



Computer Engineering Department

Offline Reinforcement Learning

Mohammad Hossein Rohban, Ph.D.

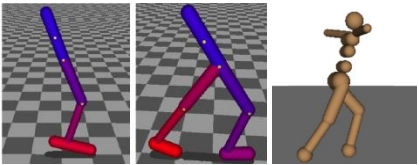
Spring 2025

Slides are adopted from CS 285, UC Berkeley.

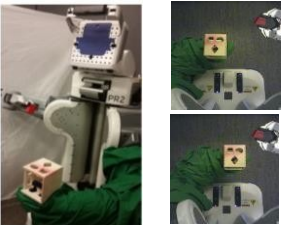
The generalization gap



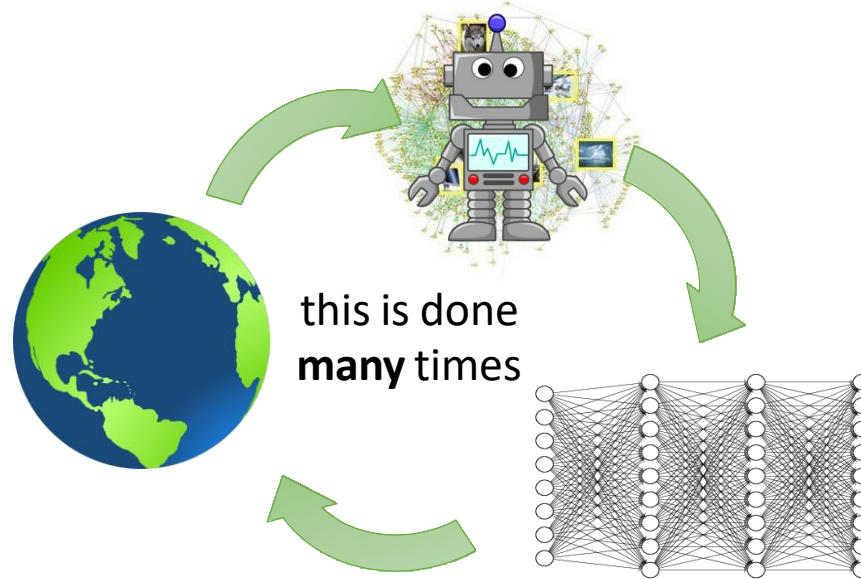
Mnih et al. '13



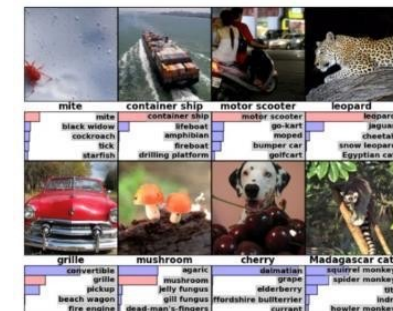
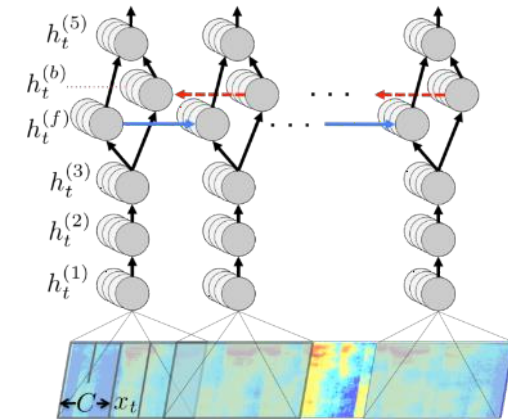
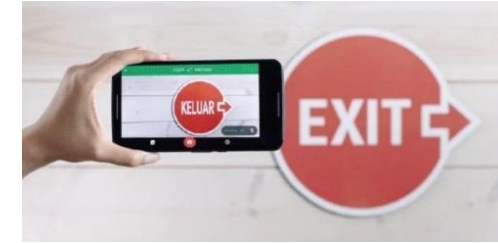
Schulman et al. '14 & '15



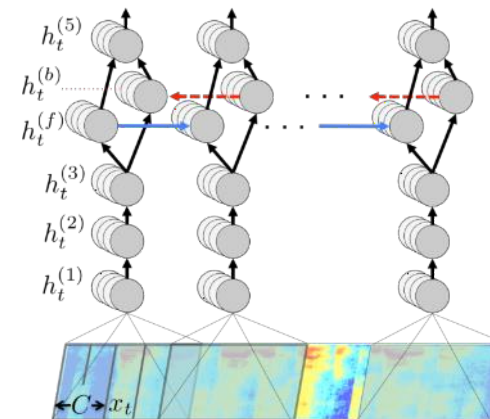
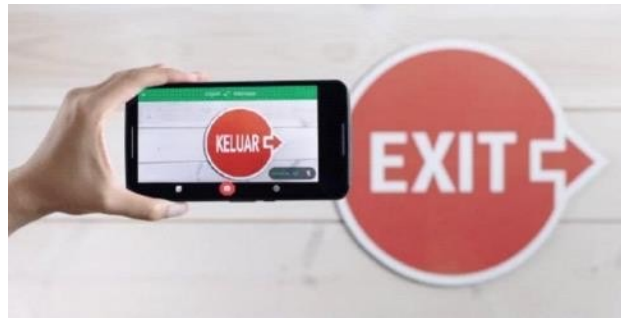
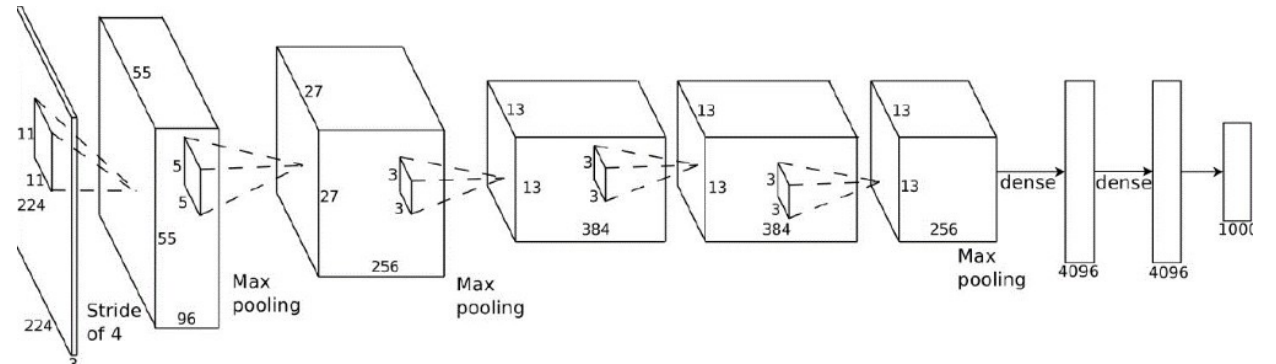
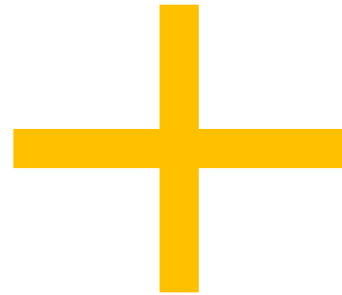
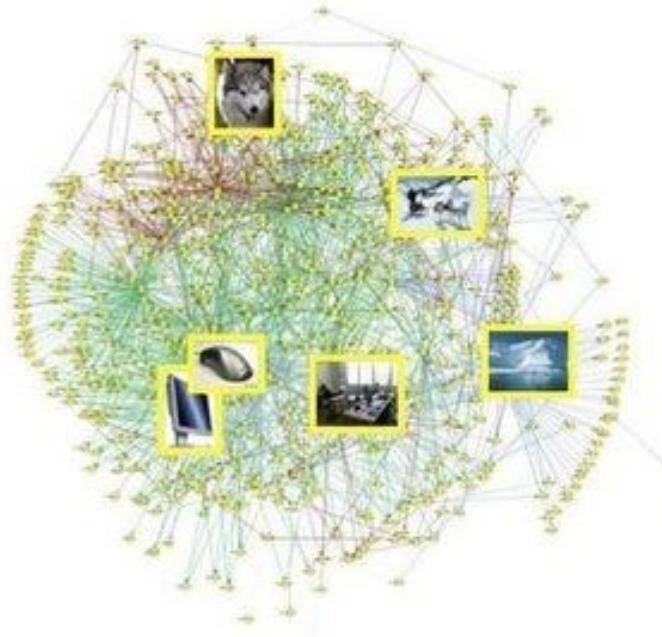
Levine*, Finn*, et al. '16



