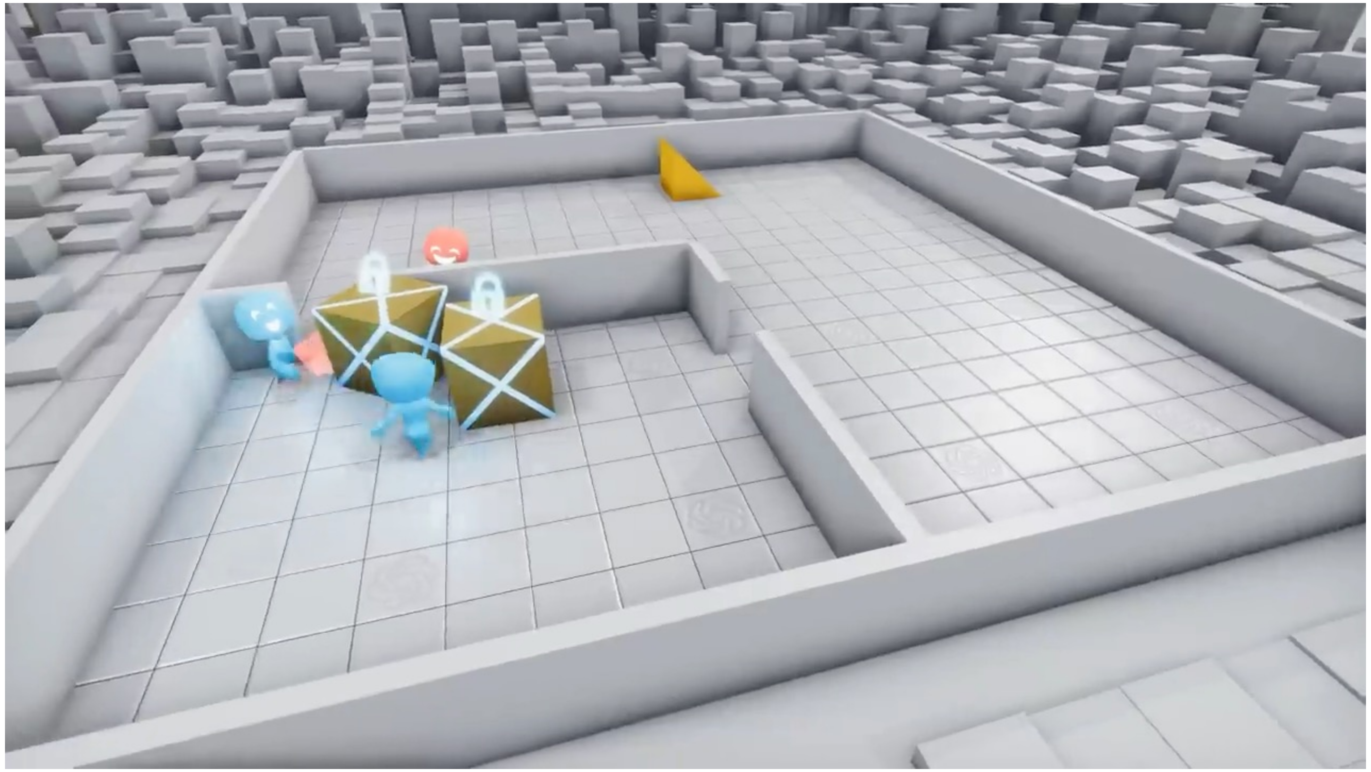
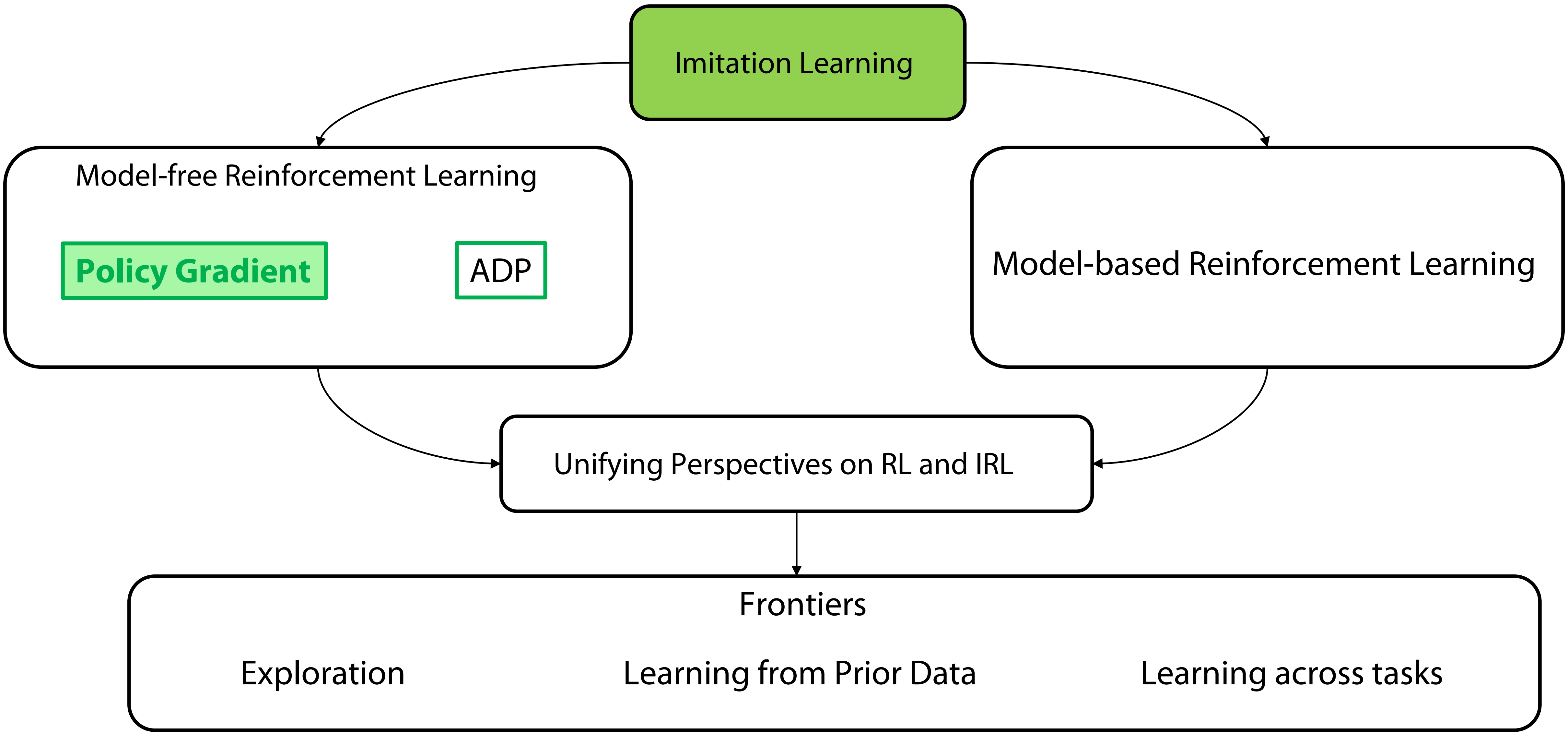
Reinforcement Learning
Spring 2024

Abhishek Gupta

TAs: Patrick Yin, Qiuyu Chen



Class Structure



Why is Policy Gradient sample inefficient?

$$\nabla_\theta J(\theta) = \int p_\theta(\tau) \nabla_\theta \log p_\theta(\tau) d\tau$$

$$\approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t} \nabla_\theta \log \pi_\theta(a_t^i | s_t^i) \sum_{t'=t}^{T} r(s_t^i, a_t^i)$$

On-policy, unable to effectively use past data

High Variance Estimator

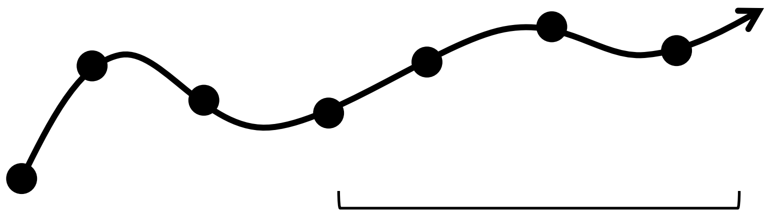
Can we develop a **<u>low variance</u>** **<u>off-policy</u>** RL algorithm that can bootstrap from prior data?

What can we do to lower variance?

$$\nabla_\theta J(\theta) = \int p_\theta(\tau) \nabla_\theta \log p_\theta(\tau) d\tau$$

$$\approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t} \nabla_\theta \log \pi_\theta(a_t^i | s_t^i) \underbrace{\sum_{t'=t}^{T} r(s_t^i, a_t^i)}$$

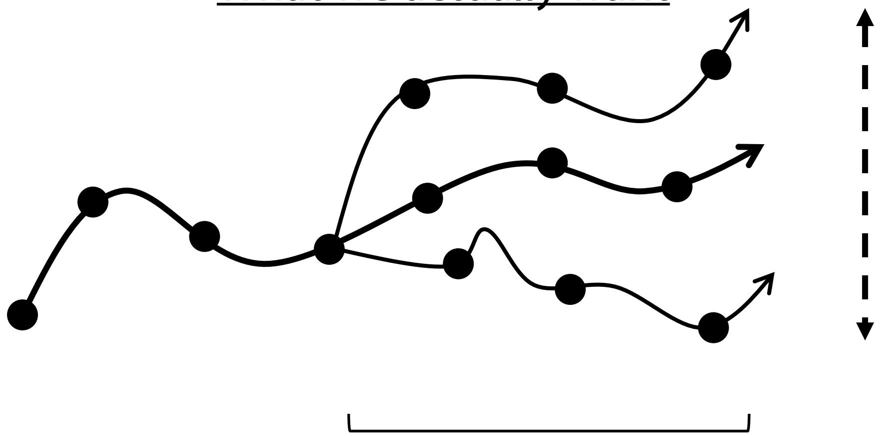
What we do



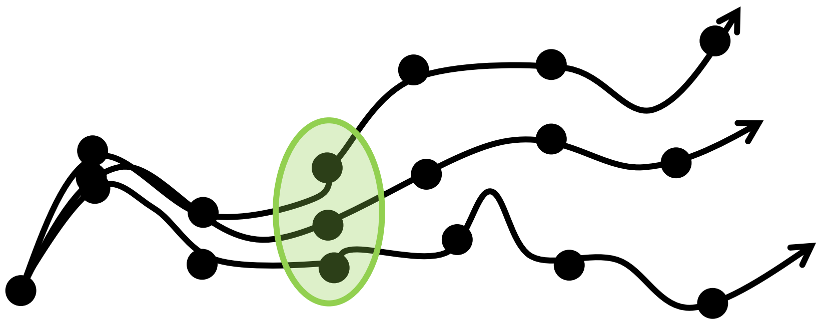
Single sample estimate

Idea: bundle this across many (s, a) with a function approximator

What we actually want



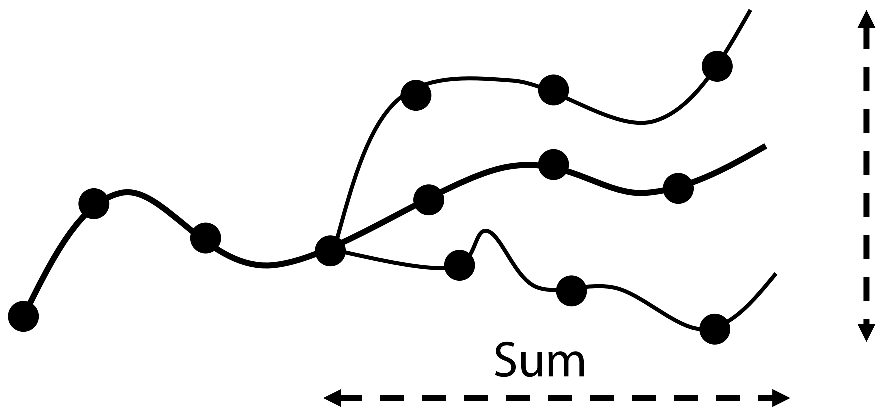
Averaged return estimate



Function approximator bundles return estimates across states

Defining Q and V functions

$$\frac{1}{N}\sum_{i=1}^{N}\sum_{t}\nabla_\theta \log \pi_\theta(a_t^i|s_t^i)\sum_{t'=t}^{T} r(s_t^i, a_t^i)$$



