Offline Reinforcement Learning

CS 224R

Course reminders

Project

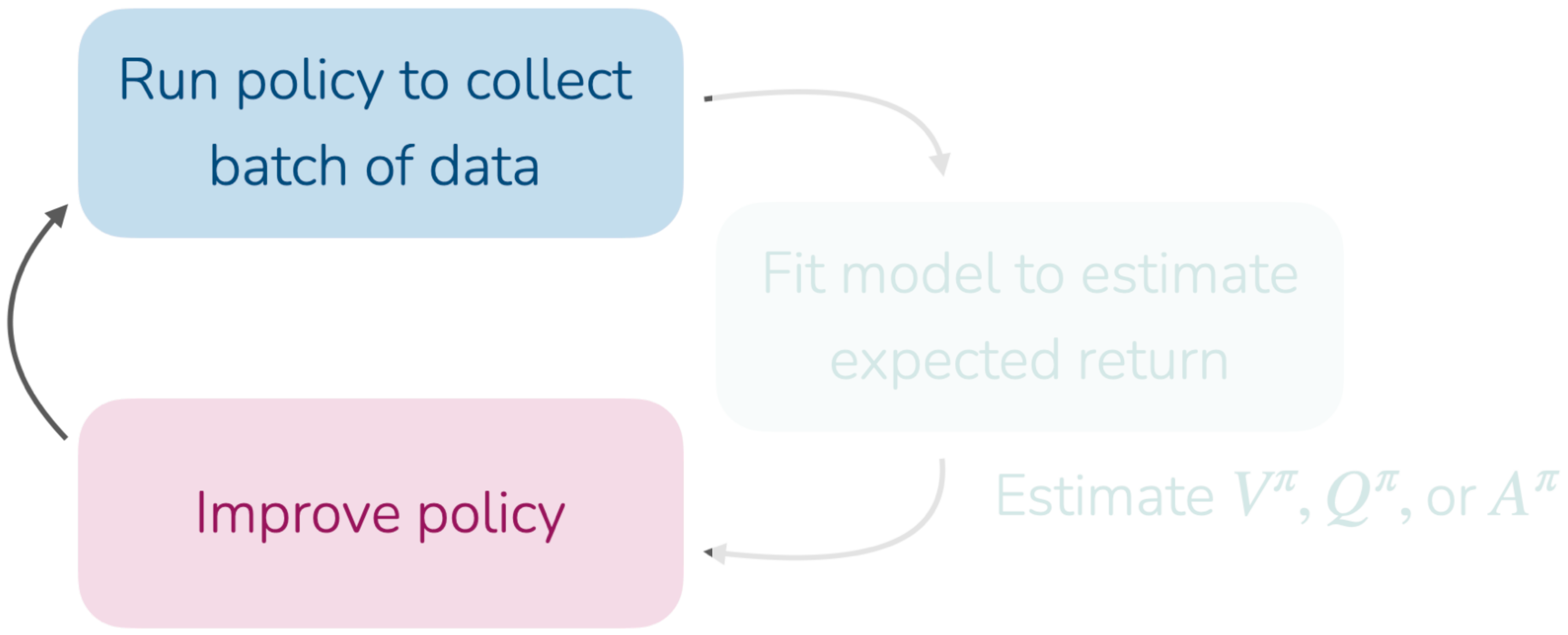
- CA mentors assigned
- Fill out AWS form for GPU quota
- Proposal due Friday

graded fairly lightly- it's for your benefit!

Homework

- Homework 2 due next Friday. (start early!!)

Recap: Methods



$$\theta \leftarrow \theta + \nabla_\theta J(\theta)$$

Online RL with policy gradients

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \underbrace{\sum_{i=1}^{N}}_{\text{samples from policy}} \sum_{t=1}^{T} \nabla_\theta \underbrace{\log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})}_{\text{policy log likelihood}} \left(\underbrace{\left(\sum_{t'=t}^{T} r(\mathbf{s}_{i,t'}, \mathbf{a}_{i,t'}) \right)}_{\text{reward to go}} \underbrace{- b}_{\text{baseline}} \right)$$

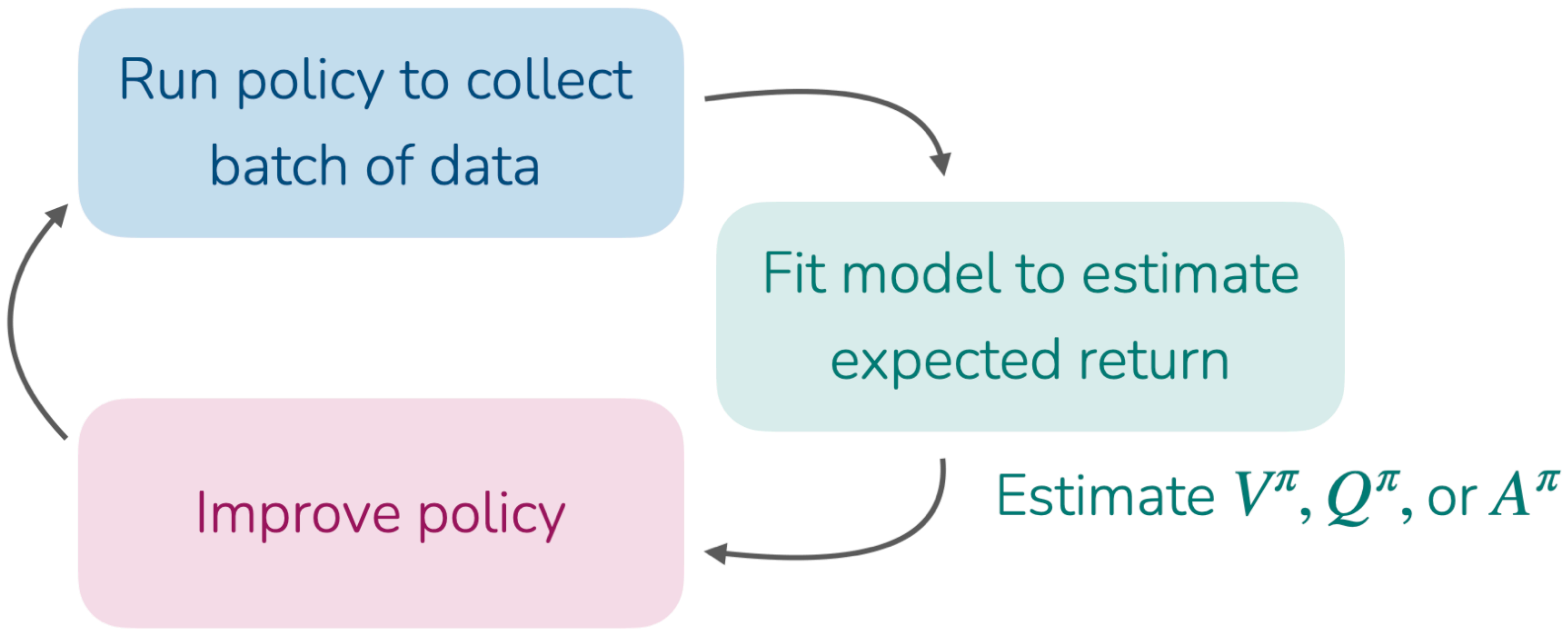
Do more of the above average stuff,
less of the below average stuff.

Online RL with actor-critic

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t}) A^\pi(\mathbf{s}_{i,t}, \mathbf{a}_{i,t})$$

Estimate what is good and bad, then
do more of the good stuff.

Recap: Methods



$\theta \leftarrow \theta + \nabla_\theta J(\theta)$

Online RL with actor-critic

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t}) A^\pi(\mathbf{s}_{i,t}, \mathbf{a}_{i,t})$$

Estimate what is good and bad, then
do more of the good stuff.