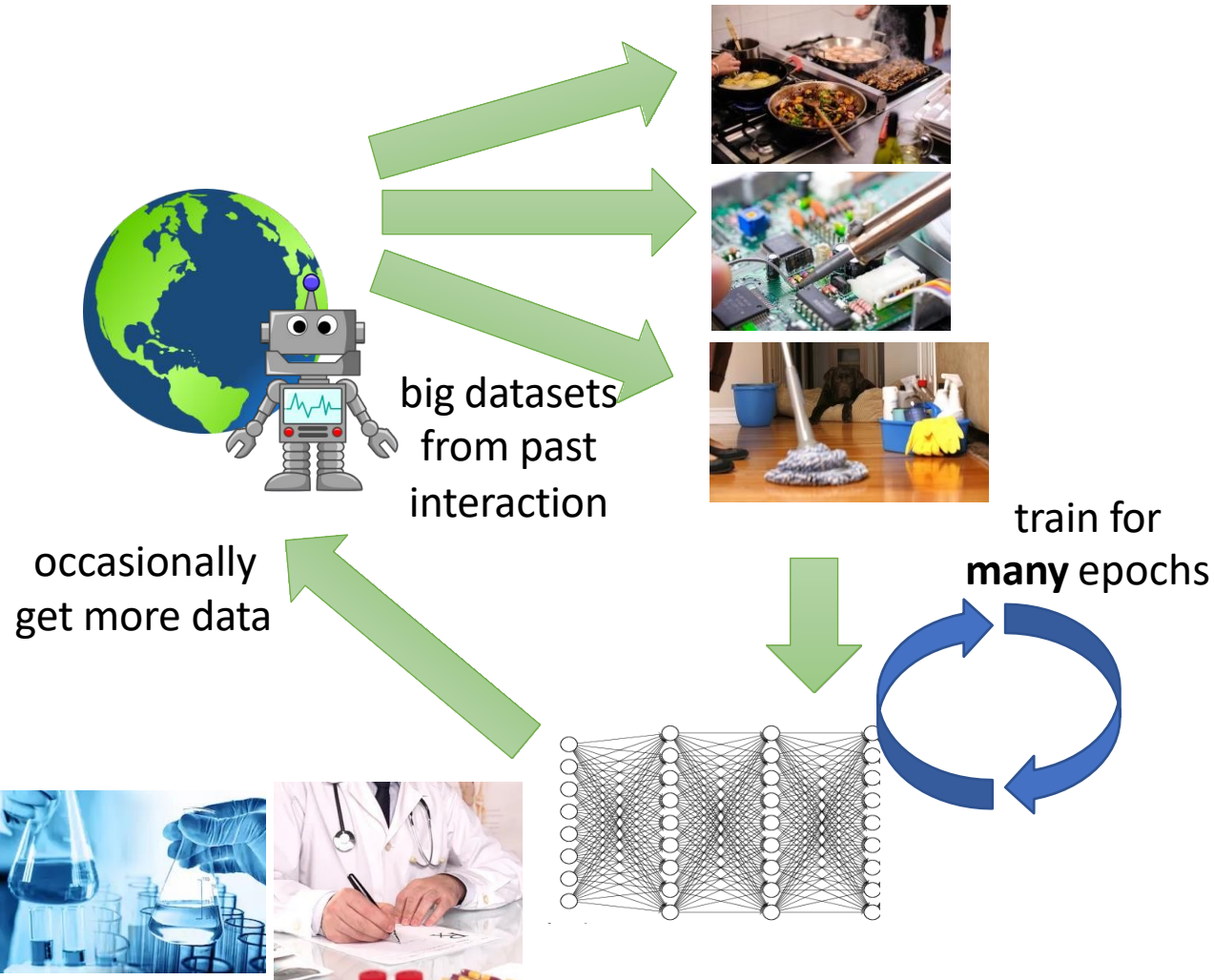
enormous gulf



What makes modern machine learning work?

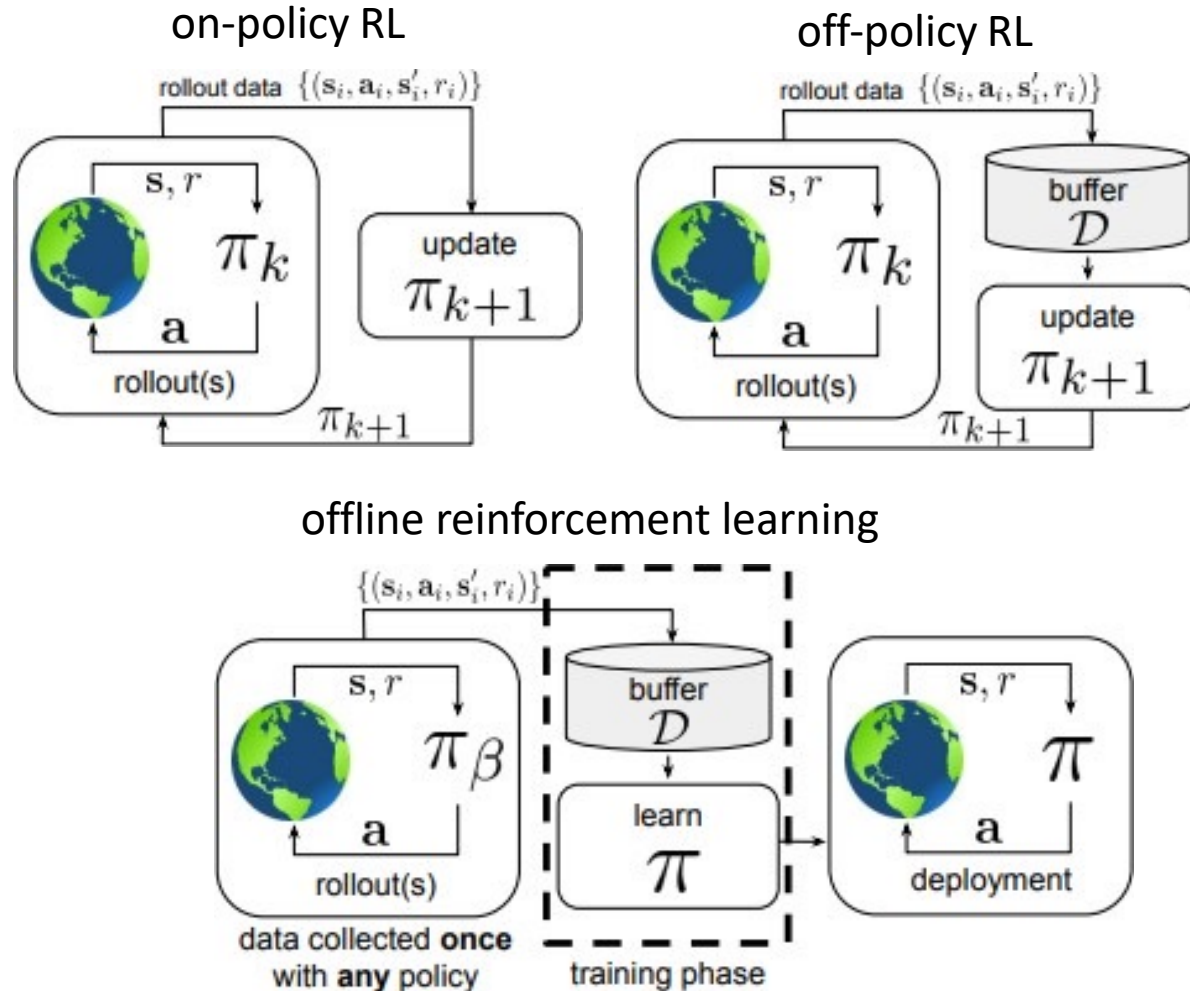


Can we develop data-driven RL methods?



Levine, Kumar, Tucker, Fu. **Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems.** '20

What does offline RL mean?



Formally:

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

$$s \sim d^{\pi_\beta}(s)$$

$$a \sim \pi_\beta(a|s)$$

$$s' \sim p(s'|s, a)$$

$$r \leftarrow r(s, a)$$

generally **not** known

$$\text{RL objective: } \max_{\pi} \sum_{t=0}^T E_{s_t \sim d^{\pi}(s), a_t \sim \pi(a|s)} [\gamma^t r(s_t, a_t)]$$

Types of offline RL problems

off-policy evaluation (OPE):

given \mathcal{D} , estimate $J(\pi) = E_{\pi} \left[\sum_{t=1}^T r(\mathbf{s}_t, \mathbf{a}_t) \right]$

$$\mathcal{D} = \{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)\}$$

$$\mathbf{s} \sim d^{\pi_{\beta}}(\mathbf{s})$$

$$\mathbf{a} \sim \pi_{\beta}(\mathbf{a}|\mathbf{s})$$

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

$$r \leftarrow r(\mathbf{s}, \mathbf{a})$$

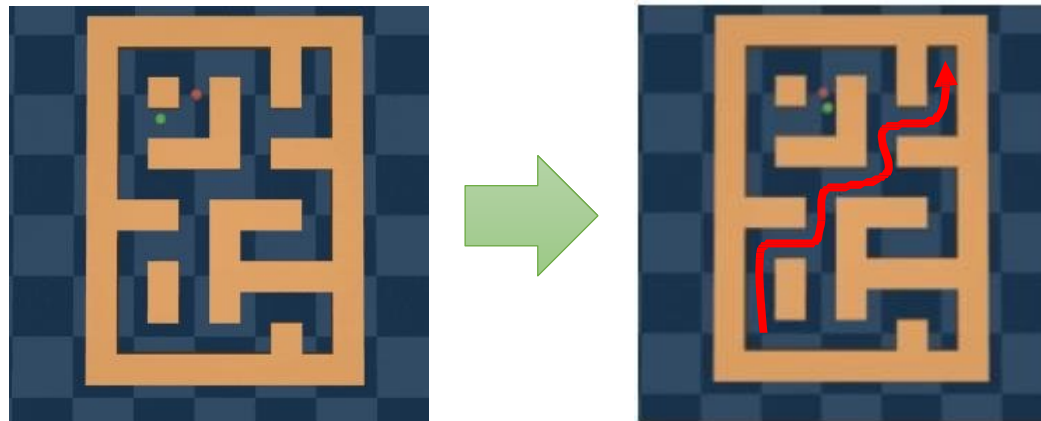
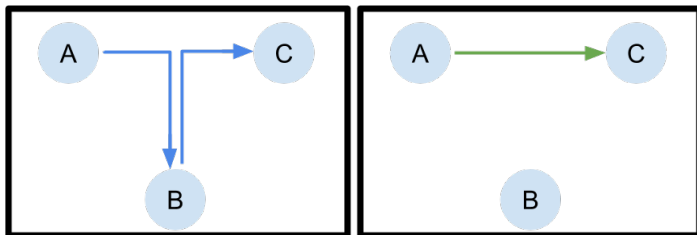
offline reinforcement learning: (a.k.a. batch RL, sometimes fully off-policy RL)

given \mathcal{D} , learn the best possible policy π_{θ}

not necessarily obvious what this means

How is this even possible?

