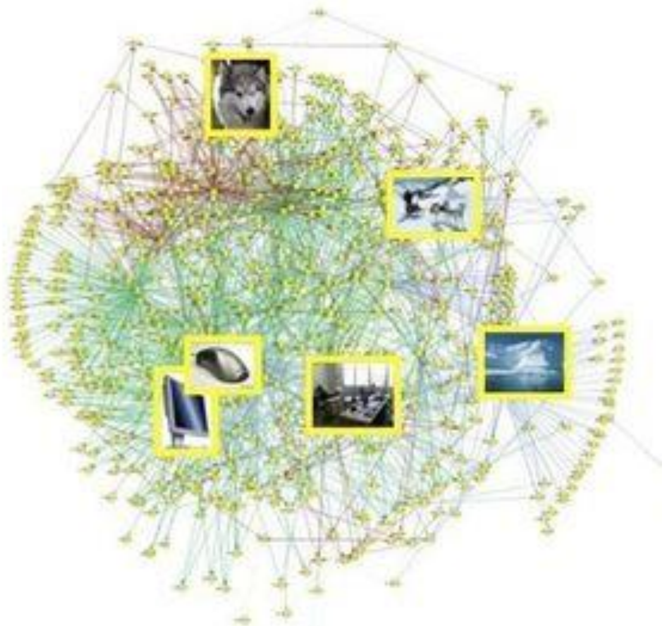
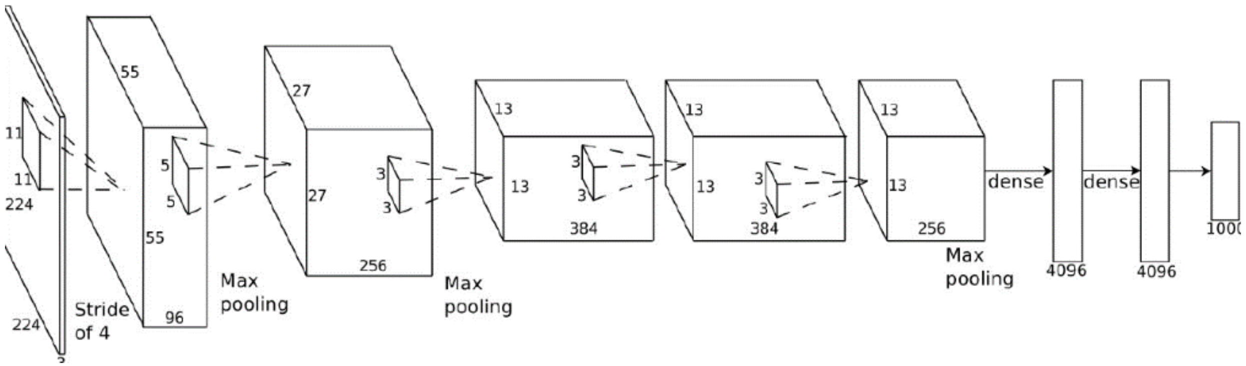
Offline Reinforcement Learning

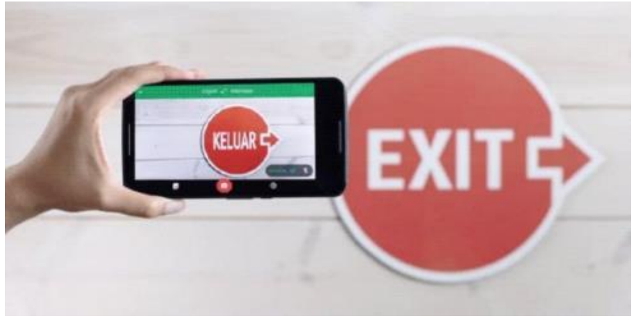
Sergey Levine

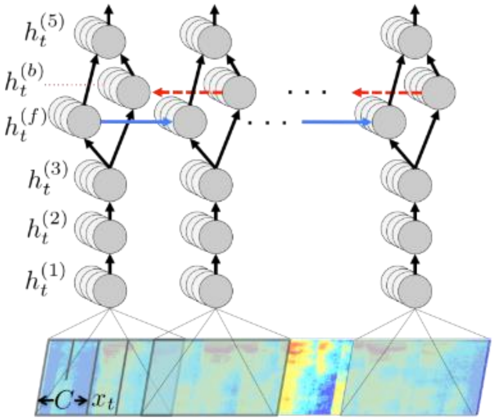
UC Berkeley

What makes modern machine learning work?

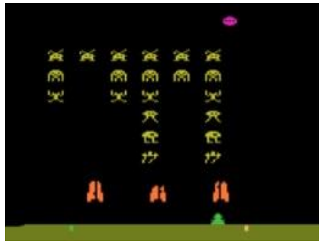




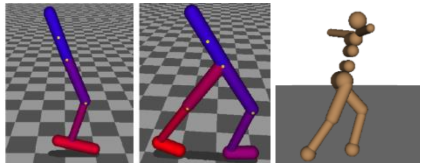




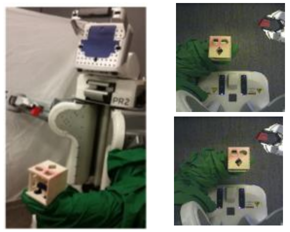
What about reinforcement learning?



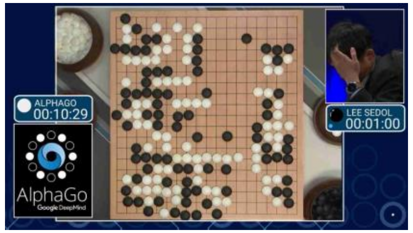
Mnih et al. '13

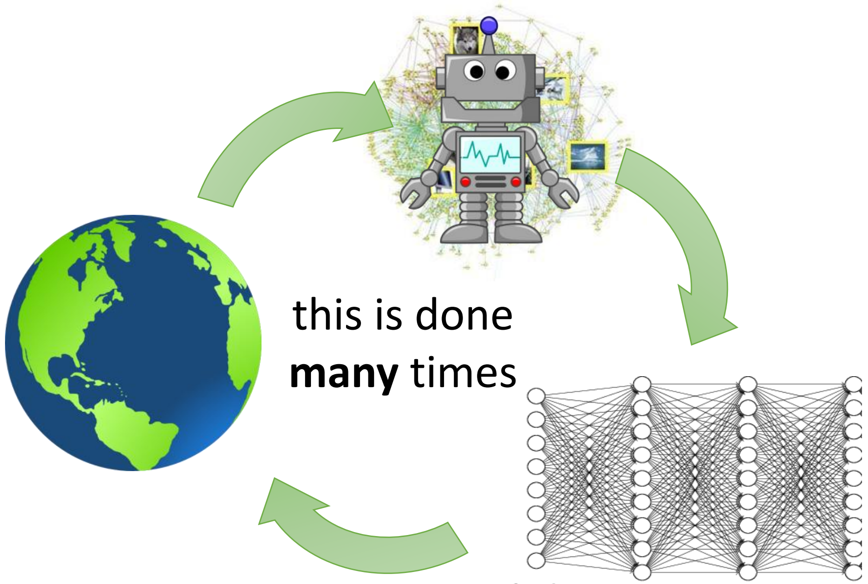


Schulman et al. '14 & '15

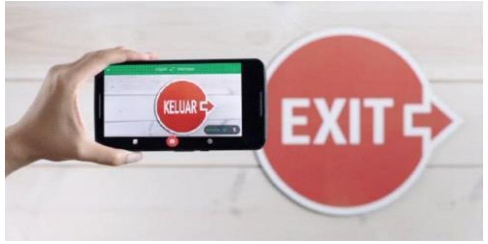


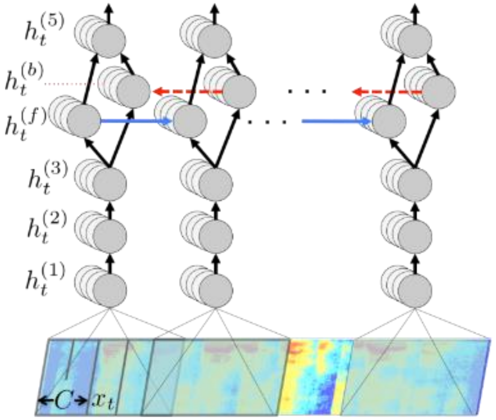
Levine*, Finn*, et al. '16



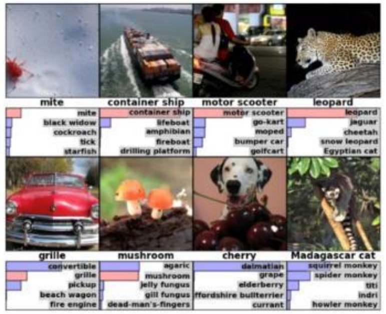


this is done
many times



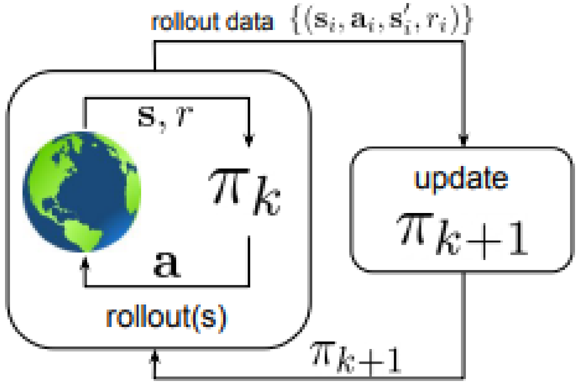


enormous gulf

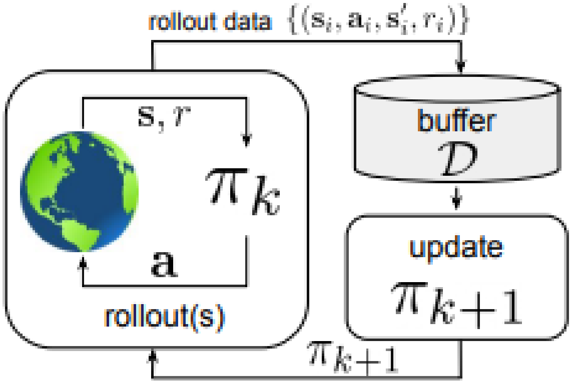


Can we develop **data-driven** RL methods?