1. Estimate V^π with Monte Carlo

$$\min_\phi \sum_{\mathbf{s}_t \sim \mathcal{D}} \|\hat{V}_\phi^{\pi_\theta}(\mathbf{s}_t) - \sum_{t'=t}^{T} r(\mathbf{s}_t, \mathbf{a}_t)\|^2$$

2. Estimate V^π with bootstrapped / TD updates

$$\min_\phi \sum_{(\mathbf{s},\mathbf{a},\mathbf{s}') \sim \mathcal{D}} \|\hat{V}_\phi^{\pi_\theta}(\mathbf{s}) - \left(r(\mathbf{s},\mathbf{a}) + \gamma \hat{V}_\phi^{\pi_\theta}(\mathbf{s}')\right)\|^2$$

3. Estimate Q^π with bootstrapped / TD updates

$$\min_\phi \sum_{(\mathbf{s},\mathbf{a},\mathbf{s}') \sim \mathcal{D}} \|\hat{Q}_\phi^{\pi_\theta}(\mathbf{s},\mathbf{a}) - \left(r(\mathbf{s},\mathbf{a}) + \gamma \mathbb{E}_{\mathbf{a}' \sim \pi_\theta(\cdot|\mathbf{s}')}[\hat{Q}_\phi^{\pi_\theta}(\mathbf{s}',\mathbf{a}')]\right)\|^2$$

Recap: Full Off-Policy Actor-Critic Method

online actor-critic algorithm:

1. take action $\mathbf{a} \sim \pi_\theta(\mathbf{a}|\mathbf{s})$, get $(\mathbf{s}, \mathbf{a}, \mathbf{s}', r)$, store in $\mathcal{R}$
2. sample a batch $\{\mathbf{s}_i, \mathbf{a}_i, r_i, \mathbf{s}'_i\}$ from buffer $\mathcal{R}$
3. update $\hat{Q}^\pi_\phi$ using targets $y_i = r_i + \gamma \hat{Q}^\pi_\phi(\mathbf{s}'_i, \mathbf{a}'_i)$ where $\mathbf{a}'_i \sim \pi_\theta(\cdot|\mathbf{s}'_i)$
4. $\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_i \nabla_\theta \log \pi_\theta(\mathbf{a}^\pi_i|\mathbf{s}_i) \hat{Q}^\pi(\mathbf{s}_i, \mathbf{a}^\pi_i)$ where $\mathbf{a}^\pi_i \sim \pi_\theta(\mathbf{a}|\mathbf{s}_i)$
5. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

The plan for today

Offline RL

1. Why offline RL? Can we just run off-policy methods?
2. Implicit policy constraint methods
3. Conservative methods

} Part of homework 3!

Key learning goals:

- the key challenges arising in offline reinforcement learning
- two approaches for offline RL (& why they work!)
- how offline RL can improve over imitation learning

Why offline RL?

Online RL process (on-policy or off-policy)

Collect data

Update policy on latest data or data so far

Offline RL process

- Given static dataset
- Train policy on provided dataset

Why, or when, might offline RL be more useful?

- leverage datasets collected by people, existing systems
- online policy collection may be risky, unsafe
- reuse previously collected data rather than recollecting
  (e.g. previous experiments, projects, robots, institutions)

Note: A blend of offline then online RL is also possible!

Why offline RL?

Offline RL process

- Given static dataset
- Train policy on provided dataset

More formally:

Offline dataset $\mathcal{D} : \{(\mathbf{s}, \mathbf{a}, \mathbf{s}', r)\}$ sampled from some *unknown policy* π_β

"behavior policy"

$$\mathbf{s} \sim p_{\pi_\beta}(\cdot)$$
$$\mathbf{a} \sim \pi_\beta(\cdot \mid \mathbf{s})$$
$$\mathbf{s}' \sim p(\cdot \mid \mathbf{s}, \mathbf{a})$$
$$r = r(\mathbf{s}, \mathbf{a})$$

(**Note**: π_β may be a mixture of policies)

Objective: $\max_\theta \mathbb{E}_{p_\theta(\tau)} \left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) \right]$ <- expectation under *the learned policy* π_θ

distribution shift

Why offline RL?

Offline RL process

- Given static dataset
- Train policy on provided dataset

Where does the data come from?

- human collected data
- data from a hand-designed system / controller
- data from previous RL run(s)
- a mixture of sources

Can we just use off-policy algorithms?

Recall: Off-policy actor & critic updates (e.g. SAC):

Q-function update: $$\min_\phi \sum_{(\mathbf{s},\mathbf{a},\mathbf{s}')\sim\mathcal{D}} \|\hat{Q}^{\pi_\theta}_\phi(\mathbf{s},\mathbf{a}) - \Big(r(\mathbf{s},\mathbf{a}) + \gamma \mathbb{E}_{\mathbf{a}'\sim\pi_\theta(\cdot|\mathbf{s}')}[\hat{Q}^{\pi_\theta}_\phi(\mathbf{s}',\mathbf{a}')]\Big)\|^2$$

Subsequent policy update: $\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_i \nabla_\theta \log \pi_\theta(\mathbf{a}_i^\pi|\mathbf{s}_i)\hat{Q}^{\pi_\theta}_\phi(\mathbf{s}_i,\mathbf{a}_i^\pi)$ where $\mathbf{a}_i^\pi \sim \pi_\theta(\mathbf{a}|\mathbf{s}_i)$

What happens if you optimize this using a static dataset?

(e.g. say data collected by a mediocre policy)

Can we just use off-policy algorithms?

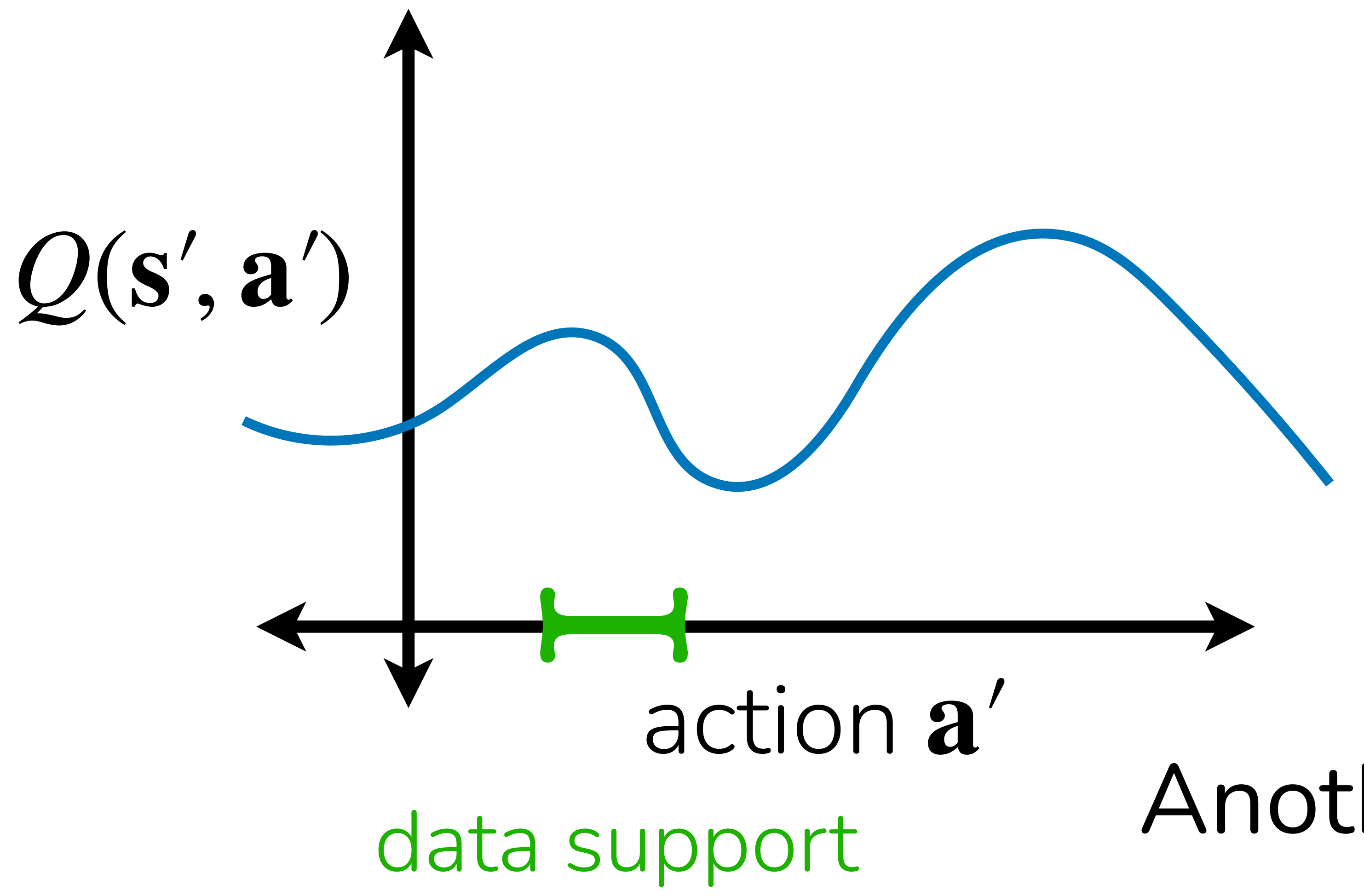
Recall: Off-policy critic objective $\min_\phi \sum_{(\mathbf{s},\mathbf{a},\mathbf{s}')\sim\mathcal{D}} \|\hat{Q}_\phi^{\pi_\theta}(\mathbf{s},\mathbf{a}) - \left(r(\mathbf{s},\mathbf{a}) + \gamma\mathbb{E}_{\mathbf{a}'\sim\pi_\theta(\cdot|\mathbf{s}')}[\hat{Q}_\phi^{\pi_\theta}(\mathbf{s}',\mathbf{a}')]\right)\|^2$

What happens if you optimize this using a static dataset?