1. Find the “good stuff” in a dataset full of good and bad behaviors
2. Generalization: good behavior in one place may suggest good behavior in another place
3. “Stitching”: parts of good behaviors can be recombined

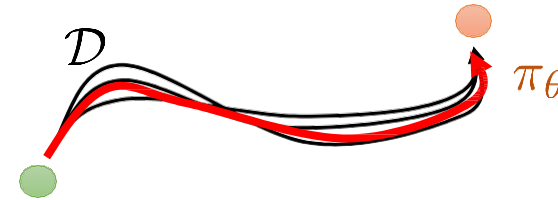


What do we expect offline RL methods to do?

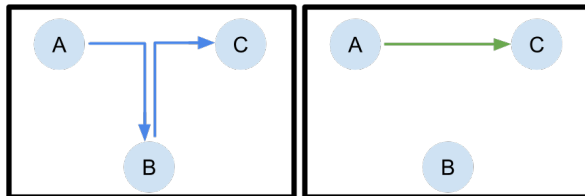
Bad intuition: it's like imitation learning

Though it can be shown to be **provably** better than imitation learning even with optimal data, under some structural assumptions!

See: Kumar, Hong, Singh, Levine. Should I Run Offline Reinforcement Learning or Behavioral Cloning?

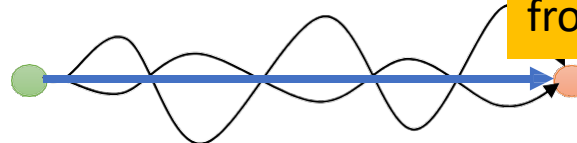


Better intuition: get order from chaos

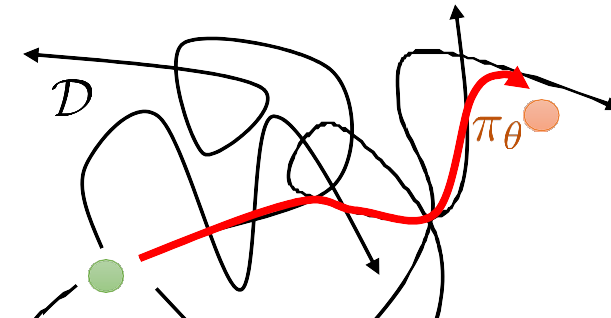


But this is just the clearest example!

“Micro-scale” stitching:



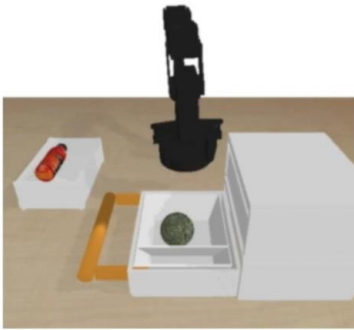
“Macro-scale” stitching



If we have algorithms that properly perform dynamic programming, we can take this idea much further and get near-optimal policies from highly suboptimal data

A vivid example

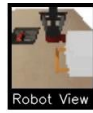
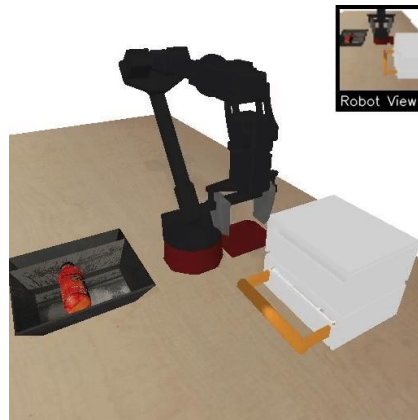
RL policies typically don't generalize to initial conditions that were not seen during training



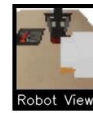
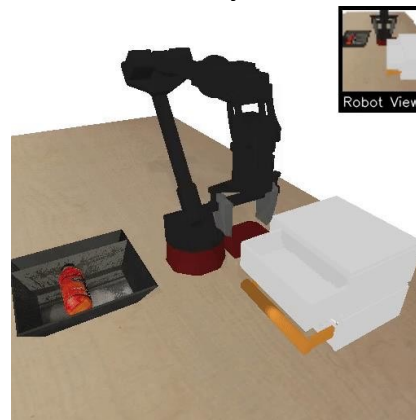
Training time

training task

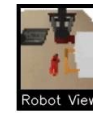
closed drawer



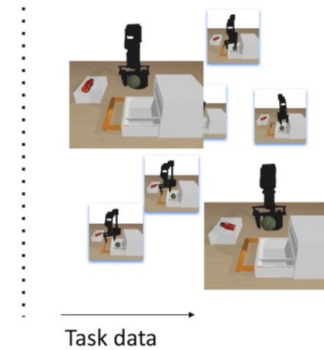
blocked by drawer



blocked by object

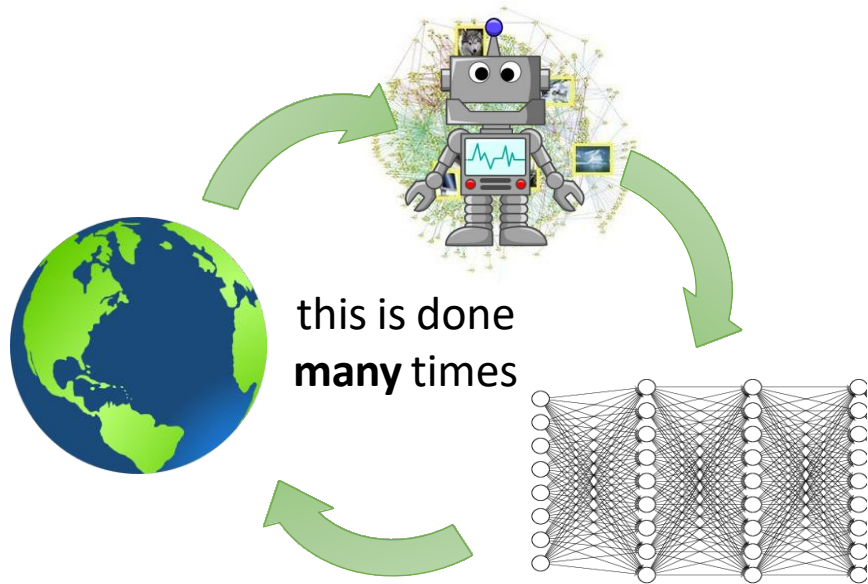


Can we use previously collected, unlabeled datasets to extend learned skills?

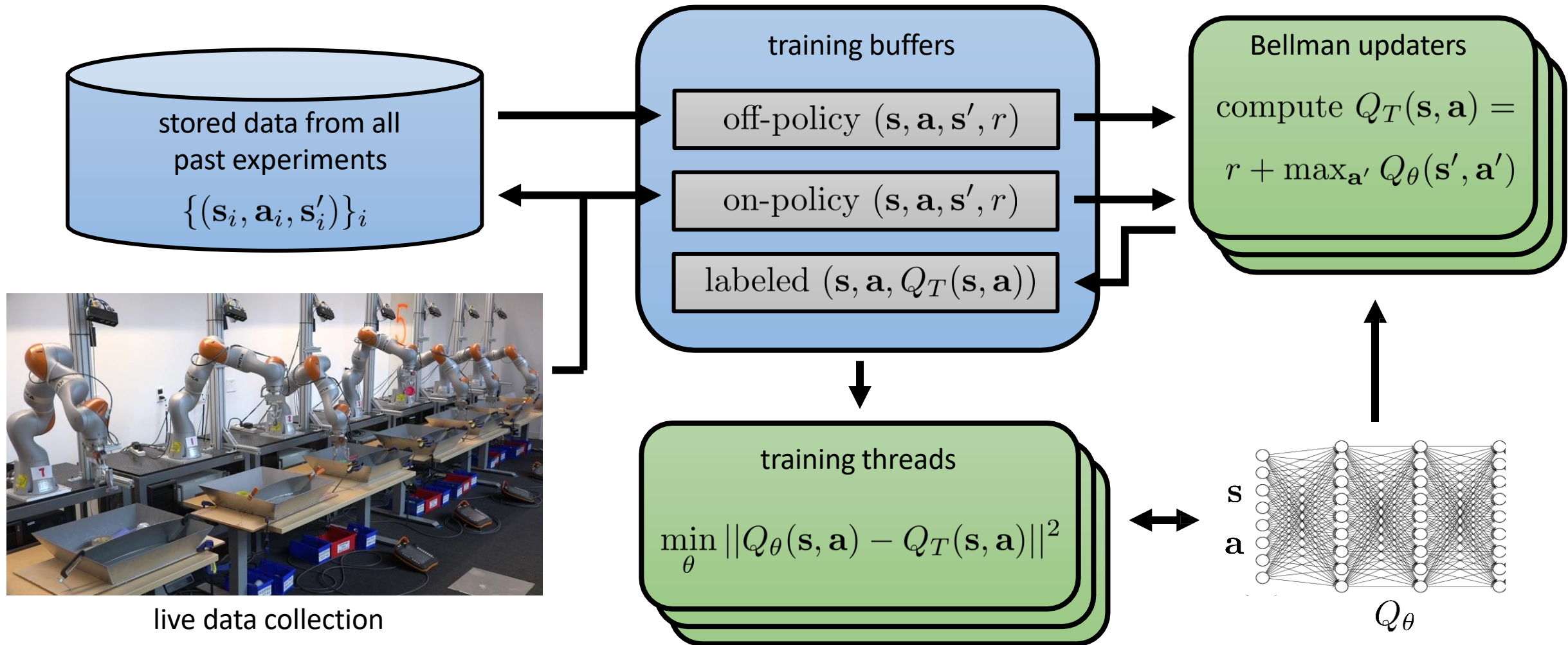


Task data

Why should we care?



Does it work?