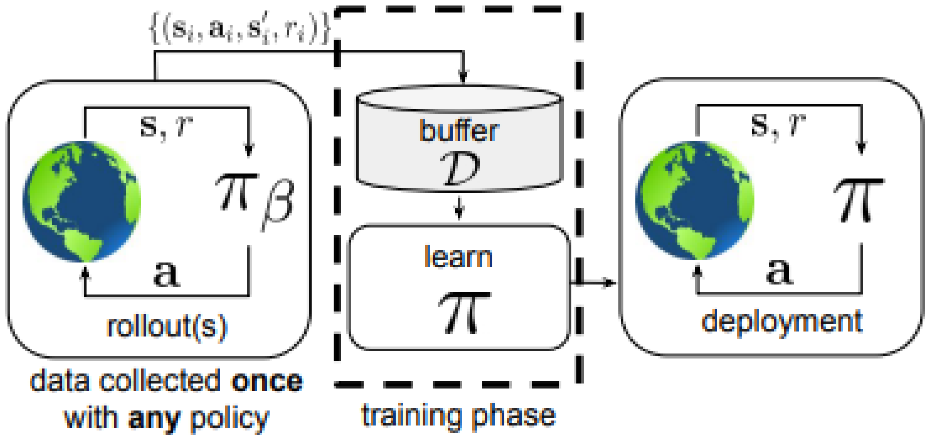
on-policy RL

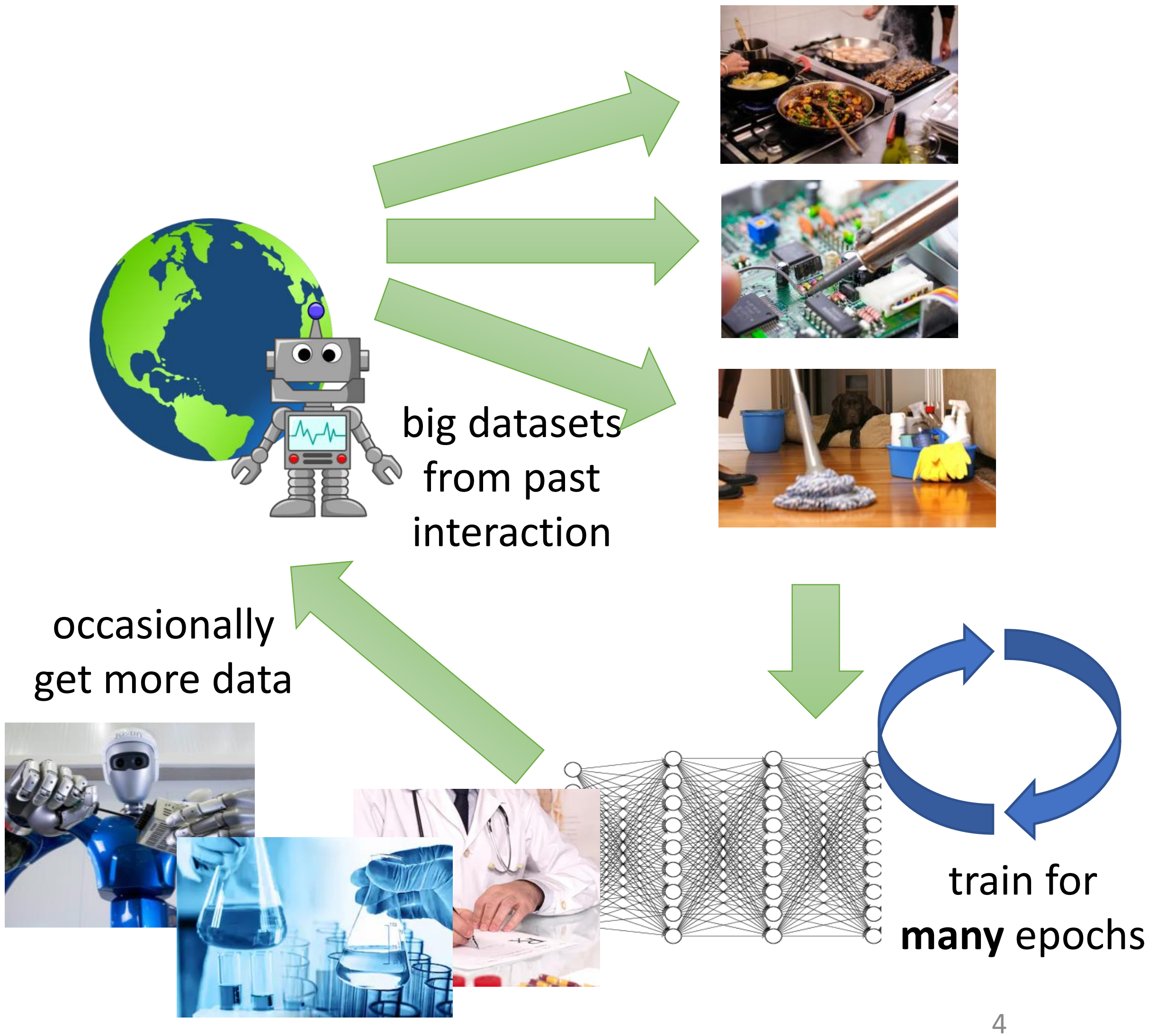


off-policy RL

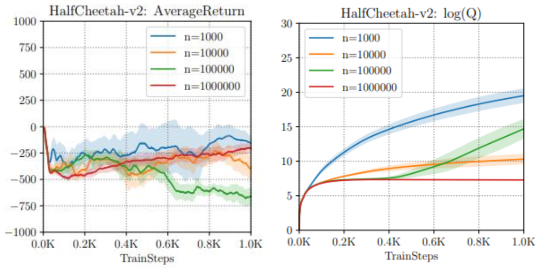


offline reinforcement learning

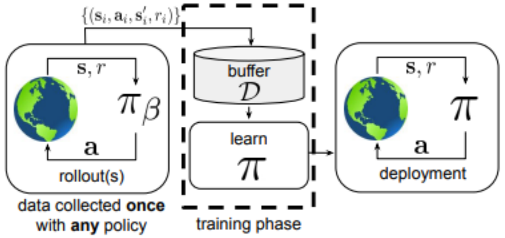




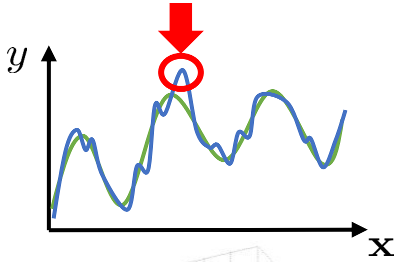
Levine, Kumar, Tucker, Fu. **Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems.** ‘20



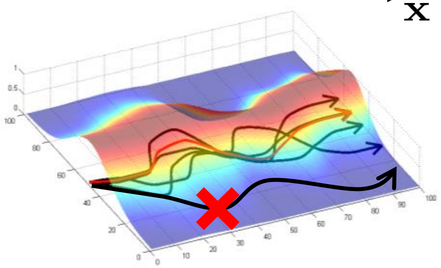
Why is offline RL difficult?



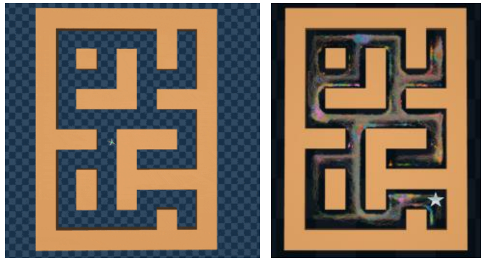
How do we design offline RL algorithms?



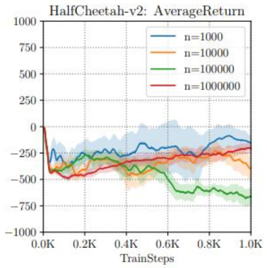
Conservative Q-Learning

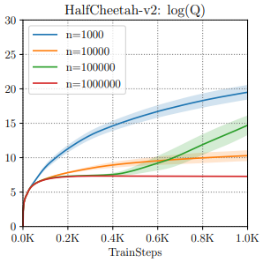


Model-based offline RL

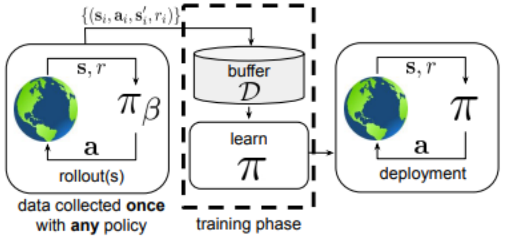


How do we evaluate offline RL methods?

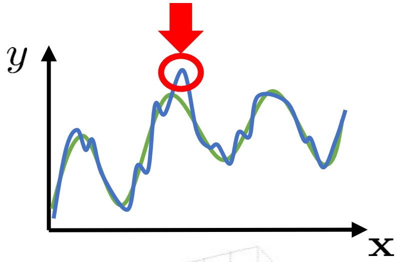




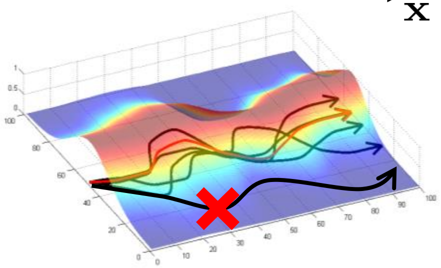
Why is offline RL difficult?



How do we design offline RL algorithms?



Conservative Q-Learning

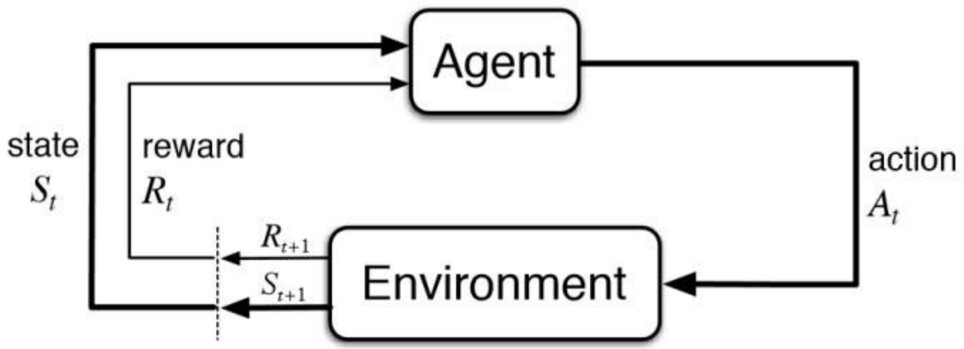


Model-based offline RL

How do we evaluate offline RL methods?