(e.g. say data collected by a mediocre policy)

What happens when evaluating Q on actions $\mathbf{a}'$ not in the dataset?

Randomly init. Q-function for state $\mathbf{s}'$



- Q-function will be unreliable on OOD actions
- policy will seek out actions where Q-function is over-optimistic
- After policy update, Q-values will become substantially overestimated.

Another perspective: learned policy deviates too much from behavior policy.

How to mitigate overestimation in offline RL?

This is the **core goal** of offline RL methods!

The plan for today

Offline RL

1. Why offline RL? Can we just run off-policy methods?
2. **Implicit policy constraint methods**
3. Conservative methods

} Part of homework 3!

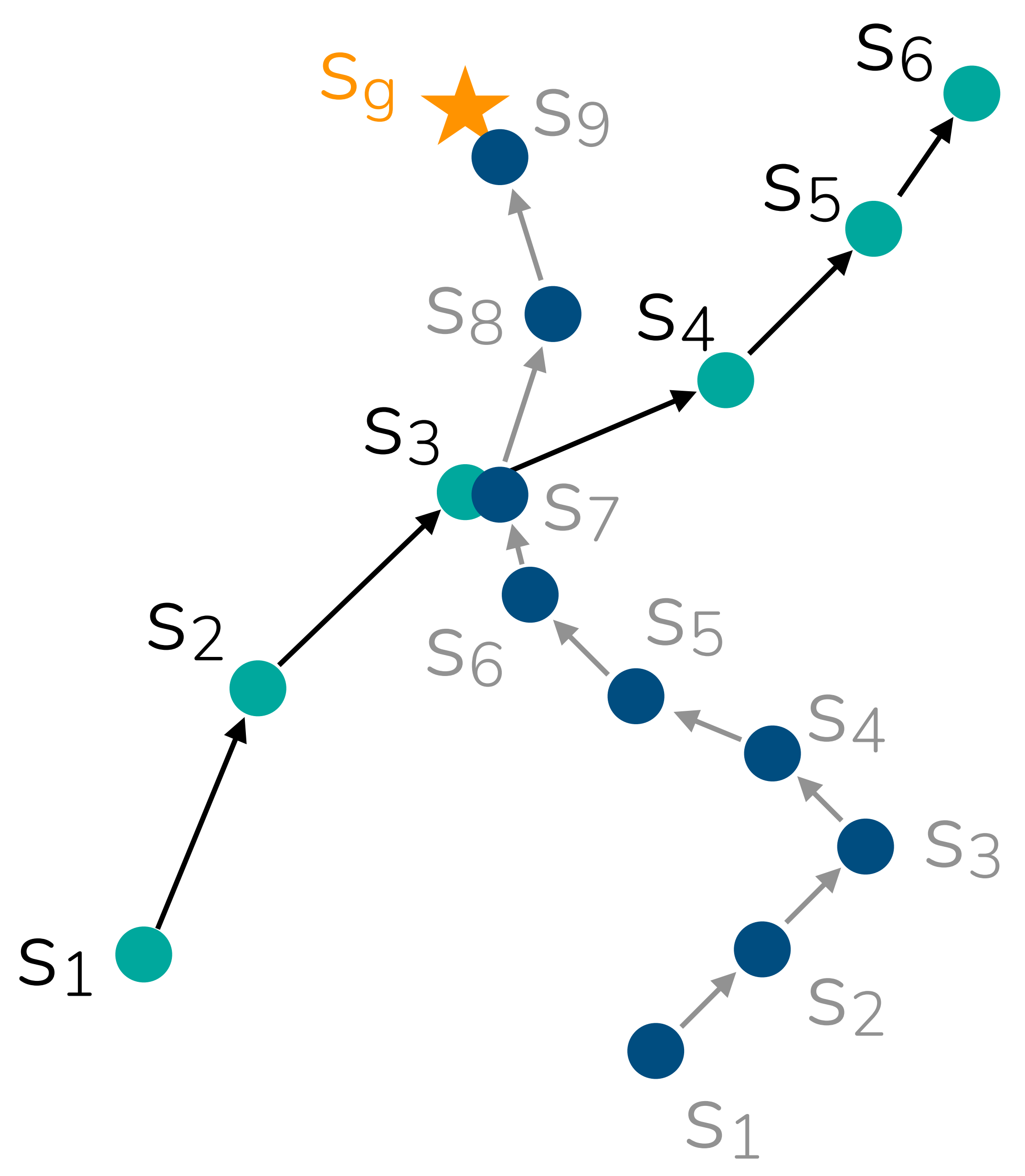
Why offline RL versus imitation learning?

(**Recall**: Imitation methods can't outperform the expert.)

Offline data may not be optimal!

—> Offline RL can leverage reward information to outperform behavior policy.

—> Good offline RL methods can *stitch* together good behaviors.



s_1 -> s_3 is good behavior

s_7 -> s_9 is good behavior

Offline RL methods can learn a policy that goes from s_1 to s_9!

A simple way to leverage rewards in imitation

If we have reward labels: imitate only the good trajectories?

Filtered behavior cloning:

1. Rank trajectories by return $r(\tau) = \sum_{(\mathbf{s}_t, \mathbf{a}_t) \in \tau} r(\mathbf{s}_t, \mathbf{a}_t)$
2. Filter dataset to include top k% of data $\tilde{D} : \{\tau \,|\, r(\tau) > \eta\}$
3. Imitate filtered dataset: $\max_\theta \sum_{(\mathbf{s}, \mathbf{a}) \in \tilde{D}} \log \pi_\theta(\mathbf{a} \,|\, \mathbf{s})$

A very primitive approach to using reward information.

Therefore, a **good baseline** to test against!

