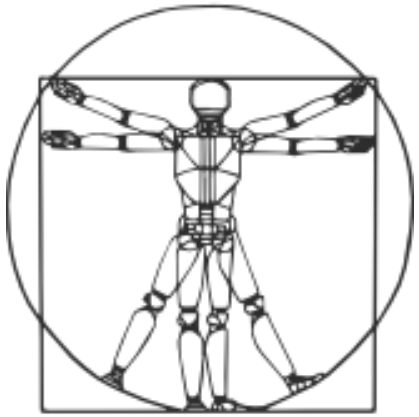


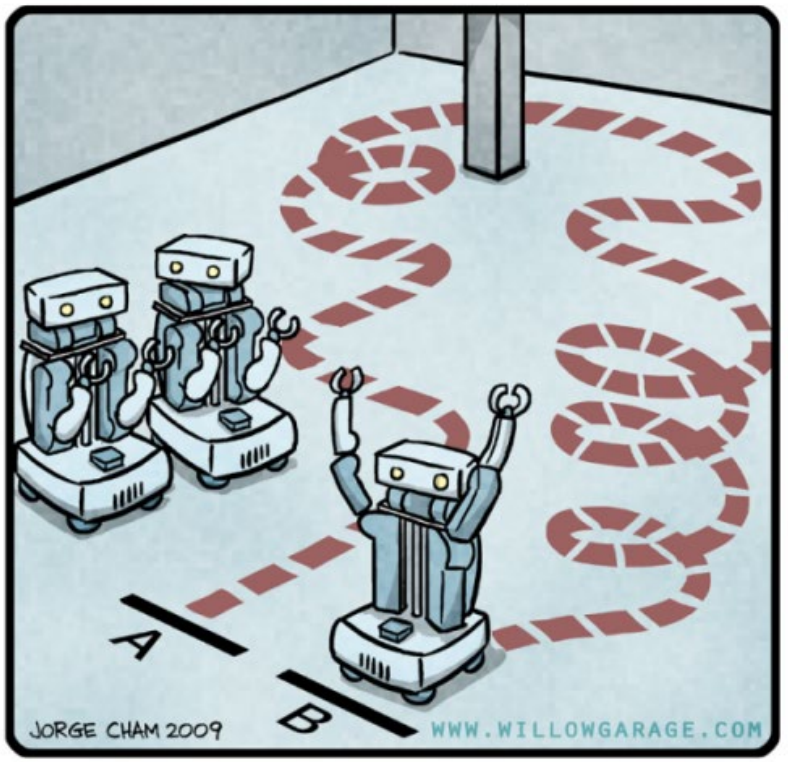
CSC2626
Imitation Learning for Robotics

Florian Shkurti

Week 3: Offline/Batch Reinforcement Learning

Today's agenda

- Reinforcement Learning Terminology
- Distribution Shift in Offline RL
- Offline RL with Policy Constraints
- Offline RL with Conservative Q-Estimates



Acknowledgments
Today's slides borrow very heavily from: Sergey Levine & Aviral Kumar (CSC285 and NeurIPS Offline RL workshop), Joelle Pineau (DLSS'16)

RL Terminology

- Episodic vs Non-episodic
- Tabular vs Function Approximation
- Exploration vs Exploitation
- Model-based vs Model-free
- Policy Optimization vs Value Function Estimation
- On-policy vs Off-Policy
- Batch (Offline) vs Online

Episodic vs Non-episodic RL methods

Episodic: optimize expected reward-to-go for finite time horizon

$$V_T^{\pi}(\mathbf{s}_0) = \mathbb{E}_{\mathbf{a}_t \sim \pi_t(\mathbf{a}|\mathbf{s}_t),\, \mathbf{s}_{t+1} \sim p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)} \left[\sum_{t=0}^{T} r(\mathbf{s}_t, \mathbf{a}_t) \right]$$

Often need to have a reset mechanism to bring back the system to state $\mathbf{s}_0$

Episodic vs Non-episodic RL methods

Episodic: optimize expected reward-to-go for finite time horizon

$$V_T^{\pi}(\mathbf{s}_0) = \mathbb{E}_{\mathbf{a}_t \sim \pi_t(\mathbf{a}|\mathbf{s}_t),\ \mathbf{s}_{t+1} \sim p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)} \left[\sum_{t=0}^{T} r(\mathbf{s}_t, \mathbf{a}_t) \right]$$

Often need to have a reset mechanism to bring back the system to state $\mathbf{s}_0$

Non-episodic: optimize expected discounted reward-to-go for infinite time horizon, i.e. a task may go on forever, no resets

$$V^{\pi}(\mathbf{s}_0) = \mathbb{E}_{\mathbf{a}_t \sim \pi(\mathbf{a}|\mathbf{s}_t),\ \mathbf{s}_{t+1} \sim p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)} \left[\sum_{t=0}^{\infty} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t) \right] \qquad \gamma \in (0, 1)$$

Tabular vs Function Approximation Methods

Tabular: discretize states and actions, represent them as a grid, and compute a policy or value function directly on states and actions (typically can enumerate small number of states and actions)

Tabular vs Function Approximation Methods

Tabular: discretize states and actions, represent them as a grid, and compute a policy or value function directly on states and actions (typically can enumerate small number of states and actions)

Function approximation: discrete or continuous states and actions, associate a feature representation $\phi(s, a)$ to each state action pair and compute a policy or value function in terms of features (typically useful for continuous or very large, but discrete, state-action spaces)

Exploration vs Exploitation in RL

Exploitation: act by using current estimates of dynamics and value function to improve task performance in the short term

Exploration vs Exploitation in RL

Exploitation: act by using current estimates of dynamics and value function to improve task performance in the short term

Exploration: act to improve current estimates of dynamics and value function to improve task performance in the long-term, even if it hurts short-term performance

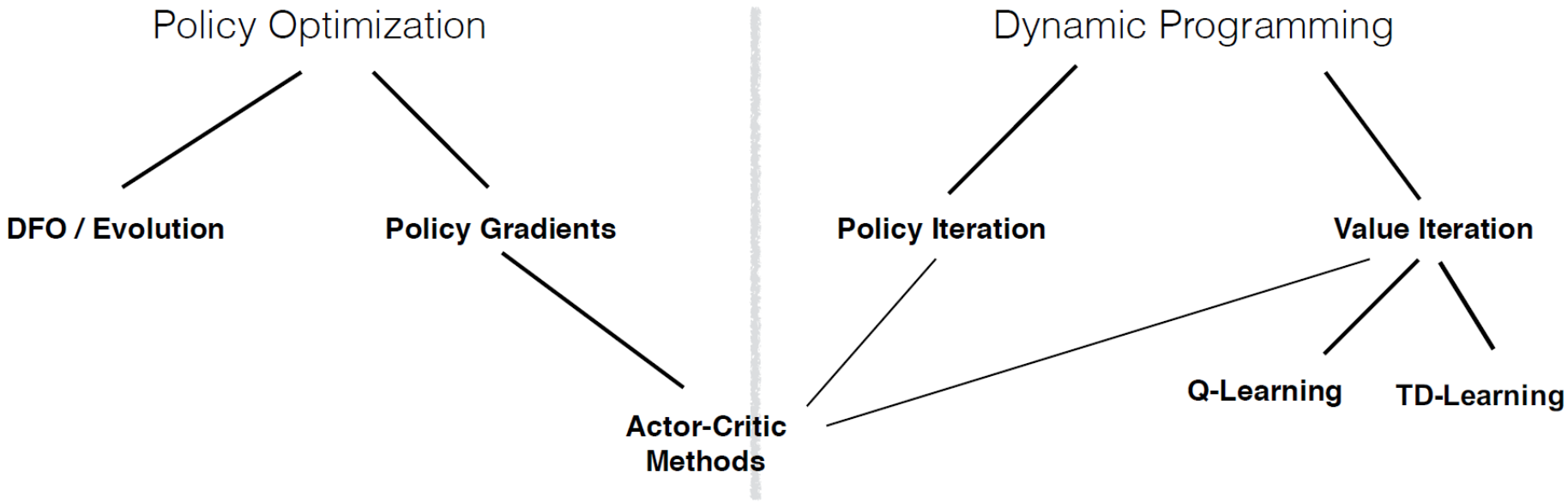
Model-based vs Model-free RL Methods

Model-based: estimate predictive models of the instantaneous reward as well as the dynamics model, and act by making value function predictions based on these models and optimizing the estimated value function. I.e. plan with "imagined" state transition and reward data.

Model-free: do not try to estimate models of reward or dynamics, but interact with the environment to optimize policy. Plan with real state transition and reward data.

Mixed: only trust your dynamics and reward model (imagined data) for a few steps in the near future, and then use real data.

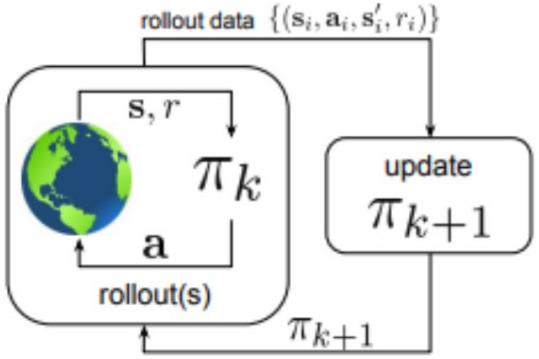
Policy Optimization vs Value Function Estimation



Credit: John Schulman

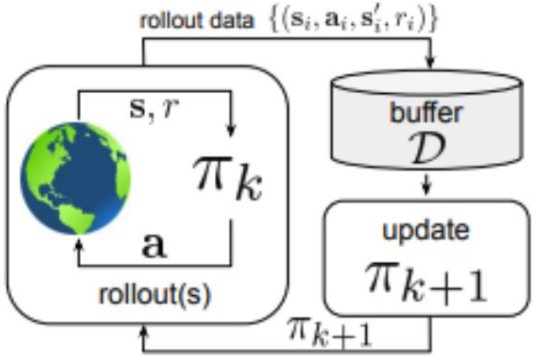
On-policy vs Off-policy Methods

on-policy RL



On-policy RL methods: improve the policy that acts on the environment using data collected from that same policy

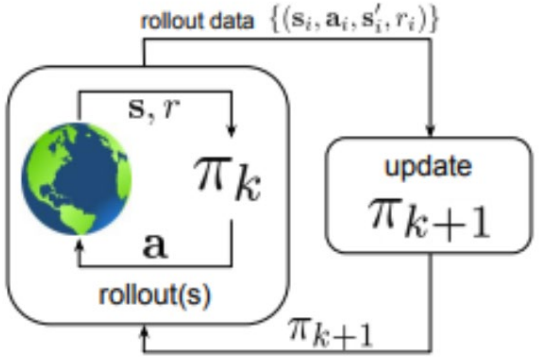
off-policy RL



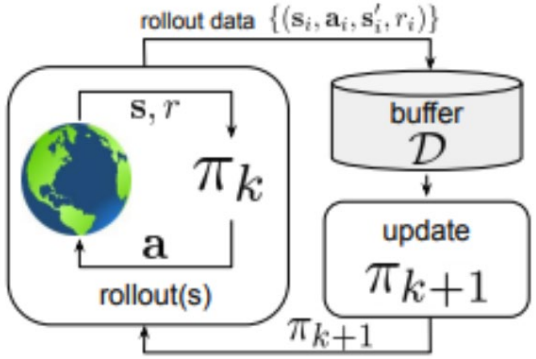
Off-policy RL methods: improve the policy that acts on the environment using data collected from ***any policy***

Batch (Offline) vs Online Methods

on-policy RL

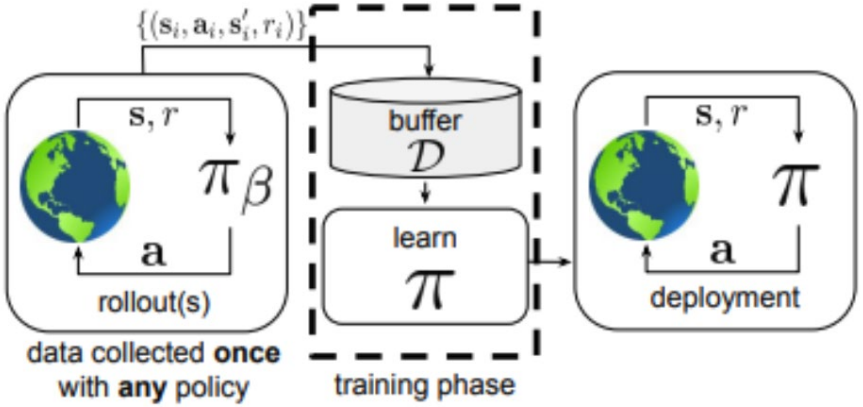


off-policy RL



Online RL methods: Can collect data over multiple rounds. Data distribution changes over time.