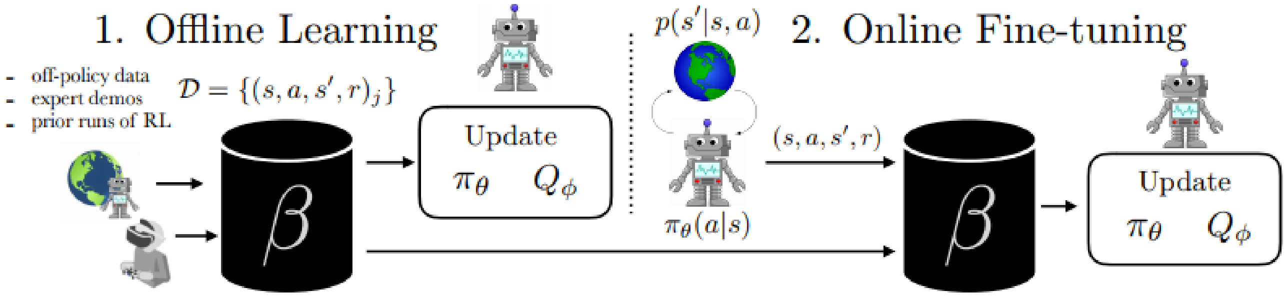
On-policy, off-policy, and offline RL

"Classic" RL diagram:

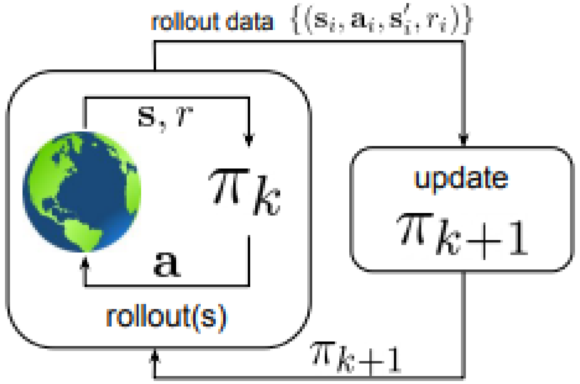


This is a very **online** view of RL

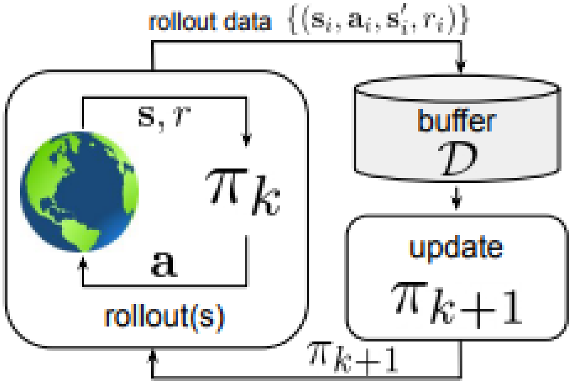
More typical use case:



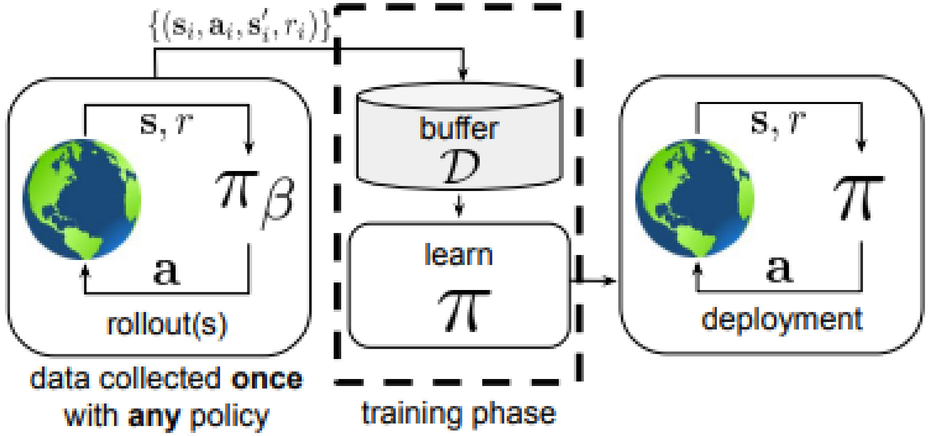
on-policy RL



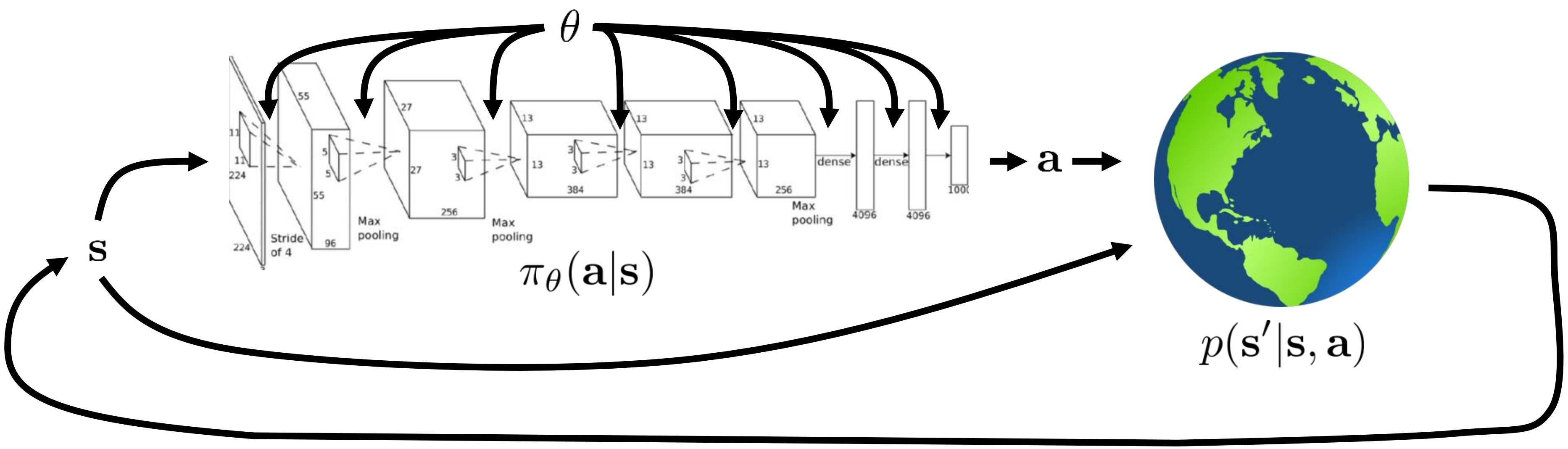
off-policy RL



offline reinforcement learning



The RL objective



$$\underbrace{p_\theta(\mathbf{s}_1,\mathbf{a}_1,\ldots,\mathbf{s}_T,\mathbf{a}_T)}_{p_\theta(\tau)} = p(\mathbf{s}_1)\prod_{t=1}^{T}\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)p(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t)$$

it is very hard to optimize this with off-policy data directly

$$\theta^\star = \arg\max_\theta E_{\tau\sim p_\theta(\tau)}\left[\sum_t \gamma^t r(\mathbf{s}_t,\mathbf{a}_t)\right]$$

The RL objective

$$E_{\tau\sim p_\theta(\tau)}\left[\sum_t \gamma^t r(\mathbf{s}_t,\mathbf{a}_t)\right] \approx \frac{1}{N}\sum_{i=1}^{N} E_{\mathbf{a}\sim\pi_\theta(\mathbf{a}|\mathbf{s}_i)}\left[\overbrace{Q^\pi(\mathbf{s}_i,\mathbf{a})}^{\sum_{t'=t}^{\infty}\gamma^{t'-t}r(\mathbf{s}_{t'},\mathbf{a}_{t'})}\right]$$

sum over all states in the dataset

if we just knew this, all would be easy
so let's learn it!

Aside: recovering the policy

could optimize the above objective w.r.t. π_θ directly

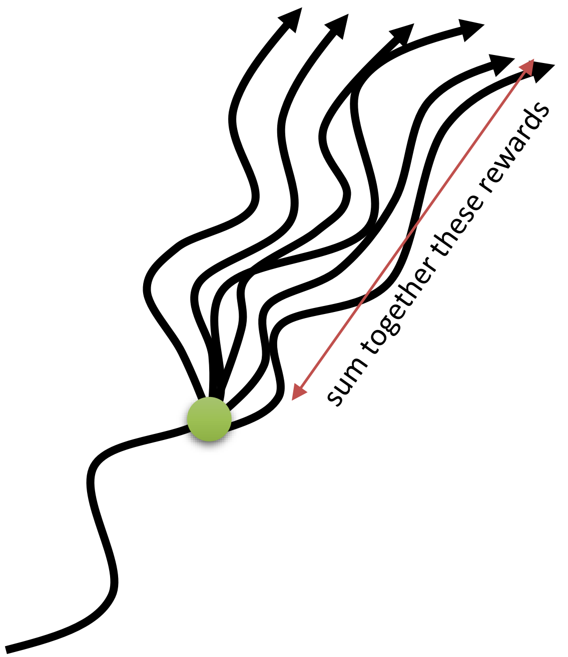
$$\pi(\mathbf{a}_t|\mathbf{s}_t) = \begin{cases} 1 \text{ if } \mathbf{a}_t = \arg\max_{\mathbf{a}_t} Q^\pi(\mathbf{s}_t,\mathbf{a}_t) \\ 0 \text{ otherwise} \end{cases}$$

"greedy" policy
can recover with optimization (e.g., CEM)

The Q-function

$$Q^\pi(\mathbf{s}_t, \mathbf{a}_t) = E[\sum_{t'=t}^{\infty} \gamma^{t'-t} r(\mathbf{s}_{t'}, \mathbf{a}_{t'})\ ] = r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \underbrace{(E[r(\mathbf{s}_{t+1}, \mathbf{a}_{t+1})] + \gamma E[r(\mathbf{s}_{t+1}, \mathbf{a}_{t+2})] \ldots}_{Q^\pi(\mathbf{s}_{t+1}, \mathbf{a}_{t+1})}$$

expectation under $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$
and $p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$



Bellman equation:

$$Q^\pi(\mathbf{s}_t, \mathbf{a}_t) = r(\mathbf{s}_t, \mathbf{a}_t) + \gamma E[Q^\pi(\mathbf{s}_{t+1}, \mathbf{a}_{t+1})]$$

let's say we have a trajectory $\mathbf{s}_1, \mathbf{a}_1, \mathbf{s}_2, \mathbf{a}_2, \ldots, \mathbf{s}_T, \mathbf{a}_T$
generated by some *other* policy π_β
can we *estimate* the Bellman equation?

these come from our trajectory

this is sampled from π_θ

$$Q^\pi(\mathbf{s}_t, \mathbf{a}_t) \approx r(\mathbf{s}_t, \mathbf{a}_t) + \gamma Q^\pi(\mathbf{s}_{t+1}, \mathbf{a}_{t+1})$$

this is a **single sample** estimate of the expectation

The Q-function

these come from our trajectory

$\text{this is sampled from } \pi_\theta$

$$Q^\pi(\mathbf{s}_t, \mathbf{a}_t) \approx r(\mathbf{s}_t, \mathbf{a}_t) + \gamma Q^\pi(\mathbf{s}_{t+1}, \mathbf{a}_{t+1})$$

Rough sketch:

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

In reality, we use a minibatch, not just one transition!

1. Load $\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}$ from buffer
2. Get $\mathbf{a}_{t+1} \sim \pi_\theta(\mathbf{a}_{t+1}|\mathbf{s}_{t+1})$

   either from explicit policy network or via max
3. Compute *target value* $y = r(\mathbf{s}_t, \mathbf{a}_t) + \gamma Q^\pi(\mathbf{s}_{t+1}, \mathbf{a}_{t+1})$
4. Take gradient step on $\mathcal{E} = (Q^\pi(\mathbf{s}_t, \mathbf{a}_t) - y)^2$

Off-policy RL summary

$Q(\mathbf{s},\mathbf{a}) \leftarrow r(\mathbf{s},\mathbf{a}) + E_{\mathbf{a}'}[Q(\mathbf{s}',\mathbf{a}')]$ ⟵ don’t need on-policy data for this!