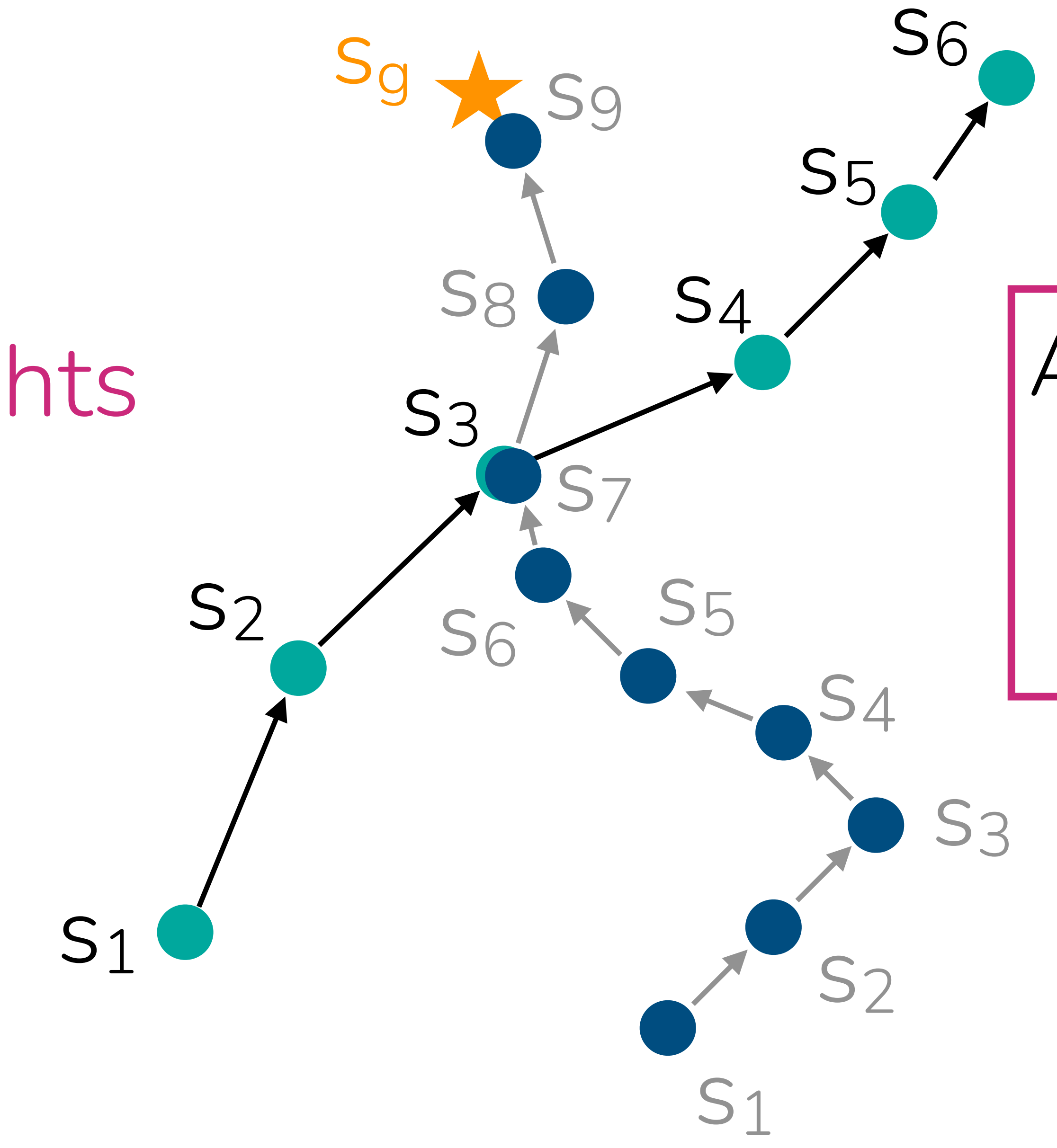
Better way to do weighted imitation learning?

Could we weight each transition depending on **how good the action is**?

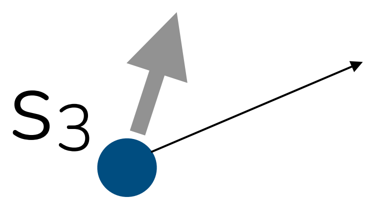
How do you measure how good an action is? Recall: advantage function A

$$\boxed{A^\pi(\mathbf{s}_t, \mathbf{a}_t) = Q^\pi(\mathbf{s}_t, \mathbf{a}_t) - V^\pi(\mathbf{s}_t)\text{: how much better } \mathbf{a}_t \text{ is}}$$

$$\theta \leftarrow \underbrace{\arg\max_\theta E_{\mathbf{s},\mathbf{a}\sim D}\left[\log \pi_\theta(\mathbf{a}\,|\,\mathbf{s})\right.}_{\text{standard imitation learning}}\underbrace{\left.\exp(A(\mathbf{s},\mathbf{a}))\right]}_{\text{with advantage weights}}$$



Adv. weights for s_3



Aside: Can show that advantage-weighted objective approximates KL-constrained objective.

$$\pi_{new} = \arg\max_\pi E_{\mathbf{a}\sim\pi(\cdot|\mathbf{s})} Q(\mathbf{s},\mathbf{a}) \text{ s.t. } D_{KL}(\pi \| \pi_\beta) < \epsilon$$

See Peters et al. (REPS), Rawlik et al. ("psi-learning")

Better way to do weighted imitation learning?

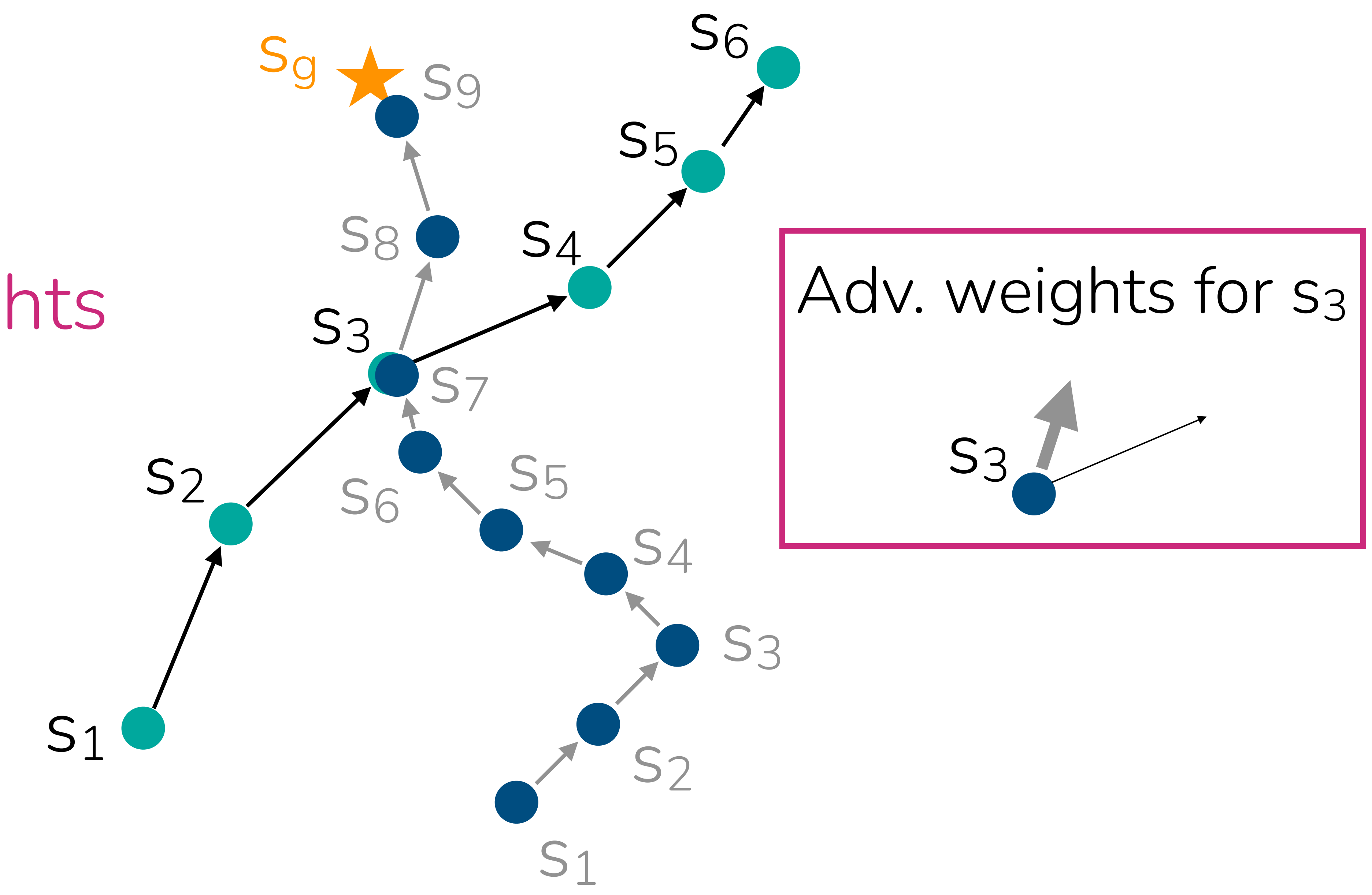
Could we weight each transition depending on **how good the action is**?

How do you measure how good an action is?

Recall: advantage function A

$$\boxed{A^\pi(\mathbf{s}_t, \mathbf{a}_t) = Q^\pi(\mathbf{s}_t, \mathbf{a}_t) - V^\pi(\mathbf{s}_t)\text{: how much better } \mathbf{a}_t \text{ is}}$$

$$\theta \leftarrow \underbrace{\arg\max_\theta E_{\mathbf{s},\mathbf{a}\sim D}\left[\log \pi_\theta(\mathbf{a}\,|\,\mathbf{s})\right.}_{\text{standard imitation learning}}\underbrace{\left.\exp(A(\mathbf{s},\mathbf{a}))\right]}_{\text{with advantage weights}}$$



Advantage of which policy? We'll use A^{π_β} for now.

Key question: How to estimate the advantage function?