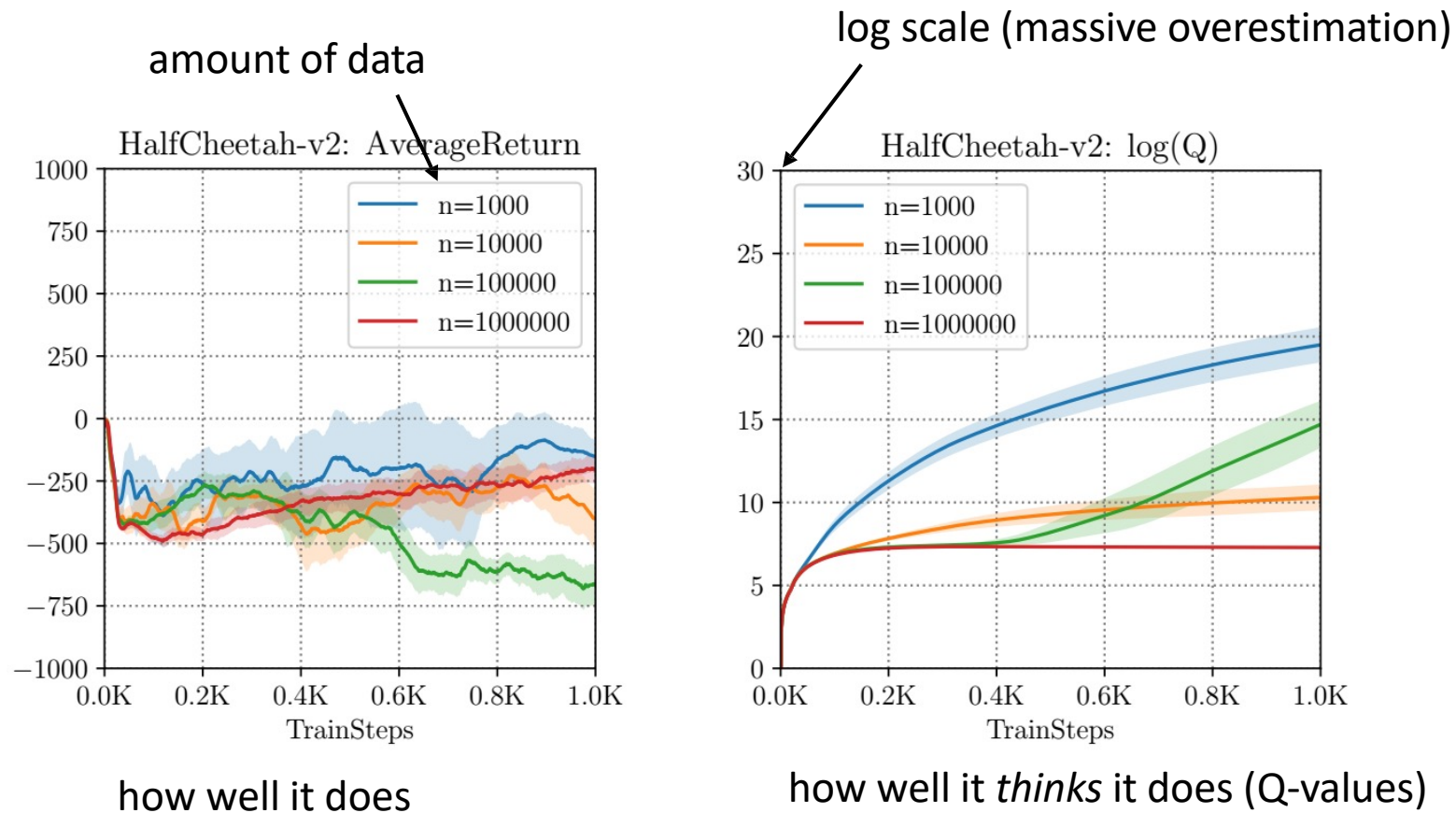


Does it work?



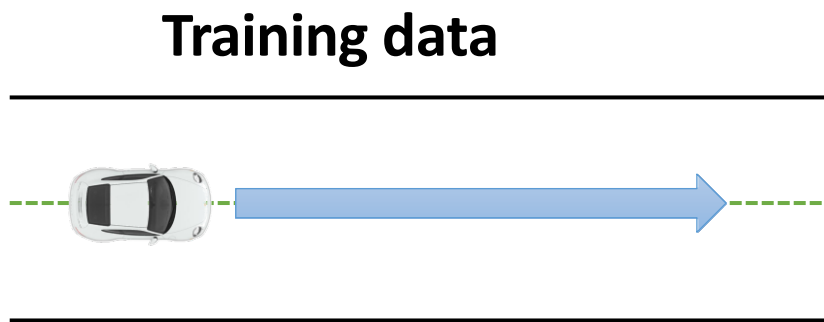
Method	Dataset	Success	Failure
Offline QT-Opt	580k offline	87%	13%
Finetuned QT-Opt	580k offline + 28k online	96%	4%

Why is offline RL hard?

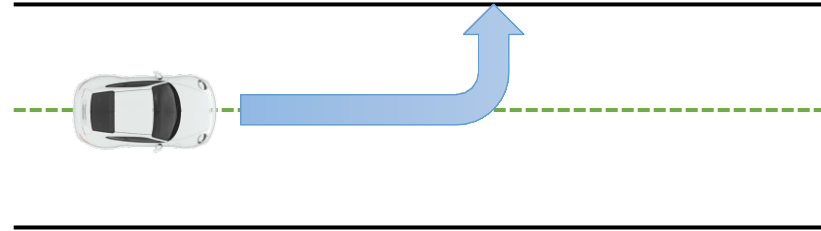


Why is offline RL hard?

Fundamental problem: counterfactual queries



What the policy wants to do



Is this good? Bad?
How do we know if
we didn't see it in
the data?

Online RL algorithms don't have to handle this, because they can simply **try** this action and see what happens

Offline RL methods must somehow account for these unseen ("out-of-distribution") actions, ideally in a safe way

...while still making use of generalization to come up with behaviors that are better than the best thing seen in the data!

Distribution shift in a nutshell

Example empirical risk minimization (ERM) problem:

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

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

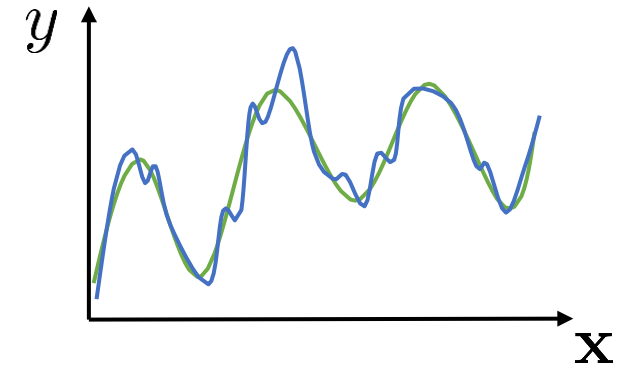
$E_{\mathbf{x} \sim p(\mathbf{x}), y \sim p(y|\mathbf{x})} [(f_{\theta}(\mathbf{x}) - y)^2]$ is low

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

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

usually we are not worried – neural nets generalize well!

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



Where do we suffer from distribution shift?

$$\cancel{Q(\mathbf{s}, \mathbf{a}) \leftarrow r(\mathbf{s}, \mathbf{a}) + \max_{\mathbf{a}'} Q(\mathbf{s}', \mathbf{a}')}$$

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

what is the objective?

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

behavior policy

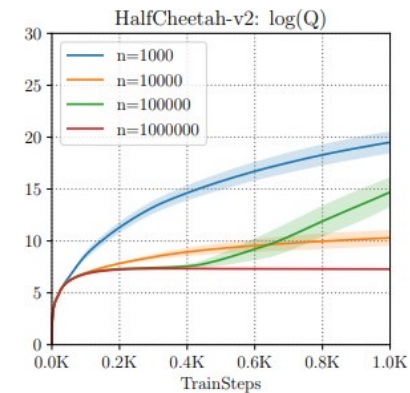
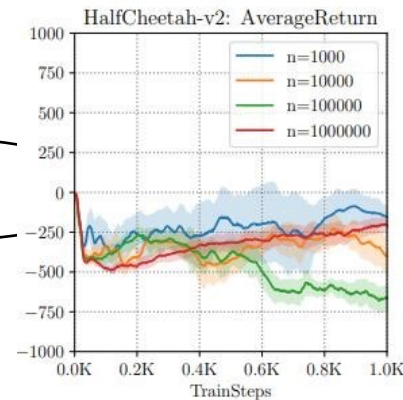
target value

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

how often does *that* happen?