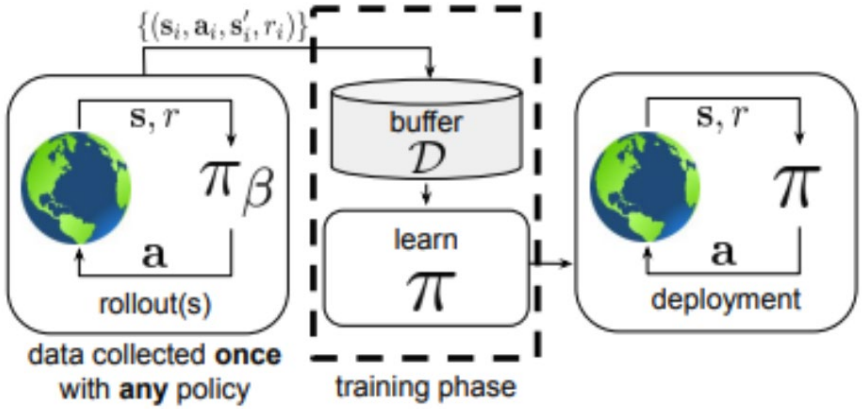
offline reinforcement learning



Batch/Offline RL methods: Can collect data only once from any policy. Data distribution is stationary.

Batch (Offline) vs Online Methods

offline reinforcement learning



Formally:

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

$$\mathbf{s} \sim d^{\pi_\beta}(\mathbf{s})$$
$$\mathbf{a} \sim \pi_\beta(\mathbf{a}|\mathbf{s})$$
$$\mathbf{s}' \sim p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$$
$$r \leftarrow r(\mathbf{s}, \mathbf{a})$$

generally **not** known

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

Supervised Learning

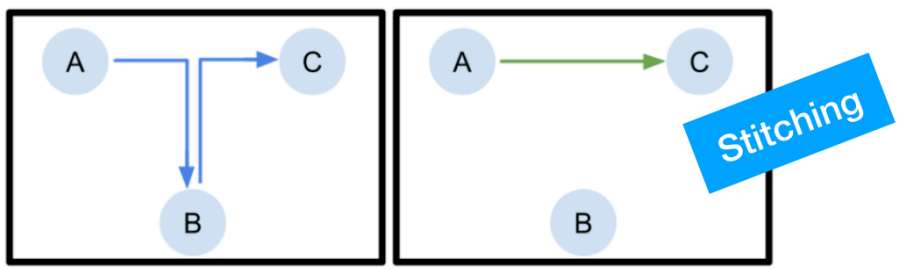
→ **Dog**

Can do as good as the dataset!

→ **Cat?**

Offline Reinforcement Learning

Can do **better** than the dataset!



Can show that Q-learning recovers optimal policy from random data.

Fu, **Kumar,** Nachum, Tucker, Levine. *D4RL: Datasets for Deep Data-Driven RL.* arXiv 2020.

Formalism and Notation

$$\max_{\pi} \sum_{t=1}^{\infty} \mathbb{E}_{s_t, a_t \sim \pi} \left[\gamma^t r\left(s_t, a_t\right) \right]$$

- Dataset construction:
  - Several trajectories:

Reward known

$$\mathcal{D} = \{\tau_1, \cdots, \tau_N\}, \quad \tau_i = \{s_i^t, a_i^t, r_i^t, s_i^{'t}\}_{t=1}^{H}$$

- Approximate "distribution" of states in the dataset: $\mathcal{D}(s)$
- Approximate distribution of actions at a given state in the dataset: $\mathcal{D}(a|s)$
- Will use notation for the behavior policy, $\pi_\beta(a|s) = \mathcal{D}(a|s)$

$\mathbf{s} \in \mathcal{S}$ – discrete or continuous state

$\mathbf{a} \in \mathcal{A}$ – discrete or continuous action

$\tau = \{\mathbf{s}_0, \mathbf{a}_0, \mathbf{s}_1, \mathbf{a}_1, ..., \mathbf{s}_T, \mathbf{a}_T\}$ – *trajectory*

$$\underbrace{\pi(\mathbf{s}_0, \mathbf{a}_0, \ldots, \mathbf{s}_T, \mathbf{a}_T)}_{\pi(\tau)} = p(\mathbf{s}_1) \prod_{t=0}^{T} \pi(\mathbf{a}_t|\mathbf{s}_t) p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$$

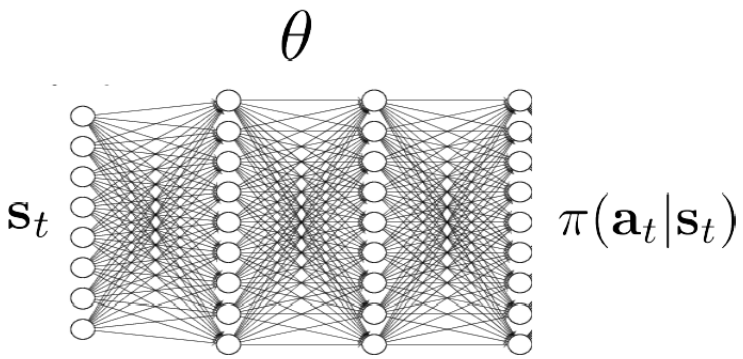
$d_t^\pi(\mathbf{s}_t)$ – state marginal of $\pi(\tau)$ at t

$d^\pi(\mathbf{s}) = \frac{1}{1-\gamma} \sum_{t=0}^{T} \gamma^t d_t^\pi(\mathbf{s}_t)$ – "visitation frequency"

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

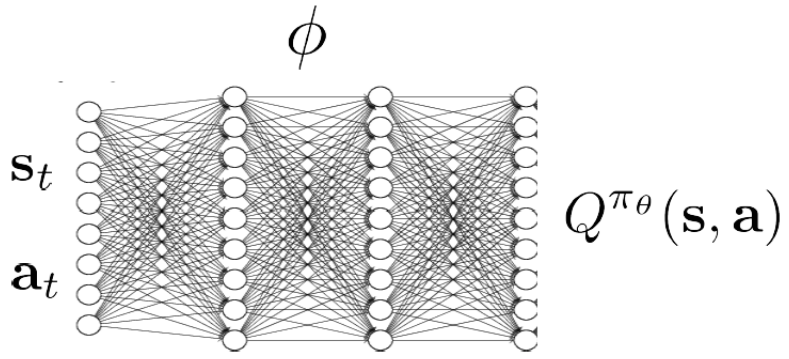
$$V^\pi(\mathbf{s}_t) = \mathbb{E}_{\mathbf{a}_t \sim \pi(\mathbf{a}_t|\mathbf{s}_t)} \left[Q^\pi(\mathbf{s}_t, \mathbf{a}_t) \right]$$

On-policy actor-critic with function approximation



$\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$

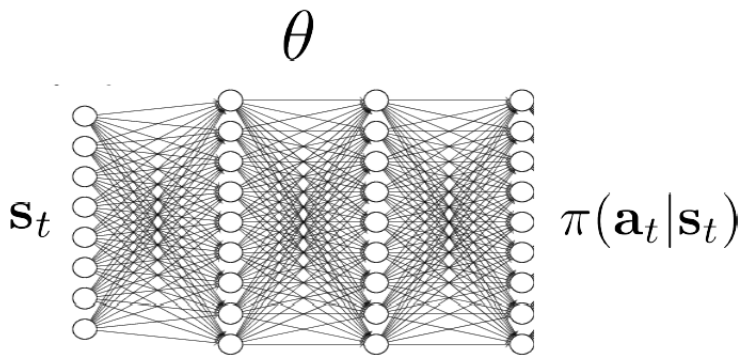
actor (policy)



$Q_\phi(\mathbf{s}, \mathbf{a})$

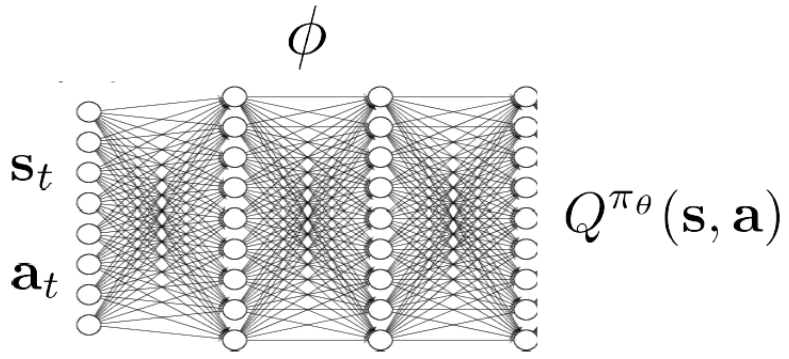
critic (state-action value function)

On-policy actor-critic with function approximation



$\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$

actor (policy)

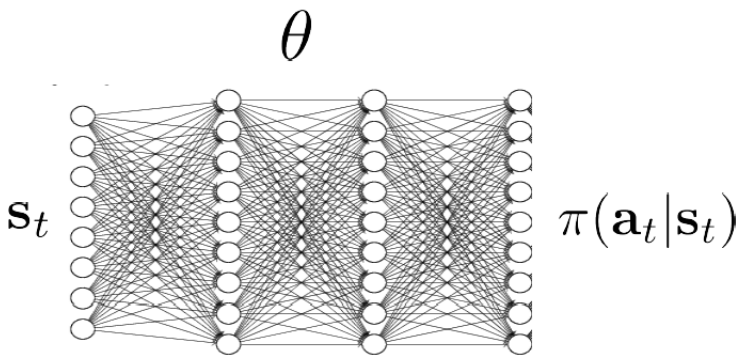


$Q_\phi(\mathbf{s},\mathbf{a})$

critic (state-action value function)

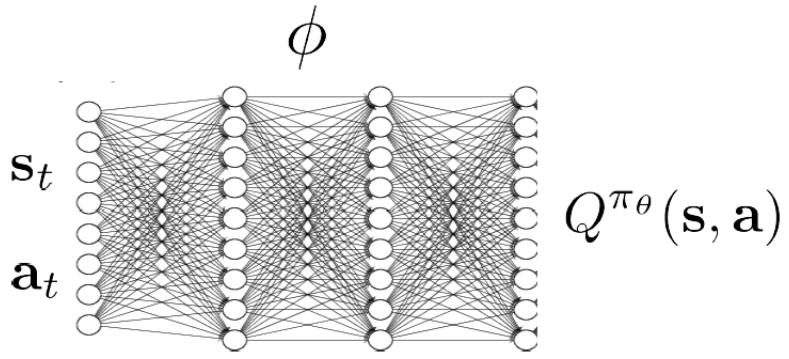
1. update Q_ϕ to decrease $E_{\mathbf{s}\sim d^{\pi_\theta}(\mathbf{s}),\, \mathbf{a}\sim\pi_\theta(\mathbf{a}|\mathbf{s})}\left[\left(Q_\phi(\mathbf{s},\mathbf{a}) - (r(\mathbf{s},\mathbf{a}) + \gamma E_{\pi_\theta}[Q_\phi(\mathbf{s}',\mathbf{a}')])\right)^2\right]$
2. update π_θ to increase $E_{\mathbf{s}\sim d^{\pi_\theta}(\mathbf{s}),\mathbf{a}\sim\pi_\theta(\mathbf{a}|\mathbf{s})}\left[Q_\phi(\mathbf{s},\mathbf{a})\right]$

Off-policy actor-critic with function approximation



$\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$

actor (policy)

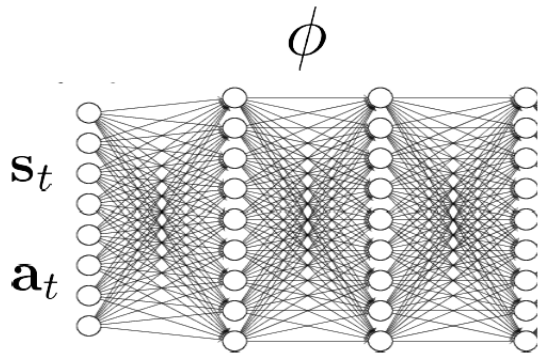


$Q_\phi(\mathbf{s},\mathbf{a})$

critic (state-action value function)