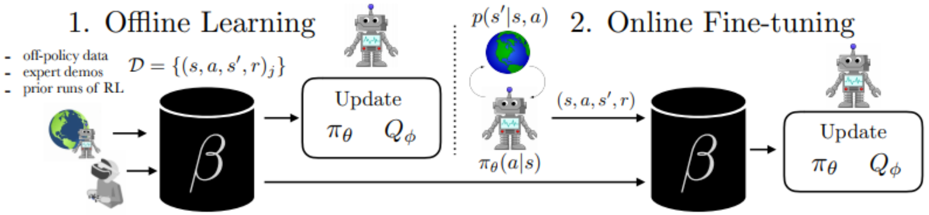
off-policy Q-learning:

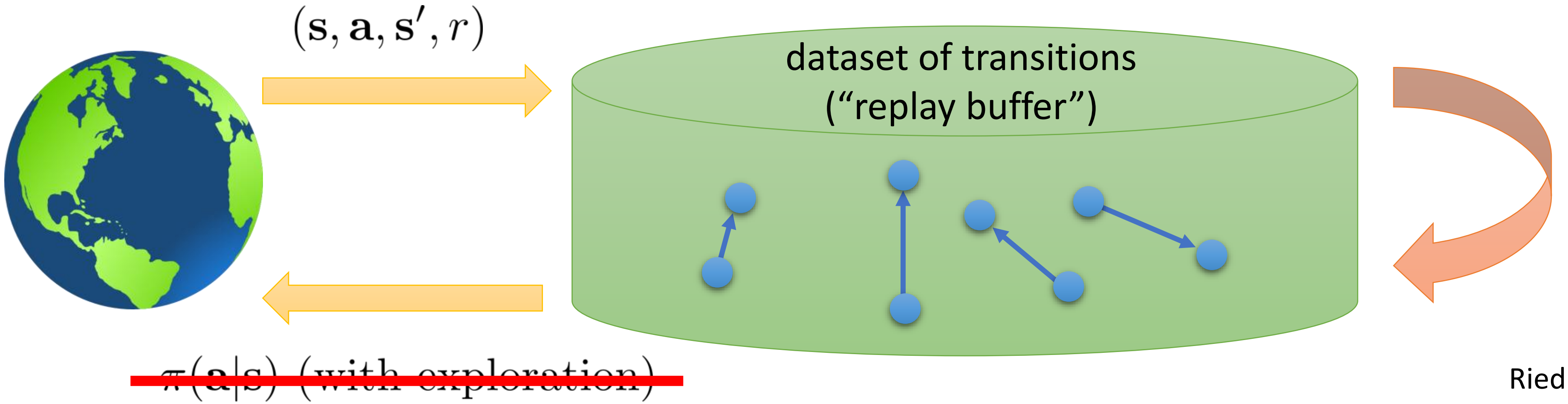
1. collect dataset $\{(\mathbf{s}_i,\mathbf{a}_i,\mathbf{s}'_i,r_i)\}$ using some policy, add it to $\mathcal{B}$

$K\times$

2. sample a batch $(\mathbf{s}_i,\mathbf{a}_i,\mathbf{s}'_i,r_i)$ from $\mathcal{B}$
3. minimize $\sum_i (Q(\mathbf{s}_i,\mathbf{a}_i) - [r(\mathbf{s}_i,\mathbf{a}_i) + E_{\mathbf{a}'_i}[Q(\mathbf{s}'_i,\mathbf{a}'_i)]])^2$

more typical use case:

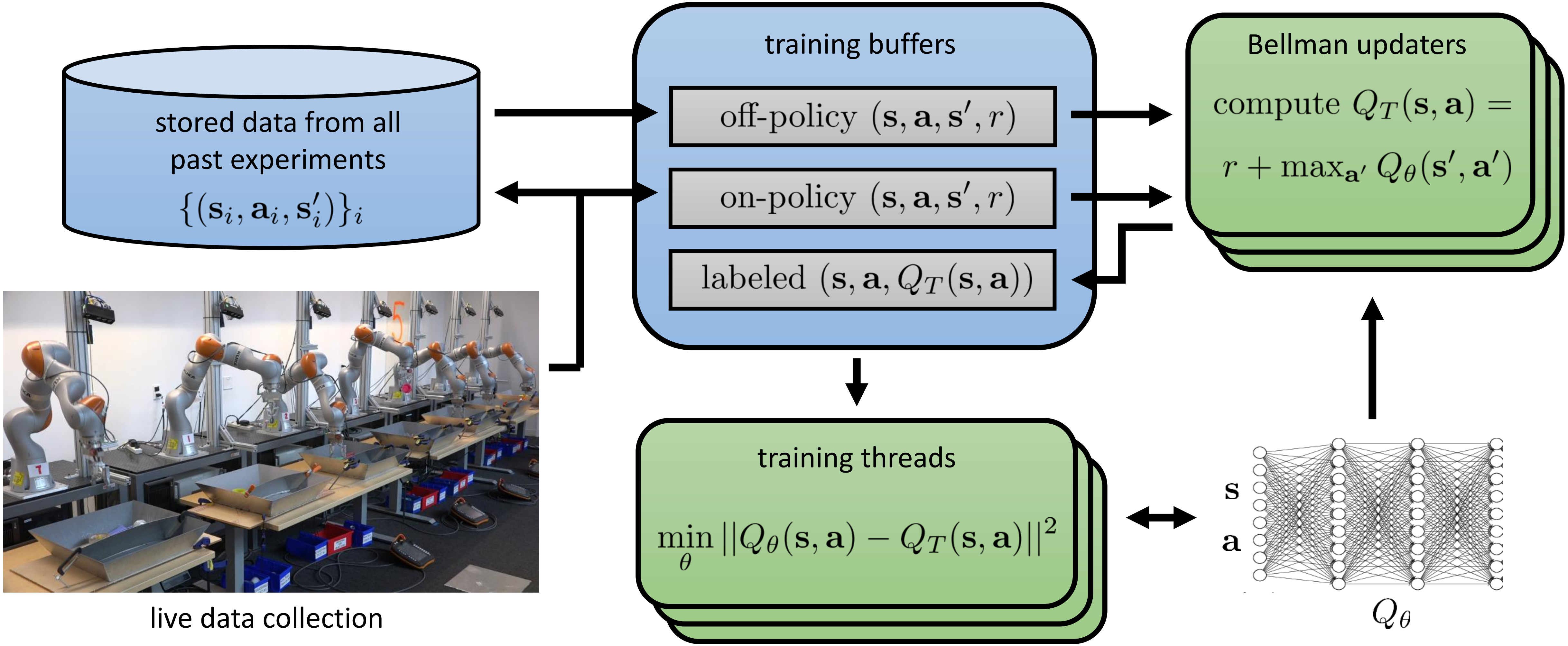




off-policy
Q-learning

See, e.g.
Riedmiller, Neural Fitted Q-Iteration ‘05
Ernst et al., Tree-Based Batch Mode RL ‘05

An instantiation of this idea…



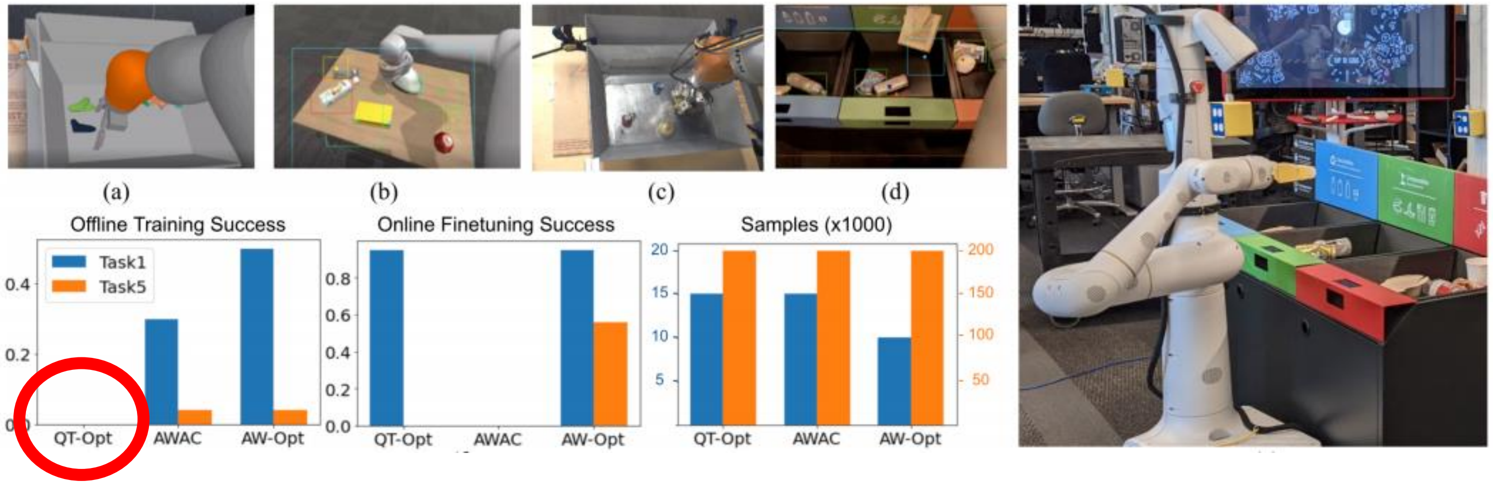
Kalashnikov, Irpan, Pastor, Ibarz, Herzong, Jang, Quillen, Holly, Kalakrishnan, Vanhoucke, Levine. **QT-Opt: Scalable Deep Reinforcement Learning of Vision-Based Robotic Manipulation Skills**

Does it work?

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

Kalashnikov, Irpan, Pastor, Ibarz, Herzong, Jang, Quillen, Holly, Kalakrishnan, Vanhoucke, Levine. **QT-Opt: Scalable Deep Reinforcement Learning of Vision-Based Robotic Manipulation Skills**

Even worse...



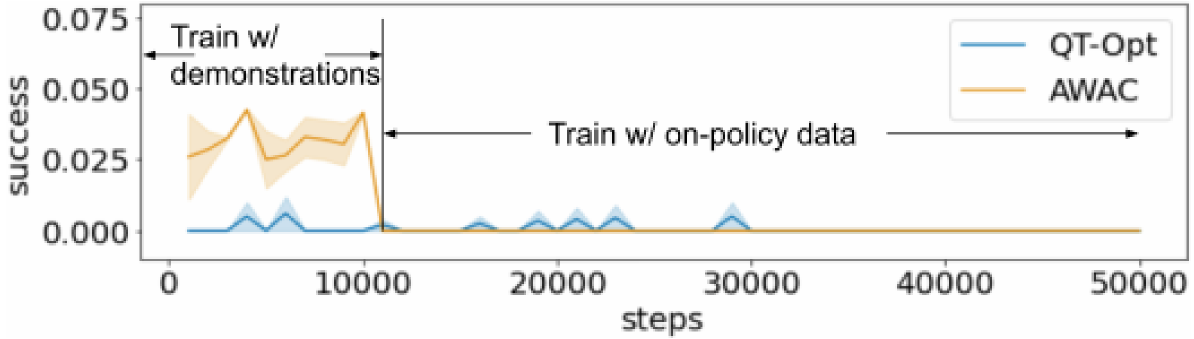
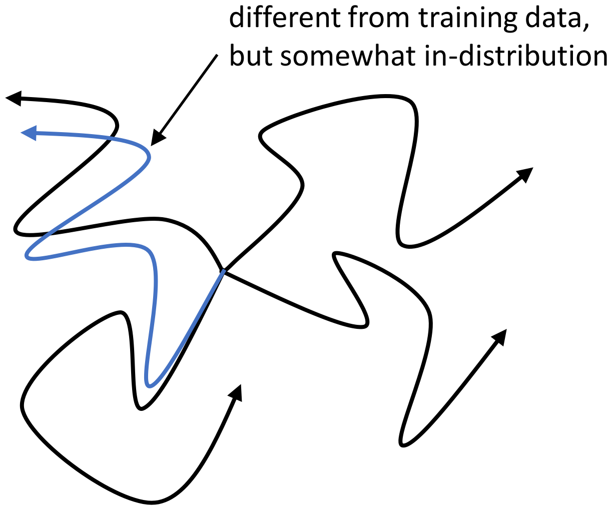
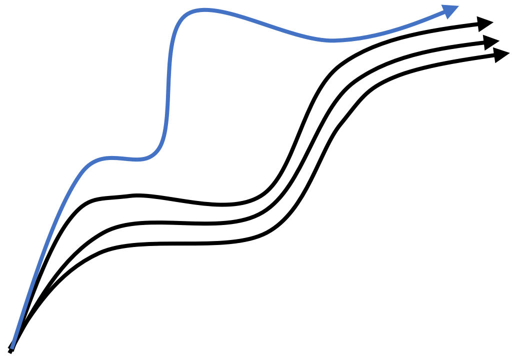


