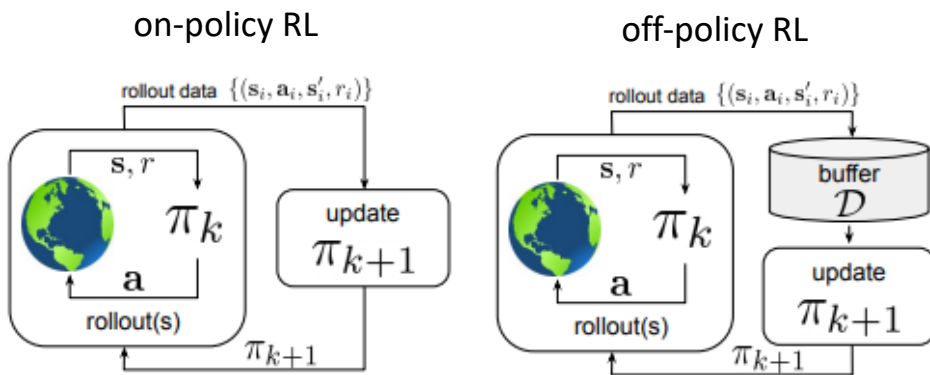


# Batch / Offline Reinforcement Learning

With thanks to Emma Brunskill, Scott Fujimoto, Pieter Abbeel, George Tucker, Sergey Levine, Bilal Piot,  
Yuxin Chen, Yuejie Chi

# On-policy vs off-policy vs offline RL



Formally:

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

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

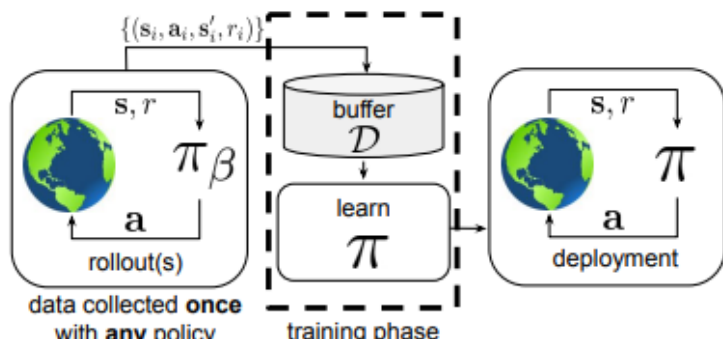
$$a \sim \pi_\beta(a|s)$$

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

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

← generally **not** known

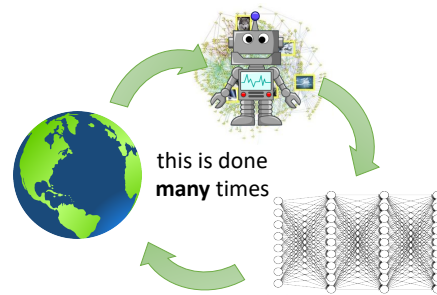
## offline reinforcement learning



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

# Why is this important?

- Collecting new data may be expensive / infeasible
- We may have access to existing/historical data instead



# Problem formulation

**A historical dataset**  $\mathcal{D} = \{(s^{(i)}, a^{(i)}, s'^{(i)})\}$ :  $N$  independent copies of

$$s \sim \rho^b, \quad a \sim \pi^b(\cdot | s), \quad s' \sim P(\cdot | s, a)$$

for some state distribution  $\rho^b$  and behavior policy  $\pi^b$

**Goal:** given some test distribution  $\rho$  and accuracy level  $\varepsilon$ , find an  $\varepsilon$ -optimal policy  $\hat{\pi}$  based on  $\mathcal{D}$  obeying

$$V^*(\rho) - V^{\hat{\pi}}(\rho) = \mathbb{E}_{s \sim \rho} [V^*(s)] - \mathbb{E}_{s \sim \rho} [V^{\hat{\pi}}(s)] \leq \varepsilon$$