1. update Q_ϕ to decrease $E_{\mathbf{s}\sim d^{\pi_\beta}(\mathbf{s}),\, \mathbf{a}\sim\pi_\beta(\mathbf{a}|\mathbf{s})}\left[\left(Q_\phi(\mathbf{s},\mathbf{a}) - (r(\mathbf{s},\mathbf{a}) + \gamma E_{\pi_\theta}[Q_\phi(\mathbf{s}',\mathbf{a}')])\right)^2\right]$
2. update π_θ to increase $E_{\mathbf{s}\sim d^{\pi_\beta}(\mathbf{s}),\, \mathbf{a}\sim\pi_\theta(\mathbf{a}|\mathbf{s})}[Q_\phi(\mathbf{s},\mathbf{a})]$

Q-Learning/Fitted Q-Iteration with function approximation (off-policy)



$Q_\phi(\mathbf{s}, \mathbf{a})$

critic (state-action value function)

1. update Q_ϕ to decrease $E_{\mathbf{s}\sim d^{\pi_\beta}(\mathbf{s}),\, \mathbf{a}\sim\pi_\beta(\mathbf{a}|\mathbf{s})}\left[\left(Q_\phi(\mathbf{s},\mathbf{a}) - (r(\mathbf{s},\mathbf{a}) + \gamma E_\pi[Q_\phi(\mathbf{s}', \mathbf{a}')])\right)^2\right]$

2. ~~update π_θ to increase $E_{\mathbf{s}\sim d^{\pi_\beta}(\mathbf{s}),\, \mathbf{a}\sim\pi_\theta(\mathbf{a}|\mathbf{s})}[Q_\phi(\mathbf{s},\mathbf{a})]$~~

choose π according to: $\pi(\mathbf{a}_t|\mathbf{s}_t) = \begin{cases} 1 \text{ if } \mathbf{a}_t = \arg\max_{\mathbf{a}_t} Q_\phi(\mathbf{s}_t, \mathbf{a}_t) \\ 0 \text{ otherwise} \end{cases}$

Policy gradients (on-policy)

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

↗ exactly the same thing!

$$J(\theta) = E_{\tau \sim \pi_\theta(\tau)} \left[\sum_{t=0}^{T} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t) \right] \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T} \gamma^t r(\mathbf{s}_{t,i}, \mathbf{a}_{t,i})$$

$$\nabla_\theta J(\theta) = E_{\tau \sim \pi_\theta(\tau)} \left[\nabla_\theta \log \pi_\theta(\tau) \sum_{t=0}^{T} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t) \right] \quad \text{simple algebraic derivation}$$

$$= E_{\tau \sim \pi_\theta(\tau)} \left[\left(\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t) \right) \left(\sum_{t=0}^{T} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t) \right) \right] \quad \text{from definition of } \tau$$

Policy gradients (on-policy)

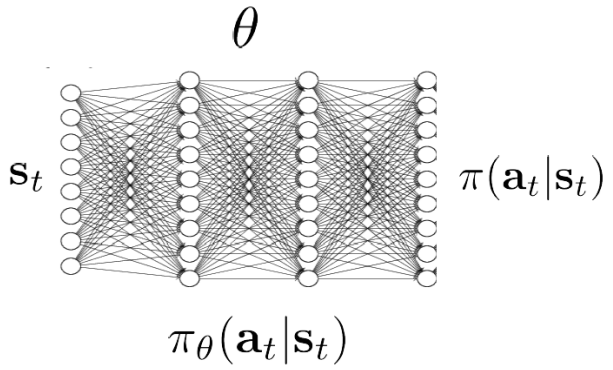
$$\text{RL objective: } \max_{\pi} \sum_{t=0}^{T} E_{\mathbf{s}_t \sim d^{\pi}(\mathbf{s}), \mathbf{a}_t \sim \pi(\mathbf{a}|\mathbf{s})}[\gamma^t r(\mathbf{s}_t, \mathbf{a}_t)]$$

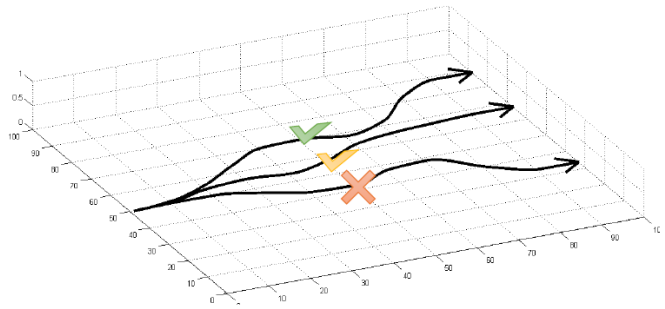
↗ exactly the same thing!

$$J(\theta) = E_{\tau \sim \pi_\theta(\tau)} \left[\sum_{t=0}^{T} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t) \right] \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T} \gamma^t r(\mathbf{s}_{t,i}, \mathbf{a}_{t,i})$$

$$\nabla_\theta J(\theta) = E_{\tau \sim \pi_\theta(\tau)} \left[\nabla_\theta \log \pi_\theta(\tau) \sum_{t=0}^{T} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t) \right]$$ simple algebraic derivation

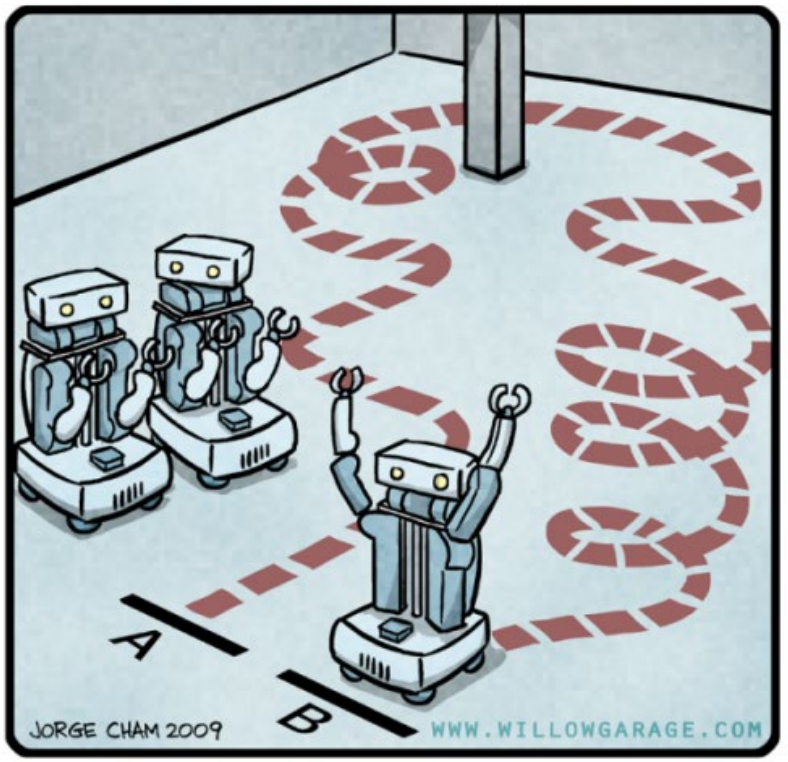
(REINFORCE gradient estimator)





Today's agenda

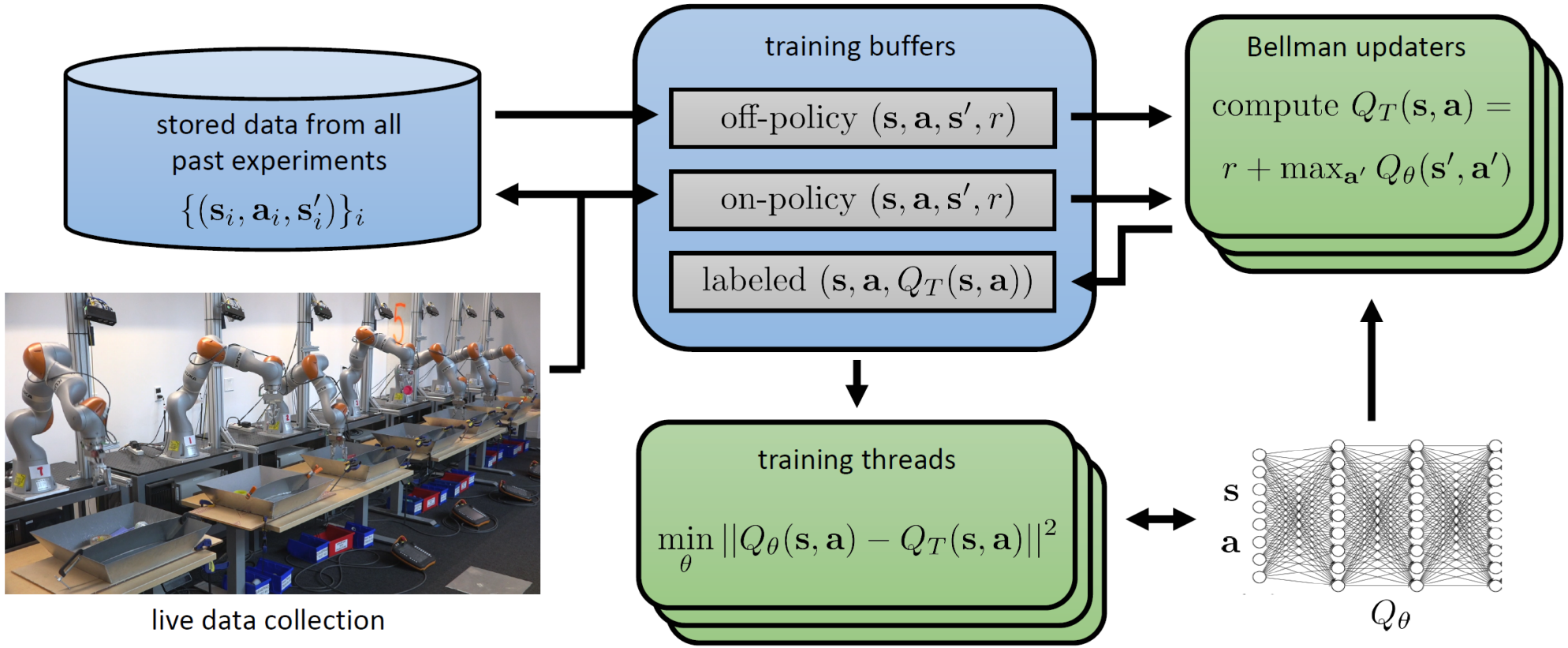
- Reinforcement Learning Terminology
- Distribution Shift in Offline RL
- Offline RL with Policy Constraints
- Offline RL with Conservative Q-Estimates



Acknowledgments
Today's slides borrow very heavily from: Sergey Levine & Aviral Kumar (CSC285 and NeurIPS Offline RL workshop), Joelle Pineau (DLSS'16)

QT-Opt (roughly: continuous-action Q-Learning)

QT-Opt (roughly: continuous-action Q-Learning)



live data collection

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?

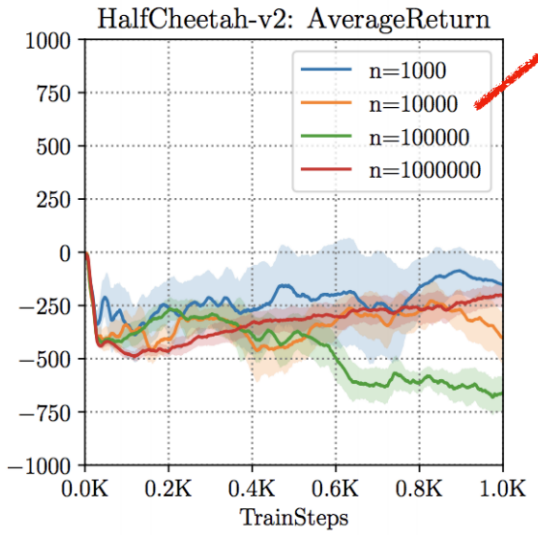
Method	Dataset	Success	Failure
Offline QT-Opt	580k offline	87%	13%
Finetuned QT-Opt	580k offline + 28k online	**96%**	**4%**

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

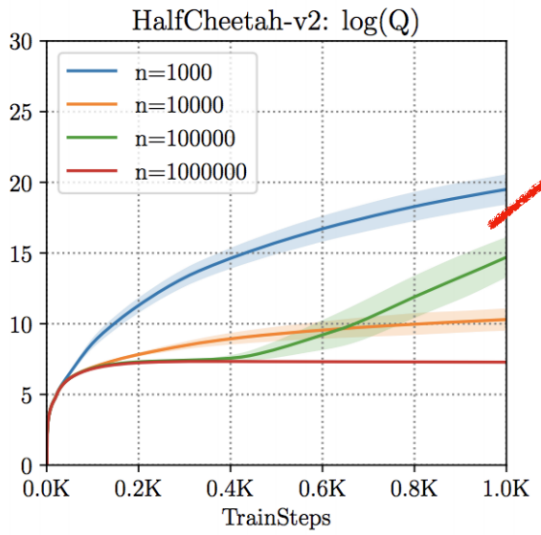
Modern Offline RL: A Simple Experiment

Collect expert data and run actor-critic algorithms on this data

Performance doesn't improve with more data



how well it does



how well **it thinks** it does

Learning diverges

Not a classical overfitting issue!