Advantage-weighted regression

Could we weight each transition depending on **how good the action is**?

$$\theta \leftarrow \underbrace{\arg\max_{\theta} E_{\mathbf{s},\mathbf{a}\sim D}\left[\log \pi_\theta(\mathbf{a}\,|\,\mathbf{s})\right.}_{\text{standard imitation learning}}\underbrace{\left.\exp(A(\mathbf{s},\mathbf{a}))\right]}_{\text{with advantage weights}}$$

Key question: How to estimate the advantage function?

First simple approach

Estimate $V^{\pi_\beta}(s)$ with Monte Carlo: $\min_{\phi} \sum_{\mathbf{s}_t \sim \mathcal{D}} \|\hat{V}_\phi^{\pi_\beta}(\mathbf{s}_t) - \sum_{t'=t}^{T} r(\mathbf{s}_t, \mathbf{a}_t)\|^2$

empirical return

Approximate $\hat{A}^{\pi_\beta}(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^{T} r(\mathbf{s}_t, \mathbf{a}_t) - \hat{V}_\phi^{\pi_\beta}(\mathbf{s}_t)$

Question: What do you learn for deterministic policy π_β?

Peng, Kumar, Zhang, Levine. Advantage-Weighted Regression. '19

Advantage-weighted regression

Full AWR algorithm

1. Fit value function: $\min_{\phi} \sum_{\mathbf{s}_t \sim \mathcal{D}} \|\hat{V}_{\phi}^{\pi_\beta}(\mathbf{s}_t) - \sum_{t'=t}^{T} r(\mathbf{s}_t, \mathbf{a}_t)\|^2$

2. Train policy: $\max_{\theta} \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t) \sim \mathcal{D}} \left[\log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t) \exp\left(\frac{1}{\alpha} \left(\sum_{t'=t}^{T} r(\mathbf{s}_t, \mathbf{a}_t) - \hat{V}_{\phi}^{\pi_\beta}(\mathbf{s}_t) \right) \right) \right]$

hyperparameter

+ Simple

+ Avoids querying or training on any OOD actions!

- Monte Carlo estimation is noisy

- $\hat{A}^{\pi_\beta}$ is for weaker policy than $\hat{A}^{\pi_\theta}$

Can we do better?

Want to estimate advantages using TD updates, without querying Q on OOD actions.

Estimate Q-function: $$\min_{\psi} E_{(\mathbf{s},\mathbf{a},\mathbf{s}')\sim D}\left[\left(\hat{Q}^{\pi}_{\psi}(\mathbf{s},\mathbf{a}) - \left(r + \gamma E_{\cancel{\mathbf{a}'\sim\pi(\cdot|\mathbf{s})}}[\hat{Q}^{\pi}_{\psi}(\mathbf{s}',\mathbf{a}')]\right)\right)^2\right]$$

$\mathbf{a}' \sim D$

advantage-weighted actor-critic (AWAC)

Note: Gives Q-function estimate for π_β underlying the data, *not* π_θ

Can we do better?

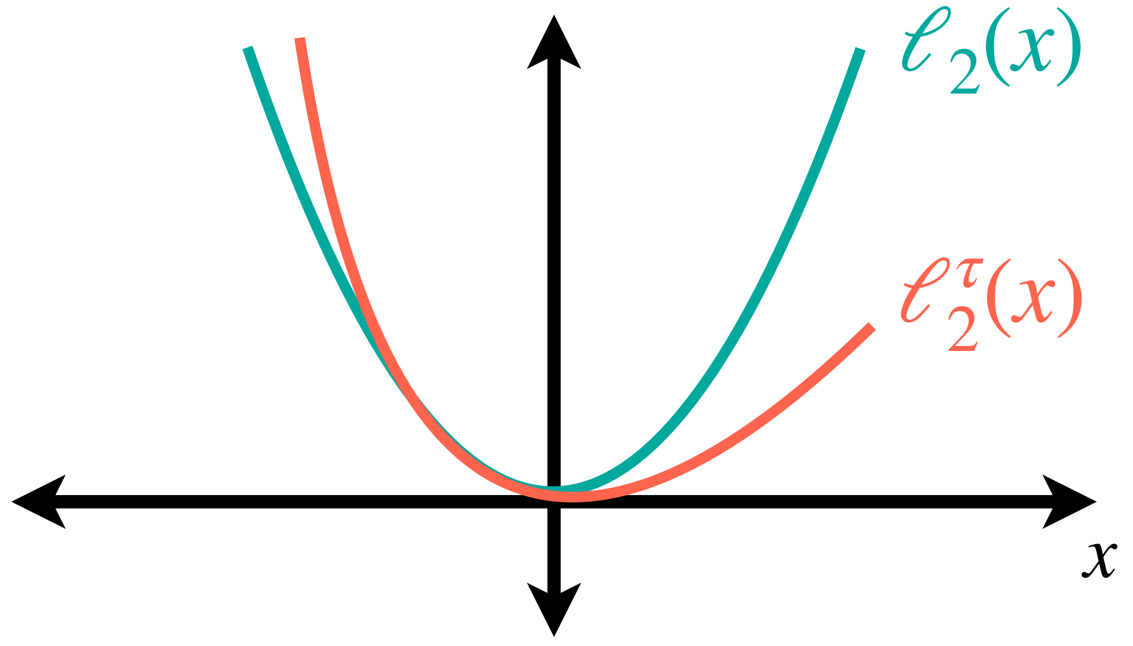
Want to estimate advantages using TD updates, without querying Q on OOD actions.

AWAC update: $\hat{Q}^{\pi_\beta} \leftarrow \arg\min_Q E_{(\mathbf{s},\mathbf{a},\mathbf{s}',\mathbf{a}')\sim D}\left[\left(Q(\mathbf{s},\mathbf{a}) - \left(r + \gamma \underline{Q(\mathbf{s}',\mathbf{a}')}\right)\right)^2\right]$

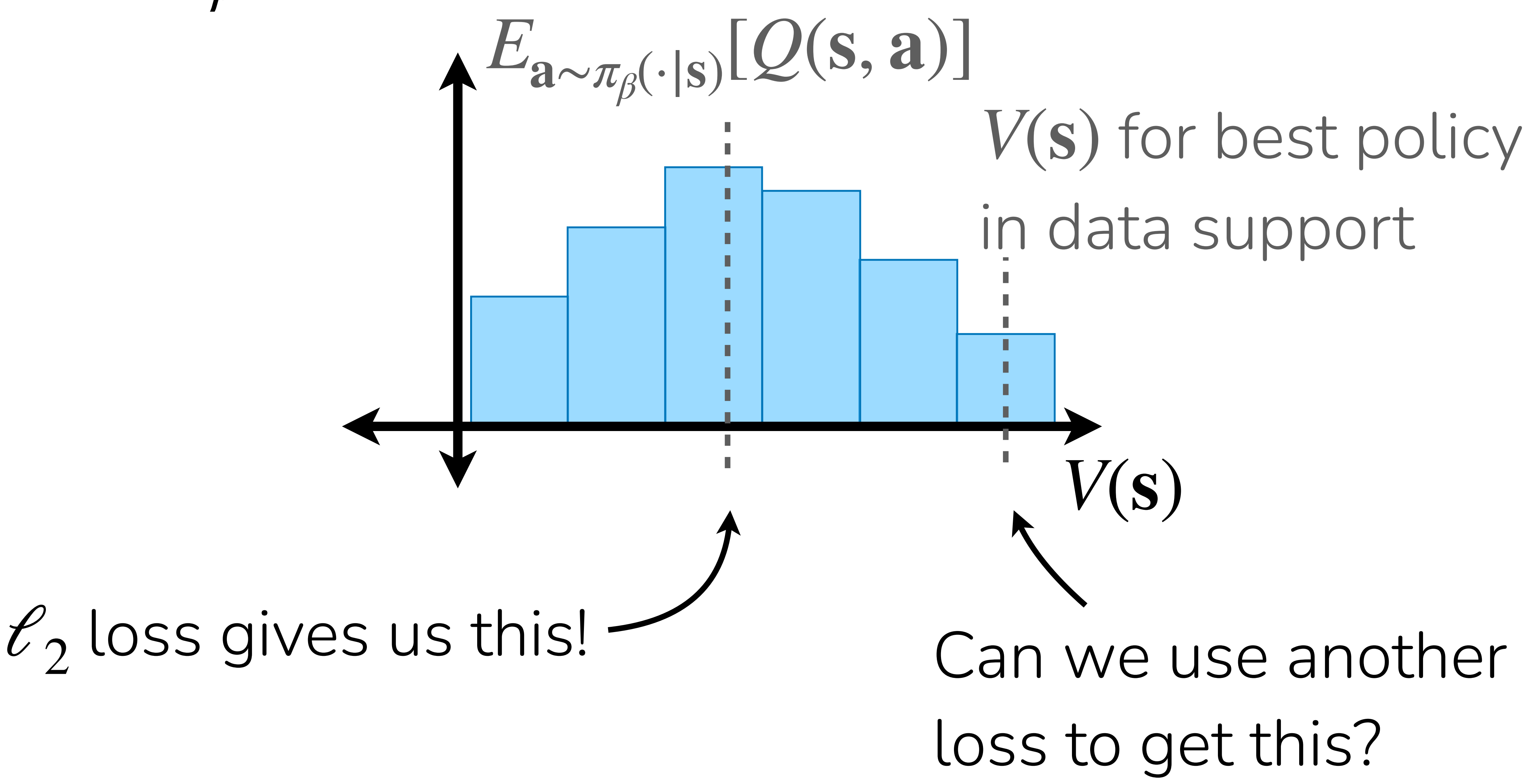
a sample of $V^{\pi_\beta}(\mathbf{s}')$