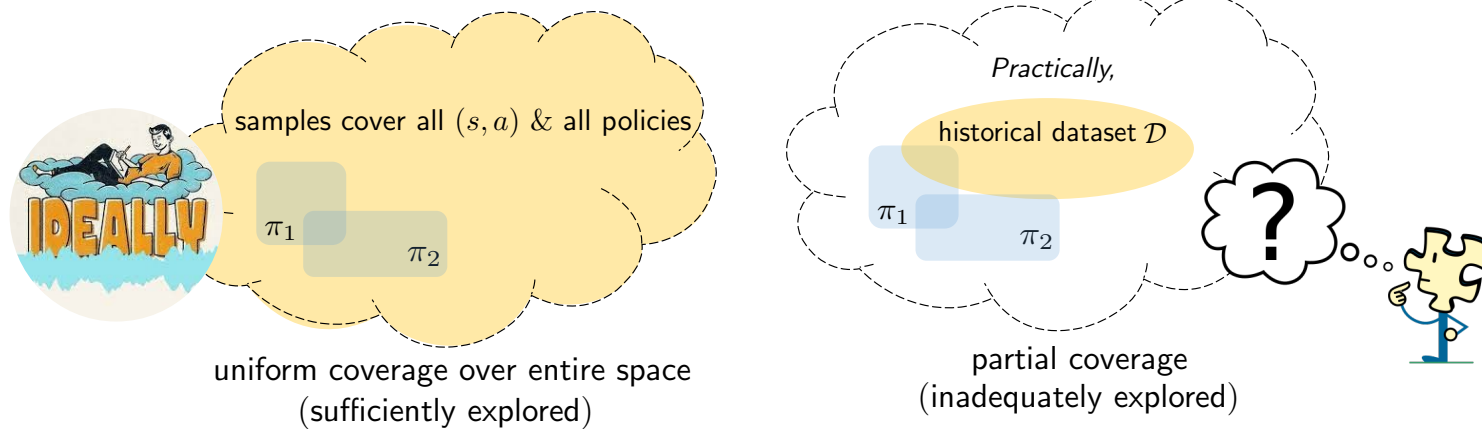
— *in a sample-efficient manner*

# Challenges of offline / batch RL (1)

- **Distribution shift:**

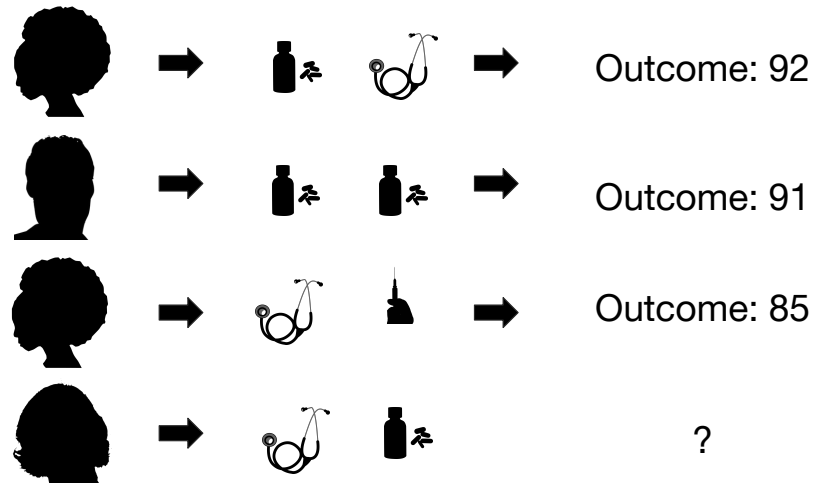
distribution( $\mathcal{D}$ )  $\neq$  target distribution under  $\pi^*$

- **Partial coverage of state-action space:**



## Challenges of offline / batch RL (2)

- Data is *censored*: we only observe outcomes for decisions made (and need to generalize from them)



- Need for *counterfactual inference*: what would happen if one would take a different action?
- Often we do not observe rewards, just states and actions!