Figure 2: Baseline comparison for the example task. Neither QT-Opt [3], nor AWAC [2] can solve the task.

training on random(ized) offline data



training on demo data



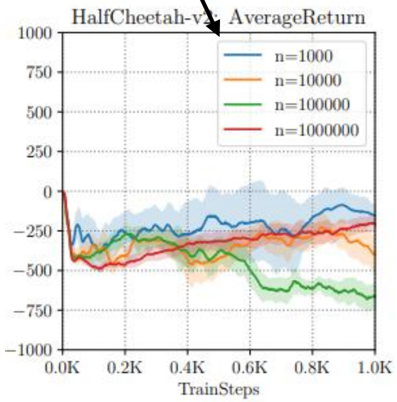
Yao Lu et al. **AW-Opt: Learning Robotic Skills with Imitation and Reinforcement at Scale**. 2021.

What’s the problem?

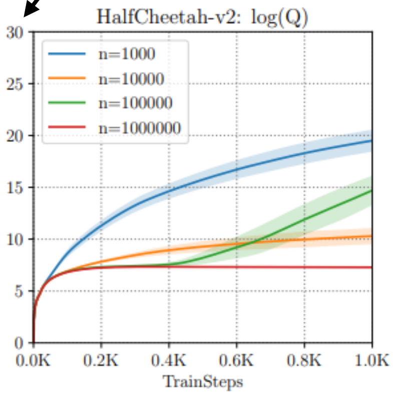
amount of data

log scale (massive overestimation)

Hypothesis 1: Overfitting



how well it does



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

Hypothesis 2: Training data is not good

Usually not the case: behavioral cloning of best data does better!

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

Distribution shift in a nutshell

Example empirical risk minimization (ERM) problem:

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

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

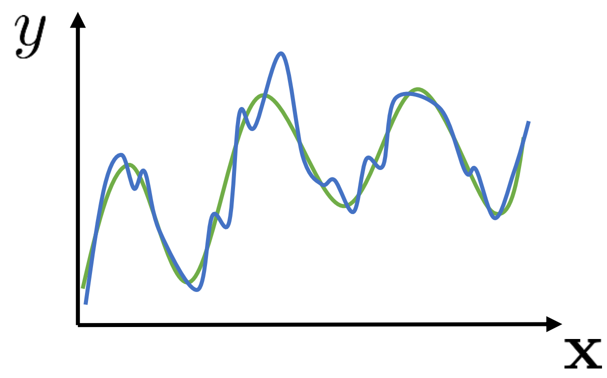
$E_{\mathbf{x}\sim p(\mathbf{x}),y\sim p(y|\mathbf{x})}\left[(f_\theta(\mathbf{x}) - y)^2\right]$ is low

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

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

usually we are not worried – neural nets generalize well!

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



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

Where do we suffer from distribution shift?

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

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

what is the objective?

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

behavior policy

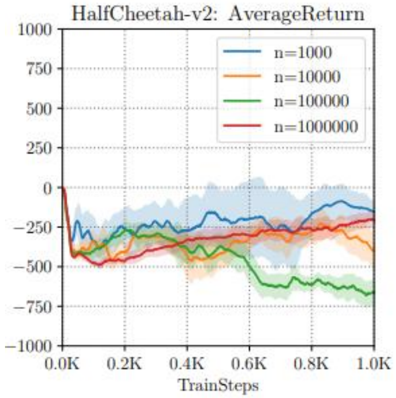
target value

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

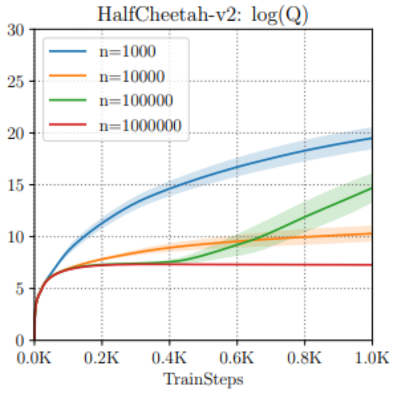
how often does *that* happen?

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 $\mathbf{x}^\star \leftarrow \arg\max_{\mathbf{x}} f_\theta(\mathbf{x})$?)

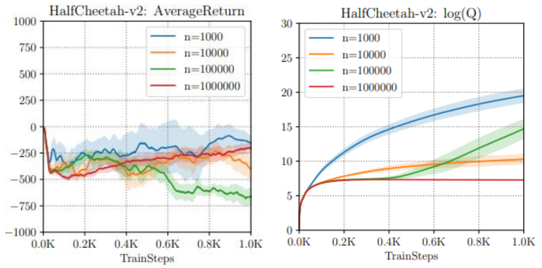


how well it does

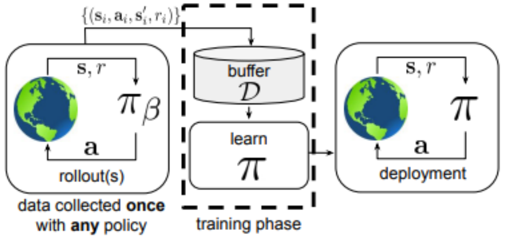


how well it *thinks* it does (Q-values)

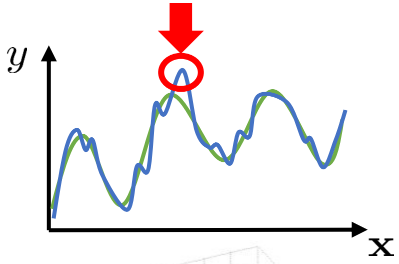
Kumar, Fu, Tucker, Levine. **Stabilizing Off-Policy Q-Learning via Bootstrapping Error Reduction.** NeurIPS '19



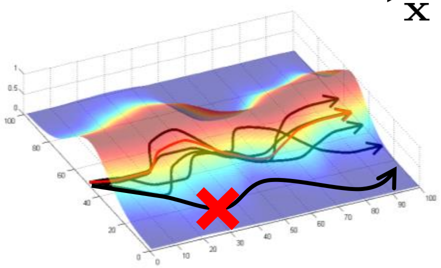
Why is offline RL difficult?



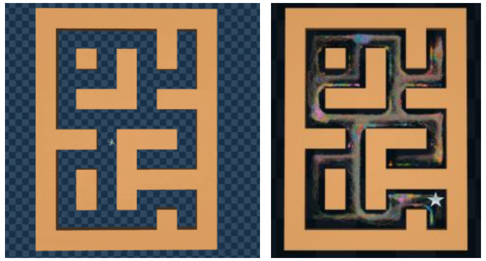
How do we design offline RL algorithms?



Conservative Q-Learning



Model-based offline RL



How do we evaluate offline RL methods?

How do prior methods address this?

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

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

This solves distribution shift, right?

No more erroneous values?

Issue 1: Estimating the behavior policy is difficult

Issue 2: This might be **too** conservative
(we'll come back to this)

"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…")

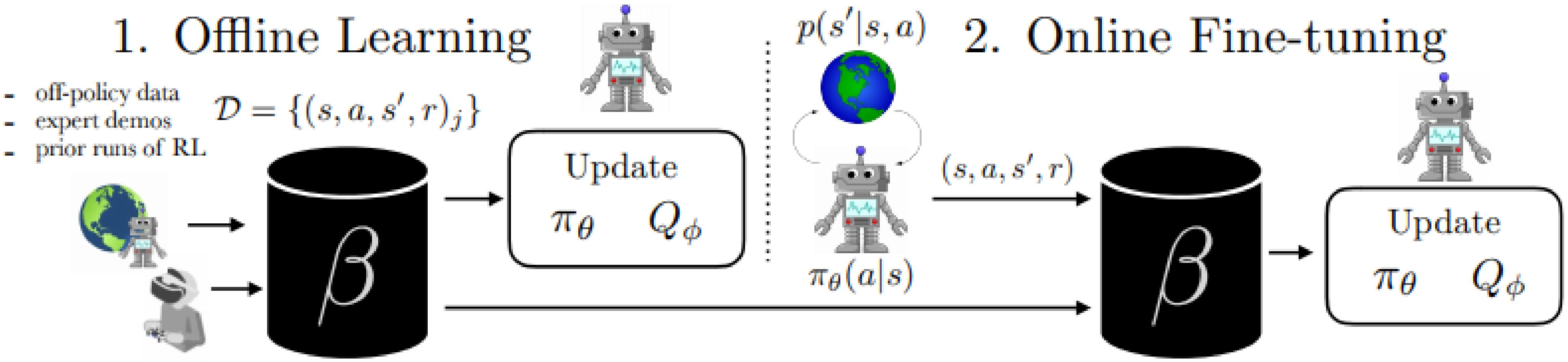
Wu et al. '19 ("Behavior Regularized…")

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

When is estimating the behavior policy hard?

Issue 1: Estimating the behavior policy is difficult

- **Easy case: all data comes from the same Markovian policy**
  - This is not very common or realistic
- **Hard case: data comes from many different policies**
  - Very common in reality (e.g., some demo data from humans, some scripted data)
  - Very common during *online finetuning*



Avoiding behavior policies with **implicit** constraints

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

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

straightforward to show via duality

See also:
Peters et al. (REPS)
Rawlik et al. (“psi-learning”)
…many follow-ups

approximate via **weighted** max likelihood!