Expected sum of rewards in the future, starting from (s, a) on first step, then π

$$Q^\pi(s_t, a_t) = \mathbb{E}_{\pi_\theta}\left[\sum_{t'=t}^{T} r(s_t', a_t')|s_t, a_t\right]$$

$$V^\pi(s_t) = \mathbb{E}_{\pi_\theta}\left[\sum_{t'=t}^{T} r(s_{t'}, a_{t'})|s_t\right] = \mathbb{E}_{a_t\sim\pi_\theta(\cdot|s_t)}\left[Q(s_t, a_t)\right]$$

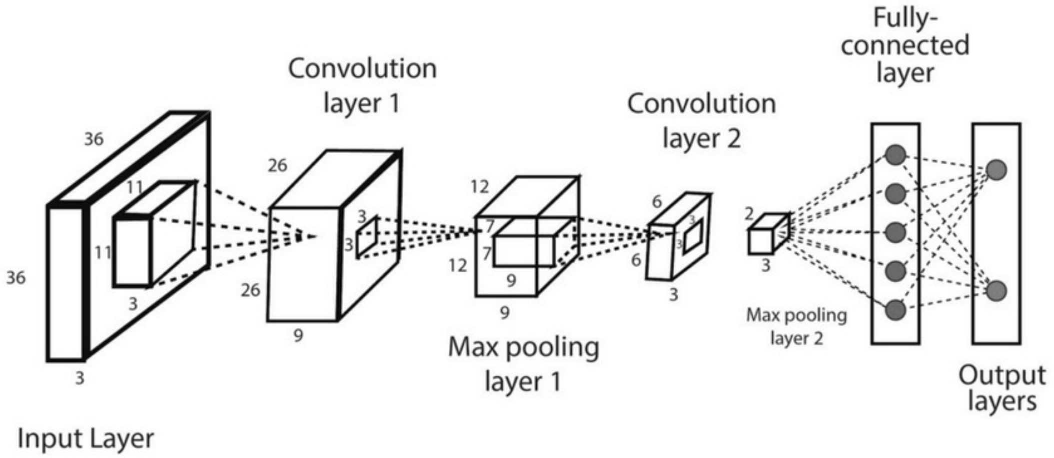
Attempt 0: Monte-Carlo Estimation of Q-Functions

$$\frac{1}{N}\sum_{i=0}^{N}\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t^i|s_t^i)Q^\pi(s_{t'}^i, a_{t'}^i)$$

$$Q^\pi(s_t, a_t) = \mathbb{E}_{\pi_\theta}\left[\sum_{t'=t}^{T} r(s'_t, a'_t)|s_t, a_t\right]$$ ← Monte-carlo approximation

Idea: Regression from (s, a) to Monte-Carlo estimate

State
Action



Return to Go

Unbiased, but high variance!

Attempt 1: Using Recursive Structure

$$\frac{1}{N}\sum_{i=0}^{N}\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t^i|s_t^i)Q^\pi(s_t^i,a_t^i)$$

$$Q^\pi(s_t,a_t)=\mathbb{E}_{\pi_\theta}\left[\sum_{t'=t}^{T} r(s'_t,a'_t)|s_t,a_t\right]$$

Note the definition of a value function $V^\pi(s_t)=\mathbb{E}_{\pi_\theta}\left[\sum_{t'=t}^{T} r(s_{t'},a_{t'})|s_t\right] = \mathbb{E}_{a_t\sim\pi_\theta(\cdot|s_t)}\left[Q(s_t,a_t)\right]$

Average Q-function over actions sampled from policy

Value functions are recursive

$$V^\pi(s_t)=\mathbb{E}_{\pi_\theta}\left[r(s_t,a_t)+\sum_{t'=t+1}^{T} r(s_{t'},a_{t'})|s_t\right]$$

$$V^\pi(s_t)=\mathbb{E}_{\pi_\theta}\left[r(s_t,a_t)+\mathbb{E}_{\pi_\theta}\left[\sum_{t'=t+1}^{T} r(s_{t'},a_{t'})|s_{t+1}\right]\right]$$

← VF!

$$V^\pi(s_t)=\mathbb{E}_{\pi_\theta}\left[r(s_t,a_t)+V^\pi(s_{t+1})\right]$$

Attempt 1: Using Recursive Structure

$$\frac{1}{N}\sum_{i=0}^{N}\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t^i|s_t^i)Q^\pi(s_{t'}^i, a_{t'}^i)$$

$$Q^\pi(s_t, a_t) = \mathbb{E}_{\pi_\theta}\left[\sum_{t'=t}^{T} r(s_t', a_t')|s_t, a_t\right]$$

Fit a value function on on-policy data

$$\min_\phi \mathbb{E}_{(s_i, a_i, {s_i}')\sim\pi}\left[(V_\phi^\pi(s_i) - y_i)^2\right]$$

$$y_i = r(s_i, a_i) + V({s_i}')$$

Compute the policy gradient

$$\nabla_\theta J(\theta) = \frac{1}{N}\sum_{i=0}^{N}\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t^i|s_t^i)(r(s_t, a_t) + V(s_{t+1}) - V(s_t))$$

Collect more data

+ lowers variance

- Still on-policy

Revisit: Generalized Advantage Estimation

Sum up all the estimators in a geometric sum

$$A_N^\theta(s_1, a_1) = r_1 + \gamma r_2 + \cdots + \gamma^{N-1} r_N - V(s_1)$$

$$A_{N-1}^\theta(s_1, a_1) = r_1 + \gamma r_2 + \cdots + \gamma^{N-2} V(s_{N-1}) - V(s_1)$$

$$A_2^\theta(s_1, a_1) = r_1 + \gamma r_2 + \cdots + \gamma^2 V(s_3) - V(s_1)$$

$$A_1^\theta(s_1, a_1) = r_1 + \gamma V(s_2) - V(s_1)$$

Geometric sum

$$A_\lambda^\theta(s_1, a_1) = \sum_{j=1}^{N} \lambda^j A_j^\theta(s, a)$$

λ controls bias-variance tradeoff

Best of both worlds – very similar idea to eligibility traces

Lecture outline

Going from On-Policy to Off-Policy AC

↓

Getting Off-Policy RL to Work

↓

Frontiers of Off-Policy RL

A Closer Look at the Value Bellman Equation

Bellman Update

$$\min_{\phi} \mathbb{E}_{(s_i, a_i, s_i{'}) \sim \pi} \left[(V_\phi^\pi(s_i) - y_i)^2 \right]$$

$$y_i = r(s_i, a_i) + V(s_i{'})$$

Fixed-point iteration algorithm

Bellman equation

$$V^\pi(s_t) = \mathbb{E}_{\pi_\theta}\left[r(s_t, a_t) + V^\pi(s_{t+1}) \right]$$

Holds at convergence

$$x_{n+1} = f(x_n) \implies x^* = f(x^*)$$

Fixed-point

Q: Does this update converge to the true value as a fixed point?

Does this converge?

Q: Does this update converge to the true value as a fixed point?

$$\min_{\phi} \mathbb{E}_{(s_i, a_i, s_i{}')\sim\pi}\left[(V_\phi^\pi(s_i) - y_i)^2\right]$$

$$y_i = r(s_i, a_i) + V(s_i{}')$$

<u>Banachs fixed point theorem</u>

Let (M, d) be a complete metric space.

Let $f : M \to M$ be a contraction.

That is, there exists $q \in [0\,..\,1)$ such that for all $x, y \in M$:

$$d(f(x), f(y)) \le q\, d(x, y)$$

Then there exists a unique fixed point of f.

Let's consider a simple version of this algorithm

$$V_{i+1}^\pi(s) \leftarrow \mathbb{E}_{\substack{s' \sim p(.|s,a) \\ a \sim \pi(\cdot|s')}}\left[r(s, a) + V_i^\pi(s')\right]$$

$$V_{i+1} \leftarrow B_p^\pi V_i^\pi$$

↑

Prove this is a contraction

Does this converge?

$$V^{\pi}_{i+1}(s) \leftarrow \mathbb{E}_{\substack{s' \sim p(.|s,a) \\ a \sim \pi(\cdot|s')}} \left[r(s,a) + V^{\pi}_{i}(s') \right]$$

$$V_{i+1} \leftarrow B^{\pi}_{p} V^{\pi}_{i}$$

Bellman operator

To prove: $d\left(f(x), f(y)\right) \leq q\, d(x, y)$

inf-norm

Value functions

$$V_{i+1} \leftarrow B^{\pi}_{p} V^{\pi}_{i} \qquad U_{i+1} \leftarrow B^{\pi}_{p} U^{\pi}_{i}$$

$$|V_{i+1} - U_{i+1}|_{\infty} \leq \gamma |V_i - U_i|_{\infty}$$

$$|V_{i+1} - U_{i+1}|_{\infty} = \max_{s} |V_{i+1}(s) - U_{i+1}(s)|$$

$$= \max_{s} | \left(\int \pi(a|s) \left(\int p(s'|s,a)(r(s,a) + \gamma U_i(s'))ds \right) da \right) - \left(\int \pi(a|s) \left(\int p(s'|s,a)(r(s,a) + \gamma V_i(s'))ds \right) da \right) |$$

$$= \gamma \max_{s} | \left(\int \pi(a|s) \left(\int p(s'|s,a)(U_i(s') - V_i(s'))ds \right) da \right) |$$

$$\leq \gamma \max_{s} | \left(\int \pi(a|s) \left(\int p(s'|s,a) \max_{x}(U_i(x) - V_i(x))ds \right) da \right) |$$

$$= \gamma \max_{s} | \left(\int \pi(a|s) \max_{x}(U_i(x) - V_i(x)da \right) |$$

$$= \gamma \max_{x} |U_i(x) - V_i(x)| = \gamma |U_i - V_i|_{\infty}$$

Contraction, hence converges to a fixed point

Does this converge for arbitrary function approximation?