Can we estimate Q for a policy that is better than π_β?

Idea: Use an asymmetric loss function



Histogram of $V(\mathbf{s})$

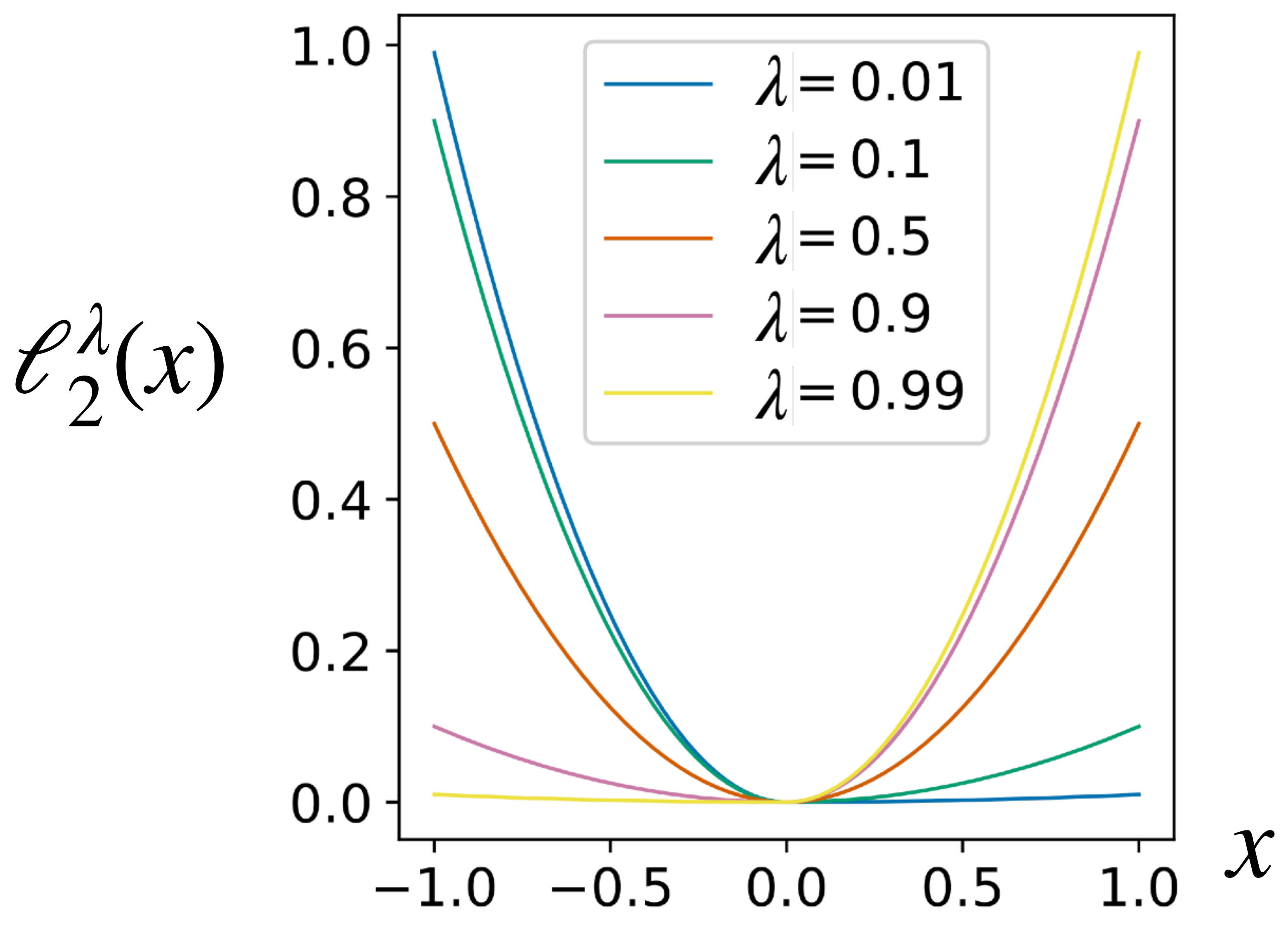


Aside: Expectile regression

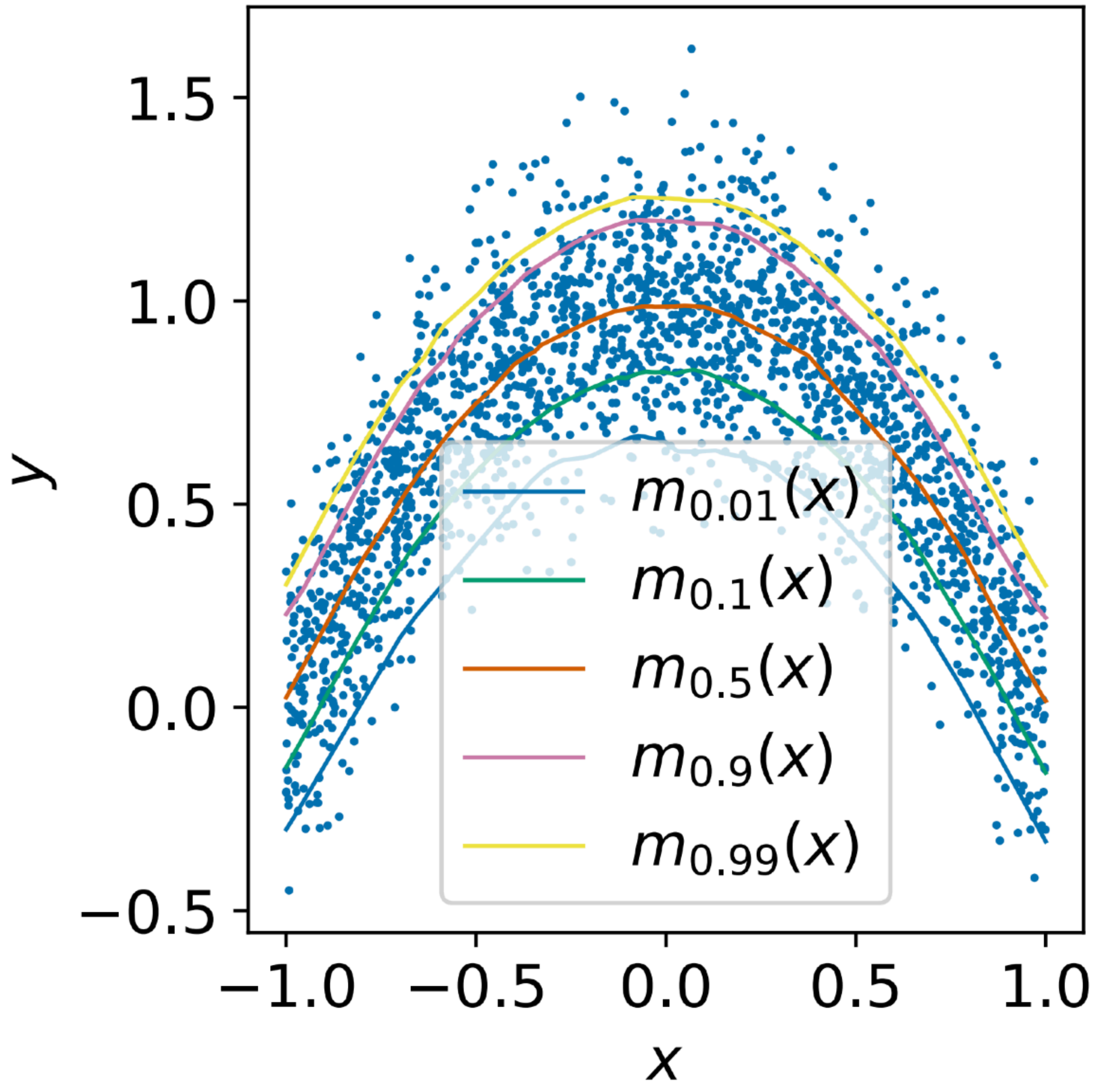
Instead of getting the mean of a random variable, can we get a higher or lower expectile?