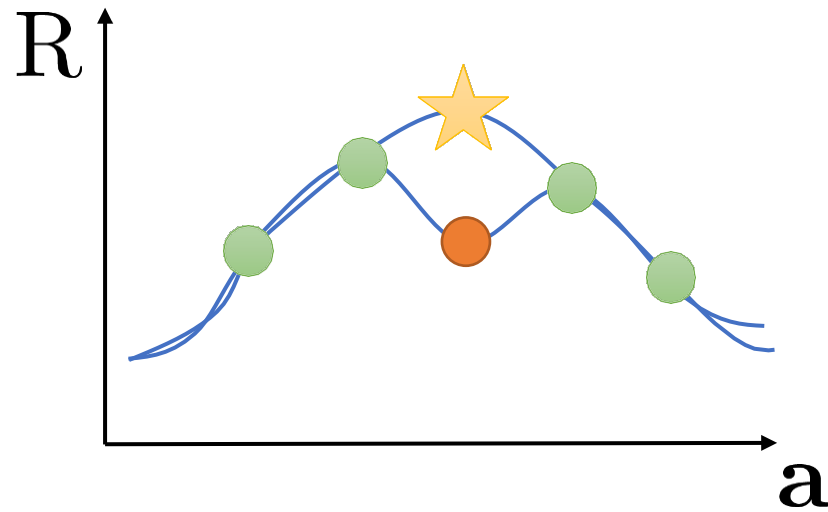
even *worse*: $\pi_{\text{new}} = \arg \max_{\pi} E_{\mathbf{a} \sim \pi(\mathbf{a}|\mathbf{s})}[Q(\mathbf{s}, \mathbf{a})]$

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

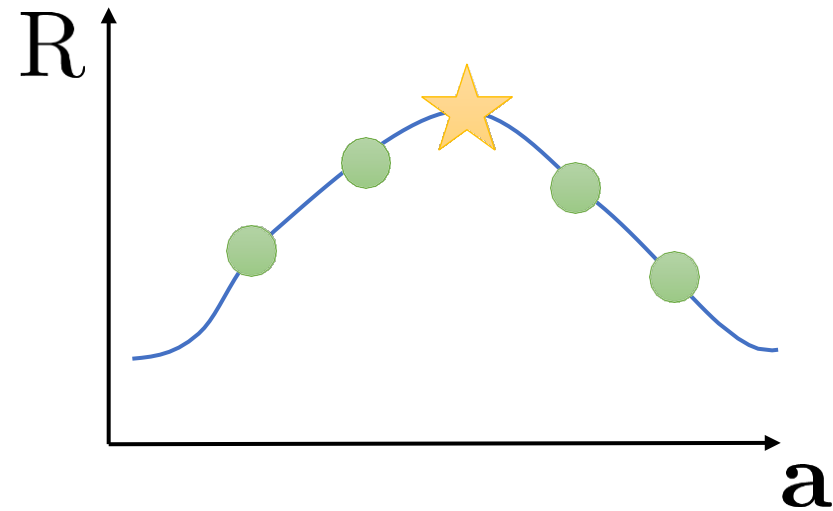


Issues with generalization are not corrected

online RL setting



offline RL setting




Existing challenges with sampling error and function approximation error in standard RL become **much more severe** in offline RL

Offline RL Solutions

Policy Constraint Methods

How do prior methods address this?


$$Q(\mathbf{s}, \mathbf{a}) \leftarrow r(\mathbf{s}, \mathbf{a}) + E_{\mathbf{a}' \sim \pi_{\text{new}}} [Q(\mathbf{s}', \mathbf{a}')] \\ \pi_{\text{new}}(\mathbf{a}|\mathbf{s}) = \arg \max_{\pi} E_{\mathbf{a} \sim \pi(\mathbf{a}|\mathbf{s})} [Q(\mathbf{s}, \mathbf{a})] \text{ s.t. } D_{\text{KL}}(\pi \| \pi_{\beta}) \leq \epsilon$$

This solves distribution shift, right?

No more erroneous values?

Issue 1: we usually don't know the behavior policy $\pi_{\beta}(\mathbf{a}|\mathbf{s})$

- human-provided data
- data from hand-designed controller
- data from many past RL runs
- all of the above

Issue 2: this is both *too pessimistic* and *not pessimistic enough*

“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...”)

Explicit policy constraint methods

What kinds of constraints can we use?

KL-divergence: $D_{\text{KL}}(\pi \parallel \pi_\beta)$

+ easy to implement (more on this later)

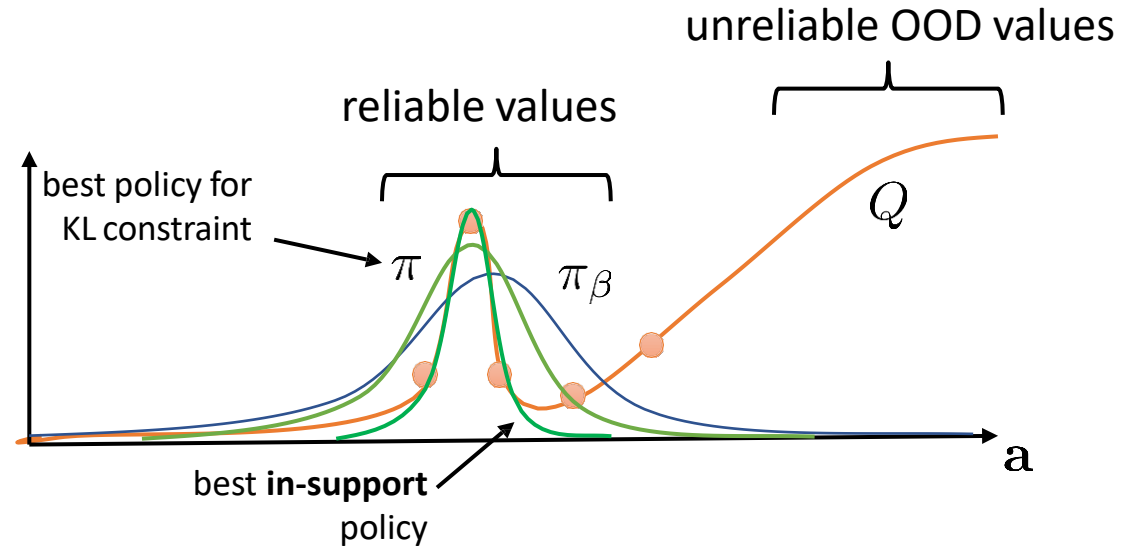
- not necessarily what we want

support constraint: $\pi(\mathbf{a}|\mathbf{s}) \geq 0$ only if $\pi_\beta(\mathbf{a}|\mathbf{s}) \geq \epsilon$

can approximate with MMD

- significantly more complex to implement

+ much closer to what we really want



For more information, see:

Levine, Kumar, Tucker, Fu. **Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems.** '20

Kumar, Fu, Tucker, Levine. **Stabilizing Off-Policy Q-Learning via Bootstrapping Error Reduction.** '19

Wu, Tucker, Nachum. **Behavior Regularized Offline Reinforcement Learning.** '19

Explicit policy constraint methods

How do we implement constraints?

1. Modify the actor objective

~~$$\theta \leftarrow \arg \max_{\theta} E_{\mathbf{s} \sim D} [E_{\mathbf{a} \sim \pi_{\theta}(\mathbf{a}|\mathbf{s})} [Q(\mathbf{s}, \mathbf{a})]]$$~~

$$\theta \leftarrow \arg \max_{\theta} E_{\mathbf{s} \sim D} [E_{\mathbf{a} \sim \pi_{\theta}(\mathbf{a}|\mathbf{s})} [Q(\mathbf{s}, \mathbf{a}) + \lambda \log \pi_{\beta}(\mathbf{a}|\mathbf{s})] + \lambda \mathcal{H}(\pi(\mathbf{a}|\mathbf{s}))]$$

Lagrange multiplier

easy to compute and differentiate
for Gaussian or categorical policies

$$D_{\text{KL}}(\pi \| \pi_{\beta}) = E_{\pi} [\log \pi(\mathbf{a}|\mathbf{s}) - \log \pi_{\beta}(\mathbf{a}|\mathbf{s})] = -E_{\pi} [\log \pi_{\beta}(\mathbf{a}|\mathbf{s})] - \mathcal{H}(\pi)$$

2. Modify the reward function

$$\bar{r}(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) - D(\pi, \pi_{\beta})$$

simple modification to directly penalize divergence
also accounts for **future** divergence

See: Wu, Tucker, Nachum. **Behavior Regularized Offline Reinforcement Learning**. '19

generally, the best modern offline RL methods do not do either of these things

Implicit policy constraint methods

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

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

straightforward to
show via duality

See also:
Peters et al. (REPS)

approximate via **weighted** max likelihood!

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


↑
samples from dataset
 $\mathbf{a} \sim \pi_{\beta}(\mathbf{a}|\mathbf{s})$

↖
critic can be used
to give us this


Implicit policy constraint methods

$$\mathcal{L}_C(\phi) = E_{(\mathbf{s}, \mathbf{a}, \mathbf{s}') \sim D} \left[\left(Q_\phi(\mathbf{s}, \mathbf{a}) - (r(\mathbf{s}, \mathbf{a}) + \gamma E_{\mathbf{a}' \sim \pi_\theta(\mathbf{a}'|\mathbf{s}')} [Q_\phi(\mathbf{s}', \mathbf{a}')]]) \right)^2 \right]$$

$$\mathcal{L}_A(\theta) = -E_{(\mathbf{s}, \mathbf{a}) \sim \pi_\beta} \left[\log \pi_\theta(\mathbf{a}|\mathbf{s}) \frac{1}{Z(\mathbf{s})} \exp \left(\frac{1}{\lambda} A^{\pi_{\text{old}}}(\mathbf{s}, \mathbf{a}) \right) \right]$$



1. $\phi \leftarrow \phi - \alpha \nabla_\phi \mathcal{L}_C(\phi)$
2. $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_A(\theta)$


$$Q(\mathbf{s}, \mathbf{a}) \leftarrow r(\mathbf{s}, \mathbf{a}) + E_{\mathbf{a}' \sim \pi_{\text{new}}} [Q(\mathbf{s}', \mathbf{a}')] \\ \pi_{\text{new}}(\mathbf{a}|\mathbf{s}) = \arg \max_{\pi} E_{\mathbf{a} \sim \pi(\mathbf{a}|\mathbf{s})} [Q(\mathbf{s}, \mathbf{a})] \text{ s.t. } D_{\text{KL}}(\pi \| \pi_\beta) \leq \epsilon$$

Can we also avoid all OOD actions in the Q update?