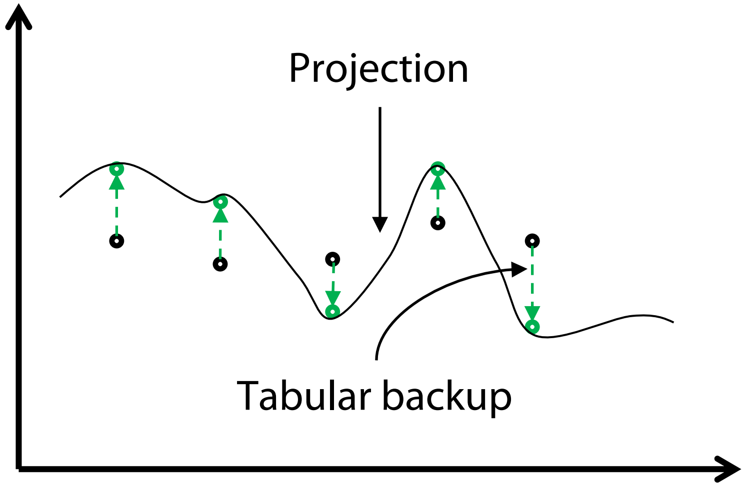
For arbitrary function approximation, it is not just a Bellman backup

$$V^{\pi}_{i+1}(s) \leftarrow \mathbb{E}_{\substack{s' \sim p(.|s,a) \\ a \sim \pi(\cdot|s')}} \left[r(s,a) + V^{\pi}_{i}(s') \right]$$

$$V_{i+1} \leftarrow B^{\pi}_{p} V^{\pi}_{i}$$

We perform a Bellman backup + a projection

Projection – find closest element of function class to approximate tabular values



$$\min_{\phi} \mathbb{E}_{(s_i, a_i, {s_i}') \sim \pi} \left[(V^{\pi}_{\phi}(s_i) - y_i)^2 \right]$$

$$y_i = r(s_i, a_i) + V({s_i}^{'})$$

Backup may be a contraction, but backup + projection may not be

Why is this not enough?

Fit a value function on on-policy data

$$\min_{\phi} \mathbb{E}_{(s_i, a_i, s_i{}')\sim\pi}\left[(V_\phi^\pi(s_i) - y_i)^2\right]$$

$$y_i = r(s_i, a_i) + V(s_i{}')$$

Compute the policy gradient

$$\nabla_\theta J(\theta) = \frac{1}{N}\sum_{i=0}^{N}\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t^i|s_t^i)(r(s_t, a_t) + V(s_{t+1}) - V(s_t))$$

Collect more data

+ lowers variance

- Still on-policy

Need to be able to use arbitrary data

Past iterates data

Other tasks data

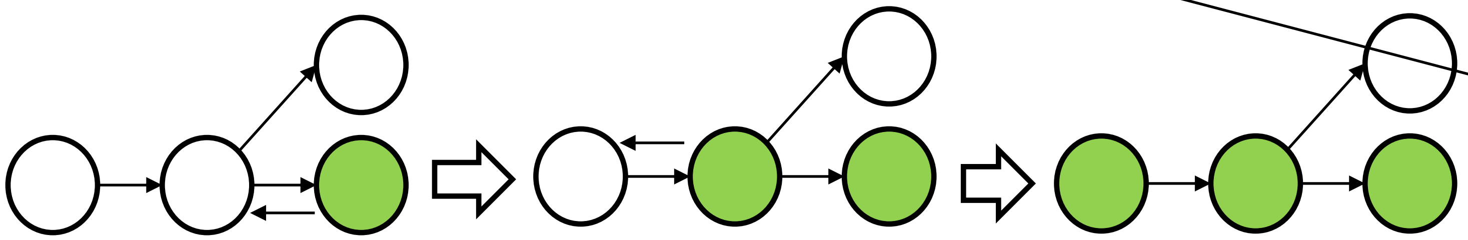
Attempt 2: Recursive structure in Q functions directly

Q functions have special recursive structure themselves!

$$Q^\pi(s_t, a_t) = \mathbb{E}_{\pi_\theta}\left[\sum_{t'=t}^{T} r(s'_t, a'_t) | s_t, a_t\right]$$

$$= r(s_t, a_t) + \mathbb{E}_\pi\left[\sum_{t'=t+1} r(s_{t'}, a_{t'}) | s_{t+1}, a_{t+1} \sim \pi(.|s_{t+1})\right]$$

Bellman equation $$Q^\pi(s_t, a_t) = r(s_t, a_t) + \mathbb{E}_{\substack{s_{t+1} \sim p(.|s_t, a_t) \\ a_{t+1} \sim \pi_\theta(.|s_{t+1})}} \left[Q^\pi(s_{t+1}, a_{t+1})\right]$$



Can be from different policies

Decompose temporally via dynamic programming

Off-policy!

Learning Q-functions via Dynamic Programming

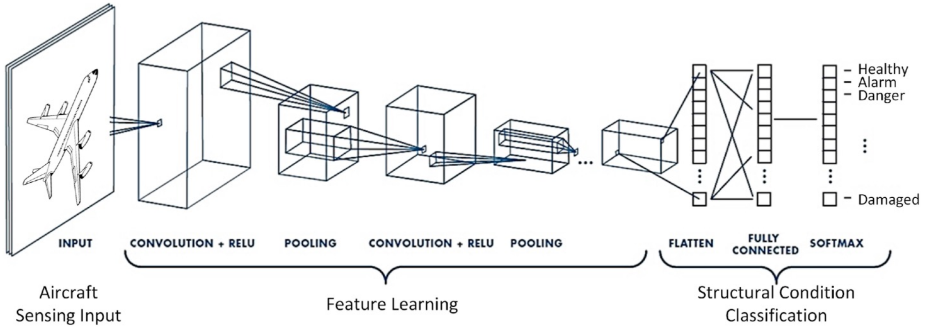
<u>Policy Evaluation:</u> Try to minimize Bellman Error (almost)

Bellman equation

$$Q^\pi(s_t, a_t) = r(s_t, a_t) + \mathbb{E}_{\substack{s_{t+1} \sim p(.|s_t, a_t) \\ a_{t+1} \sim \pi_\theta(.|s_{t+1})}} [Q^\pi(s_{t+1}, a_{t+1}]$$

Same function approximator

How can we convert this recursion into an off-policy learning objective?

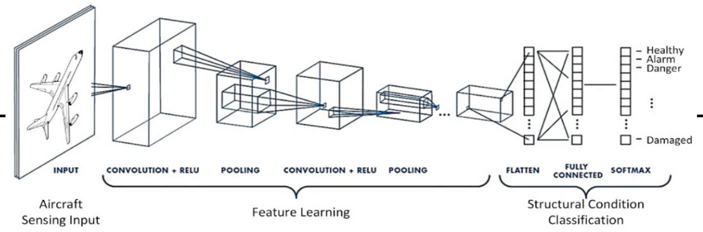


Why is this not just the gradient of the Bellman Error?

$$\min_{\phi} \mathbb{E}_{(s_t,a_t,s_{t+1})\sim\mathcal{D}} \left(Q^{\pi}_{\phi}(s_t,a_t) - \left(r(s_t,a_t) + \mathbb{E}_{a_{t+1}\sim\pi_\theta(a_{t+1}|s_{t+1})} \left[Q^{\pi}_{\hat{\phi}}(s_{t+1},a_{t+1}) \right]\right)\right)^2$$

Approximate using stochastic optimization

$$\min_{\phi} \mathbb{E}_{(s_t,a_t,s_{t+1})\sim\mathcal{D}} \left(Q^{\pi}_{\phi}(s_t,a_t) - \left(r(s_t,a_t) + Q^{\pi}_{\hat{\phi}}(s_{t+1},a_{t+1})\right)\right)^2 \quad a_{t+1} \sim \pi(\cdot|s_{t+1})$$



"Target" moves too much

Often tough empirically with function approximators

Expectation inside the square, hard to be unbiased

Note: this may look like gradient descent on Bellman error, it is not!

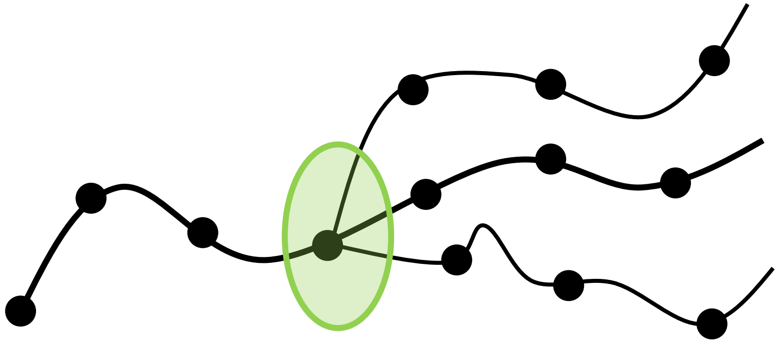
Improving Policies with Learned Q-functions

<u>Policy Improvement:</u> Improve policy with **<u>policy gradient</u>**

$$\max_{\theta} \mathbb{E}_{s \sim \mathcal{D}, a \sim \pi_\theta(a|s)} \left[Q^{\pi_\theta}(s, a) \right]$$

Replace Monte-Carlo sum of rewards with learned Q function

Lowers variance compared to policy gradient!



+ off-policy

Policy Updates – REINFORCE or Reparameterization

Let's look a little deeper into the policy update

$$\max_\theta J(\theta) = \max_\theta \mathbb{E}_{s \sim \mathcal{D}} \mathbb{E}_{a \sim \pi_\theta(.|s)} \left[Q^\pi(s, a) \right]$$

<u>Likelihood Ratio/Score Function</u>

$$\nabla_\theta J(\theta) = \mathbb{E}_{s \sim \mathcal{D}} \mathbb{E}_{a \sim \pi_\theta(.|s)} \left[\nabla_\theta \log \pi_\theta(a|s) Q^\pi(s, a) \right]$$

Easier to Apply to Broad Policy Class

<u>Pathwise derivative/Reparameterization</u>

$$\nabla_\theta J(\theta) = \mathbb{E}_{s \sim \mathcal{D}} \mathbb{E}_{z \sim p(z)} \left[\nabla_a Q^\pi(s, a)|_{a = \mu_\theta + z\sigma_\theta} \nabla_\theta (\mu_\theta + z\sigma_\theta) \right]$$

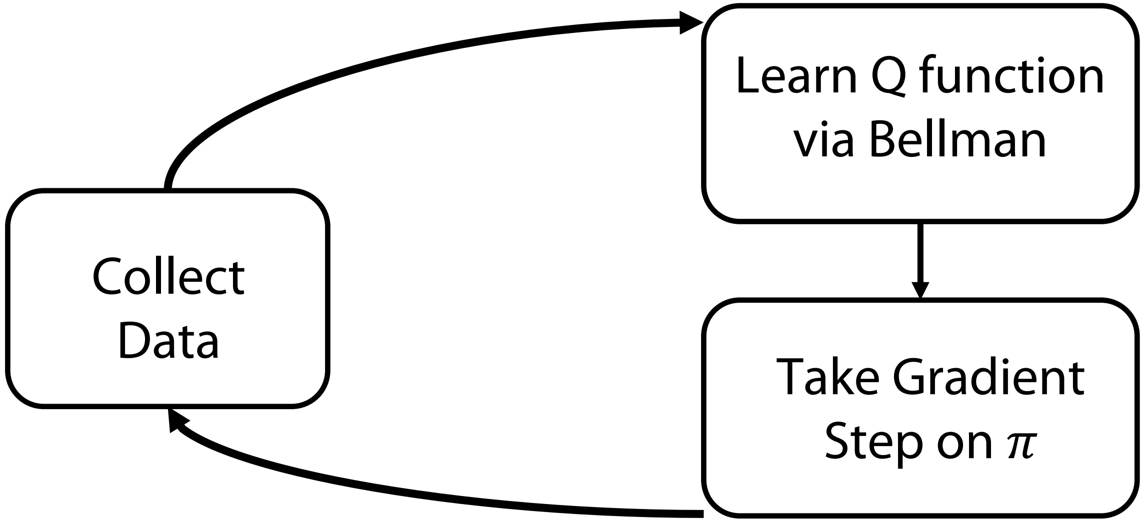
Lower variance (empirically)