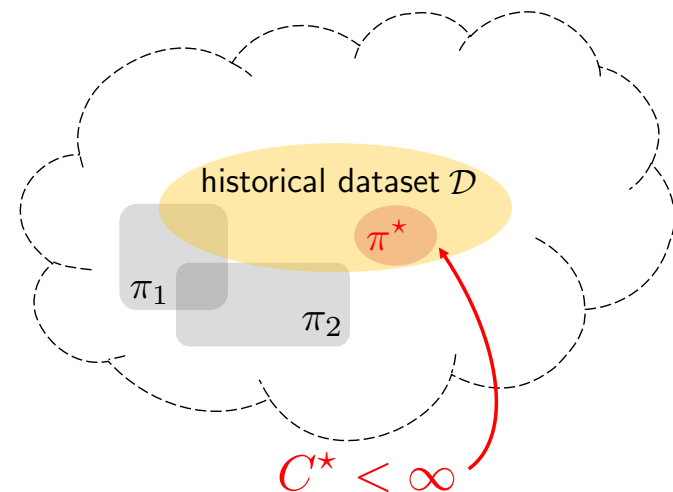
# Dataset quality assessment

## Single-policy concentrability coefficient

$$C^* := \max_{s,a} \frac{d^{\pi^*}(s,a)}{d^{\pi^b}(s,a)} = \left\| \frac{\text{occupancy density of } \pi^*}{\text{occupancy density of } \pi^b} \right\|_{\infty} \geq 1$$

where  $d^{\pi}(s,a) = (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t \mathbb{P}((s^t, a^t) = (s, a) | \pi)$

- captures distributional shift
- allows for partial coverage

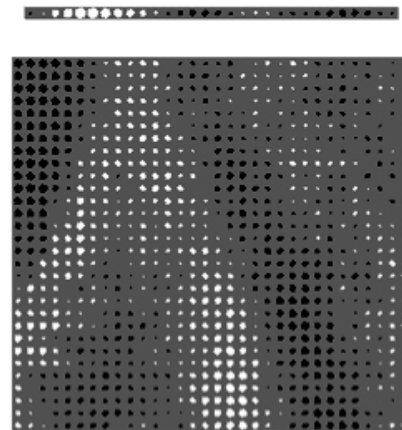
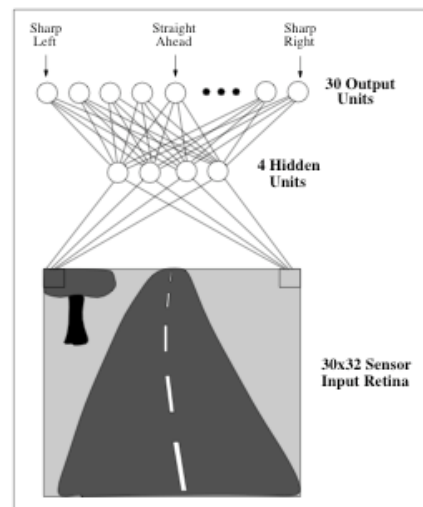