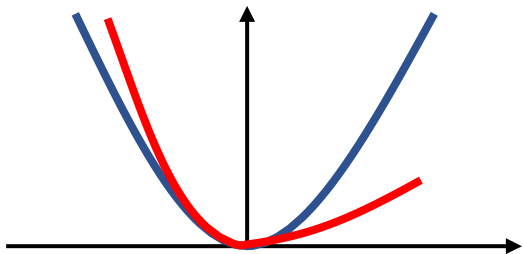
$$Q(s, a) \leftarrow r(s, a) + \underbrace{E_{a' \sim \pi_{\text{new}}}[Q(s', a')]}_{V(s')}$$

$V(s')$ ← just another neural network

$$V \leftarrow \arg \min_V \frac{1}{N} \sum_{i=1}^N \ell(V(s_i), Q(s_i, a_i))$$

e.g., MSE loss $(V(s_i) - Q(s_i, a_i))^2$ this action comes from π_β not from π_{new}

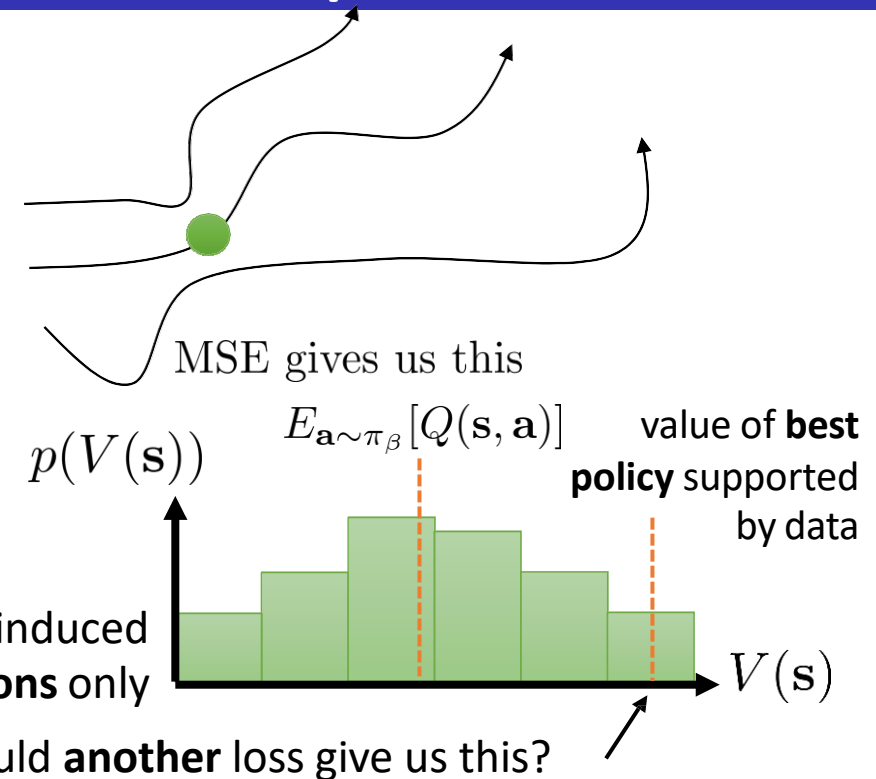
expectile: $\ell_2^\tau(x) = \begin{cases} (1 - \tau)x^2 & \text{if } x > 0 \\ \tau x^2 & \text{else} \end{cases}$



$$V(s) \leftarrow \max_{a \in \Omega(s)} Q(s, a)$$

$$\Omega(s) = \{a : \pi_\beta(a|s) \geq \epsilon\}$$

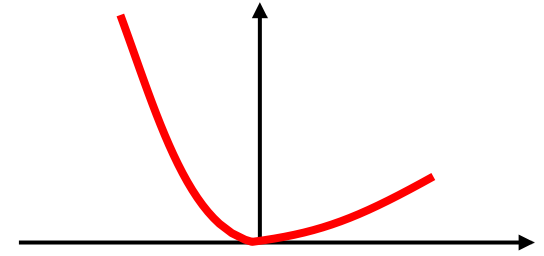
if we use ℓ_2^τ for big τ



Kostrikov, Nair, Levine. **Offline Reinforcement Learning with Implicit Q-Learning.** '21

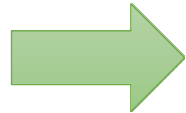
Implicit Q-learning (IQL)

$$Q(\mathbf{s}, \mathbf{a}) \leftarrow r(\mathbf{s}, \mathbf{a}) + V(\mathbf{s}') \quad V \leftarrow \arg \min_V \frac{1}{N} \sum_{i=1}^N \ell_2^\tau(V(\mathbf{s}_i), Q(\mathbf{s}_i, \mathbf{a}_i))$$



$$V(\mathbf{s}) \leftarrow \max_{\mathbf{a} \in \Omega(\mathbf{s})} Q(\mathbf{s}, \mathbf{a})$$

$$\Omega(\mathbf{s}) = \{\mathbf{a} : \pi_\beta(\mathbf{a}|\mathbf{s}) \geq \epsilon\}$$



$$Q(\mathbf{s}, \mathbf{a}) \leftarrow r(\mathbf{s}, \mathbf{a}) + \max_{\mathbf{a}' \in \Omega(\mathbf{s}')} Q(\mathbf{s}', \mathbf{a}')$$

“implicit” policy

$$\pi_{\text{new}}(\mathbf{a}|\mathbf{s}) = \delta(\mathbf{a} = \arg \max_{\mathbf{a} \in \Omega(\mathbf{s})} Q(\mathbf{s}, \mathbf{a}))$$

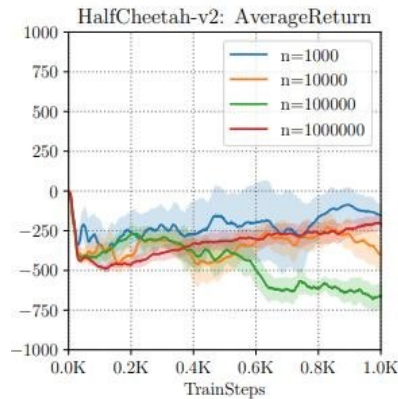
if we use ℓ_2^τ for big τ

Now we can do value function updates without ever risking out-of-distribution actions!

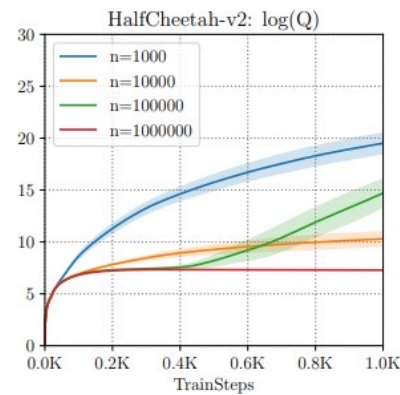
Offline RL Solutions

Conservative Q-Learning

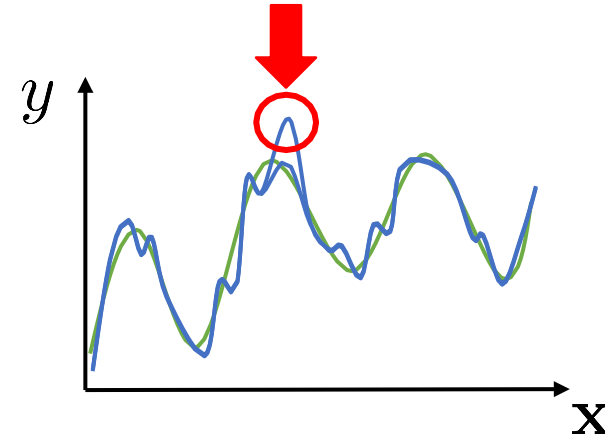
Conservative Q-learning (CQL)



how well it does



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



$$\hat{Q}^\pi = \arg \min_Q \max_{\mu} \alpha E_{s \sim D, a \sim \mu(a|s)} [Q(s, a)] \quad \left. \vphantom{\arg \min_Q} \right\} \text{ term to push down big Q-values}$$

$$\text{regular objective} \quad \left\{ + E_{(s, a, s') \sim D} \left[(Q(s, a) - (r(s, a) + E_\pi [Q(s', a')]))^2 \right] \right\}$$

can show that $\hat{Q}^\pi \leq Q^\pi$ for large enough α
 \uparrow
 true Q-function

Conservative Q-learning (CQL)


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

$$\hat{Q}^\pi = \arg \min_Q \max_\mu \left\{ \begin{aligned} & \alpha E_{\mathbf{s} \sim D, \mathbf{a} \sim \mu(\mathbf{a}|\mathbf{s})} [Q(\mathbf{s}, \mathbf{a})] - \alpha E_{(\mathbf{s}, \mathbf{a}) \sim D} [Q(\mathbf{s}, \mathbf{a})] \\ & + E_{(\mathbf{s}, \mathbf{a}, \mathbf{s}') \sim D} \left[(Q(\mathbf{s}, \mathbf{a}) - (r(\mathbf{s}, \mathbf{a}) + E_\pi [Q(\mathbf{s}', \mathbf{a}')]))^2 \right] \end{aligned} \right\} \mathcal{L}_{\text{CQL}}(\hat{Q}^\pi)$$

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

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

if actions are discrete:

$$\pi(\mathbf{a}|\mathbf{s}) = \begin{cases} 1 & \text{if } \mathbf{a} = \arg \max_{\mathbf{a}} \hat{Q}(\mathbf{s}, \mathbf{a}) \\ 0 & \text{otherwise} \end{cases}$$

if actions are continuous:

$$\theta \leftarrow \theta + \alpha \nabla_{\theta} \sum_i E_{\mathbf{a} \sim \pi_{\theta}(\mathbf{a}|\mathbf{s}_i)} [\hat{Q}(\mathbf{s}_i, \mathbf{a})]$$

Conservative Q-learning (CQL)

$$\hat{Q}^\pi = \arg \min_Q \max_\mu \left\{ \begin{aligned} &\alpha E_{\mathbf{s} \sim D, \mathbf{a} \sim \mu(\mathbf{a}|\mathbf{s})} [Q(\mathbf{s}, \mathbf{a})] - \alpha E_{(\mathbf{s}, \mathbf{a}) \sim D} [Q(\mathbf{s}, \mathbf{a})] - \mathcal{R}(\mu) \\ &+ E_{(\mathbf{s}, \mathbf{a}, \mathbf{s}') \sim D} \left[(Q(\mathbf{s}, \mathbf{a}) - (r(\mathbf{s}, \mathbf{a}) + E_\pi [Q(\mathbf{s}', \mathbf{a}')]))^2 \right] \end{aligned} \right\} \mathcal{L}_{\text{CQL}}(\hat{Q}^\pi)$$