Remember Lecture 2 and discussion of when gradients can be moved inside

Actor-Critic: Policy Gradient in terms of Q functions

Critic: learned via the Bellman update (Policy Evaluation)

$$\min_{\phi} \mathbb{E}_{(s_t, a_t, s_{t+1}) \sim \mathcal{D}} \left(Q^{\pi}_{\phi}(s_t, a_t) - (r(s_t, a_t) + Q^{\pi}_{\hat{\phi}}(s_{t+1}, a_{t+1})) \right)^2 \quad a_{t+1} \sim \pi(\cdot | s_{t+1})$$

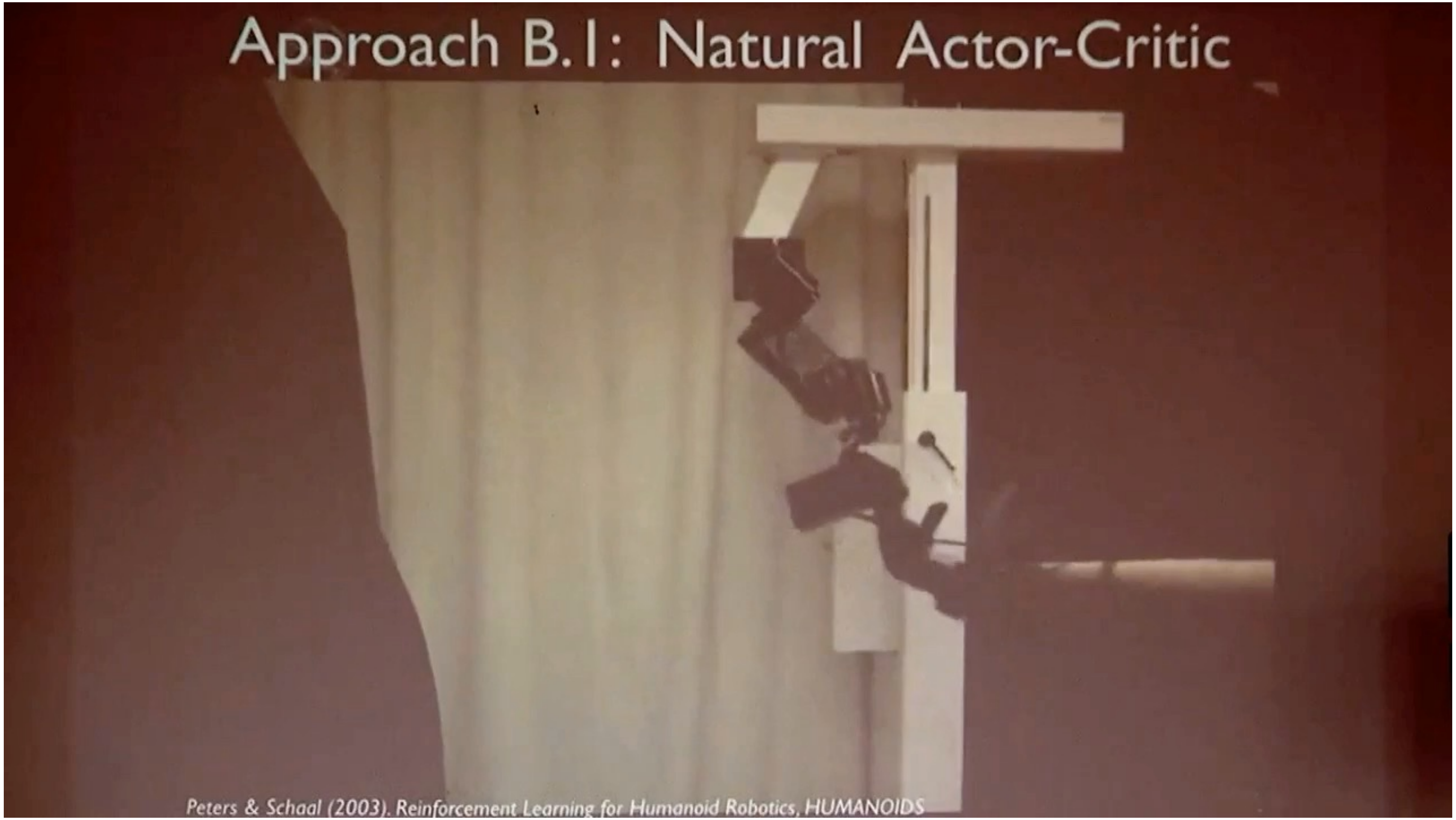


Lowers variance and is off-policy!

Actor: updated using learned critic (Policy Improvement)

$$\max_{\pi} \mathbb{E}_{s \sim \mathcal{D}} \mathbb{E}_{a \sim \pi(.|s)} \left[Q^{\pi}(s, a) \right]$$

Actor-Critic in Action



Lecture outline

Going from On-Policy to Off-Policy AC

↓

Getting Off-Policy RL to Work

↓

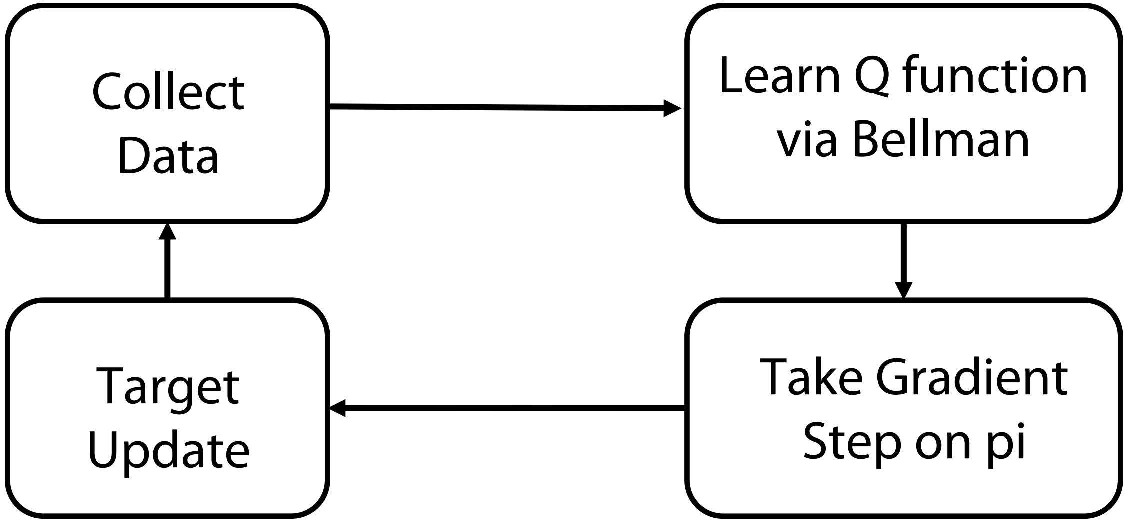
Frontiers of Off-Policy RL

What can we do to make off-policy algorithms work in practice?

Going from Batch Updates to Online Updates

This algorithm can go from full batch mode to fully online updates

1 sample



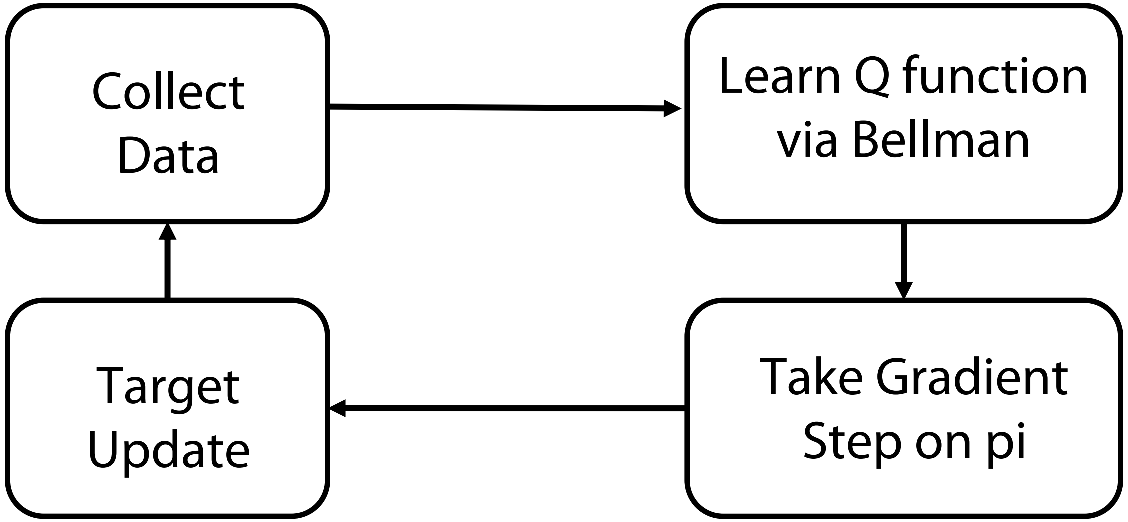
Critic: 1 gradient step on Bellman error

Actor: 1 gradient step on policy gradient

Allows for much more immediate updates

Challenges of doing online updates

1 sample



Critic: 1 gradient step on Bellman error

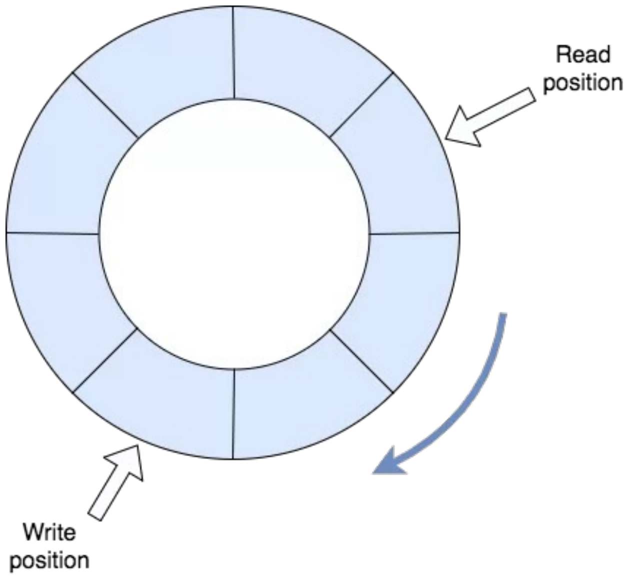
Actor: 1 gradient step on policy gradient

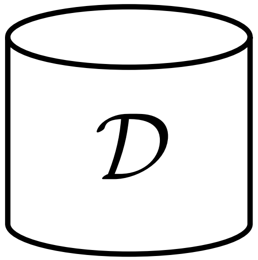
When updates are performed online, two issues persist:

1. Correlated updates since samples are correlated
2. Optimization objective changes constantly, unstable

Decorrelating updates with replay buffers

Updates can be decorrelated by storing and shuffling data in a replay buffer





Instead of doing updates in order, sample batches from replay buffer

How?