Expectile regression loss:

$$\ell_2^\lambda(x) = \begin{cases} (1-\lambda)x^2 & \text{if } x < 0 \\ \lambda x^2 & \text{otherwise} \end{cases}$$



Example with a 2D random variable



Kostrikov, Nair, Levine. Implicit Q-Learning. ICLR '22

Can we do better?

Want to estimate advantages using TD updates, without querying Q on OOD actions.

Full algorithm

Fit V with expectile loss: $\hat{V}(\mathbf{s}) \leftarrow \arg\min_V E_{(\mathbf{s},\mathbf{a})\sim D}\left[\ell_2^\lambda\left(V(\mathbf{s}) - \hat{Q}(\mathbf{s},\mathbf{a})\right)\right]$ using small $\lambda < 0.5$

Update Q with typical MSE loss: $\hat{Q}(\mathbf{s},\mathbf{a}) \leftarrow \arg\min_Q E_{(\mathbf{s},\mathbf{a},\mathbf{s}')\sim D}\left[\left(Q(\mathbf{s},\mathbf{a}) - \left(r + \gamma \hat{V}(\mathbf{s}')\right)\right)^2\right]$

Extract policy with AWR: $\hat{\pi} \leftarrow \arg\max_\pi E_{\mathbf{s},\mathbf{a}\sim D}\left[\log \pi(\mathbf{a}\,|\,\mathbf{s}) \exp\left(\frac{1}{\alpha}\left(\hat{Q}(\mathbf{s},\mathbf{a}) - \hat{V}(\mathbf{s})\right)\right)\right]$

+ Never need to query OOD actions!

+ Policy (still) only trained on actions in data.

+ Decoupling actor & critic training —> computationally fast

policy improvement is implicit

-> implicit Q-learning (IQL)

You will implement IQL in homework 3!

Kostrikov, Nair, Levine. Implicit Q-Learning. ICLR '22

The plan for today

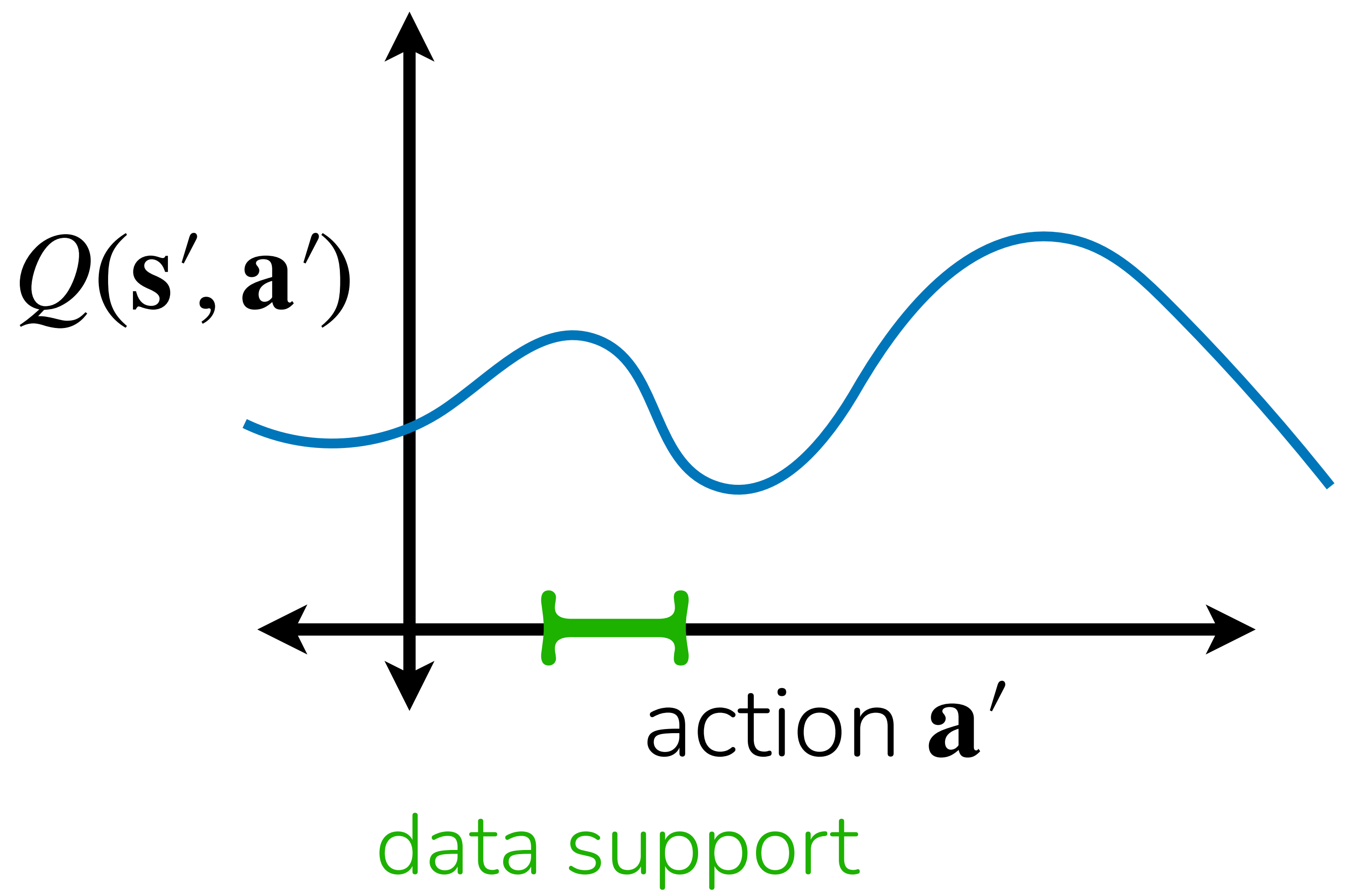
Offline RL

1. Why offline RL? Can we just run off-policy methods?
2. Implicit policy constraint methods
3. **Conservative methods**

Part of homework 3!

How to mitigate overestimation in offline RL?

Recall: Randomly init. Q-function for state $\mathbf{s}'$



Can we discourage overestimation?
without explicitly modeling the behavior policy

What if we just push down on large Q-values?

$$\hat{Q}^{\pi} = \arg\min_{Q} \max_{\mu} E_{(\mathbf{s},\mathbf{a},\mathbf{s}')\sim D}\left[\Big(\underbrace{Q(\mathbf{s},\mathbf{a}) - \big(r(\mathbf{s},\mathbf{a}) + \gamma E_{\pi}[Q(\mathbf{s}',\mathbf{a}')]\big)}_{\text{standard critic update}}\Big)^2\right] + \underbrace{\alpha E_{\mathbf{s}\sim D, \mathbf{a}\sim\mu(\cdot|\mathbf{s})}[Q(\mathbf{s},\mathbf{a})]}_{\text{push down on large Q-values}}$$

Can show that $\hat{Q}^{\pi} \leq Q^{\pi}$ for large enough α

How to mitigate overestimation in offline RL?

Can we discourage overestimation?

without explicitly modeling the behavior policy

$$\hat{Q}^\pi = \arg\min_Q \max_\mu E_{(\mathbf{s},\mathbf{a},\mathbf{s}')\sim D}\left[\left(\underbrace{Q(\mathbf{s},\mathbf{a}) - \left(r(\mathbf{s},\mathbf{a}) + \gamma E_\pi[Q(\mathbf{s}',\mathbf{a}')]\right)}_{\text{standard critic update}}\right)^2\right] + \underbrace{\alpha E_{\mathbf{s}\sim D,\mathbf{a}\sim\mu(\cdot|\mathbf{s})}[Q(\mathbf{s},\mathbf{a})]}_{\text{push down on large Q-values}}$$

$$- \underbrace{\alpha E_{(\mathbf{s},\mathbf{a})\sim D}[Q(\mathbf{s},\mathbf{a})]}_{\text{push up on Q-values for } (\mathbf{s},\mathbf{a}) \text{ in the data}}$$

No longer guaranteed that $\hat{Q}^\pi \leq Q^\pi$ for all $(\mathbf{s},\mathbf{a})$.