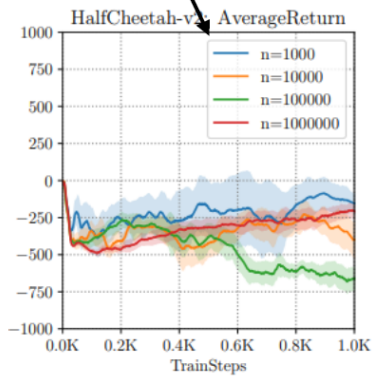
Kumar, Fu, Tucker, Levine. *Stabilizing Off-Policy RL via Bootstrapping Error Reduction*, NeurIPS 2019.
Levine, **Kumar,** Tucker, Fu. *Offline RL Tutorial and Perspectives on Open Problems.* arXiv 2020.

Why is offline reinforcement learning hard?

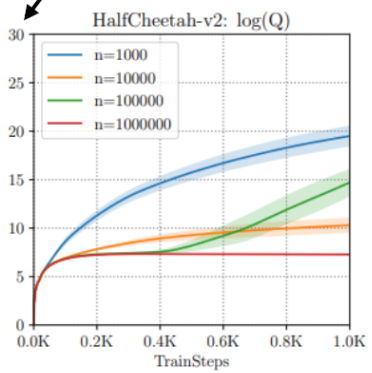
Hypothesis 1: Overfitting

amount of data

log scale (massive overestimation)



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

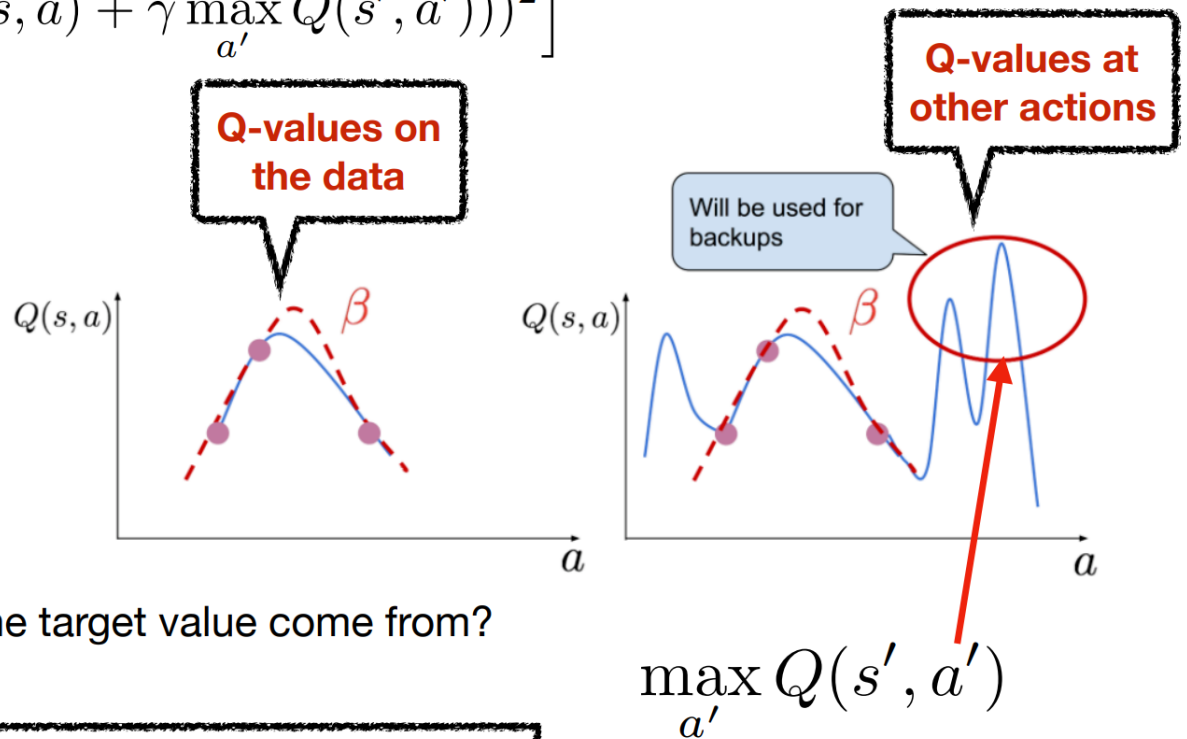
Let's see how the Q-function is updated

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

$$\mathbb{E}_{s,a,s' \sim \mathcal{D}} \left[(Q(s,a) - (r(s,a) + \gamma \max_{a'} Q(s',a')))^2 \right]$$

Which actions does the Q-function train on?

$$s, a \sim \mathcal{D}$$



Where does the action a' for the target value come from?

Q-learning queries values at unseen action targets, which are never trained during training

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?

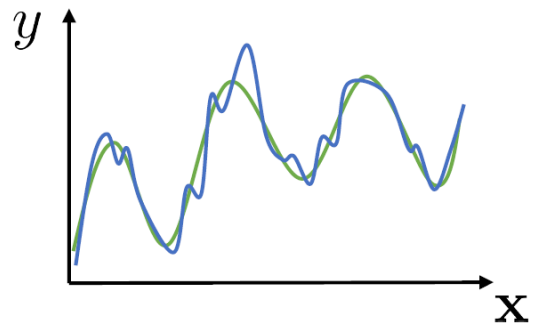
$E_{\mathbf{x}\sim p(\mathbf{x}),y\sim p(y|\mathbf{x})}\left[(f_\theta(\mathbf{x})-y)^2\right]$ is low

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

Why are erroneous backups a big deal?

- This phenomenon also happens in online RL settings, where the Q-function is erroneously optimistic

- But Boltzmann or epsilon-greedy exploration on this overoptimistic Q-function (generally) leads to "error correction"

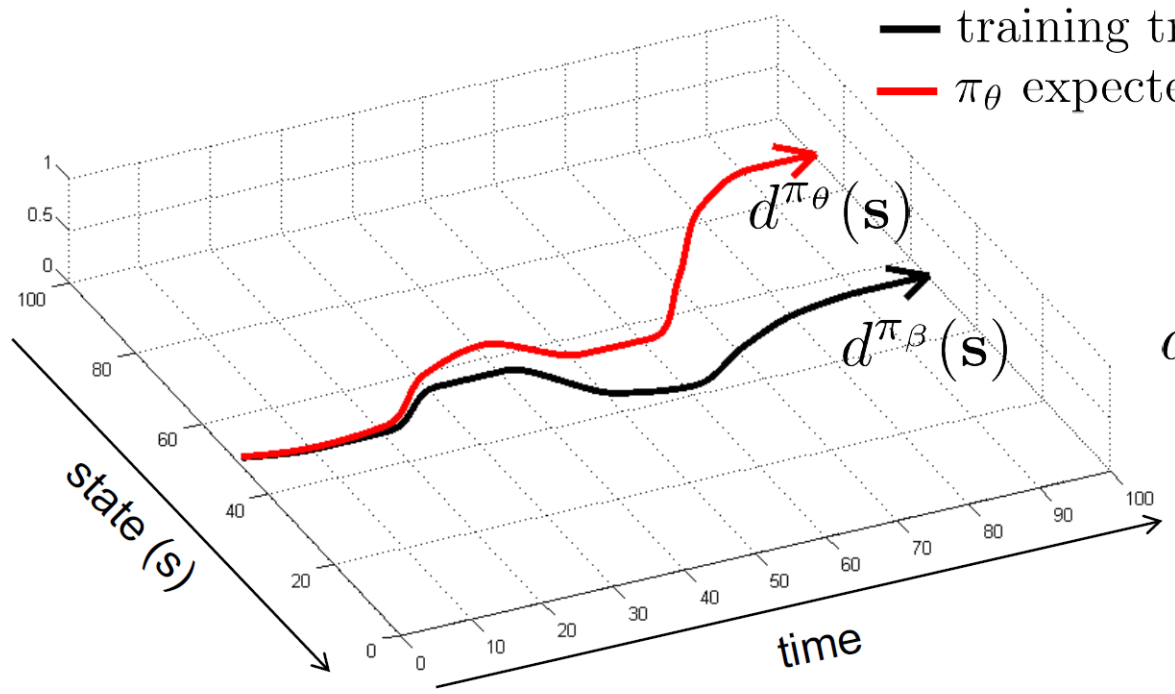
$$\pi_{\text{explore}}(a|s) \propto \exp(Q(s,a))$$

  Error correction is **not** necessarily guaranteed with online data collection when using deep neural nets, but mostly works fine in practice (trick: use replay buffers, perform distribution correction, etc)

- **But the primary ability of error correction, i.e., exploration,** is impossible in offline RL, due to no access to an environment….

Kumar, Fu, Tucker, Levine. *Stabilizing Off-Policy RL via Bootstrapping Error Reduction*, NeurIPS 2019.
Levine, **Kumar,** Tucker, Fu. *Offline RL Tutorial and Perspectives on Open Problems.* arXiv 2020.
Kumar, Gupta, Levine. DisCor: Corrective-Feedback in RL via Distribution Correction. NeurIPS 2020.
Kumar, Gupta. Does On-Policy Data Collection Fix Errors in Off-Policy Reinforcement Learning?, BAIR blog.

Where **else** do we suffer from distribution shift?



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

Even if π_θ is *great* on training data, it might be bad when used for control!

Distributional Shift in Offline RL

- Distribution shift between the behavior policy (the policy that collected the data) and the policy during learning

$$Q(s,a) \leftarrow r(s,a) + \gamma \max_{a'} Q(s',a') \qquad \neq \pi_\beta(a|s)$$

$$Q(s,a) \leftarrow r(s,a) + \gamma \mathbb{E}_{a' \sim \pi(a'|s')} Q(s',a') \qquad = \pi_\beta(a|s)$$

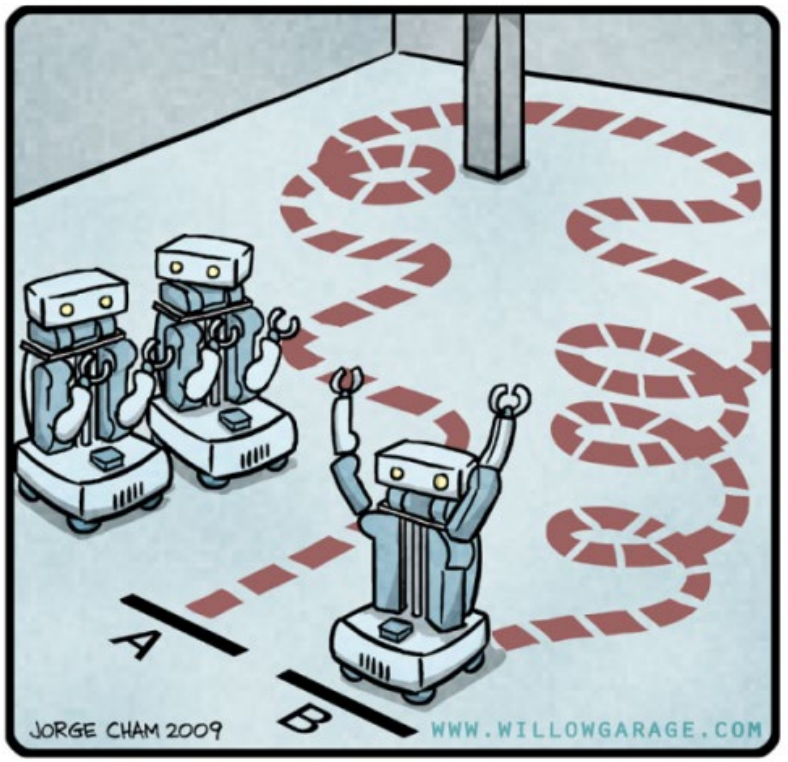
Training: $\mathbb{E}_{s,a \sim d^{\pi_\beta}(s,a)} \left[(Q(s,a) - \mathcal{B}\bar{Q}(s,a))^2 \right]$

Offline Q-Learning algorithms can overestimate the value of unseen actions and can thus be **falsely optimistic**

Kumar, Fu, Tucker, Levine. *Stabilizing Off-Policy RL via Bootstrapping Error Reduction*, NeurIPS 2019.
Levine, **Kumar,** Tucker, Fu. *Offline RL Tutorial and Perspectives on Open Problems.* arXiv 2020.