Sampled from replay buffer

$$\min_{\phi} \mathbb{E}_{\substack{(s_t,a_t,s_{t+1})\sim\mathcal{D} \\ a_{t+1}\sim\pi(\cdot|s_{t+1})}}\left[\left(Q^{\pi}_{\phi}(s_t,a_t) - (r(s_t,a_t) + Q^{\pi}_{\hat{\phi}}(s_{t+1},a_{t+1}))\right)^2\right]$$

$$\max_{\pi} \mathbb{E}_{s\sim\mathcal{D}}\mathbb{E}_{a\sim\pi(.|s)}\left[Q^{\pi}(s,a)\right]$$

1. Sample uniformly
2. Prioritize by TD-error
3. Prioritize by target error
4. … open area of research!

Slowing moving targets with target networks

Continuous updates can be unstable since there is a churn of projection and backup

$$\min_{\phi} \mathbb{E}_{\substack{(s_t, a_t, s_{t+1}) \sim \mathcal{D} \\ a_{t+1} \sim \pi(\cdot | s_{t+1})}} \left[\left(Q^{\pi}_{\phi}(s_t, a_t) - (r(s_t, a_t) + Q^{\pi}_{\hat{\phi}}(s_{t+1}, a_{t+1})) \right)^2 \right]$$

If we set $\bar{\phi}$ to ϕ every update, the update becomes very unstable

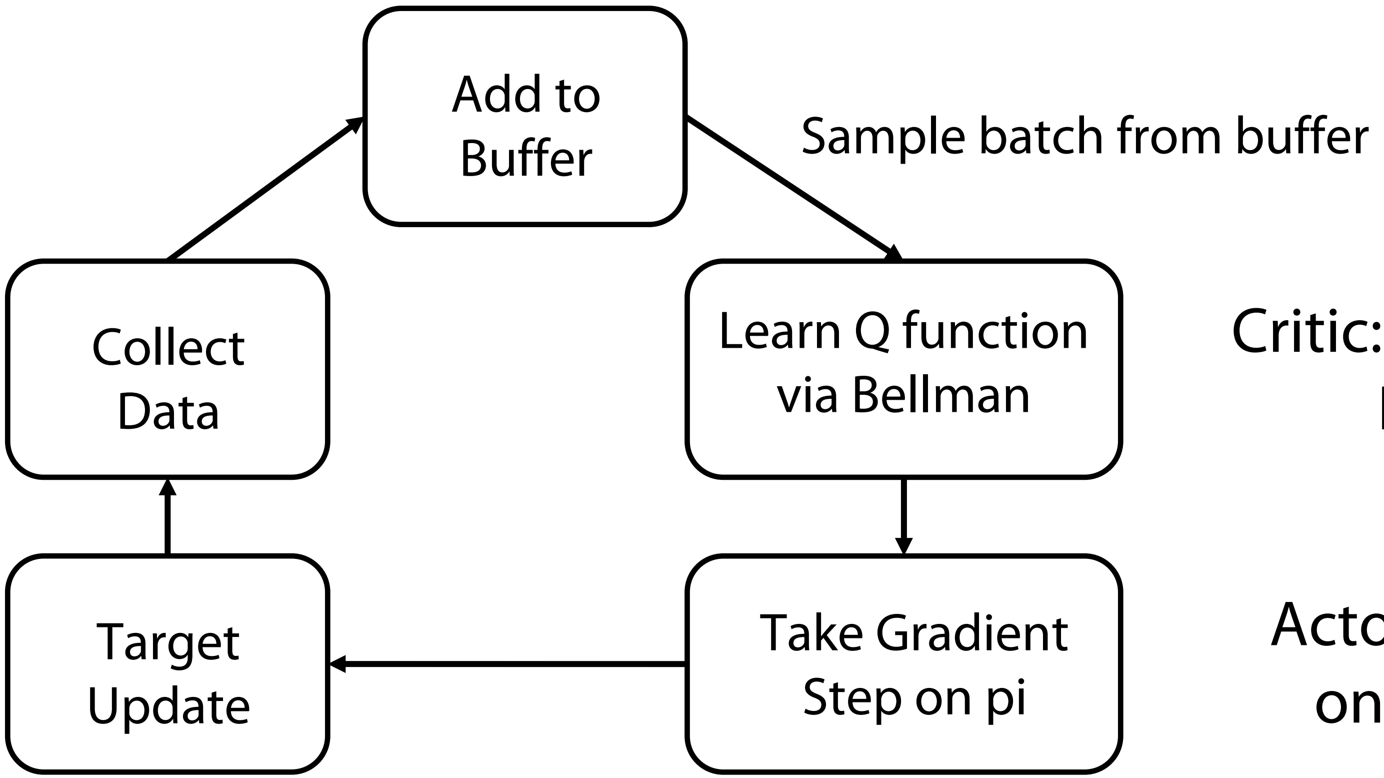
⇓

Move $\bar{\phi}$ to ϕ slowly!

$$\bar{\phi} = (1 - \tau)\phi + \tau\bar{\phi}$$

Polyak averaging

A Practical Off-Policy RL Algorithm



1 sample

Polyak Averaging

Critic: 1 gradient step on Bellman error

Actor: 1 gradient step on policy gradient

Simplify -- Can we get rid of a parametric actor?

Critic Update

$$\min_{\phi} \mathbb{E}_{(s,a,s')\sim\mathcal{D}} \left[Q^{\pi}_{\phi}(s_t, a_t) - (r(s_t, a_t) + \mathbb{E}_{a_{t+1}\sim\pi(.|s_{t+1})} \left[Q_{\bar{\phi}}(s_{t+1}, a_{t+1})\right]\right]^2$$

Actor Update

$$\max_{\pi} \mathbb{E}_{s\sim\mathcal{D}}\mathbb{E}_{a\sim\pi(.|s)} \left[Q^{\pi}(s, a)\right]$$

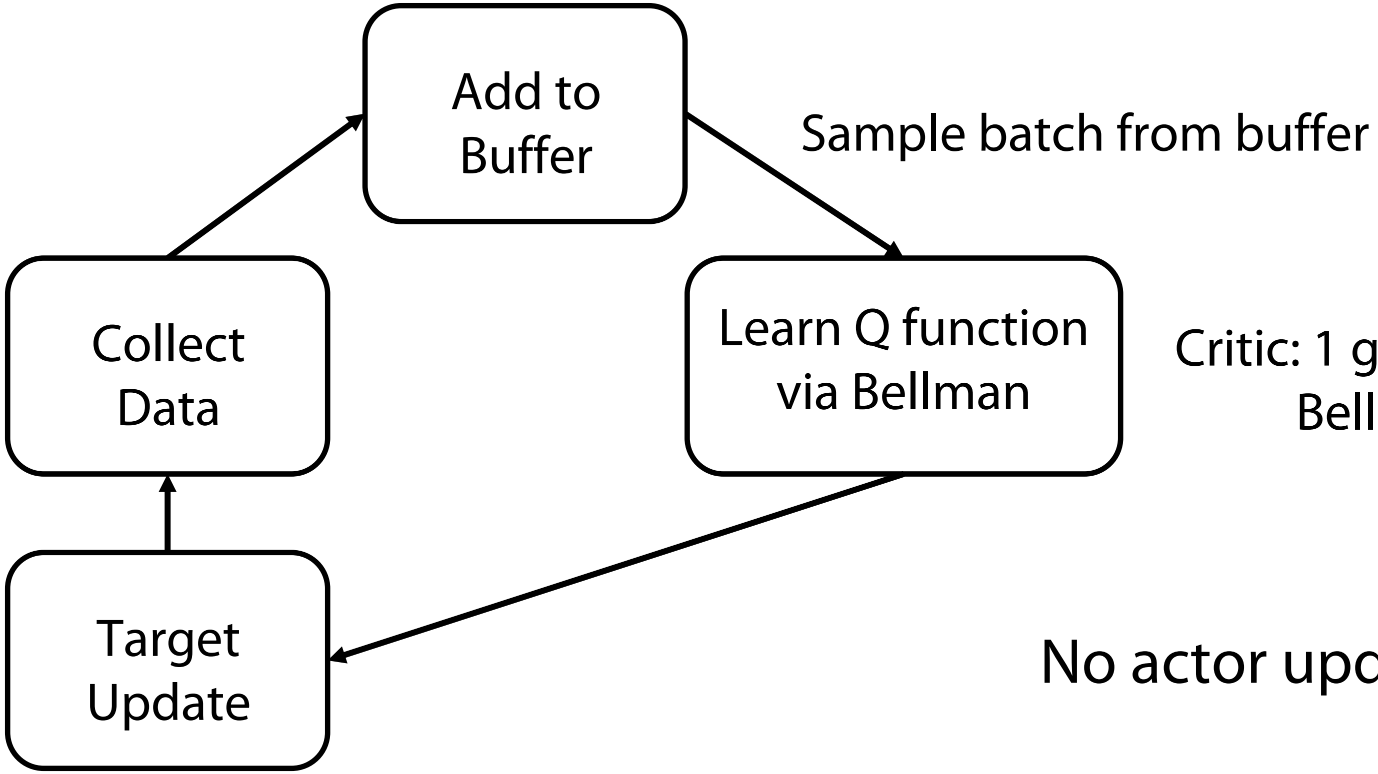
What if we removed this explicit actor completely?

Directly Learning Q*

$$\min_{\phi} \mathbb{E}_{(s,a,s')\sim\mathcal{D}} \left[\left[Q^{\pi}_{\phi}(s_t, a_t) - (r(s_t, a_t) + \max_{a_{t+1}} \left[Q_{\bar{\phi}}(s_{t+1}, a_{t+1}) \right]) \right]^2 \right]$$

$$\pi(a|s) = \max_a Q(s, a)$$

Directly do max in the Bellman update



Critic: 1 gradient step on Bellman error

1 sample

Polyak Averaging

No actor updates, just learn Q!

How can we maximize w.r.t a?

$$\pi(a|s) = \max_a Q(s, a)$$

↑

Analytic maximization can be very difficult to perform in continuous action spaces
Much easier in discrete spaces! → just do categorical max!