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

Conservative Q-learning (CQL)

Slide adapted from Sergey Levine

How to mitigate overestimation in offline RL?

Conservative Q-learning (CQL)

Full algorithm

1. Update $\hat{Q}^\pi$ using L_{CQL} using D

2. Update policy π

If actions are discrete: $\pi(\mathbf{a} \mid \mathbf{s}) = \begin{cases} 1 \text{ if } \mathbf{a} = \arg\max_{\bar{\mathbf{a}}} \hat{Q}^\pi(\mathbf{s}, \bar{\mathbf{a}}) \\ 0 \text{ otherwise} \end{cases}$

If actions are continuous: $\theta \leftarrow \theta + \eta \nabla_\theta E_{\mathbf{s} \sim D, \mathbf{a} \sim \pi_\theta(\cdot|\mathbf{s})} \left[\hat{Q}^\pi(\mathbf{s}, \mathbf{a}) \right]$

How to mitigate overestimation in offline RL?

Conservative Q-learning (CQL)

1. Update $\hat{Q}^{\pi}$ using L_{CQL} using D

2. Update policy π

How compute objective L_{CQL}?

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

$$- \alpha E_{(\mathbf{s},\mathbf{a})\sim D}[Q(\mathbf{s},\mathbf{a})] + \underbrace{R(\mu)}_{\text{regularizer}}$$

Common choice: $R(\mu) = E_{\mathbf{s}\sim D}[\mathcal{H}(\mu(\cdot|\mathbf{s}))]$

With max entropy regularizer R, optimal $\mu(\mathbf{a}\,|\,\mathbf{s}) \propto \exp\left(Q(\mathbf{s},\mathbf{a})\right)$

Then: $\underline{E_{\mathbf{s}\sim D,\mathbf{a}\sim\mu(\cdot|\mathbf{s})}[Q(\mathbf{s},\mathbf{a})]} = \log \sum_{\mathbf{a}} \exp\left(Q(\mathbf{s},\mathbf{a})\right)$

Don't need to construct μ directly.

You will implement CQL in homework 3!

Slide adapted from Sergey Levine

The plan for today

Offline RL

1. Why offline RL? Can we just run off-policy methods?
2. Implicit policy constraint methods
3. Conservative methods

Part of homework 3!

Summary

Why offline RL? Online data is expensive. *Reusing offline data* is good!

Key challenge: Overestimating Q-values because of shift between π_β and π_θ

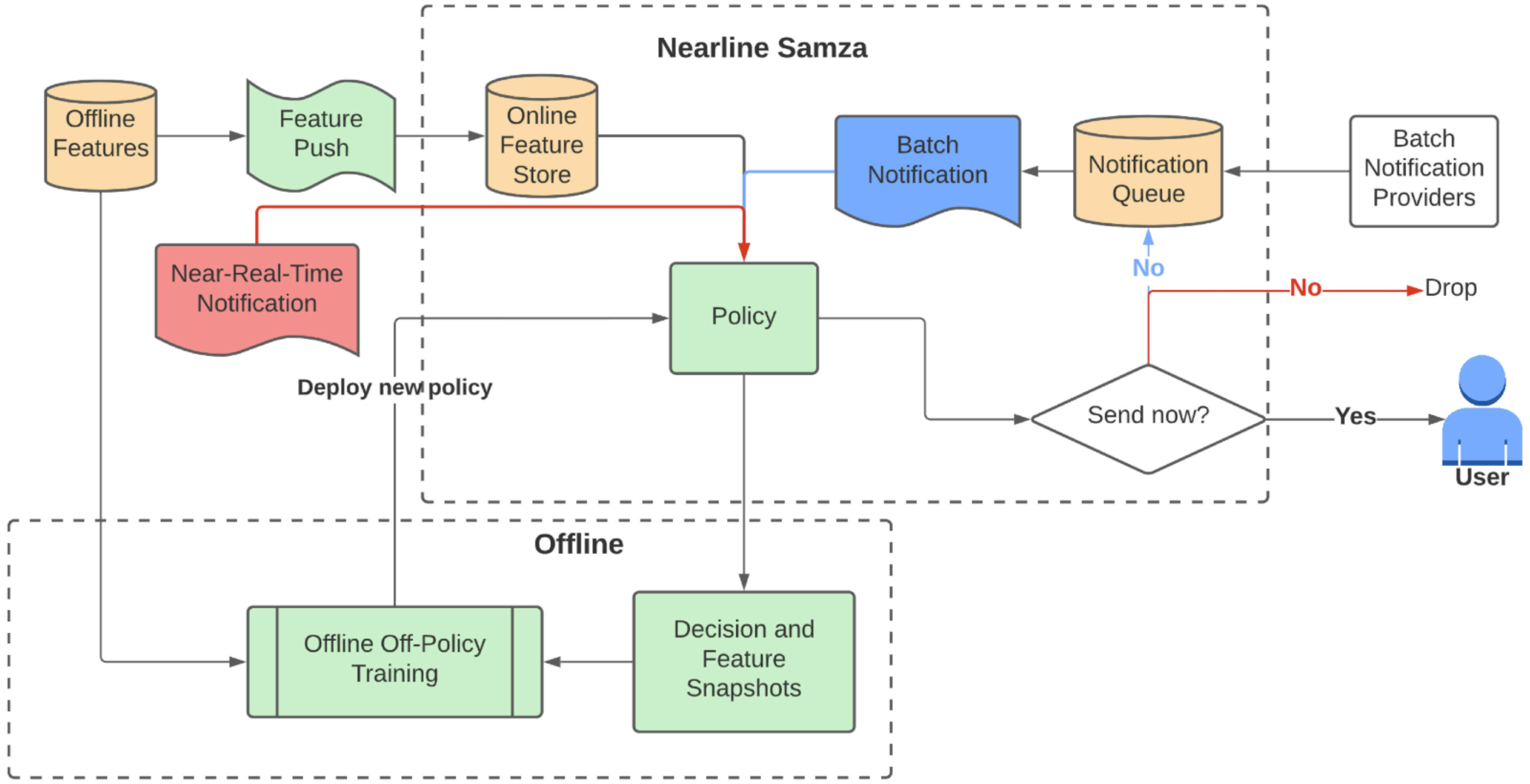
Approaches:

1. filtered or weighted imitation learning is a simple baseline
2. **implicitly constrain** the policy to π_β by only supervising on actions in data
3. construct conservative objective by **penalizing Q-values**

Trajectory stitching allows offline RL methods to improve over imitation.

An example application

Optimizing policy for sending notifications to users on LinkedIn



WAU: weekly active users
Volume: total # of notifications

CTR: click-through-rate of notifications

Metric	DDQN vs. Baseline	DDQN + CQL vs. Baseline
Sessions	not stat sig	+ 0.24%
WAU	-0.69%	+ 0.18%
Volume	+7.72%	-1.73%
CTR	-7.79%	+2.26%

Table 1: Online A/B test results for DDQN with and without CQL

Prabhakar, Yuan, Yang, Sun, Muralidharan. Multi-Objective Optimization of Notifications Using Offline RL. ‘22

Which offline RL algorithm to use?

If you only want to train offline:

- **Filtered behavior cloning**: Good first approach to using offline data.
- **Implicit Q-learning**: Can stitch data & explicitly constrained to data support
- **Conservative Q-learning**: Just one hyperparameter

If you want offline pre-training + online fine-tuning:

- **Implicit Q-learning**: Seems most performant.

See also: IDQL (Hansen-Estruch et al. IDQL: Implicit Q-Learning as an Actor-Critic Method with Diffusion Policies. 2023)

Note: Still an active area of research!

Next time

Friday lecture: How do we get rewards??

Course reminders

Project

- CA mentors assigned
- Fill out AWS form for GPU quota
- Proposal due Friday

  graded fairly lightly, for your benefit!

Homework

- Homework 2 due next Friday.
  (start early!!)