Today's agenda

- Reinforcement Learning Terminology
- Distribution Shift in Offline RL
- Offline RL with Policy Constraints
- Offline RL with Conservative Q-Estimates



Acknowledgments
Today's slides borrow very heavily from: Sergey Levine & Aviral Kumar (CSC285 and NeurIPS Offline RL workshop), Joelle Pineau (DLSS'16)

Addressing Distribution Shift via Pessimism

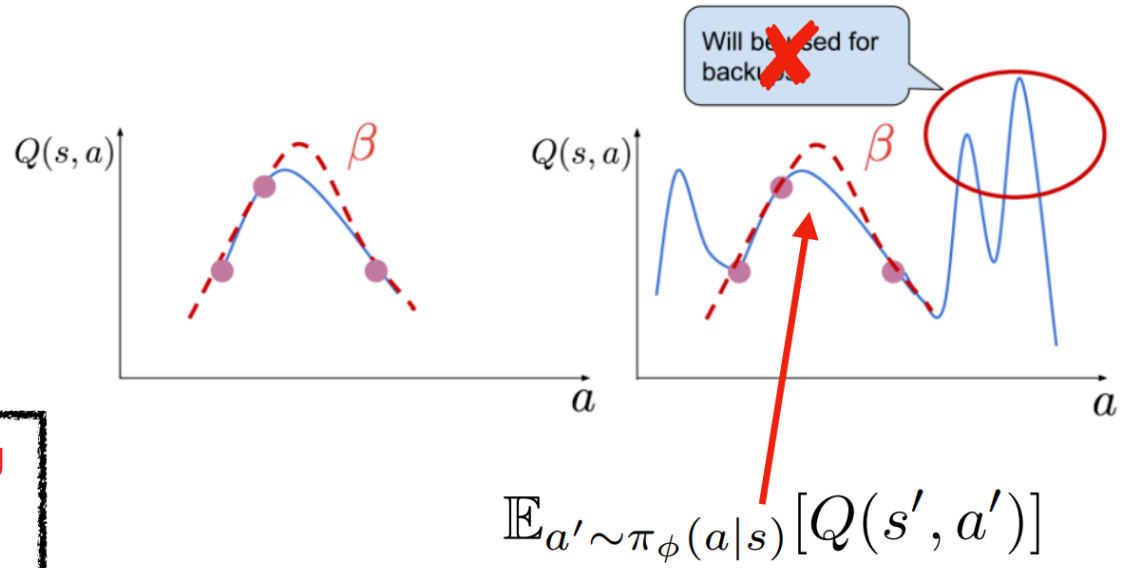
$$Q(s,a) \leftarrow r(s,a) + \gamma \mathbb{E}_{a' \sim \pi_\phi(a|s)}[Q(s',a')]$$

"Policy Constraint"

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

Out-of-distribution action values are no longer used for the backup

Hence, all values used during training are also trained, leading to better learning



Levine, **Kumar,** Tucker, Fu. *Offline RL Tutorial and Perspectives on Open Problems.* arXiv 2020.

Different Types of Policy Constraint Methods

$$\pi_\theta := \arg\max_{\pi_\theta} \mathbb{E}_{\mathbf{s}\sim\mathcal{D},\mathbf{a}\sim\pi_\theta(\mathbf{a}|\mathbf{s})}[Q(\mathbf{s},\mathbf{a})] \quad \text{s.t. } D(\pi_\theta(\mathbf{a}|\mathbf{s}),\pi_\beta(\mathbf{a}|\mathbf{s})) \le \varepsilon$$

Distribution constraints

(Fujimoto et al. 2019: action clipping)
(Wu et al. 2019, Jaques et al. 2019)

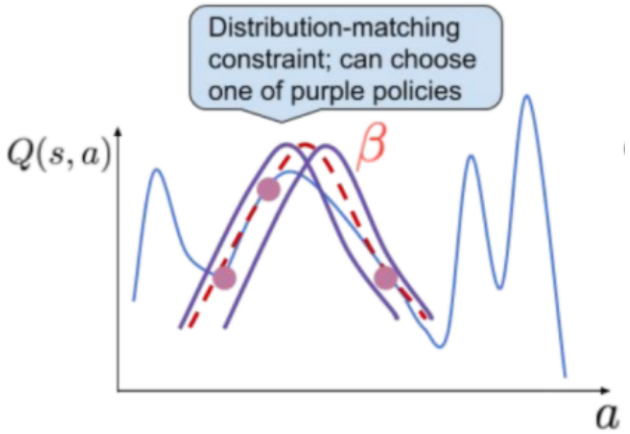
$$D(\pi_\theta,\pi_\beta) = \mathrm{D}_{\mathrm{KL}}(\pi_\theta,\pi_\beta)$$

KL divergence

$$D(\pi_\theta,\pi_\beta) = \mathrm{D}_{\mathrm{f}}(\pi_\theta,\pi_\beta) = \mathbb{E}_{\pi_\beta}\left[f\left(\frac{\pi_\theta}{\pi_\beta}\right)\right]$$

f-divergence

(Wu et al. 2019)



$$\max_{\pi_\theta(\mathbf{a}|\mathbf{s})} \mathbb{E}_{\pi_\theta}[Q(\mathbf{s},\mathbf{a})] - \alpha \mathrm{D}_{\mathrm{KL}}(\pi_\theta(\mathbf{a}|\mathbf{s}),\pi_\beta(\mathbf{a}|\mathbf{s}))$$

(solve in closed form)

$$\pi_\theta(\mathbf{a}|\mathbf{s}) \propto \pi_\beta(\mathbf{a}|\mathbf{s}) \exp(Q(\mathbf{s},\mathbf{a})/\alpha)$$

(Peng et al. 2019, Seigel et al. 2019, Wang et al. 2020, Nair et al. 2020)

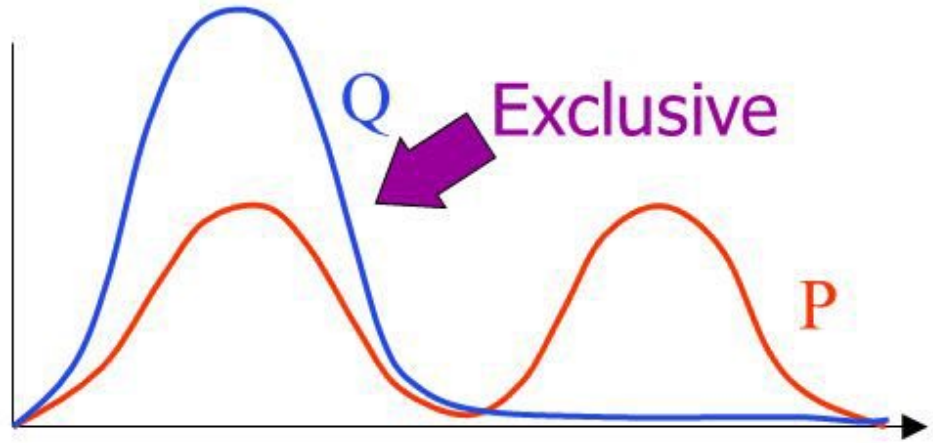
"Implicit" distributional constraint

Note: KL divergence is not symmetric

Minimising

$$\mathrm{KL}(Q||P)$$

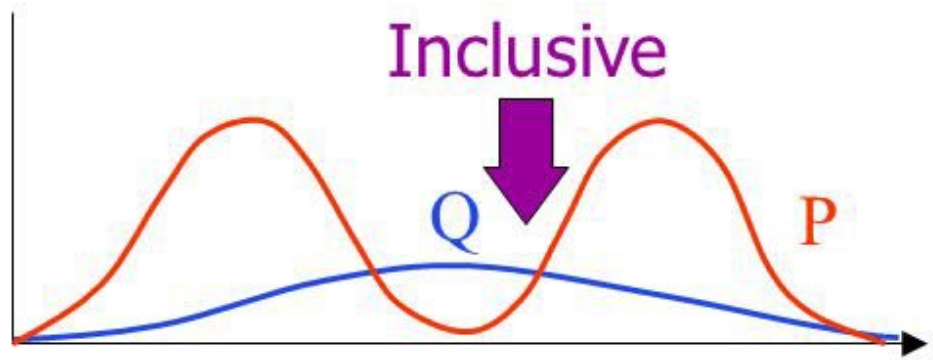
$$= \sum_H Q(H) \ln \frac{Q(H)}{P(H|V)}$$



Minimising

$$\mathrm{KL}(P||Q)$$

$$= \sum_H P(H|V) \ln \frac{P(H|V)}{Q(H)}$$



Different Types of Policy Constraint Methods

$$\pi_\theta := \arg\max_{\pi_\theta} \mathbb{E}_{\mathbf{s}\sim\mathcal{D},\mathbf{a}\sim\pi_\theta(\mathbf{a}|\mathbf{s})}[Q(\mathbf{s},\mathbf{a})] \quad \text{s.t. } D(\pi_\theta(\mathbf{a}|\mathbf{s}),\pi_\beta(\mathbf{a}|\mathbf{s})) \le \varepsilon$$

Support constraints

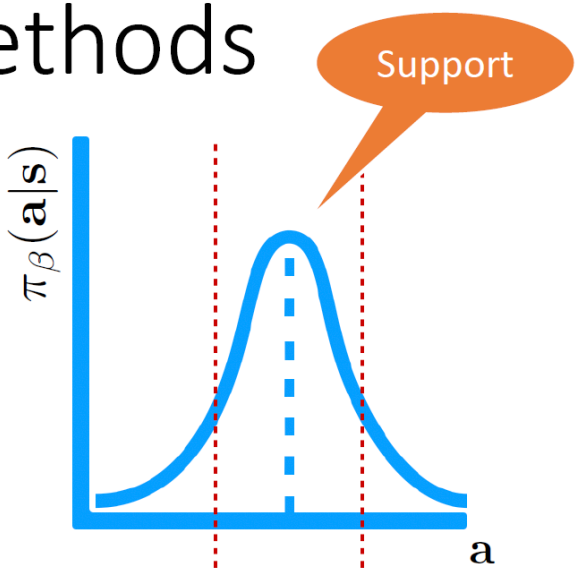
$$D(\pi_\theta,\pi_\beta) = \mathbf{D}(\pi_\theta, \text{supp}(\pi_\beta))$$

$$D(\pi_\theta,\pi_\beta) = \text{MMD}(\pi_\theta,\pi_\beta)$$

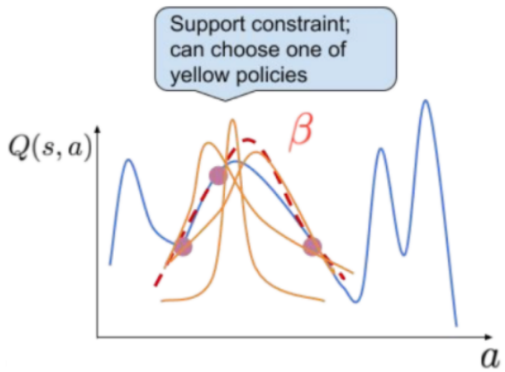
(Kumar et al. 2019)

Maximum mean discrepancy

(Gretton et al. 2012)



Intuition: compare samples regardless of density



$$MMD[\mathcal{F},p,q] := sup_{f\in\mathcal{F}}(\mathbb{E}[f(x)] - \mathbb{E}[f(y)])$$

$$MMD[\mathcal{F},X,Y] := sup_{f\in\mathcal{F}}(\frac{1}{m}\sum_{i=1}^{m} f(x_i) - \frac{1}{n}\sum_{i=1}^{n} f(y_i))$$

How should we evaluate offline RL methods?

