756.7</td>
<td>1.91</td>
<td>12.2</td>
</tr>
<tr>
<td>expert / jump</td>
<td>5000</td>
<td>1575.2</td>
<td>885.1</td>
<td>3.12</td>
<td>27.5</td>
</tr>
<tr>
<td>average task performance</td>
<td>N/A</td>
<td>926.6</td>
<td>781</td>
<td>2.17</td>
<td>15.82</td>
</tr>
</tbody>
</table>
Sharing data while reducing distributional shift - Conservative Data Sharing

We assume that relabeling data $D_j$ from task $j$ to task $i$ generates a dataset $D_{j \rightarrow i}$, which is then additionally used to train on task $i$. Thus, the effective dataset for task $i$ after relabeling is given by: $D_i^{\text{eff}} := D_i \cup (\bigcup_{j \neq i} D_{j \rightarrow i})$.

That means that we can control the dataset/behavior policy itself!
Can we automatically identify how to share data?

Standard offline RL:
\[
\pi^*(a|s) = \arg \max_{\pi} J_D(\pi) - \alpha D(\pi, \pi_\beta)
\]
Maximize reward Regularize towards the data (behavior policy \(\pi_\beta\))

Can we optimize the data distribution?
\[
\pi^*(&cdot;|cdot; , i) := \arg \max_{\pi} \max_{\pi_\beta \in \Pi_{\text{relabel}}} [J_{D_{i}^{\text{eff}}} (\pi) - \alpha D(\pi, \pi_\beta^{\text{eff}} ; i)]
\]
Optimize for the effective behavior policy to maximize reward and minimizes distribution shift

Conservative data sharing (CDS)
Share data \((s, a)\) when conservative Q-value will increase for that task
\[
\hat{Q}^\pi(s, a, i) - E_{s', a' \sim D_i} \left[ \hat{Q}^\pi(s', a', i) \right] \geq 0
\]
Does CDS prevent excessive distributional shift?

CDS reduces the KL divergence between the data distribution and the learned policy

This translates to improved performance

Yu*, Kumar*, Chebotar, Hausman, Levine, Finn. *Multi-Task Offline Reinforcement Learning with Conservative Data Sharing*, 2021
Experiments on vision-based robotic manipulation

Simulated object manipulation tasks

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>lift-banana</td>
<td>54.0%</td>
<td>39.7%</td>
<td>33.6%</td>
<td>45.6%</td>
<td>12.6%</td>
</tr>
<tr>
<td>lift-bottle</td>
<td>76.3%</td>
<td>58.5%</td>
<td>53.3%</td>
<td>42.8%</td>
<td>44.5%</td>
</tr>
<tr>
<td>lift-sausage</td>
<td>75.9%</td>
<td>65.6%</td>
<td>62.5%</td>
<td>73.8%</td>
<td>55.2%</td>
</tr>
<tr>
<td>lift-milk</td>
<td>82.7%</td>
<td>75.3%</td>
<td>62.8%</td>
<td>68.9%</td>
<td>58.9%</td>
</tr>
<tr>
<td>lift-food</td>
<td>70.3%</td>
<td>64.6%</td>
<td>23.1%</td>
<td>64.9%</td>
<td>29.5%</td>
</tr>
<tr>
<td>lift-can</td>
<td>76.1%</td>
<td>70.8%</td>
<td>37.6%</td>
<td>49.4%</td>
<td>41.8%</td>
</tr>
<tr>
<td>lift-carrot</td>
<td>80.4%</td>
<td>70.1%</td>
<td>69.4%</td>
<td>72.2%</td>
<td>63.1%</td>
</tr>
<tr>
<td>place-bowl</td>
<td>84.4%</td>
<td>72.0%</td>
<td>84.5%</td>
<td>64.7%</td>
<td>77.0%</td>
</tr>
<tr>
<td>place-plate</td>
<td>86.9%</td>
<td>82.2%</td>
<td>79.5%</td>
<td>75.1%</td>
<td>82.2%</td>
</tr>
<tr>
<td>place-divider-plate</td>
<td>89.4%</td>
<td>72.6%</td>
<td>81.0%</td>
<td>79.3%</td>
<td>84.7%</td>
</tr>
<tr>
<td><strong>average</strong></td>
<td><strong>77.6%</strong></td>
<td><strong>67.2%</strong></td>
<td><strong>58.7%</strong></td>
<td><strong>63.7%</strong></td>
<td><strong>55.0%</strong></td>
</tr>
</tbody>
</table>

Comparisons:
- **HIPI**: GCRL sharing strategy based on highest return
- **Skill**: domain-specific approach based on human intuition

Yu*, Kumar*, Chebotar, Hausman, Levine, Finn. Multi-Task Offline Reinforcement Learning with Conservative Data Sharing, 2021
The Plan