## Classes of algorithms

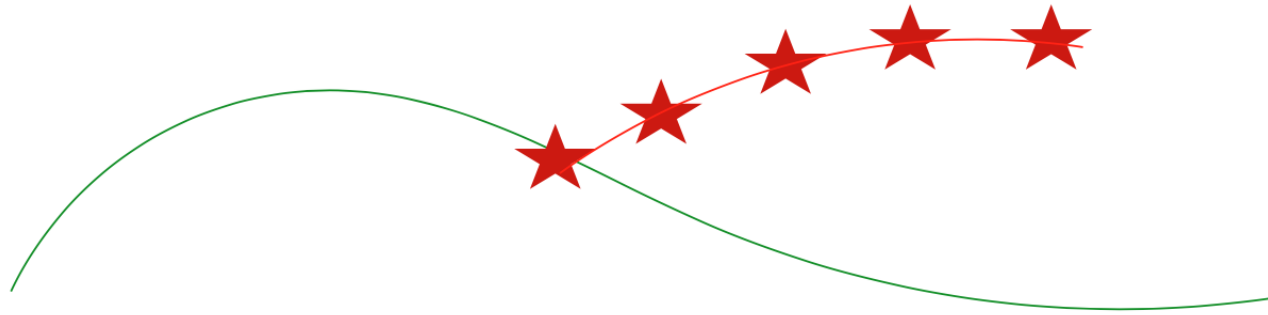
1. Behavior cloning (no rewards required)
2. Learn a model, use it for model-based RL (LSTD, LSPI)
3. Pessimistic algorithms (require rewards)
4. Inverse RL (learn reward function from data, use it for RL agent)

# Part 1: Behavior cloning

- Take dataset  $\mathcal{D}$ , learn a policy from states to actions
- Often uses a rich policy class (neural net)



## Problem: compounding errors



- Error at time  $t$  with probability  $\epsilon$
- Approximate intuition:  $\mathbb{E}[\text{Total errors}] \leq \epsilon(T + (T - 1) + (T - 2) \dots + 1) \propto \epsilon T^2$

# One solution: dataset aggregation

Initialize  $\mathcal{D} \leftarrow \emptyset$ .  
Initialize  $\hat{\pi}_1$  to any policy in  $\Pi$ .  
**for**  $i = 1$  **to**  $N$  **do**  
    Let  $\pi_i = \beta_i \pi^* + (1 - \beta_i) \hat{\pi}_i$ .  
    Sample  $T$ -step trajectories using  $\pi_i$ .  
    Get dataset  $\mathcal{D}_i = \{(s, \pi^*(s))\}$  of visited states by  $\pi_i$   
    and actions given by expert.  
    Aggregate datasets:  $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_i$ .  
    Train classifier  $\hat{\pi}_{i+1}$  on  $\mathcal{D}$ .  
**end for**  
**Return** best  $\hat{\pi}_i$  on validation.

- Idea: Get more labels of the expert action along the path taken by the policy computed by behavior cloning
- Obtains a stationary deterministic policy with good performance under its induced state distribution

## Part 2: Model-based algorithms for least-squares regression

- Given value function approximation  $\hat{v}(s, \mathbf{w}) \approx v_\pi(s)$
- And *experience*  $\mathcal{D}$  consisting of  $\langle \text{state}, \text{value} \rangle$  pairs

$$\mathcal{D} = \{ \langle s_1, v_1^\pi \rangle, \langle s_2, v_2^\pi \rangle, \dots, \langle s_T, v_T^\pi \rangle \}$$

- Which parameters  $\mathbf{w}$  give the *best fitting* value fn  $\hat{v}(s, \mathbf{w})$ ?
- **Least squares** algorithms find parameter vector  $\mathbf{w}$  minimising sum-squared error between  $\hat{v}(s_t, \mathbf{w})$  and target values  $v_t^\pi$ ,

$$\begin{aligned} LS(\mathbf{w}) &= \sum_{t=1}^T (v_t^\pi - \hat{v}(s_t, \mathbf{w}))^2 \\ &= \mathbb{E}_{\mathcal{D}} [(v^\pi - \hat{v}(s, \mathbf{w}))^2] \end{aligned}$$

## Least-squares regression

- At minimum of  $LS(\mathbf{w})$ , the expected update must be zero

$$\mathbb{E}_{\mathcal{D}} [\Delta \mathbf{w}] = 0$$

$$\alpha \sum_{t=1}^T \mathbf{x}(s_t) (v_t^\pi - \mathbf{x}(s_t)^\top \mathbf{w}) = 0$$

$$\sum_{t=1}^T \mathbf{x}(s_t) v_t^\pi = \sum_{t=1}^T \mathbf{x}(s_t) \mathbf{x}(s_t)^\top \mathbf{w}$$

$$\mathbf{w} = \left( \sum_{t=1}^T \mathbf{x}(s_t) \mathbf{x}(s_t)^\top \right)^{-1} \sum_{t=1}^T \mathbf{x}(s_t) v_t^\pi$$