Let's revisit the main motivation for offline RL

Use real-data collected from various different sources (e.g., human demonstrations, runs of hardcoded policies, etc.) for training good policies

Can train directly on real data, but how do we test the policy?

Since testing a policy completely offline is hard (unless we actually run the policy on the real-domain), we would want benchmarks!

What properties should a benchmark for offline RL have?

1. It should be realistic: should mimic what we would see in the real-world

2. Should provide a method to compare methods in a standardized way, under the actual evaluation scheme

Evaluating Offline RL – D4RL

D4RL: Datasets for Deep Data-Driven Reinforcement Learning

Justin Fu
UC Berkeley
justinjfu@eecs.berkeley.edu

Aviral Kumar
UC Berkeley
aviralk@berkeley.edu

Ofir Nachum
Google Brain
ofirnachum@google.com

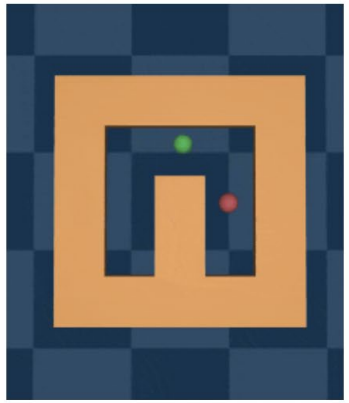
George Tucker
Google Brain
gjt@google.com

Sergey Levine
UC Berkeley, Google Brain
svlevine@eecs.berkeley.edu

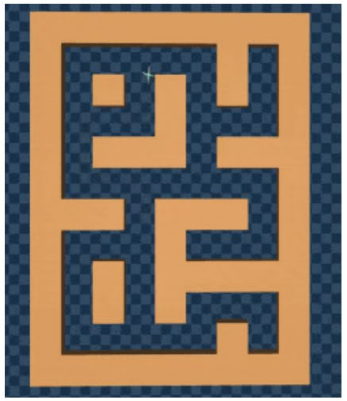
Abstract

The offline reinforcement learning (RL) problem, also known as batch RL, refers to the setting where a policy must be learned from a static dataset, without additional online data collection. This setting is compelling as potentially it allows RL methods to take advantage of large, pre-collected datasets, much like how the rise of large datasets has fueled results in supervised learning in recent years. However, existing *online* RL benchmarks are not tailored towards the *offline* setting, making progress in offline RL difficult to measure. In this work, we introduce benchmarks specifically designed for the offline setting, guided by key properties of datasets relevant to real-world applications of offline RL. Examples of such properties include: datasets generated via hand-designed controllers and human demonstrators, multi-objective datasets where an agent can perform different tasks in the same environment, and datasets consisting of a mixtures of policies. To facilitate research, we release our benchmark tasks and datasets with a comprehensive evaluation of existing algorithms and an evaluation protocol together with an open-source codebase. We hope that our benchmark will focus research effort on methods that drive improvements not just on simulated tasks, but ultimately on the kinds of real-world problems where offline RL will have the largest impact.

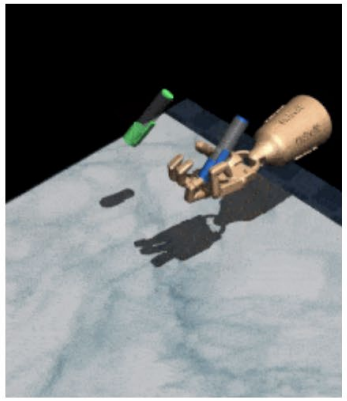
Evaluating Offline RL – D4RL



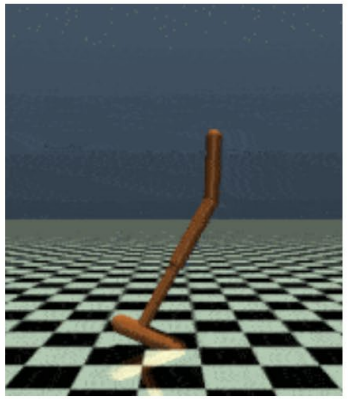
Maze2D



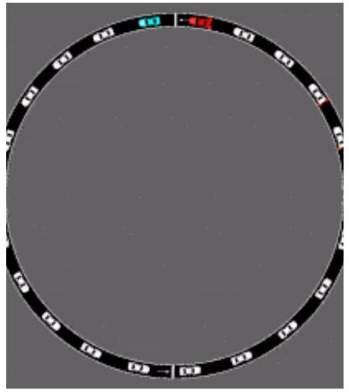
AntMaze



Adroit

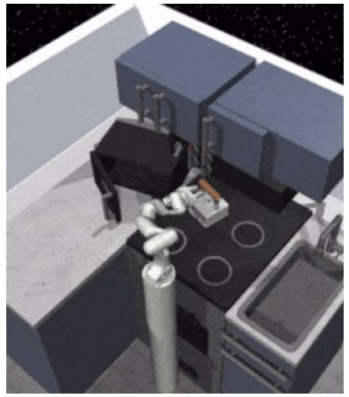


Gym



Flow

CARLA

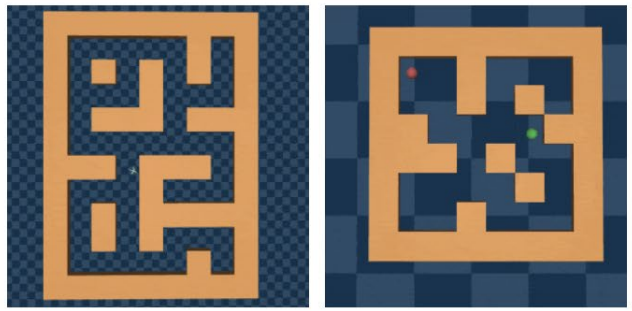
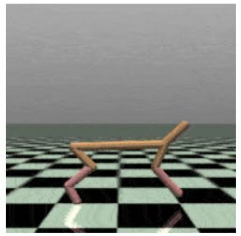
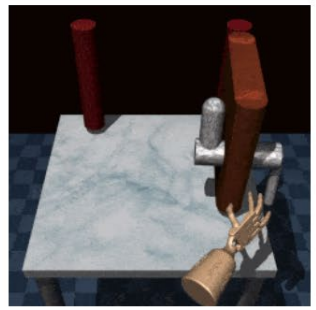


FrankaKitchen

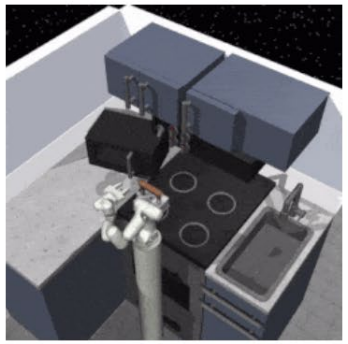
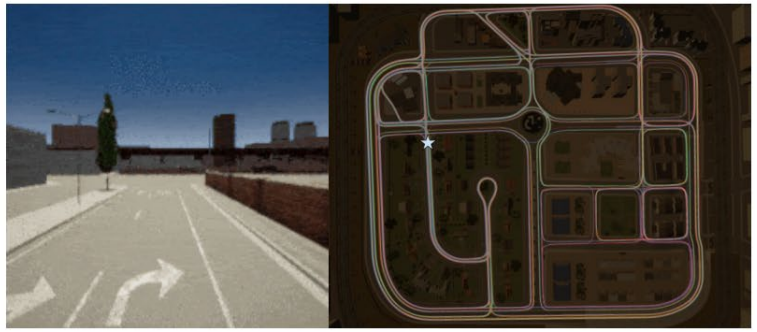
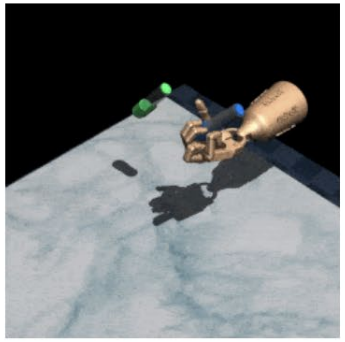
Standardized Benchmark for Offline RL

Most evaluation so far has used RL policies or replay buffers, which tend to be substantially easier and different from “real-world” scenarios

D4RL benchmark

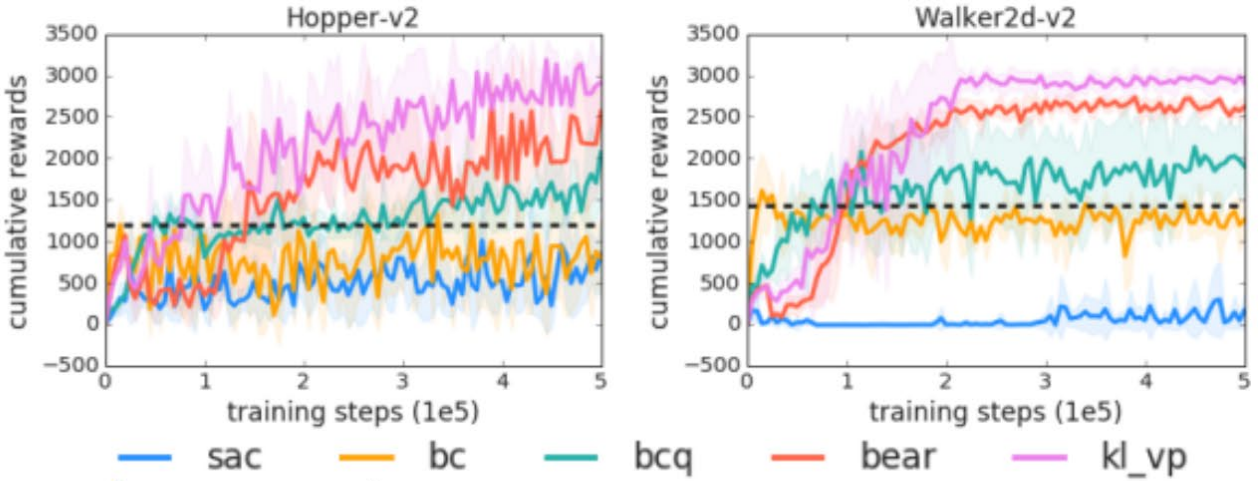
Properties: **(1)** non-representable behavior policies **(2)** narrow distributions **(3)** undirected/multi-task behavior **(4)** visual perception **(5)** human demos.

Fu, **Kumar,** Nachum, Tucker, Levine. *D4RL: Datasets for Deep Data-Driven RL.* arXiv 2020.

How do policy constraint methods perform?

Dataset collected from a mixture of random and "mediocre" policies



Naïve off-policy RL

Behavioral cloning

Policy constraint methods (BEAR: support, BCQ, BRAC: distribution)

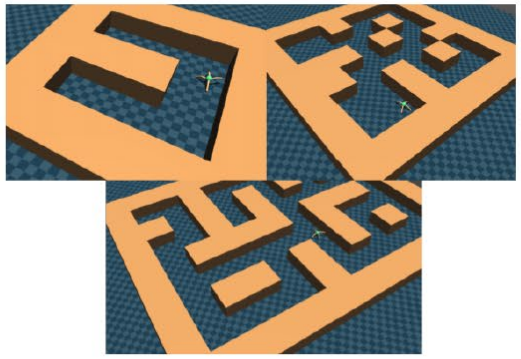
Task Name	BC	SAC	BEAR	BRAC-p
antmaze-umaze	65.0	0.0	**73.0**	50.0
antmaze-umaze-diverse	55.0	0.0	61.0	40.0
antmaze-medium-play	0.0	0.0	0.0	0.0
antmaze-medium-diverse	0.0	0.0	8.0	0.0
antmaze-large-play	0.0	0.0	0.0	0.0
antmaze-large-diverse	0.0	0.0	0.0	0.0

Results from BRAC (Wu et al. 2020) and D4RL (Fu et al. 2020)

Better than BC! (shows offline RL can do better than supervised learning)

Different choices of D matter, though not totally as expected, so other factors at play

But do not solve all tasks, especially harder ones, so work to do!



Value Function Regularization for Offline RL

So far, we have looked at handling overestimation via penalizing the reward function

Can we penalize value functions or Q-functions directly?