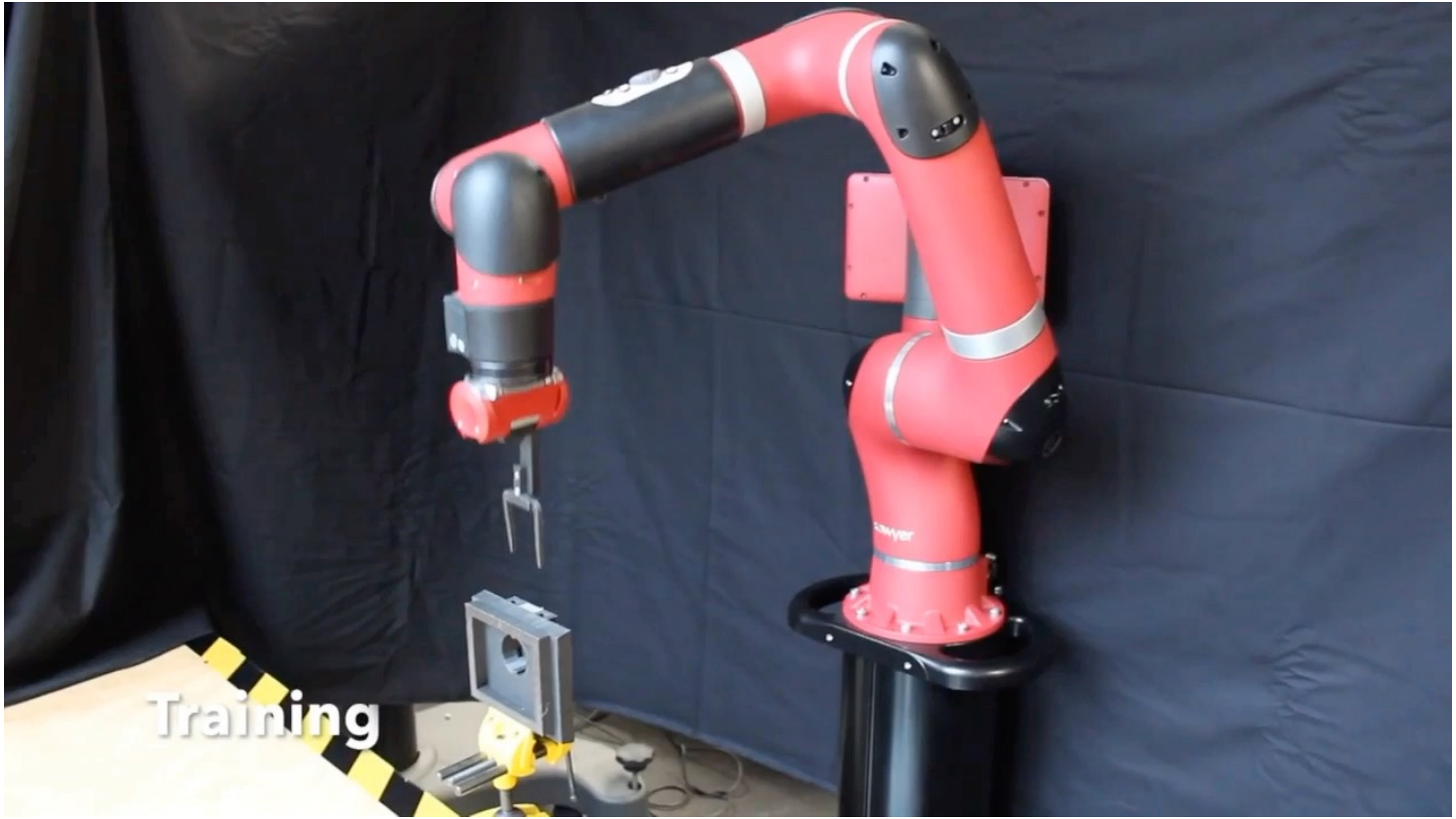
Some ideas to do maximization:

1. Sampling based (QT-opt (Kalashnikov et al))
2. Optimization based (NAF, Gu et al)

Practical Actor-Critic in Action

Trained using QT-Opt

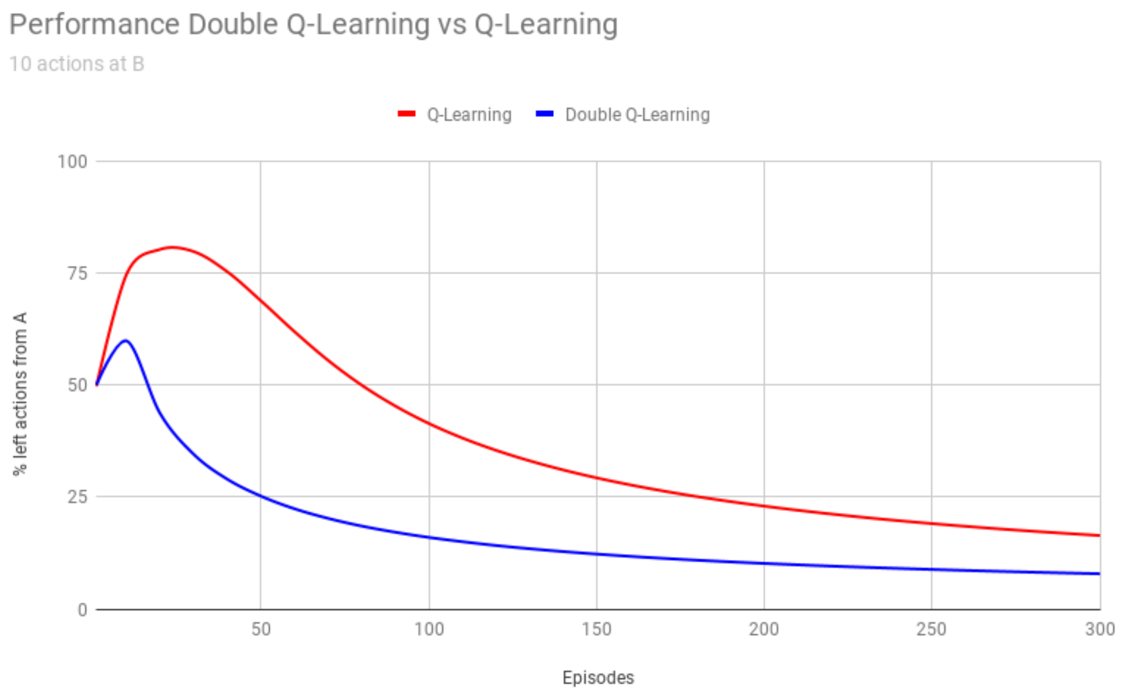
Practical Actor-Critic in Action



Trained using DDPG

What can we do to make them match on-policy algorithms in asymptotic performance?

Where does this fail?



Some issues remain:

1. **Overestimation bias**
2. Insufficient exploration

Let’s try and understand these!

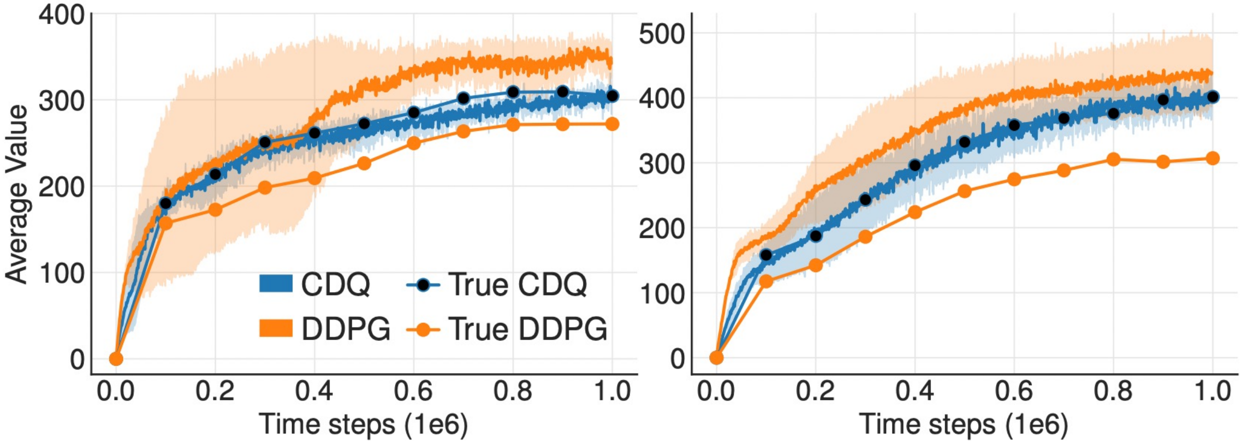
Overestimation Bias in Actor-Critic

Optimized Q's are often overly optimistic

$$\min_{\phi} \mathbb{E}_{(s,a,s')\sim\mathcal{D}} \left[\left[Q^{\pi}_{\phi}(s_t, a_t) - \left(r(s_t, a_t) + \max_{a_{t+1}} \left[Q_{\bar{\phi}}(s_{t+1}, a_{t+1}) \right] \right) \right]^2 \right]$$

Q is meant to be an expectation

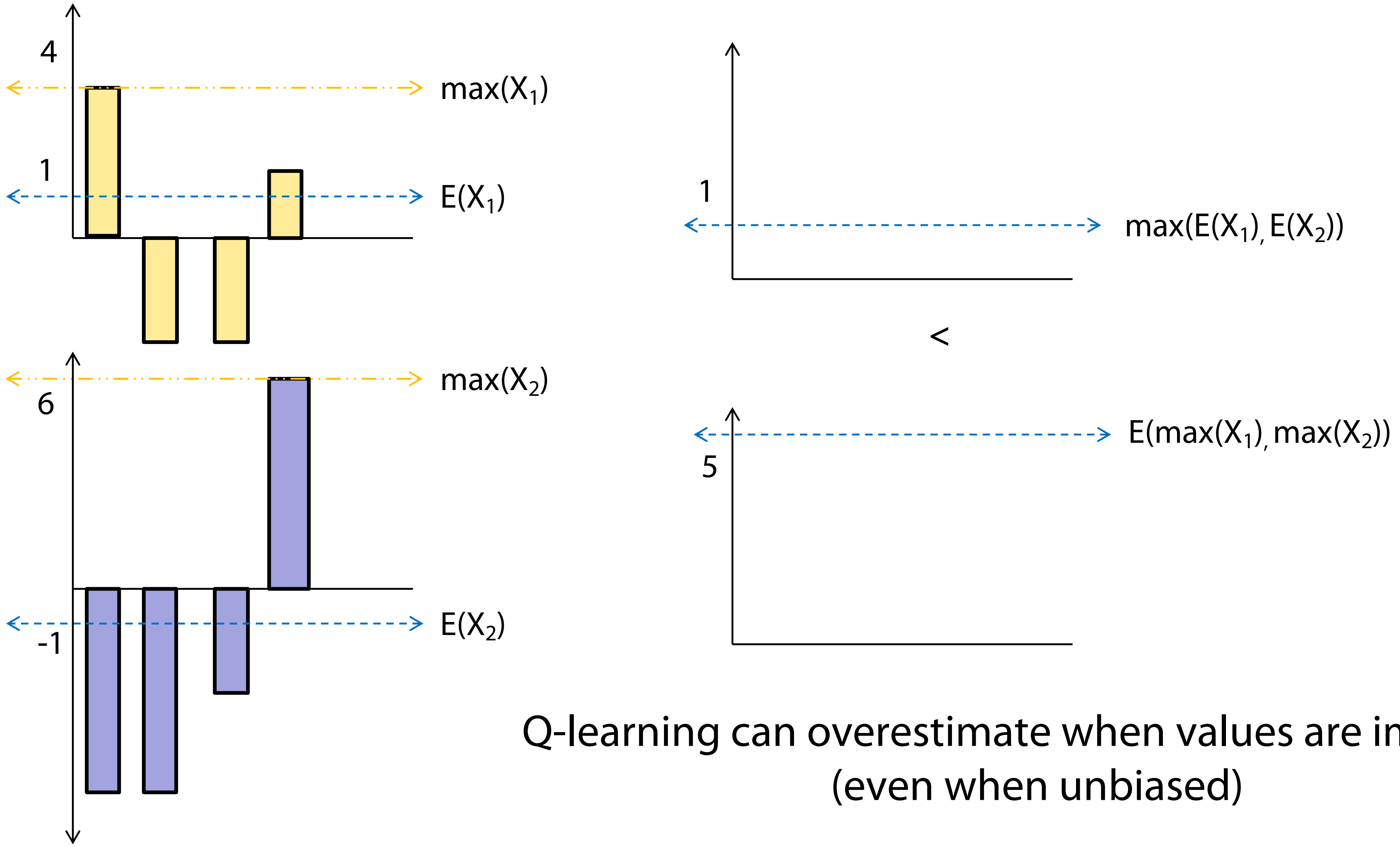
→ actually a random variable because of limited data/stochasticity