- For  $N$  features, direct solution time is  $O(N^3)$
- Incremental solution time is  $O(N^2)$  using Sherman-Morrison

## Least-squares algorithms RL-style

- We do not know true values  $v_t^\pi$
- In practice, our “training data” must use noisy or biased samples of  $v_t^\pi$ 
  - LSMC Least Squares Monte-Carlo uses return  
 $v_t^\pi \approx G_t$
  - LSTD Least Squares Temporal-Difference uses TD target  
 $v_t^\pi \approx R_{t+1} + \gamma \hat{v}(S_{t+1}, \mathbf{w})$
  - LSTD( $\lambda$ ) Least Squares TD( $\lambda$ ) uses  $\lambda$ -return  
 $v_t^\pi \approx G_t^\lambda$
- In each case solve directly for fixed point of MC / TD / TD( $\lambda$ )

## LSMC and LSTD

LSMC

$$0 = \sum_{t=1}^T \alpha (G_t - \hat{v}(S_t, \mathbf{w})) \mathbf{x}(S_t)$$

$$\mathbf{w} = \left( \sum_{t=1}^T \mathbf{x}(S_t) \mathbf{x}(S_t)^\top \right)^{-1} \sum_{t=1}^T \mathbf{x}(S_t) G_t$$

LSTD

$$0 = \sum_{t=1}^T \alpha (R_{t+1} + \gamma \hat{v}(S_{t+1}, \mathbf{w}) - \hat{v}(S_t, \mathbf{w})) \mathbf{x}(S_t)$$

$$\mathbf{w} = \left( \sum_{t=1}^T \mathbf{x}(S_t) (\mathbf{x}(S_t) - \gamma \mathbf{x}(S_{t+1}))^\top \right)^{-1} \sum_{t=1}^T \mathbf{x}(S_t) R_{t+1}$$

## LSMC and LSTD

LSMC

$$0 = \sum_{t=1}^T \alpha (G_t - \hat{v}(S_t, \mathbf{w})) \mathbf{x}(S_t)$$

$$\mathbf{w} = \left( \sum_{t=1}^T \mathbf{x}(S_t) \mathbf{x}(S_t)^\top \right)^{-1} \sum_{t=1}^T \mathbf{x}(S_t) G_t$$

LSTD

$$0 = \sum_{t=1}^T \alpha (R_{t+1} + \gamma \hat{v}(S_{t+1}, \mathbf{w}) - \hat{v}(S_t, \mathbf{w})) \mathbf{x}(S_t)$$

$$\mathbf{w} = \left( \sum_{t=1}^T \mathbf{x}(S_t) (\mathbf{x}(S_t) - \gamma \mathbf{x}(S_{t+1}))^\top \right)^{-1} \sum_{t=1}^T \mathbf{x}(S_t) R_{t+1}$$

## Theoretical properties: Policy evaluation

On/Off-Policy	Algorithm	Table Lookup	Linear	Non-Linear
On-Policy	MC	✓	✓	✓
	LSMC	✓	✓	-
	TD	✓	✓	X
	LSTD	✓	✓	-
Off-Policy	MC	✓	✓	✓
	LSMC	✓	✓	-
	TD	✓	X	X
	LSTD	✓	✓	-

## Theoretical properties: Control

Algorithm	Table Lookup	Linear	Non-Linear
Monte-Carlo Control	✓	(✓)	✗
Sarsa	✓	(✓)	✗
Q-learning	✓	✗	✗
LSPI	✓	(✓)	-