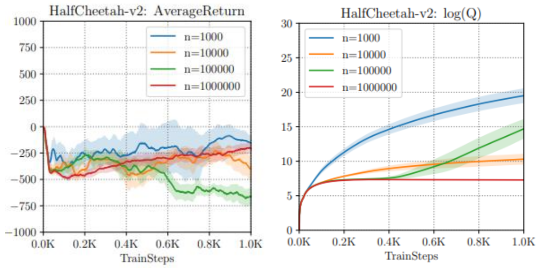
$$\pi_{\text{new}}(\mathbf{a}|\mathbf{s}) = \arg\max_{\pi} E_{(\mathbf{s},\mathbf{a})\sim\pi_\beta}\left[\log\pi(\mathbf{a}|\mathbf{s})\overbrace{\frac{1}{Z(\mathbf{s})}\exp\left(\frac{1}{\lambda}A^{\pi_{\text{old}}}(\mathbf{s},\mathbf{a})\right)}^{w(\mathbf{s},\mathbf{a})}\right]$$

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

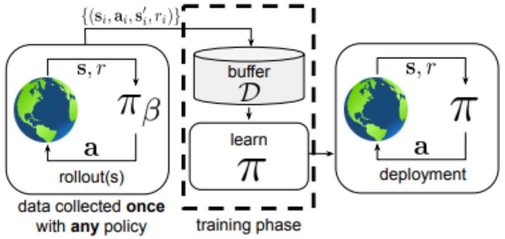
critic can be used
to give us this

Peng*, Kumar*, Levine. **Advantage-Weighted Regression.** ‘19

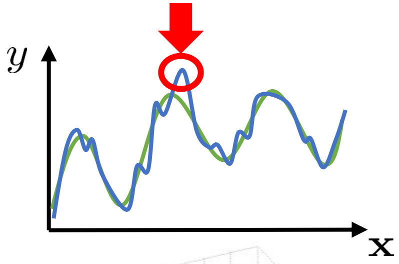
Nair, Dalal, Gupta, Levine. **Accelerating Online Reinforcement Learning with Offline Datasets.** ‘20



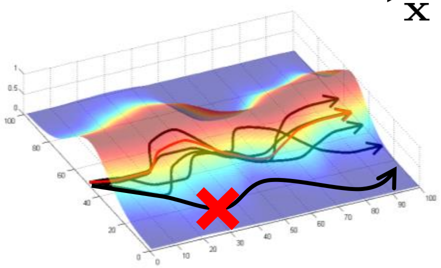
Why is offline RL difficult?



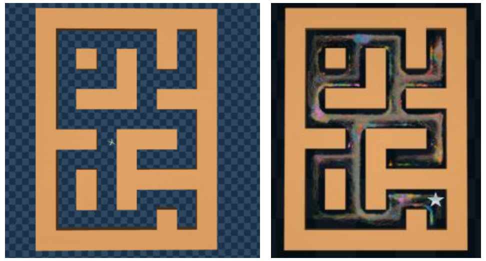
How do we design offline RL algorithms?



Conservative Q-Learning

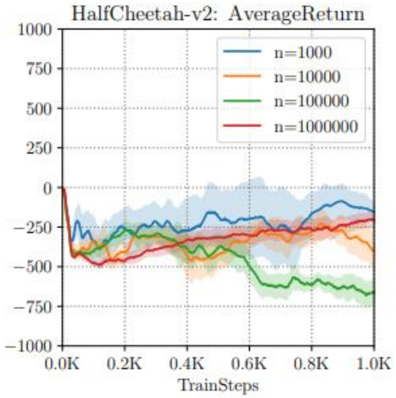


Model-based offline RL

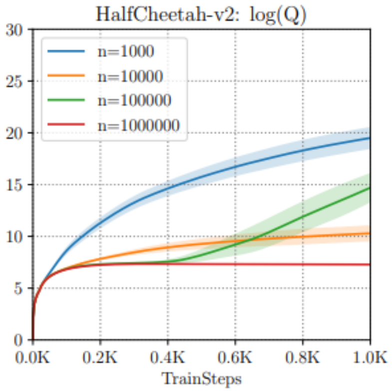


How do we evaluate offline RL methods?

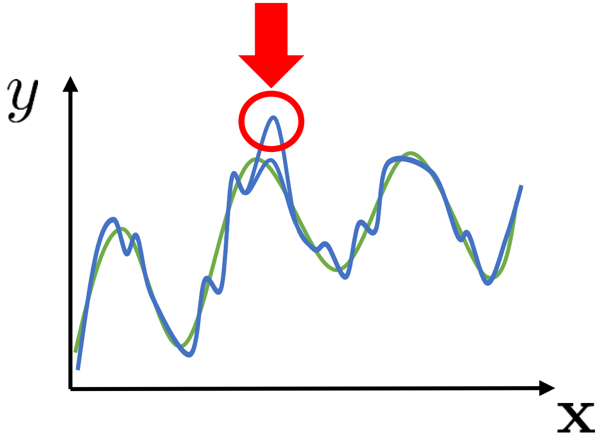
What about those Q-value errors?



how well it does



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



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

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

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

true Q-function

Learning with Q-function lower bounds

A *better* bound:

<u>always</u> pushes Q-values down

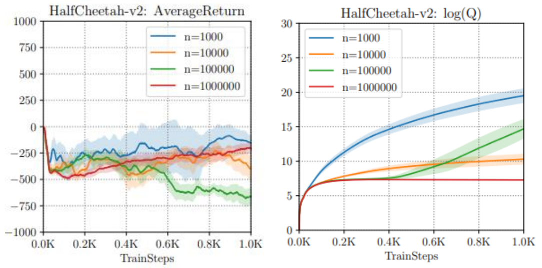
push <u>up</u> on (**s**, **a**) samples in data

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

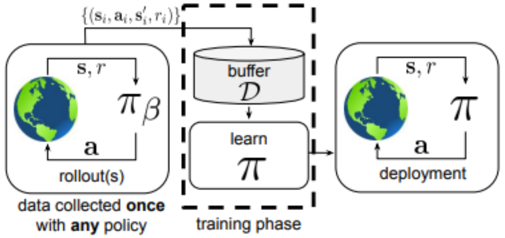
no longer guaranteed that $\hat{Q}^\pi(\mathbf{s},\mathbf{a}) \leq Q^\pi(\mathbf{s},\mathbf{a})$ *for all* $(\mathbf{s},\mathbf{a})$

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* $\mathbf{s} \in D$

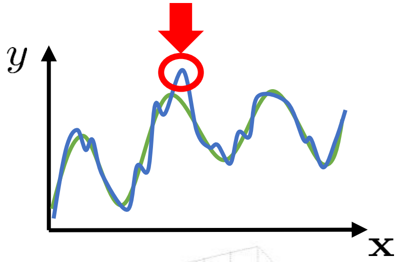
Kumar, Zhou, Tucker, Levine. **Conservative Q-Learning for Offline Reinforcement Learning.** ‘20



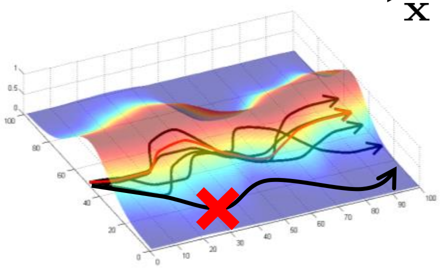
Why is offline RL difficult?



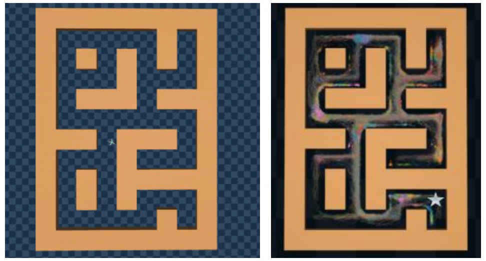
How do we design offline RL algorithms?



Conservative Q-Learning

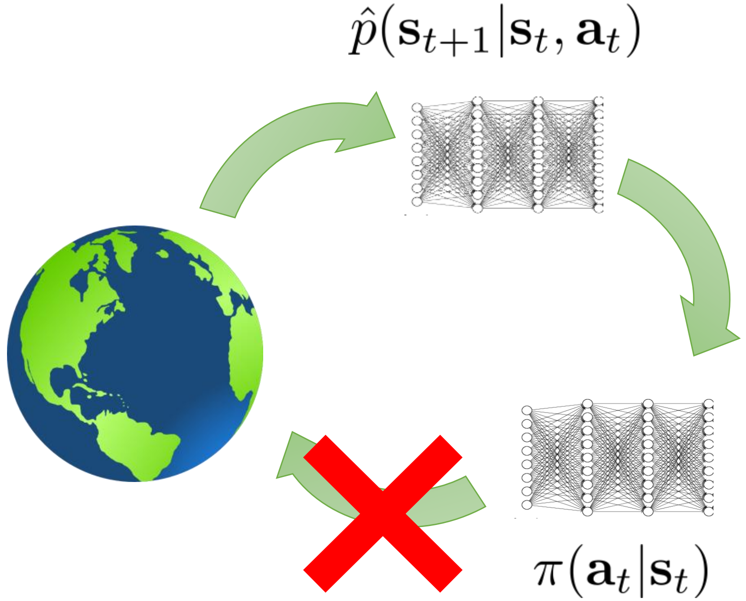


Model-based offline RL

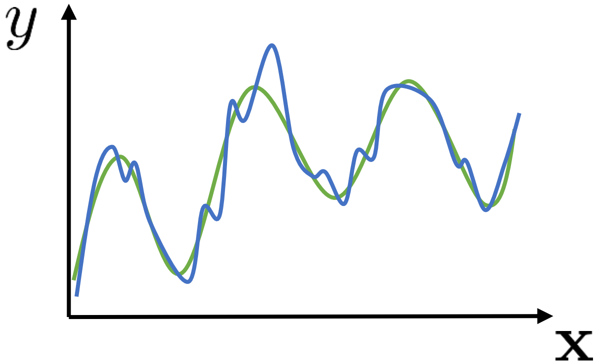


How do we evaluate offline RL methods?

How does **model-based** RL work?

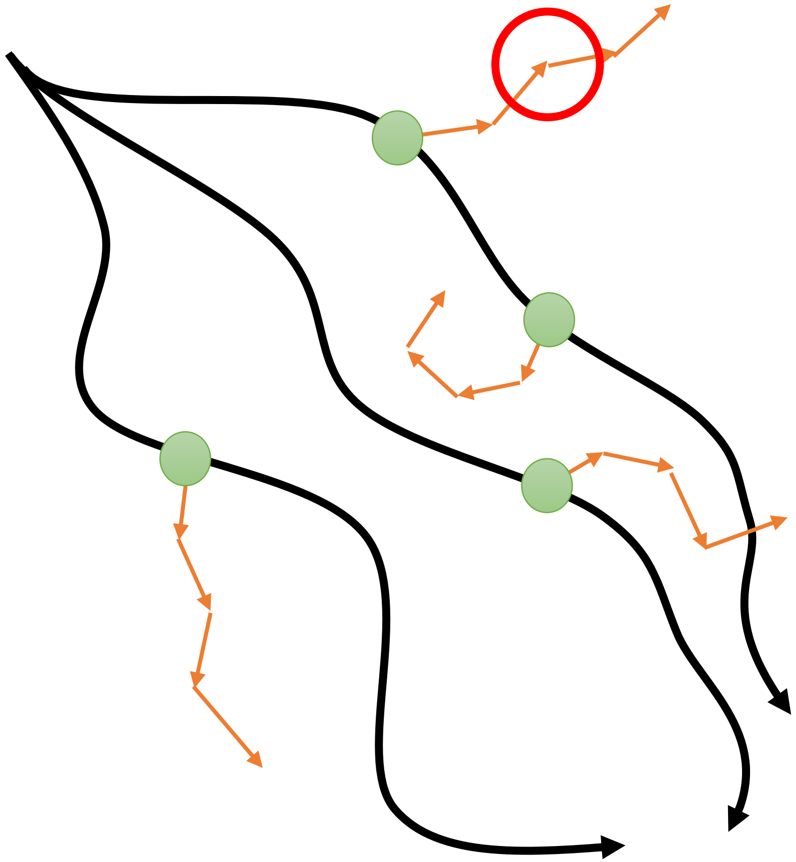


what goes wrong when we can't collect more data?