KL-control (Kakade '02, Fox et al. '16, Haarnoja et al. '17)

BRAC-v (Wu et al. 2019), KL-control (Jaques et al. 2019)

Way 1: Add penalty based on policy constraint to the value function

$$Q(\mathbf{s},\mathbf{a}) \leftarrow r(\mathbf{s},\mathbf{a}) + \gamma \mathbb{E}_{\mathbf{a}' \sim \pi_\theta(\mathbf{a}'|\mathbf{s}')}[Q(\mathbf{s}',\mathbf{a}')] - \alpha D(\pi_\theta, \pi_\beta)$$

But this still uses a policy constraint…..

Why can this work?

TD error will push values on **seen** state-action pairs up, while we minimize others

Way 2: Change the objective to make this behavior automatically kick in!

Minimize the **"big" Q-values**

Standard TD error objective

$$\min_Q \max_\mu \quad \mathbb{E}_{\mathbf{s}\sim\mathcal{D}} \mathbb{E}_{\mathbf{a}\sim\mu(\mathbf{a}|\mathbf{s})}[Q(\mathbf{s},\mathbf{a})] + \frac{1}{2\alpha} \mathbb{E}_{\mathbf{s},\mathbf{a},\mathbf{s}'\sim\mathcal{D}} \left[(Q(\mathbf{s},\mathbf{a}) - y(\mathbf{s},\mathbf{a}))^2 \right]$$

$$y(\mathbf{s},\mathbf{a}) = r(\mathbf{s},\mathbf{a}) + \gamma \mathbb{E}_{\mathbf{a}' \sim \pi_\theta(\mathbf{a}'|\mathbf{s}')}[Q(\mathbf{s}',\mathbf{a}')]$$

Target values

Learning Lower-Bounded Q-values

Conservative Q-Learning (CQL) Algorithm

Since learned Q-values (our belief of policy values) are overestimated, let's make them provably lower bound the true value

Minimize big Q-values

Standard Bellman Error

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

$$\hat{Q}^{\pi}_{\text{CQL}}(s,a) \leq Q(s,a) \quad \forall s \in \mathcal{D}, a$$

CQL-v1

CQL Algorithm:

1. Learn $\hat{Q}^{\pi}_{\text{CQL}}$ using offline data $\mathcal{D}$.
2. Optimize policy w.r.t. $\hat{Q}^{\pi}_{\text{CQL}}$: $\pi \leftarrow \arg\max_{\pi} \mathbb{E}_{\pi}[\hat{Q}^{\pi}_{\text{CQL}}]$.

Kumar, Zhou, Tucker, Levine. *Conservative Q-Learning for Offline RL.* NeurIPS 2020.

A Tighter Lower Bound

Minimize big Q-values

Maximize Data Q-values

$$\hat{Q}^{\pi}_{\text{CQL}} := \min_{Q} \max_{\mu} \mathbb{E}_{a\sim\mu(a|s)}[Q(s,a)] - \mathbb{E}_{a\sim\mathcal{D}(a|s)}[Q(s,a)]$$

$$+\frac{1}{2\alpha}\mathbb{E}_{s,a,s'\sim\mathcal{D}}\left[(Q(s,a) - (r(s,a) + \gamma\mathbb{E}_{a\sim\pi_\phi(a|s)}[\bar{Q}(s',a')]))^2\right]$$

Standard Bellman Error

$$\hat{Q}^{\pi}_{\text{CQL}}(s,a) \leq Q(s,a) \quad \forall s \in \mathcal{D}, a$$

$$\hat{V}^{\pi}_{\text{CQL}}(s) := \mathbb{E}_{a\sim\pi_k}[\hat{Q}^{\pi}_{\text{CQL}}(s,a)] \leq V^{\pi}(s) \quad \forall\ s \in \mathcal{D}$$

CQL-v2

CQL Algorithm:

1. Learn $\hat{Q}^{\pi}_{\text{CQL}}$ using offline data $\mathcal{D}$.
2. Optimize policy w.r.t. $\hat{Q}^{\pi}_{\text{CQL}}$: $\pi \leftarrow \arg\max_{\pi} \mathbb{E}_{\pi}[\hat{Q}^{\pi}_{\text{CQL}}]$.

Kumar, Zhou, Tucker, Levine. *Conservative Q-Learning for Offline RL.* NeurIPS 2020.

Practical CQL Algorithm

CQL(H)

$$\min_Q \alpha \mathbb{E}_{\mathbf{s}\sim\mathcal{D}}\left[\log \sum_{\mathbf{a}} \exp(Q(\mathbf{s},\mathbf{a})) - \mathbb{E}_{\mathbf{a}\sim\hat{\pi}_\beta(\mathbf{a}|\mathbf{s})}\left[Q(\mathbf{s},\mathbf{a})\right]\right] + \frac{1}{2}\mathbb{E}_{\mathbf{s},\mathbf{a},\mathbf{s}'\sim\mathcal{D}}\left[\left(Q - \hat{\mathcal{B}}^{\pi_k}\hat{Q}^k\right)^2\right].$$

Algorithm 1 Conservative Q-Learning (both variants)

1: Initialize Q-function, Q_θ, and optionally a policy, π_ϕ.
2: **for** step t in $\{1, \ldots, N\}$ **do**
3: Train the Q-function using G_Q gradient steps on objective from Equation 4
$\theta_t := \theta_{t-1} - \eta_Q \nabla_\theta \mathrm{CQL}(\mathcal{R})(\theta)$
(Use $\mathcal{B}^*$ for Q-learning, $\mathcal{B}^{\pi_{\phi_t}}$ for actor-critic)
4: (only with actor-critic) Improve policy π_ϕ via G_π gradient steps on ϕ with SAC-style entropy regularization:
$\phi_t := \phi_{t-1} + \eta_\pi \mathbb{E}_{\mathbf{s}\sim\mathcal{D},\mathbf{a}\sim\pi_\phi(\cdot|\mathbf{s})}\left[Q_\theta(\mathbf{s},\mathbf{a}) - \log \pi_\phi(\mathbf{a}|\mathbf{s})\right]$
5: **end for**

Only change on top of standard Deep Q-Learning

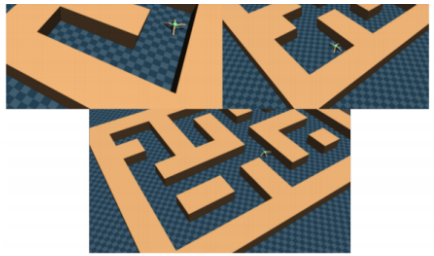
CQL Algorithm:

1. Learn $\hat{Q}^\pi_{\mathrm{CQL}}$ using offline data $\mathcal{D}$.
2. Optimize policy w.r.t. $\hat{Q}^\pi_{\mathrm{CQL}}$: $\pi \leftarrow \arg\max_\pi \mathbb{E}_\pi[\hat{Q}^\pi_{\mathrm{CQL}}]$.

$$\min_Q \max_\mu \alpha\left(\mathbb{E}_{\mathbf{s}\sim\mathcal{D},\mathbf{a}\sim\mu(\mathbf{a}|\mathbf{s})}\left[Q(\mathbf{s},\mathbf{a})\right] - \mathbb{E}_{\mathbf{s}\sim\mathcal{D},\mathbf{a}\sim\hat{\pi}_\beta(\mathbf{a}|\mathbf{s})}\left[Q(\mathbf{s},\mathbf{a})\right]\right) + \frac{1}{2}\mathbb{E}_{\mathbf{s},\mathbf{a},\mathbf{s}'\sim\mathcal{D}}\left[\left(Q(\mathbf{s},\mathbf{a}) - \hat{\mathcal{B}}^{\pi_k}\hat{Q}^k(\mathbf{s},\mathbf{a})\right)^2\right] + \mathcal{R}(\mu) \quad (\mathrm{CQL}(\mathcal{R})).$$

Kumar, Zhou, Tucker, Levine. *Conservative Q-Learning for Offline RL.* NeurIPS 2020.

CQL, Empirically



Learned policy value - Actual policy value

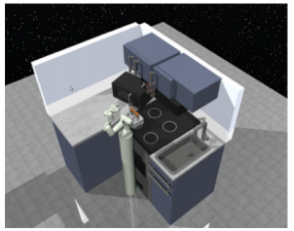
Task Name	CQL($\mathcal{H}$)	CQL (Eqn. 1)	Ensemble(2)	Ens.(4)	Ens.(10)	Ens.(20)	BEAR
hopper-medium-expert	**-43.20**	-151.36	3.71e6	2.93e6	0.32e6	24.05e3	65.93
hopper-mixed	**-10.93**	-22.87	15.00e6	59.93e3	8.92e3	2.47e3	1399.46
hopper-medium	**-7.48**	-156.70	26.03e12	437.57e6	1.12e12	885e3	4.32

Behavior cloning

Naive off-policy RL

Policy constraint methods

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-medium-play	0.0	0.0	0.0	0.0	0.0	**61.2**	4.6
	antmaze-medium-diverse	0.0	0.0	8.0	0.0	0.0	**53.7**	5.1
	antmaze-large-play	0.0	0.0	0.0	0.0	0.0	**15.8**	3.2
	antmaze-large-diverse	0.0	0.0	0.0	0.0	0.0	**14.9**	2.3
Adroit	pen-human	34.4	6.3	-1.0	8.1	0.6	37.5	**55.8**
	hammer-human	1.5	0.5	0.3	0.3	0.2	**4.4**	2.1
	door-human	0.5	3.9	-0.3	-0.3	-0.3	**9.9**	**9.1**
	relocate-human	0.0	0.0	-0.3	-0.3	-0.3	0.20	**0.35**
	pen-cloned	**56.9**	23.5	26.5	1.6	-2.5	39.2	40.3
	hammer-cloned	0.8	0.2	0.3	0.3	0.3	2.1	**5.7**
	door-cloned	-0.1	0.0	-0.1	-0.1	-0.1	0.4	**3.5**
	relocate-cloned	**-0.1**	-0.2	-0.3	-0.3	-0.3	**-0.1**	**-0.1**
Kitchen	kitchen-complete	33.8	15.0	0.0	0.0	0.0	**43.8**	31.3
	kitchen-partial	33.8	0.0	13.1	0.0	0.0	**49.8**	**50.1**
	kitchen-undirected	47.5	2.5	47.2	0.0	0.0	**51.0**	**52.4**



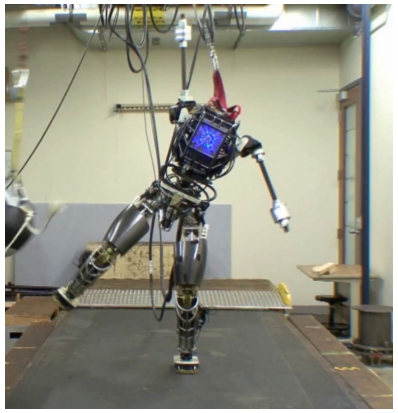
"Stitching"

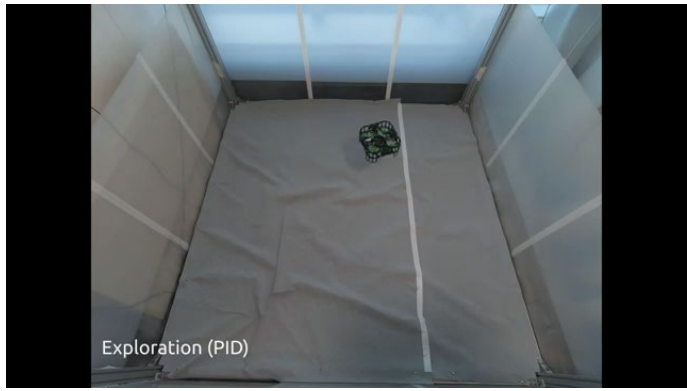
Better than other methods, not the best in each case

Only method to outperform BC

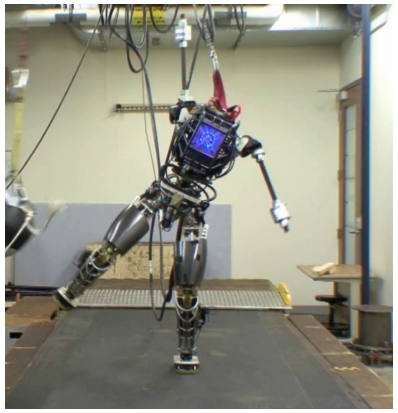
Kumar, Zhou, Tucker, Levine. *Conservative Q-Learning for Offline RL.* NeurIPS 2020.