(✓) = chatters around near-optimal value function

## More generally: Model-based RL

- Take the data (some of it not labelled by rewards possibly) and fit a model
- Use your favourite model-based RL algorithm to plan/learn

## Part 3: Pessimism in the face of uncertainty

- *Conservative* approach
- Assume that states or state-action pairs not visited are bad
- Use a penalty to avoid the new policy visiting them

# Value iteration with lower confidence bounds

**Pessimism in the face of uncertainty:** penalize value estimate of those  $(s, a)$  pairs that were poorly visited [Jin et al., 2021, Rashidinejad et al., 2021]

**Algorithm:** value iteration w/ lower confidence bounds

- compute empirical estimate  $\hat{P}$  of  $P$
- initialize  $\hat{Q} = 0$ , and repeat

$$\hat{Q}(s, a) \leftarrow \max \left\{ r(s, a) + \gamma \langle \hat{P}(\cdot | s, a), \hat{V} \rangle - \underbrace{b(s, a; \hat{V})}_{\text{Bernstein-style confidence bound}}, 0 \right\}$$

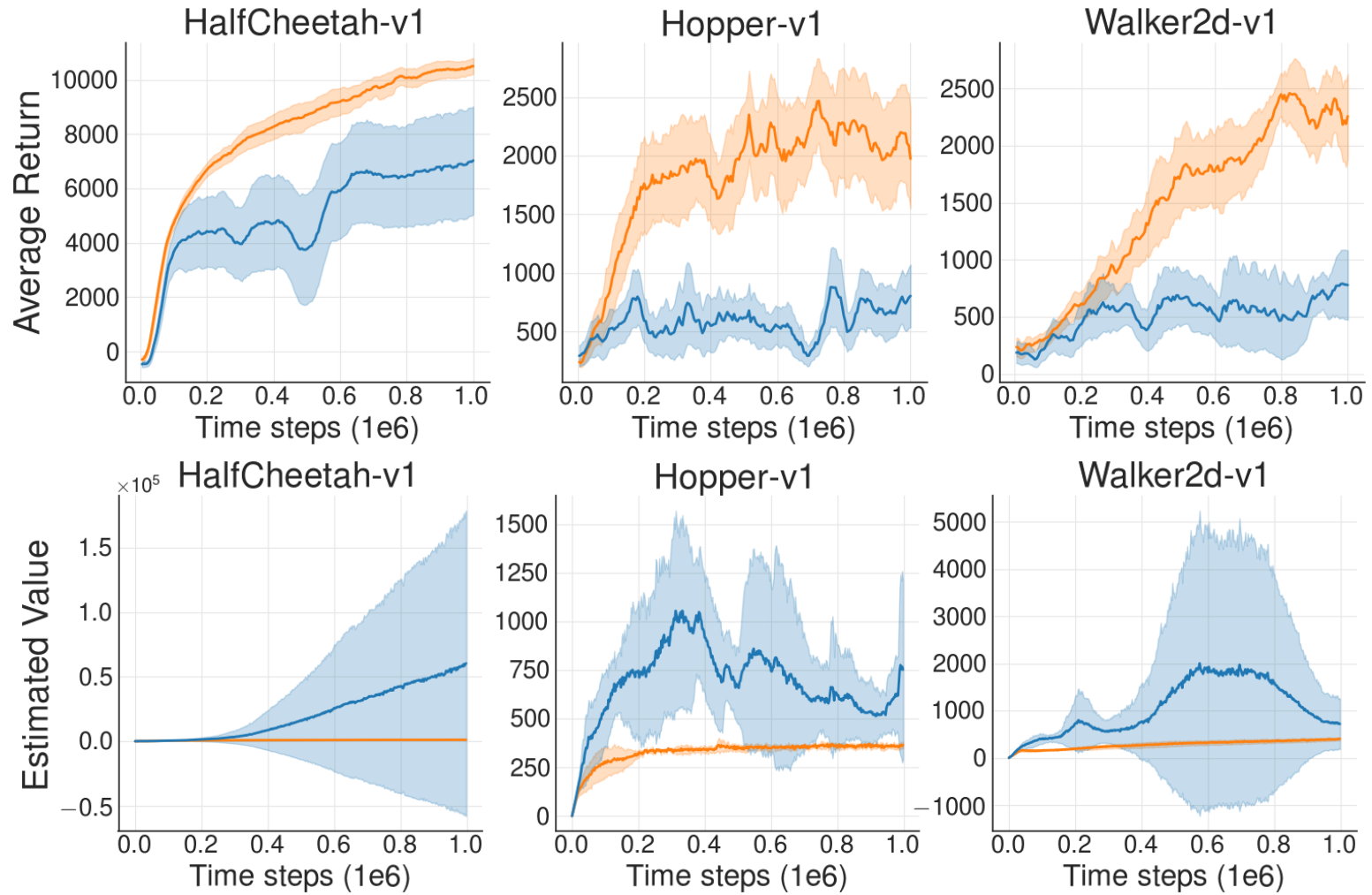
for all  $(s, a)$ , where  $\hat{V}(s) = \max_a \hat{Q}(s, a)$