(a) Hopper-v1 (b) Walker2d-v1

E(max) > max(E), so values are optimistic

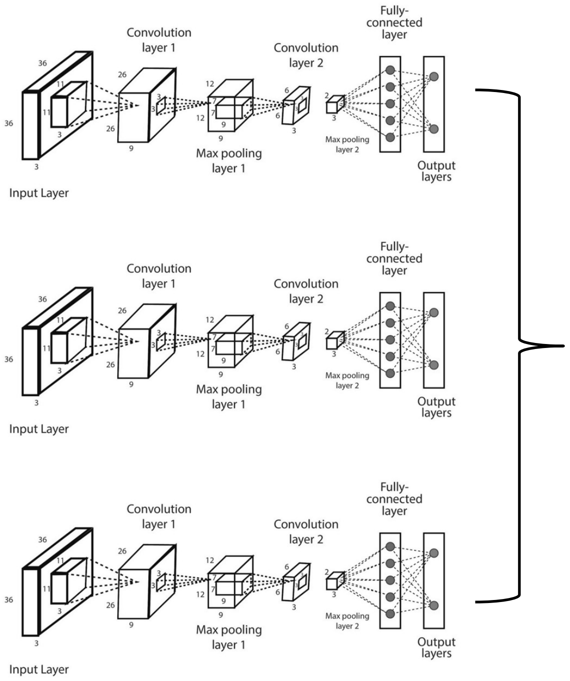
Overestimation Bias in Actor-Critic



Q-learning can overestimate when values are imperfect (even when unbiased)

Overestimation Bias in Actor-Critic – Ensemble Q

Learn two (or N) independent measures of Q, take the minimum
→ pessimistic on random variable



Independent updates

Critic

$$y_j = r(s,a) + \gamma \min_{i=1,\dots,N} Q_{\bar{\phi}_i}(s', \pi_\theta(s'))$$

$$\min_{\phi_j} \mathbb{E}_{(s,a,s')\sim\mathcal{D}} \left[(Q_{\phi_j}(s,a) - y_j)^2 \right]$$

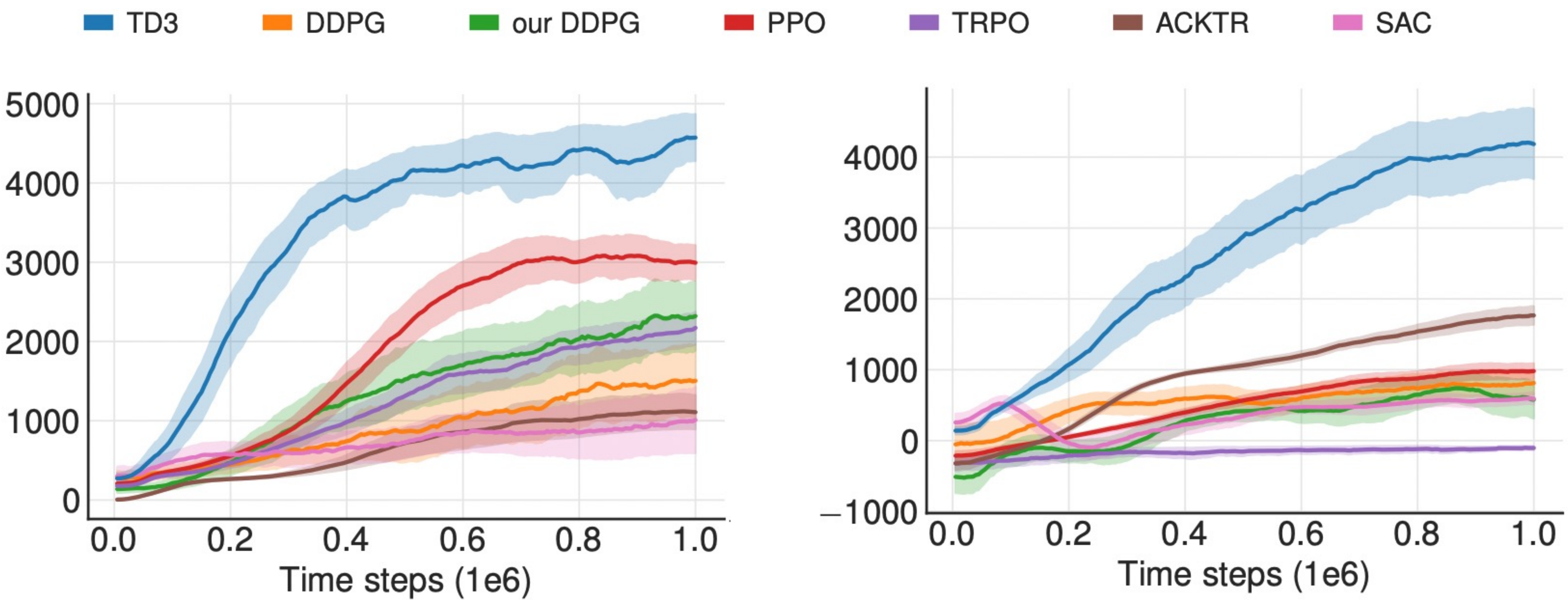
Actor

$$\max_{\theta_j} \mathbb{E}_{s\sim\mathcal{D}} \mathbb{E}_{a\sim\pi_{\theta_j}} \left[Q_{\phi_j}(s,a) \right]$$

Significantly improves overestimation and in turn sample efficiency!

Overestimation Bias in Actor-Critic

Significantly improves overestimation and in turn sample efficiency!



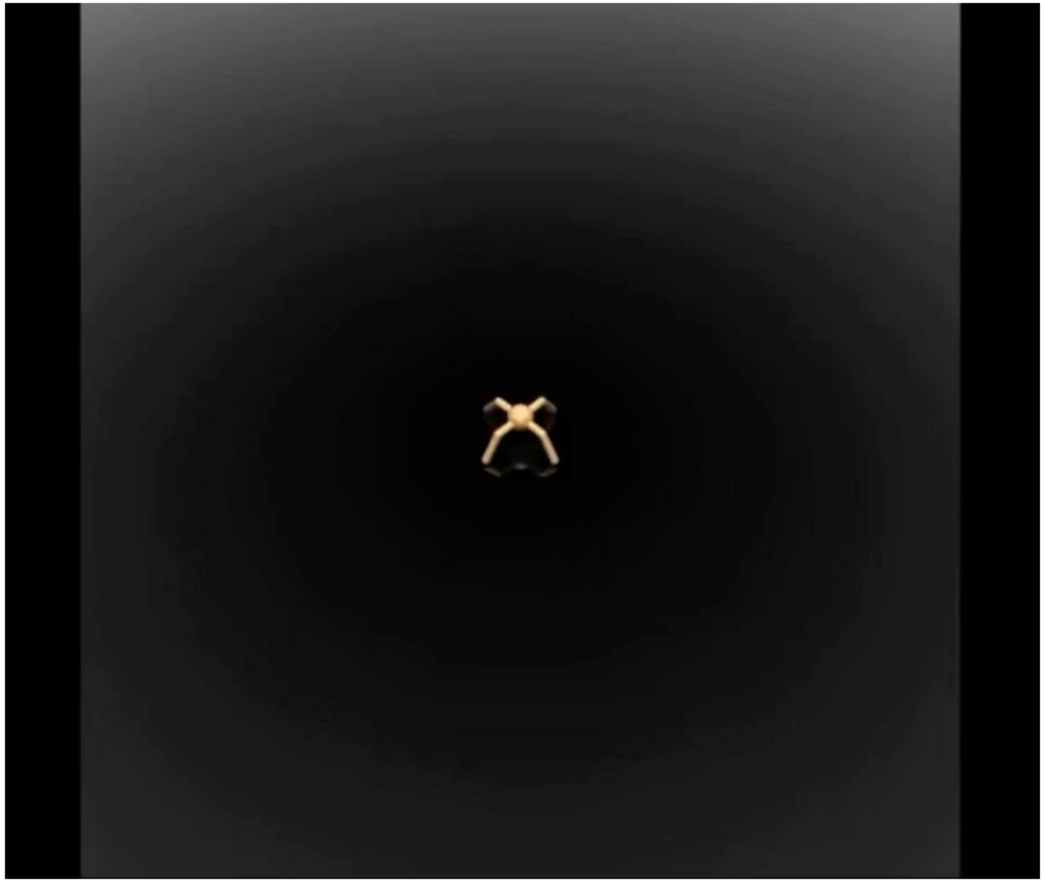
(c) Walker2d-v1

(d) Ant-v1

Double Actor Critic in Action

Double Actor Critic in Action

Where does this fail?