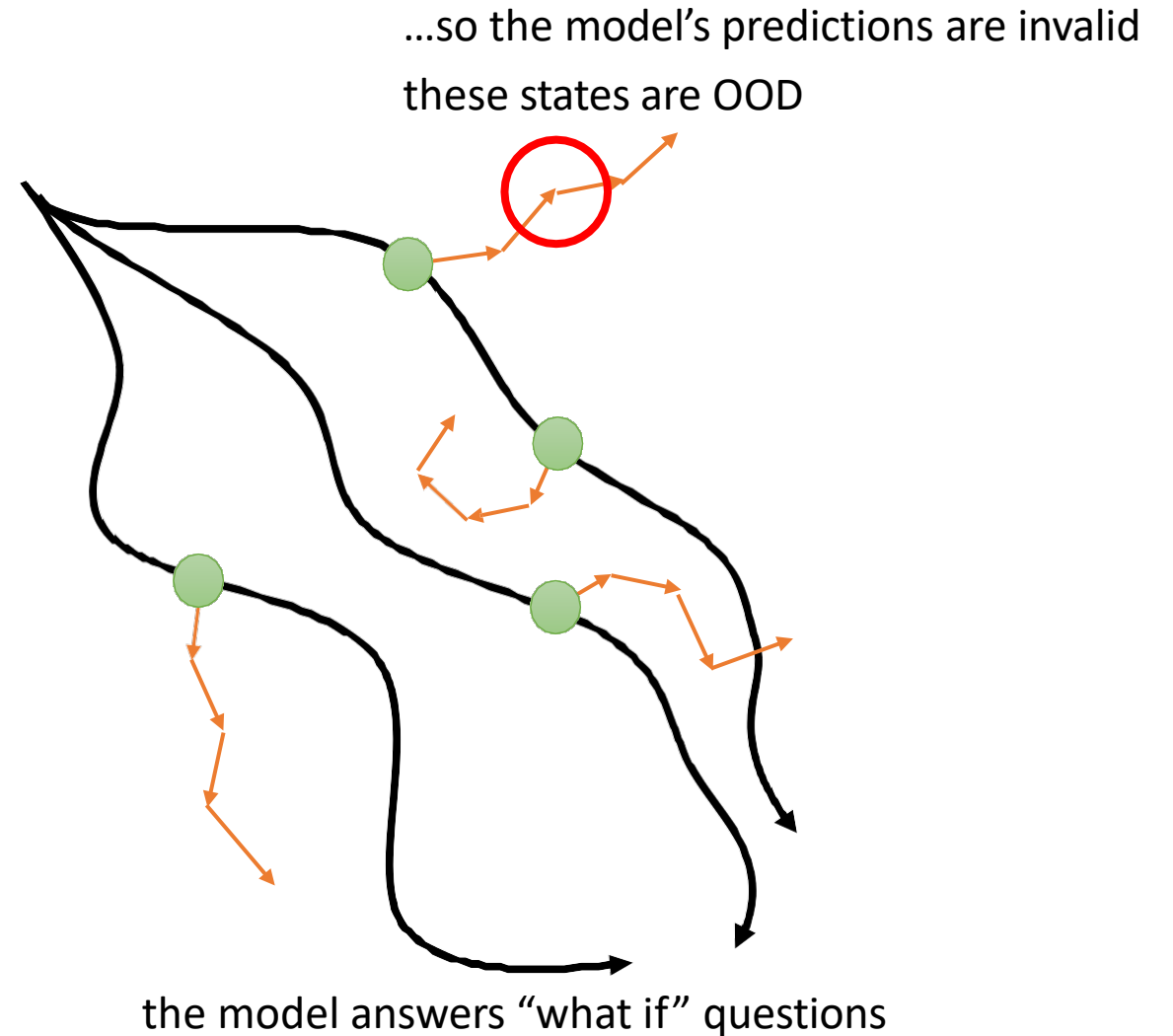
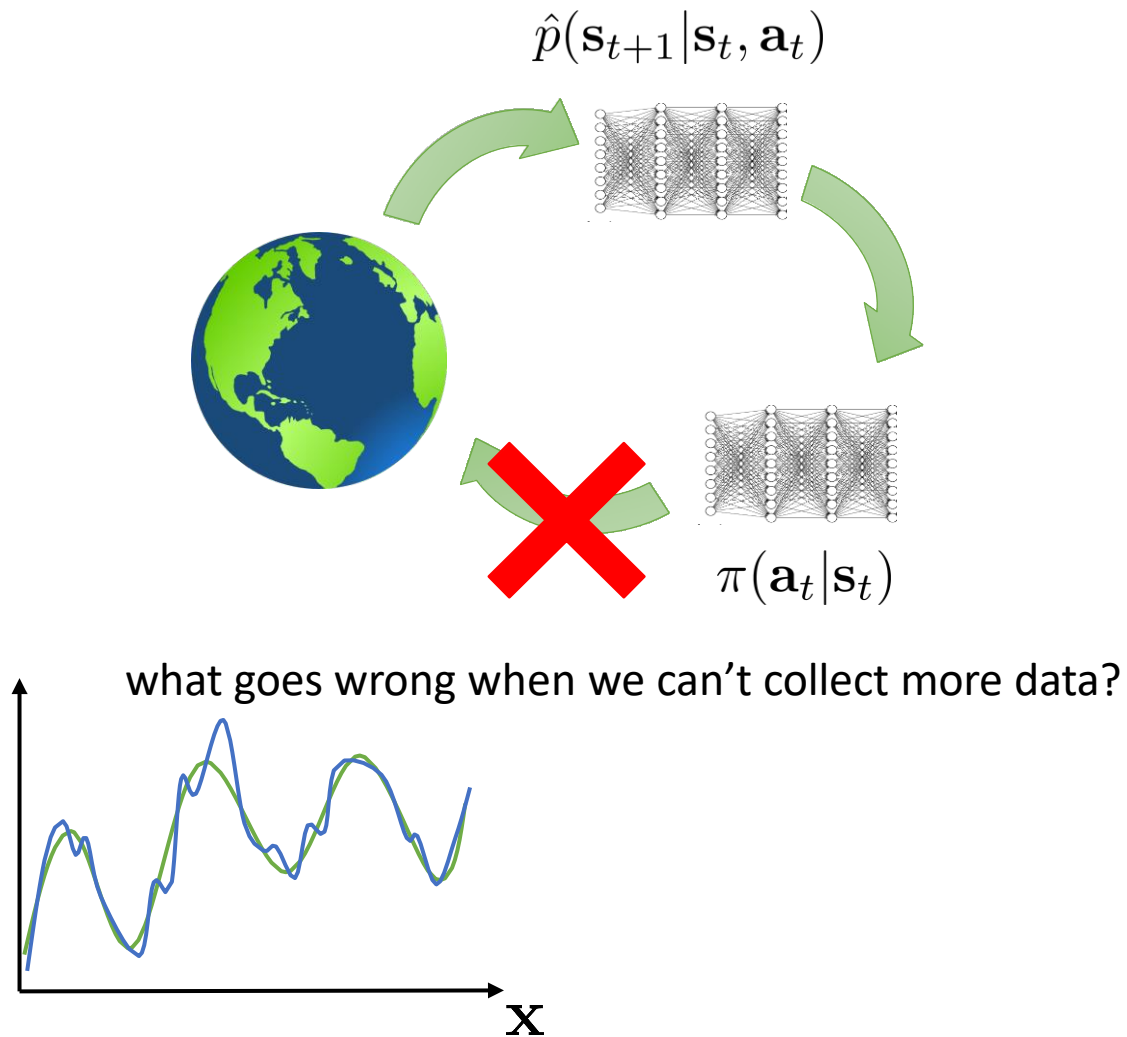
regularization

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

Offline RL Solutions

Model-Based Offline RL

How does model-based RL work?



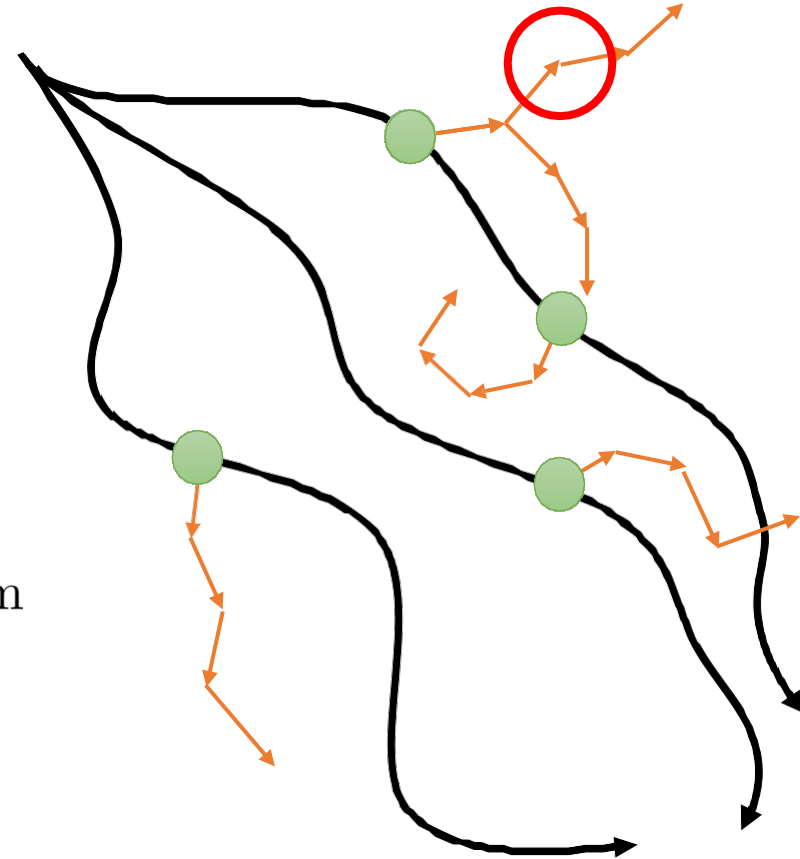
Model-Based Offline RL

solution: “punish” the policy for exploiting

$$\tilde{r}(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) - \lambda u(\mathbf{s}, \mathbf{a})$$