…so the model's predictions are invalid

these states are OOD



the model answers "what if" questions

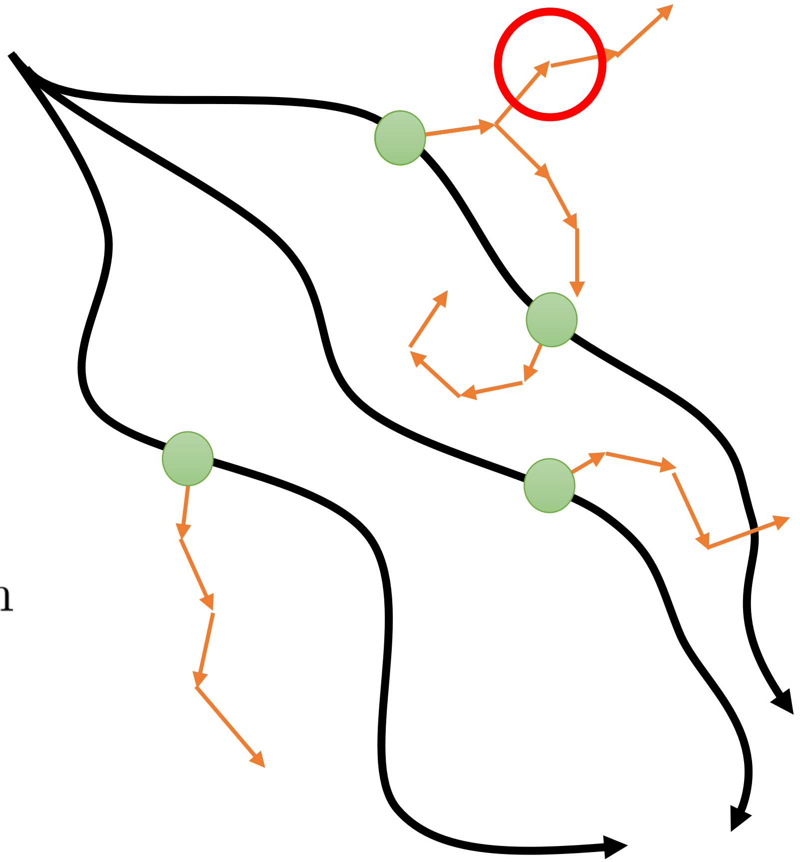
MOPO: Model-Based Offline Policy Optimization

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)

MOPO: Theoretical Analysis

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

we can represent the value function

model error is bounded (above) by $u(\mathbf{s},\mathbf{a})$

Theorem 4.4. *Under Assumption 4.2 and 4.3, the learned policy $\hat{\pi}$ in MOPO (Algorithm 1) satisfies*

true return of policy trained under model

$$\eta_M(\hat{\pi}) \geq \sup_{\pi}\{\eta_M(\pi) - 2\lambda\epsilon_u(\pi)\} \tag{11}$$

$$\epsilon_u(\pi) := \mathop{\bar{\mathbb{E}}}_{(s,a)\sim\rho_{\hat{T}}^{\pi}}[u(s,a)]$$

In particular, for all $\delta \geq \delta_{\min}$,

$$\eta_M(\hat{\pi}) \geq \eta_M(\pi^{\delta}) - 2\lambda\delta \tag{12}$$

$$\pi^{\delta} := \mathop{\arg\max}_{\pi:\epsilon_u(\pi)\leq\delta} \eta_M(\pi)$$

some implications:

$$\eta_M(\hat{\pi}) \geq \eta_M(\pi^{\mathrm{B}}) - 2\lambda\epsilon_u(\pi^{\mathrm{B}})$$

- improves over behavior policy

$$\eta_M(\hat{\pi}) \geq \eta_M(\pi^{\star}) - 2\lambda\epsilon_u(\pi^{\star})$$

- quantifies "optimality gap" in terms of model error

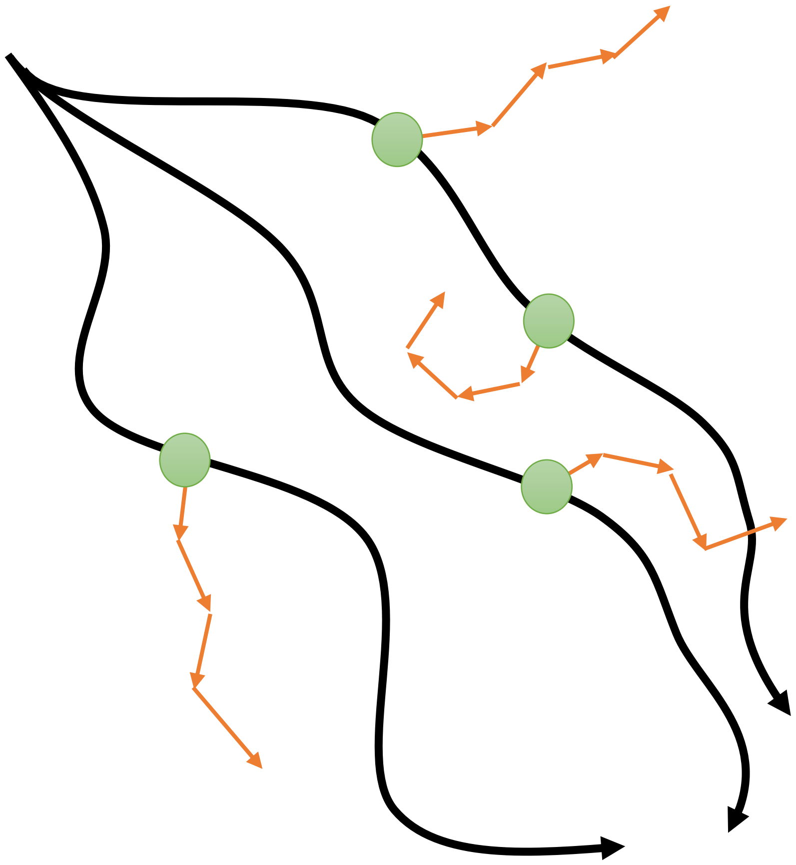
Yu*, Thomas*, Yu, Ermon, Zou, Levine, Finn, Ma. **MOPO: Model-Based Offline Policy Optimization.** '20

COMBO: 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

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

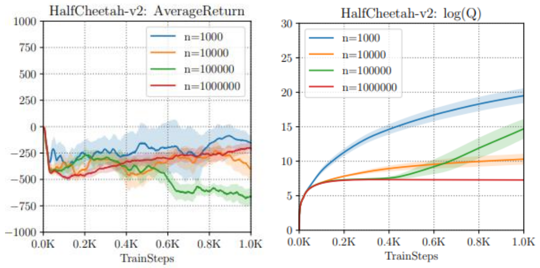
state-action tuples from the model



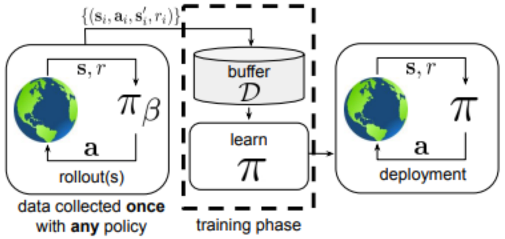
Intuition: if the model produces something that looks clearly different from real data, it's easy for the Q-function to make it look bad

Dataset type	Environment	BC	COMBO (ours)	MOPO	CQL	SAC-off	BEAR	BRAC-p	BRAC-v
random	halfcheetah	2.1	**38.8**	35.4	35.4	30.5	25.1	24.1	31.2
random	hopper	1.6	**17.9**	11.7	10.8	11.3	11.4	11.0	12.2
random	walker2d	9.8	7.0	**13.6**	7.0	4.1	7.3	-0.2	1.9
medium	halfcheetah	36.1	**54.2**	42.3	44.4	-4.3	41.7	43.8	46.3
medium	hopper	29.0	**94.9**	28.0	86.6	0.8	52.1	32.7	31.1
medium	walker2d	6.6	75.5	17.8	74.5	0.9	59.1	77.5	**81.1**
medium-replay	halfcheetah	38.4	**55.1**	53.1	46.2	-2.4	38.6	45.4	47.7
medium-replay	hopper	11.8	**73.1**	67.5	48.6	3.5	33.7	0.6	0.6
medium-replay	walker2d	11.3	**56.0**	39.0	32.6	1.9	19.2	-0.3	0.9
med-expert	halfcheetah	35.8	**90.0**	63.3	62.4	1.8	53.4	44.2	41.9
med-expert	hopper	111.9	**111.1**	23.7	111.0	1.6	96.3	1.9	0.8
med-expert	walker2d	6.4	96.1	44.6	**98.7**	-0.1	40.1	76.9	81.6

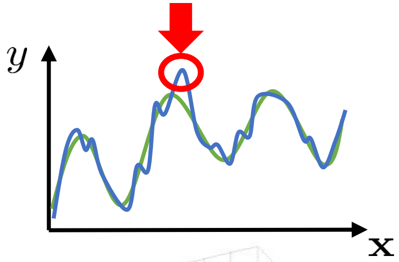
Yu, Kumar, Rafailov, Rajeswaran, Levine, Finn. **COMBO: Conservative Offline Model-Based Policy Optimization**. 2021.



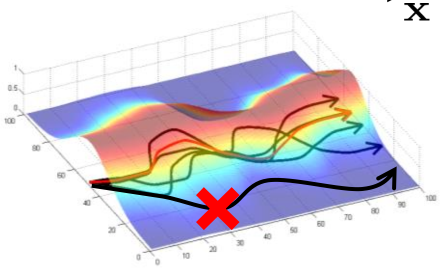
Why is offline RL difficult?



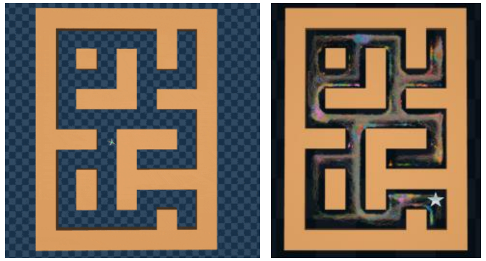
How do we design offline RL algorithms?