Offline RL problem formulation

Offline RL solutions

Offline multi-task RL and data sharing

Offline goal-conditioned RL
Goal-conditioned RL with hindsight relabeling

1. Collect data $\mathcal{D}_k = \{(s_{1:T}, a_{1:T}, s_g, r_{1:T})\}$ using some policy
2. Store data in replay buffer $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_k$
3. Perform hindsight relabeling:
   a. Relabel experience in $\mathcal{D}_k$ using last state as goal:
      $$\mathcal{D}_k' = \{(s_{1:T}, a_{1:T}, s_T, r_{1:T}')\} \text{ where } r_{t}' = -d(s_t, s_T)$$
   b. Store relabeled data in replay buffer $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_k'$
4. Update policy using replay buffer $\mathcal{D}$

What if we do it fully offline?

Kaelbling. Learning to Achieve Goals. IJCAI ’93
Actionable Models: Moving Beyond Tasks

- At scale, task definitions become a bottleneck
- **Goal** state is a task!
  Rewards through **hindsight relabeling**
- **Conservative Q-learning** to create artificial negative examples
Actionable Models: Moving Beyond Tasks

- At scale, task definitions become a bottleneck
- **Goal** state is a task!
  Rewards through **hindsight relabeling**
- **Conservative Q-learning** to create artificial negative examples

- **Functional understanding** of the world: a world model that also provides an **actionable policy**
- **Unsupervised** objective for Robotics?
  - Zero-shot visual tasks
  - Downstream fine-tuning

---

Actionable Models, Chebotar, Hausman, Lu, Xiao, Kalashnikov, Varley, Irpan, Eysenbach, Julian, Finn, Levine, ICML 2021
Actionable Models: Hindsight Relabeling

Offline dataset of robotic experience

Trajectories

Relabel all sub-sequences with goals and mark as successes
Actionable Models: Artificial Negatives

- Offline hindsight relabeling: only positive examples → need negatives

- Conservative strategy: minimize Q-values of unseen actions

- Sample contrastive artificial negative actions: $\tilde{a}_t \sim \exp(Q^\pi(s_t, \tilde{a}_t, g))$
Actionable Models: Goal Chaining

Offline dataset of robotic experience

Trajectories

Relabel all sub-sequences with goals and mark as successes

Random goals for goal chaining

Add conservative action negatives and mark as failures

Relabeled sequences
Actionable Models: Goal Chaining

- Recondition on random goals to enable chaining goals across episodes
Actionable Models: Goal Chaining

- Recondition on random goals to enable **chaining** goals **across episodes**
- If **pathway to a goal** exists: dynamic programming will propagate reward
- No pathway to the goal: conservative strategy will minimize Q-values
Actionable Models

Offline dataset of robotic experience

Trajectories

Relabel all sub-sequences with goals and mark as successes

Relabeled sequences

Random goals for goal chaining

Train goal-conditioned Q-function

\[ Q(s, a, g) \]

Add conservative action negatives and mark as failures

\[ g_R \]

\[ g \]
Actionable Models

Offline dataset of robotic experience

Train goal-conditioned Q-function

Relabel all sub-sequences with goals and mark as successes

Trajectories

Relabeled sequences

Random goals for goal chaining

Add conservative action negatives and mark as failures

Add conservative action negatives and mark as failures

$Q(s, a, g)$

Goal reaching
Actionable Models: Real world goal reaching

<table>
<thead>
<tr>
<th>Task</th>
<th>Success rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instance grasping</td>
<td>92%</td>
</tr>
<tr>
<td>Rearrangement</td>
<td>74%</td>
</tr>
<tr>
<td>Container placing</td>
<td>66%</td>
</tr>
</tbody>
</table>
Actionable Models

Offline dataset of robotic experience

Train goal-conditioned Q-function

Relabel all sub-sequences with goals and mark as successes

Add conservative action negatives and mark as failures

Random goals for goal chaining

Relabeled sequences

Trajectories

Goal reaching

Downstream tasks

$Q(s, a, g)$
Actionable Models: Downstream tasks

Simulated ablations

Real-world fine-tuning with a small amount of data

<table>
<thead>
<tr>
<th>Task</th>
<th>No pre-training</th>
<th>With pre-training</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grasp box</td>
<td>0%</td>
<td>27%</td>
</tr>
<tr>
<td>Grasp banana</td>
<td>4%</td>
<td>20%</td>
</tr>
<tr>
<td>Grasp milk</td>
<td>1%</td>
<td>20%</td>
</tr>
</tbody>
</table>
The Plan

Offline RL problem formulation

Offline RL solutions

Offline multi-task RL and data sharing

Offline goal-conditioned RL
Next time

What about long-horizon tasks?
Hierarchical RL
Skill discovery

Reminder

Homework 4 (optional) is due next Monday!