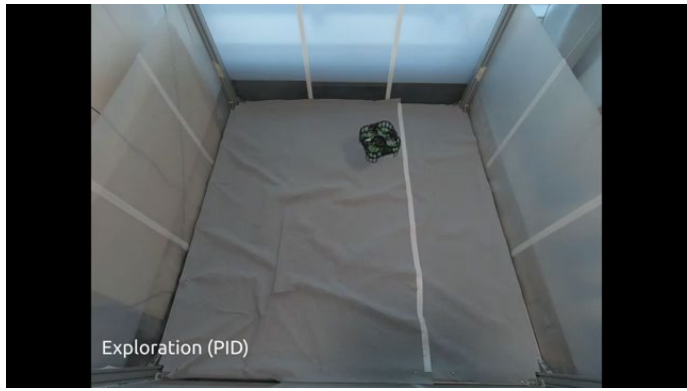
The need for safe exploration in RL for robotics





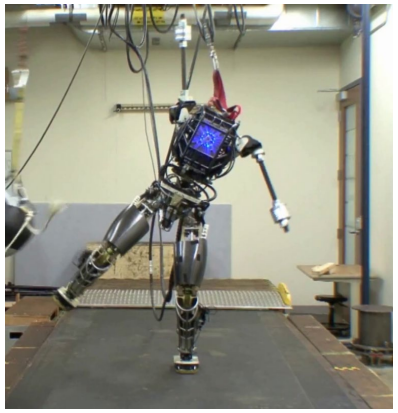
The need for safe exploration in RL for robotics

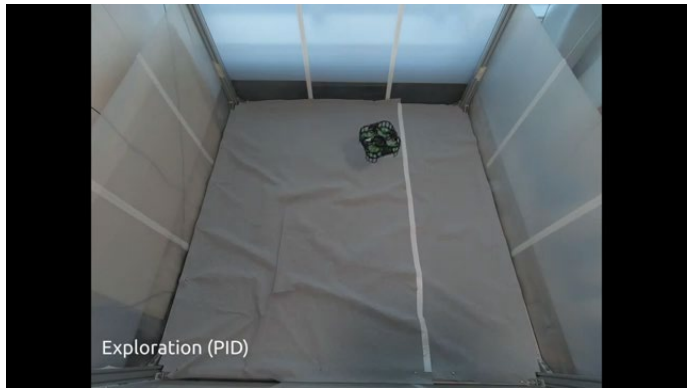




When applying RL to robotics we need to guarantee that the algorithm will not visit **unsafe states** very often during learning.

The need for safe exploration in RL for robotics





When applying RL to robotics we need to guarantee that the algorithm will not visit **unsafe states** very often during learning.

Easy

Control theory community:

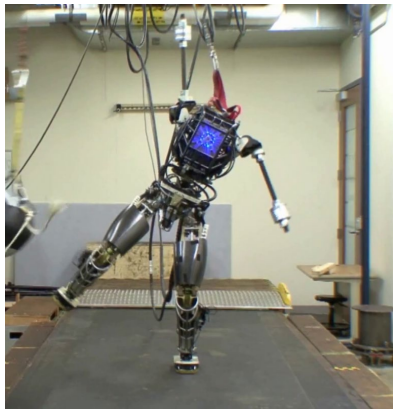
"Unsafe states are fully known"

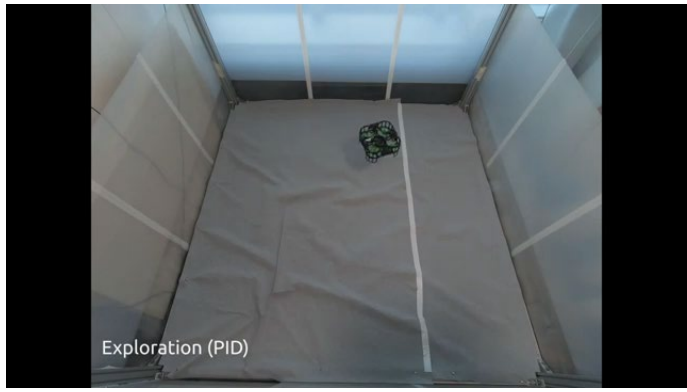
Hard

RL community:

"Unsafe states are fully unknown"

The need for safe exploration in RL for robotics





When applying RL to robotics we need to guarantee that the algorithm will not visit **unsafe states** very often during learning.

Easy

Control theory community:

"Unsafe states are fully known"

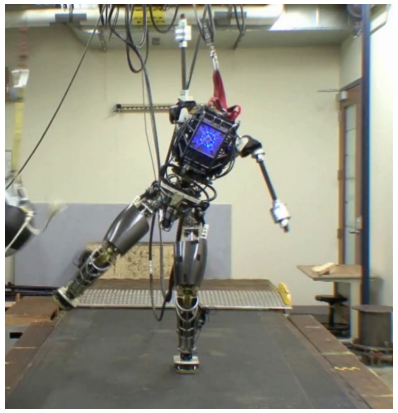
Hard

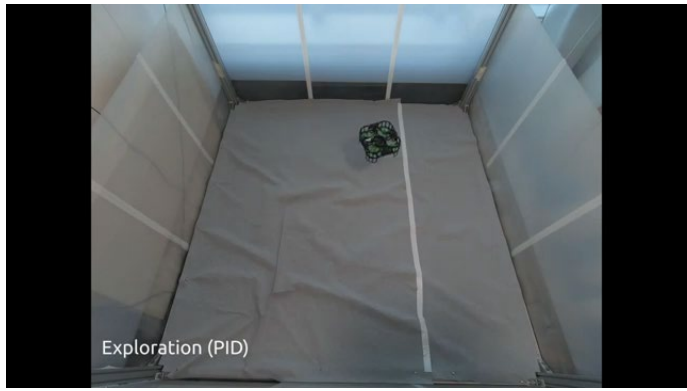
RL community:

"Unsafe states are fully unknown"

Our proposed method

The need for safe exploration in RL for robotics





When applying RL to robotics we need to guarantee that the algorithm will not visit **unsafe states** very often **during learning**.

Our proposed method:

The learned policy should be safe at each iteration, not just when optimization has converged

Our solution: Constrained Safety Critics (CSC)

$$\underset{\theta}{\text{maximize}} \quad \mathcal{V}^{\pi_\theta}_{\text{task}}(\mathbf{s}_0)$$

$$\text{subject to} \quad \mathcal{V}^{\pi_\theta}_{\text{accident}}(\mathbf{s}_0) < \epsilon$$

$$\text{KL}(\pi_\theta || \pi_{\text{prev}}) < \delta$$

Probability of an accident should be upper bounded

Conservative Safety Critics for Exploration, https://arxiv.org/abs/2010.14497, Bharadhwaj, Kumar, Rhinehart, Levine, Shkurti, Garg

Our solution: Constrained Safety Critics (CSC)

$$\underset{\theta}{\text{maximize}} \quad \mathcal{V}^{\pi_\theta}_{\text{task}}(\mathbf{s}_0)$$

$$\text{subject to} \quad \boxed{\mathcal{V}^{\pi_\theta}_{\text{accident}}(\mathbf{s}_0) < \epsilon}$$

$$\text{KL}(\pi_\theta || \pi_{\text{prev}}) < \delta$$

Probability of an accident should be upper bounded

Problem: estimation errors for value function $\mathcal{V}^{\pi_\theta}_{\text{accident}}(\mathbf{s}_0)$ can make the constraint falsely confident

Conservative Safety Critics for Exploration, https://arxiv.org/abs/2010.14497, Bharadhwaj, Kumar, Rhinehart, Levine, Shkurti, Garg

Our solution: Constrained Safety Critics (CSC)

$$\underset{\theta}{\text{maximize}} \quad \mathcal{V}^{\pi_\theta}_{\text{task}}(\mathbf{s}_0)$$

$$\text{subject to} \quad \mathcal{V}^{\pi_\theta}_{\text{accident}}(\mathbf{s}_0) < \epsilon$$

$$\text{KL}(\pi_\theta || \pi_{\text{prev}}) < \delta$$

Probability of an accident should be upper bounded

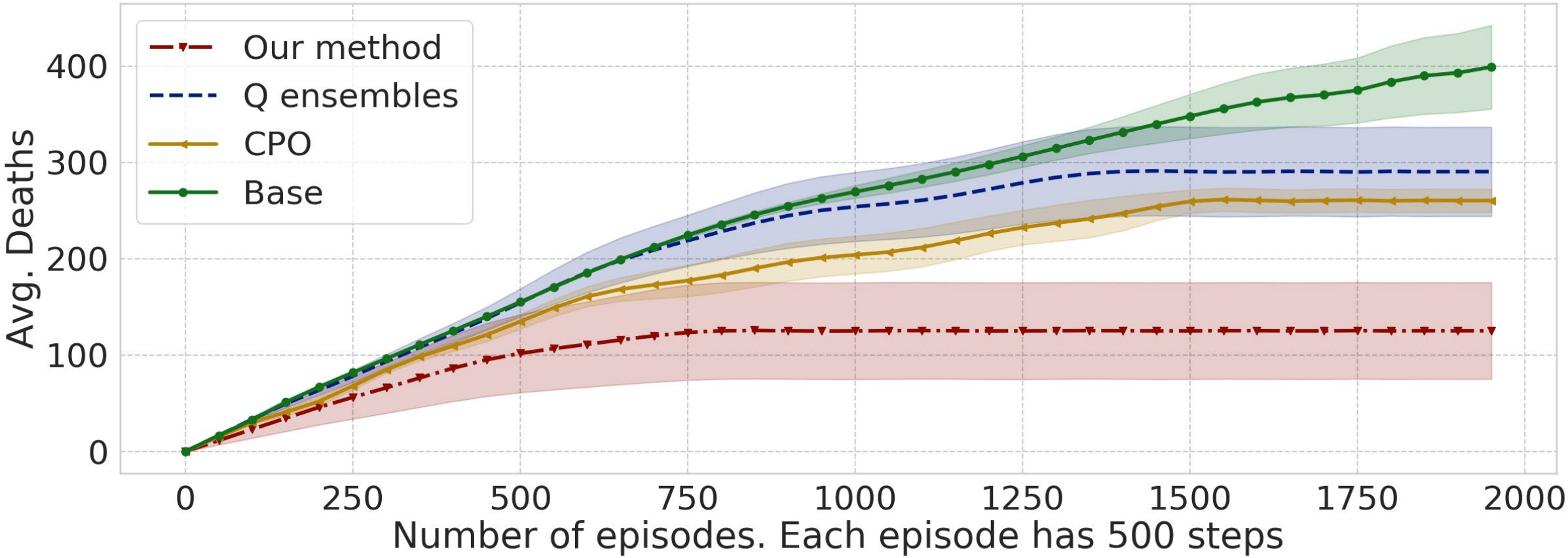
Problem: estimation errors for value function $\mathcal{V}^{\pi_\theta}_{\text{accident}}(\mathbf{s}_0)$ can make the constraint falsely confident

Fix: use the CQL update rule to guarantee that the probability of an accident is overestimated

$$\min_Q \alpha \mathbb{E}_{\mathbf{s}\sim\mathcal{D}}\left[\log \sum_{\mathbf{a}} \exp(Q(\mathbf{s},\mathbf{a})) - \mathbb{E}_{\mathbf{a}\sim\hat{\pi}_\beta(\mathbf{a}|\mathbf{s})}\left[Q(\mathbf{s},\mathbf{a})\right]\right] + \frac{1}{2}\mathbb{E}_{\mathbf{s},\mathbf{a},\mathbf{s}'\sim\mathcal{D}}\left[\left(Q - \hat{\mathcal{B}}^{\pi_k}\hat{Q}^k\right)^2\right] \Rightarrow \mathcal{V}^{\pi_\theta}_{\text{accident}}(\mathbf{s}_0) \leq \hat{\mathcal{V}}^{\pi_\theta}_{\text{accident}}(\mathbf{s}_0) < \epsilon$$

Conservative Safety Critics for Exploration, https://arxiv.org/abs/2010.14497, Bharadhwaj, Kumar, Rhinehart, Levine, Shkurti, Garg

Results: fewer accidents



Constrained Policy Optimization, https://arxiv.org/abs/1705.10528, Achiam, Held, Tamam, Abbeel

Results: task value vs safety