Conservative Q-Learning

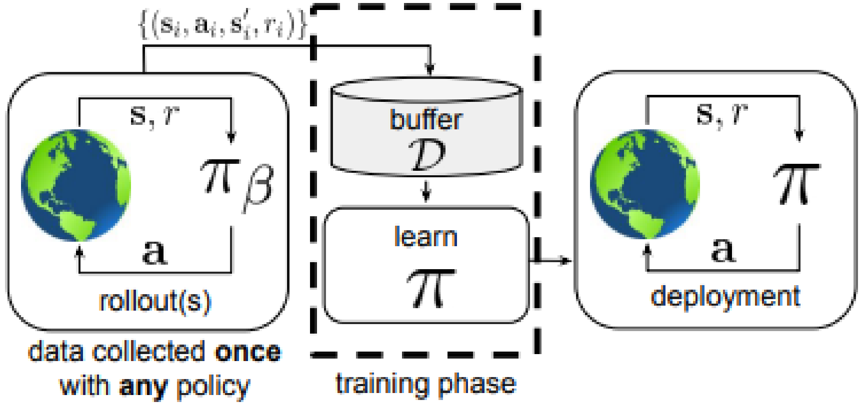


Model-based offline RL



How do we evaluate offline RL methods?

How do we evaluate offline RL methods?



maybe just train a reference policy with RL?

typical protocol in prior work:

1. train π_β with *online* RL
2. either collect data throughout training

OR

2. collect data from final policy π_β

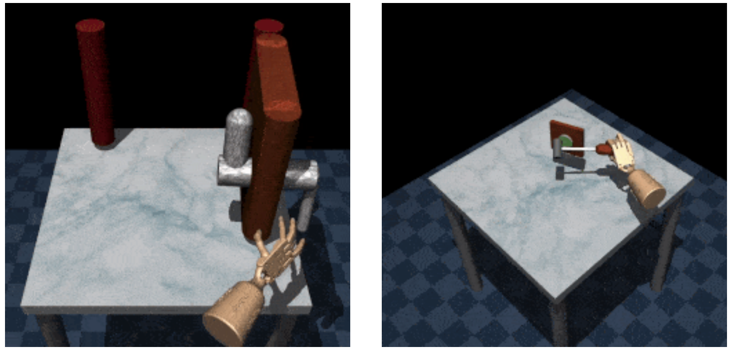
this is a really bad idea

- If you already have a good policy, why bother with offline RL?
- In the real world, data might come from non-Markovian "policies"
  - Human users
  - Hand-engineered policies
- Must use data that is **representative of real-world settings** and **leaves lots of room for improvement**
- Offline RL **must learn policies that are much better than the behavior policy!**

without testing these properties, we **cannot** trust that our algorithms are good!

D4RL: Datasets for Data-Driven Deep RL

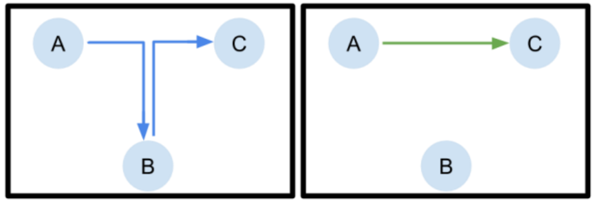
What are some important principles to keep in mind?



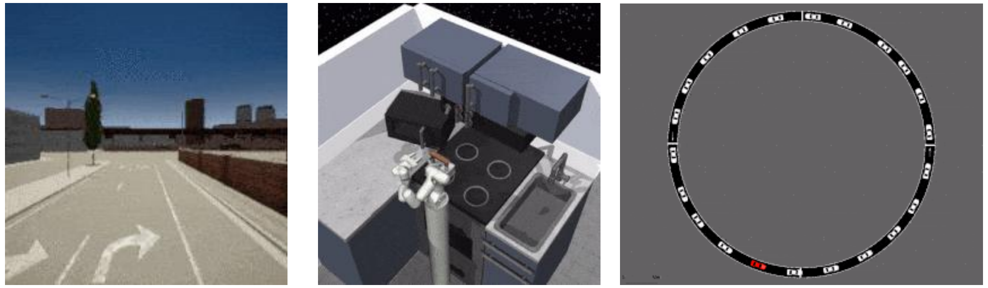
simulation & human data from Rajeswaran et al.

Data from non-RL policies, including data from humans

Stitching: data where dynamic programming can find much better solutions



Realistic tasks



Fu, Kumar, Nachum Tucker, Levine. **D4RL: Datasets for Data-Driven Deep Reinforcement Learning.** ‘20

How does CQL compare?

"**1%**" dataset from Agarwal et al.

Task Name	QR-DQN	REM	CQL($\mathcal{H}$)
Pong (1%)	-13.8	-6.9	**19.3**
Breakout	7.9	11.0	**61.1**
Q*bert	383.6	[illegible]	[illegible]
Seaquest	672.9	[illegible]	[illegible]
Asterix*	166.3	386.5	[illegible]

1.5 – 6x better

baseline: just clone the data

Domain	Task Name	BC	SAC	BEAR	BRAC-p	BRAC-v	CQL($\mathcal{H}$)	CQL(ρ)
AntMaze	antmaze-umaze	65.0	0.0	**73.0**	50.0	70.0	**74.0**	**73.5**
	antmaze-umaze-diverse	55.0	0.0	61.0	40.0	70.0	**84.0**	61.0
	antmaze-medi	[illegible]	[illegible]	[illegible]	[illegible]	0.0	**61.2**	4.6
	antmaze-medi	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	**53.7**	5.1
	antmaze-large	[illegible]	[illegible]	[illegible]	0.0	[illegible]	**15.8**	3.2
	antmaze-large	[illegible]	[illegible]	[illegible]	0.0	0.0	**14.9**	2.3
Adroit	pen-human	[illegible]	[illegible]	[illegible]	8.1	0.6	37.5	**55.8**
	hammer-huma	[illegible]	[illegible]	[illegible]	0.3	[illegible]	**4.4**	2.1
	door-human	[illegible]	[illegible]	[illegible]	-0.3	[illegible]	**9.9**	**9.1**
	relocate-huma	[illegible]	[illegible]	[illegible]	-0.3	-0.3	0.20	**0.35**
	pen-cloned	[illegible]	[illegible]	[illegible]	1.6	-2.5	39.2	40.3
	hammer-cloned	[illegible]	[illegible]	[illegible]	0.3	[illegible]	2.1	**5.7**
	door-cloned	[illegible]	[illegible]	[illegible]	-0.1	[illegible]	0.4	**3.5**
	relocate-clone	[illegible]	[illegible]	[illegible]	-0.3	-0.3	**-0.1**	**-0.1**
Kitchen	kitchen-comp	[illegible]	[illegible]	[illegible]	0.0	[illegible]	**43.8**	31.3
	kitchen-partia	[illegible]	[illegible]	[illegible]	0.0	[illegible]	**49.8**	**50.1**
	kitchen-undirected	47.5	2.5	47.2	0.0	[illegible]	**51.0**	**52.4**

nothing works on the harder mazes?

nothing beats behavioral cloning?

"**infinitely**" better

1.5-3x better

up to 5x better

1.1 – 1.3x better

CQL seems to work quite well across many tasks!

And we seem to know *why* it works!

But there is still plenty of room for improvement…

Kumar, Zhou, Tucker, Levine. **Conservative Q-Learning for Offline Reinforcement Learning.** '20

Which offline RL method should I use?

CQL-like methods

seems to get best results on external benchmarks (e.g., D4RL)

from my experience, harder to use with **online finetuning** (too conservative)

modifies the critic

these are purely empirical observations, and they might change with better implementations!

seems to imply we can combine to get the best of both worlds

we have not been successful at this so far

AWR-like methods

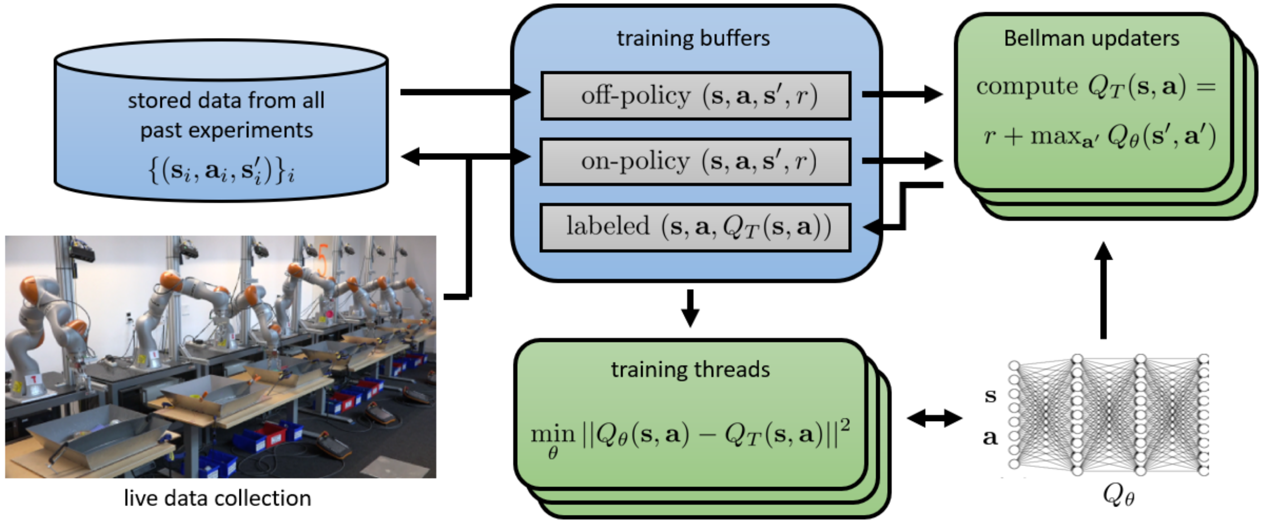
seems to get best results on external benchmarks when **finetuning**

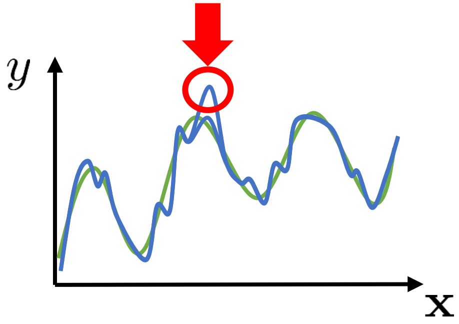
seems to be much worse than CQL on benchmarks (e.g., D4RL) in fully offline mode

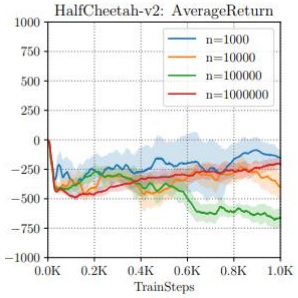
modifies the actor

Summary and takeaways

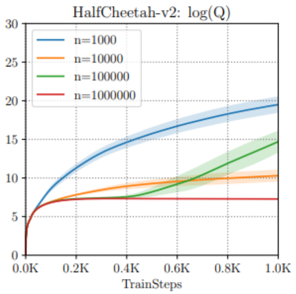
- Offline RL algorithms can be built out of Q-Learning methods
- But this can fail if there is **narrow coverage** (often the case in IL+RL)
- Offline RL is difficult because of **distributional shift**
- Solutions typically mitigate this in some way
- AWR & AWAC: **implicit constraint** formed by using a weighted imitation learning objective (weighted using the critic!)
- CQL: **conservative** critic objective that directly avoids overestimation
- Model-based offline RL: **similar principle**, avoid overestimating by penalizing value far from data







how well it does



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