Q-learning version exists as well

## Alternative approach: Batch-Constrained RL

- Do NOT try to optimize value function/policy everywhere, if you only have a limited batch
- Instead, only look at policies  $\pi$  such that the batch contains pairs  $(s, a)$  that  $\pi$  visits

# Motivation: Two versions of DDPG



## Motivation: Two versions of DDPG

- Orange agent interacts with the environment in a standard RL loop: collect data, put it in replay buffer, train, repeat
- Blue agent is trained with data collected by the orange agent concurrently
- Even though the data and algorithm are the same, being off-policy throws the learning off!!!

## Why? Extrapolation error

$$Q(s, a) \leftarrow r + \gamma Q(s', a')$$

GIVEN

GENERATED

$$Q(s, a) \leftarrow r + \gamma Q(s', a')$$

$(s', a') \notin \text{Dataset} \rightarrow Q(s', a') = \mathbf{bad}$   
 $\rightarrow Q(s, a) = \mathbf{bad}$

## Batch-constrained RL

1.  $a \sim \pi(s)$  such that  $(s, a) \in Dataset$ .
2.  $a \sim \pi(s)$  such that  $(s', \pi(s')) \in Dataset$ .
3.  $a \sim \pi(s)$  such that  $Q(s, a)$  is maxed.

## Batch-constrained Q-learning

First imitate dataset via generative model:

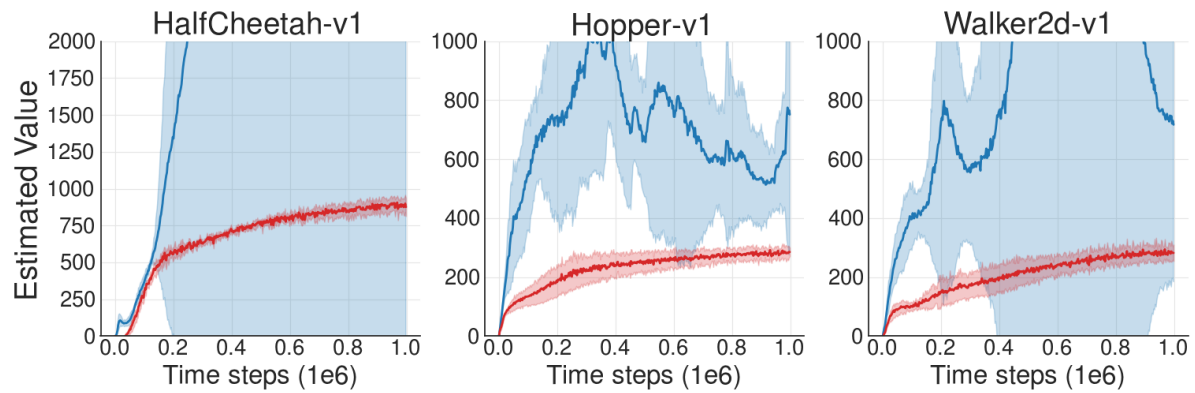
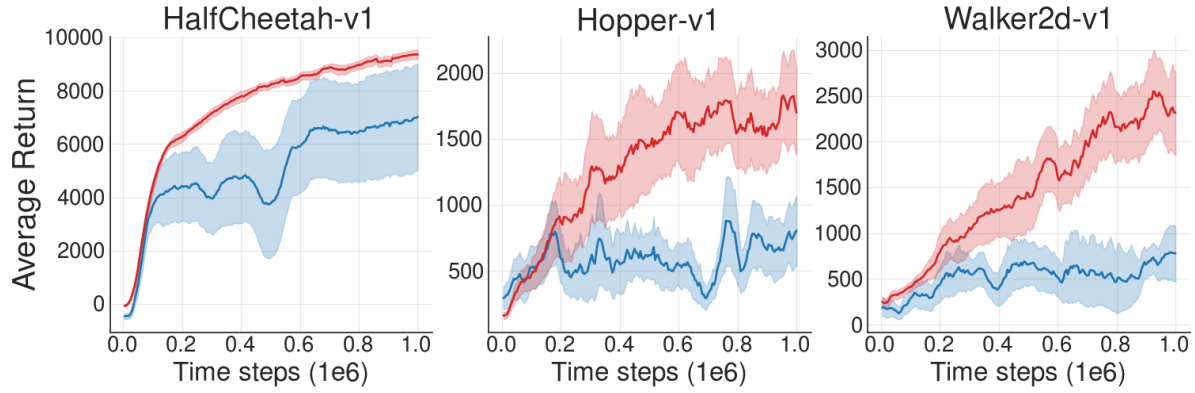
$$G(a|s) \approx P_{Dataset}(a|s).$$

$$\pi(s) = \operatorname{argmax}_{a_i} Q(s, a_i), \text{ where } a_i \sim G$$

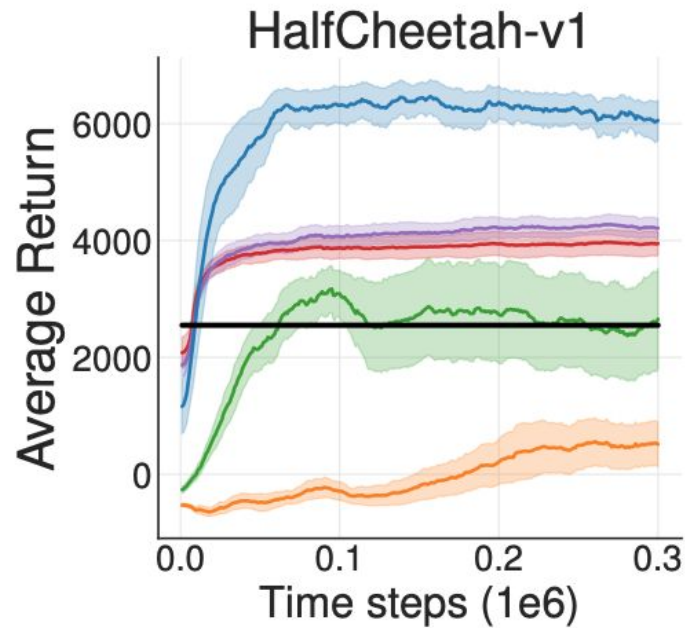
(i.e. select the best action that is likely under the dataset)

(+ some additional deep RL **magic**)

# Revisiting previous example



# BCQ comparison



BCQ figure from Fujimoto, Meger, Precup ICML 2019

BCQ DDPG DQN BC VAE-BC Behavioral<sub>17</sub>