← uncertainty penalty

...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)

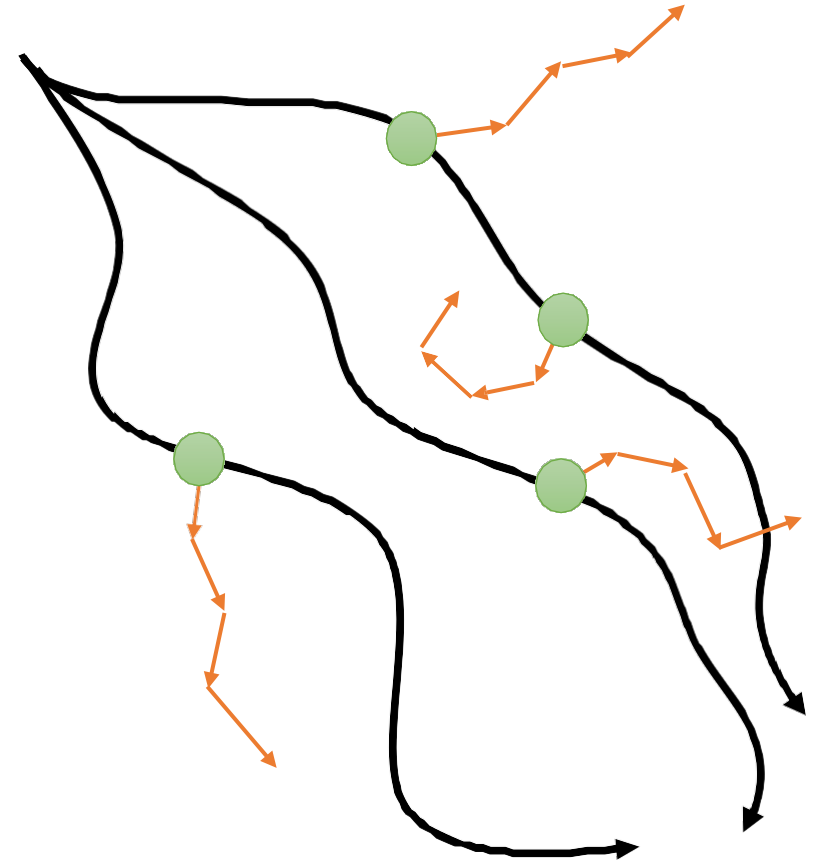
Conservative Model-Based RL

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

state-action tuples from the model

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

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



Summary, Applications, Open Questions

Which offline RL algorithm do I use?

If you want to *only* train offline...

Conservative Q-learning	+ just one hyperparameter	+ well understood and widely tested
Implicit Q-learning	+ more flexible (offline + online)	- more hyperparameters

If you want to *only* train offline and finetune online

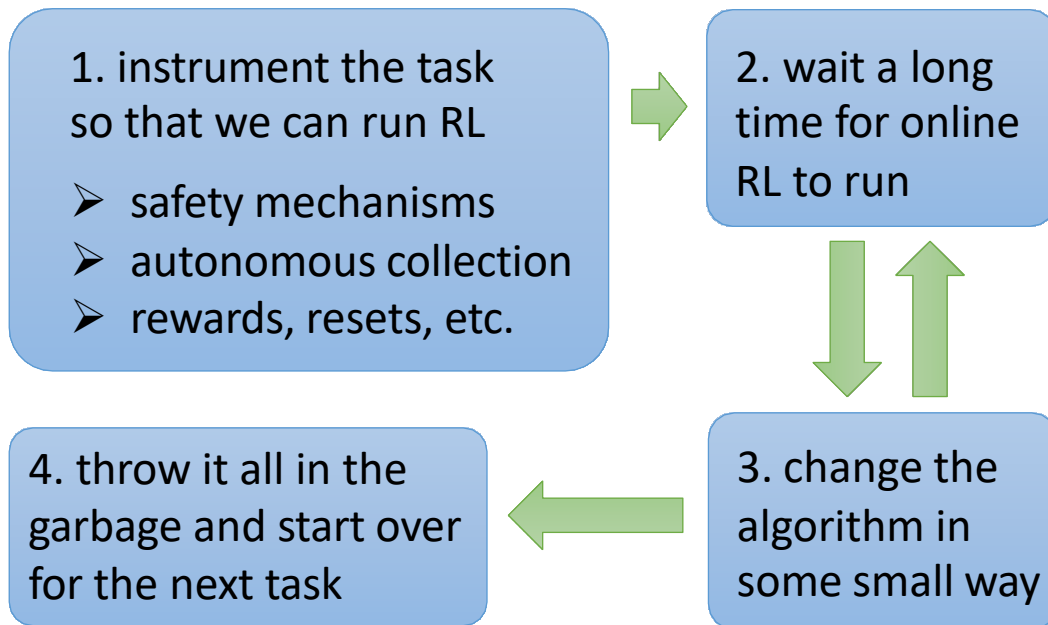
Advantage-weighted actor-critic (AWAC)	+ widely used and well tested
Implicit Q-learning	+ seems to perform much better!

If you have a good way to train models in your domain

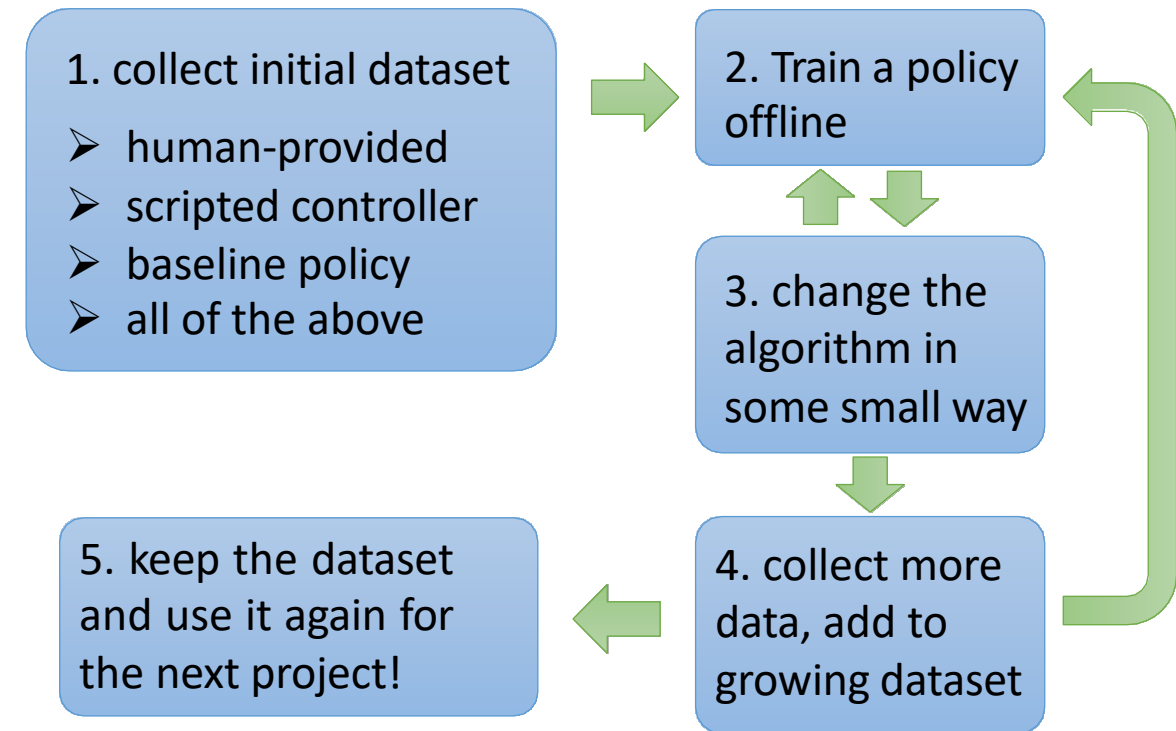
COMBO	+ similar properties as CQL, but benefits from models
	- not always easy to train a good model in your domain!

The power of offline RL

standard real-world RL process



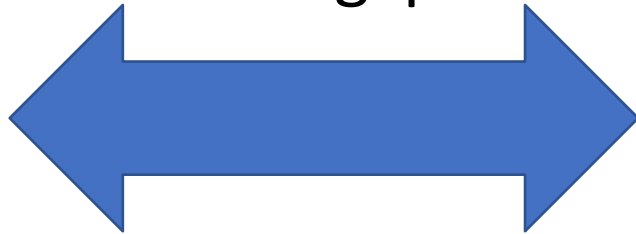
offline RL process



Takeaways, conclusions, future directions

current offline RL algorithms

“the gap”



“the dream”

1. Collect a dataset using any policy or mixture of policies
2. Run offline RL on this dataset to learn a policy
3. Deploy the policy in the real world



- An offline RL **workflow**
 - Supervised learning workflow: train/test split
 - Offline RL workflow: ???
- Statistical **guarantees**
 - Biggest challenge: distributional shift/counterfactuals
 - Can we make any guarantees?
- Scalable methods, large-scale applications
 - Dialogue systems
 - Data-driven navigation and driving