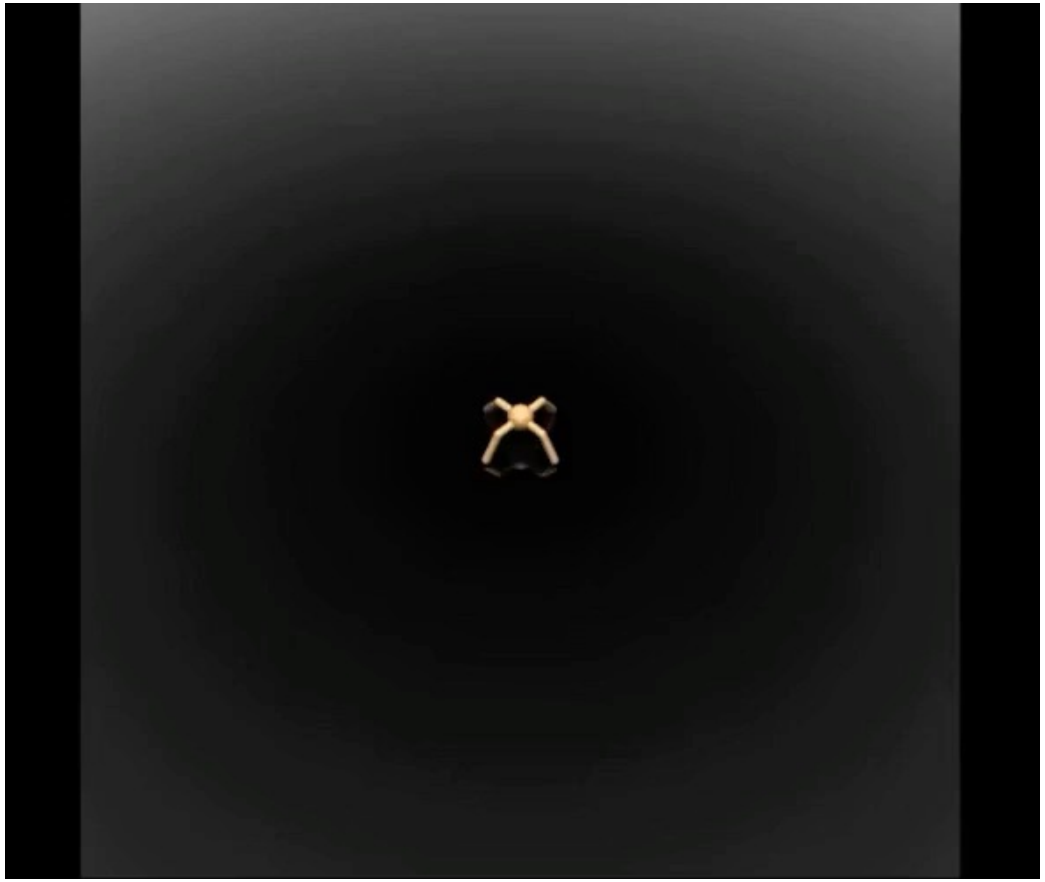


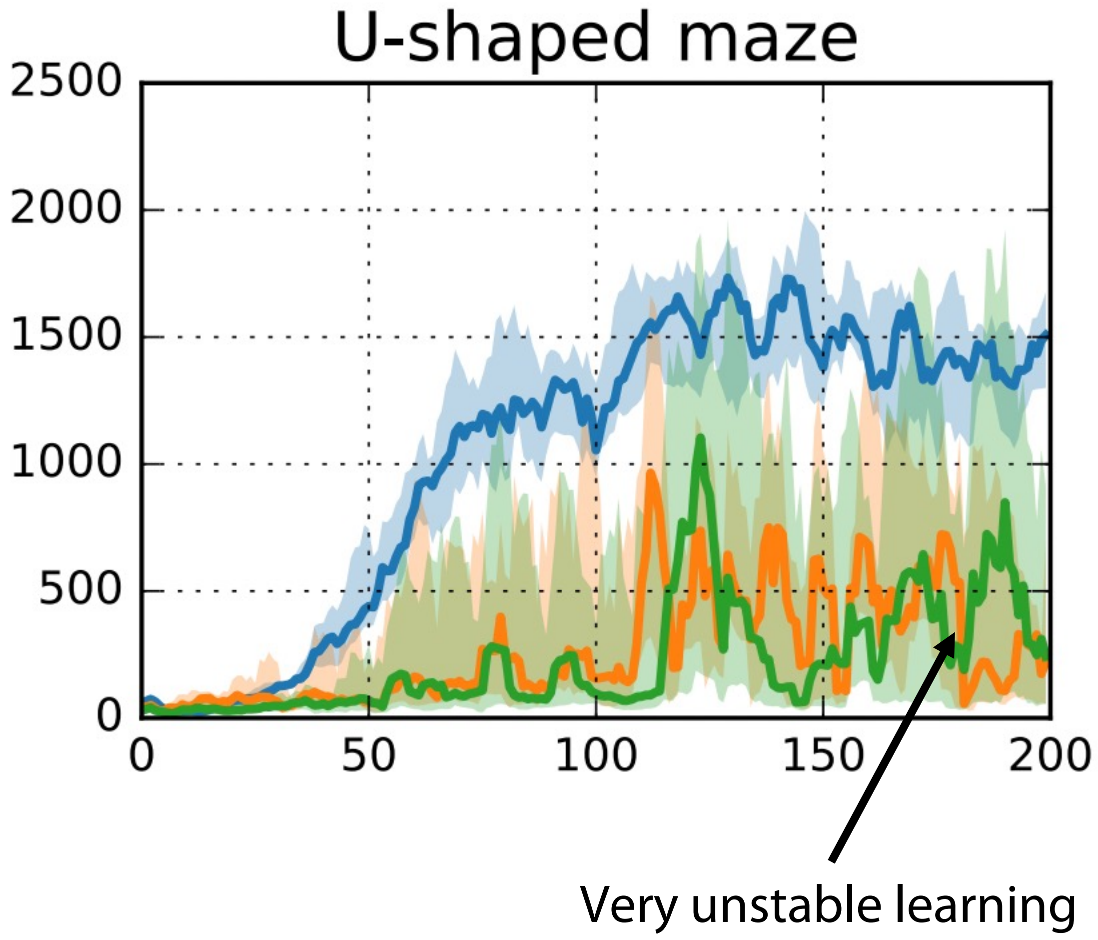
Some issues remain:
1. Overestimation bias
2. **Insufficient exploration**

Let's try and understand these!

Collapse of Exploration in Off-Policy RL

Deep RL policies will often converge prematurely or explore insufficiently





Addressing Policy Collapse in Off-Policy RL

Adding entropy to the RL objective can help significantly

$$\max_{\pi} \mathbb{E}_{\pi}\left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t)\right]$$

$$\Downarrow$$

$$\max_{\pi} \mathbb{E}_{\pi}\left[\sum_{t=0}^{T} \gamma^t \left(r(s_t, a_t) + \alpha \mathcal{H}(\pi(.|s_t))\right)\right]$$

Simple change in on-policy RL

$$\mathbb{E}_{\pi}\left[\sum_{t=0}^{T}\left[\nabla_\theta \log \pi_\theta(a_t|s_t) \sum_{t'=t}^{T} \gamma^{t'-t}(r(s_t, a_t) + \alpha \mathcal{H}(\pi(.|s_t))\right] + \alpha \nabla_\theta \mathcal{H}(\pi_\theta(.|s_t))\right]$$

(via chain rule)

Max-Ent Off-Policy RL

$$\max_{\pi} \mathbb{E}_{\pi} \left[\sum_{t=0}^{T} \gamma^t (r(s_t, a_t) + \alpha \mathcal{H}(\pi(.|s_t)) \right]$$

Work through the recursion, same as with the regular Bellman

Critic – Policy Evaluation

$$\min_{\phi} \mathbb{E}_{\substack{(s_t, a_t, s_{t+1}) \sim \mathcal{D} \\ a_{t+1} \sim \pi(\cdot|s_{t+1})}} \left[(Q_{\phi}^{\pi}(s_t, a_t) - (r(s_t, a_t) + Q_{\hat{\phi}}^{\pi}(s_{t+1}, a_{t+1})) - \alpha \log \pi(a_{t+1}|s_{t+1}))^2 \right]$$

Actor – Policy Improvement

$$\max_{\pi} \mathbb{E}_{s \sim \mathcal{D}} \left[\mathbb{E}_{a \sim \pi(\cdot|s)} \left[Q_{\phi}^{\pi}(s, a) - \alpha \log \pi(a|s) \right] \right]$$

Soft Bellman Equation from Max-Ent RL

Optimize a ”soft” Bellman equation

$$Q(s_t, a_t) \leftarrow r_t + \gamma \mathbb{E}_{s_{t+1} \sim p_s}\left[V(s_{t+1})\right]$$

$$V(s_t) \leftarrow \max_a Q(s_t, a)$$

$$\pi(a|s_t) \leftarrow \arg\max_a Q(s_t, a)$$

$$Q_{\text{soft}}(s_t, a_t) \leftarrow r_t + \gamma \mathbb{E}_{s_{t+1} \sim p_s}\left[V_{\text{soft}}(s_{t+1})\right]$$

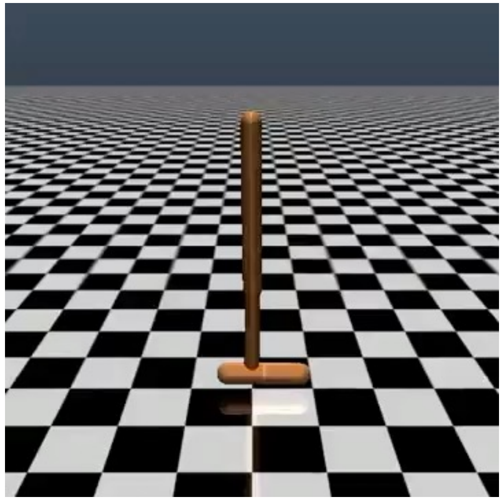
$$V_{\text{soft}}(s_t) \leftarrow \alpha \log \int_{\mathcal{A}} \exp\left(\frac{1}{\alpha} Q_{\text{soft}}(s_t, a')\right) da'$$

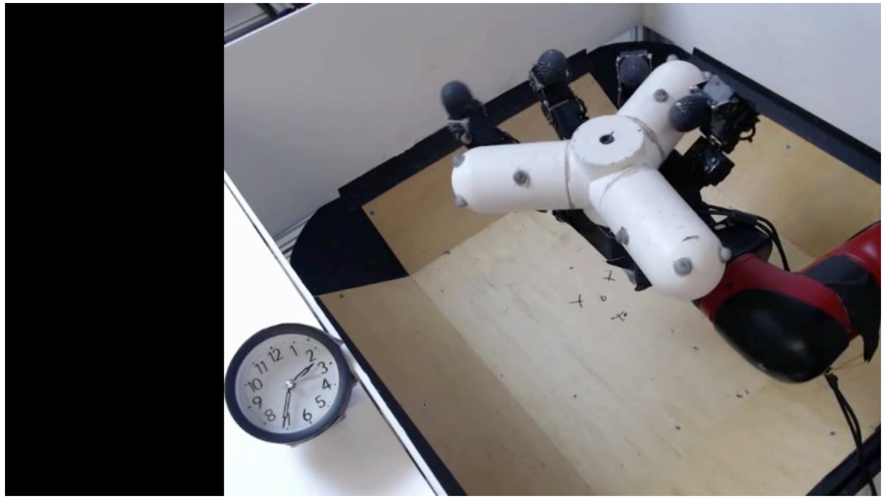
$$\pi_{\text{soft}}(a|s_t) = \exp\left(\frac{1}{\alpha}\left(Q_{\text{soft}}(s_t, a) - V_{\text{soft}}(s_t)\right)\right)$$

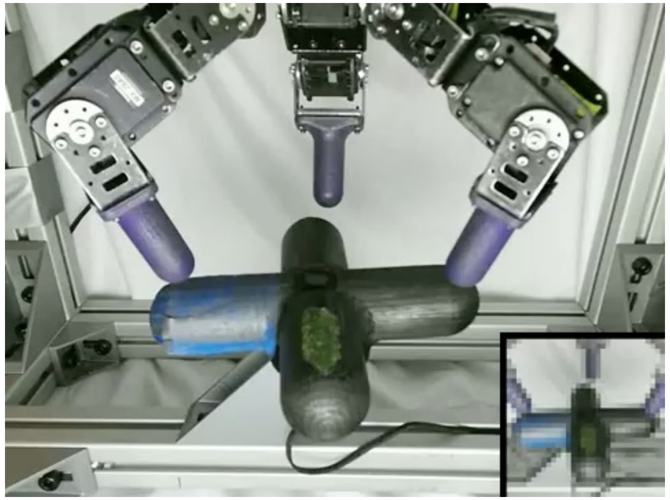
Go from max to “softmax” (imagine if α goes to 0, it becomes a max)

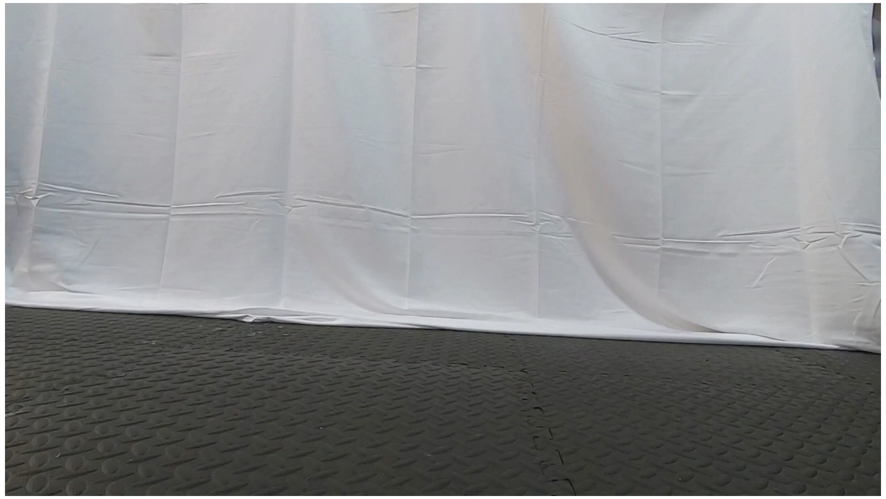
Prevents premature collapse of exploration while smoothing out optimization landscape!

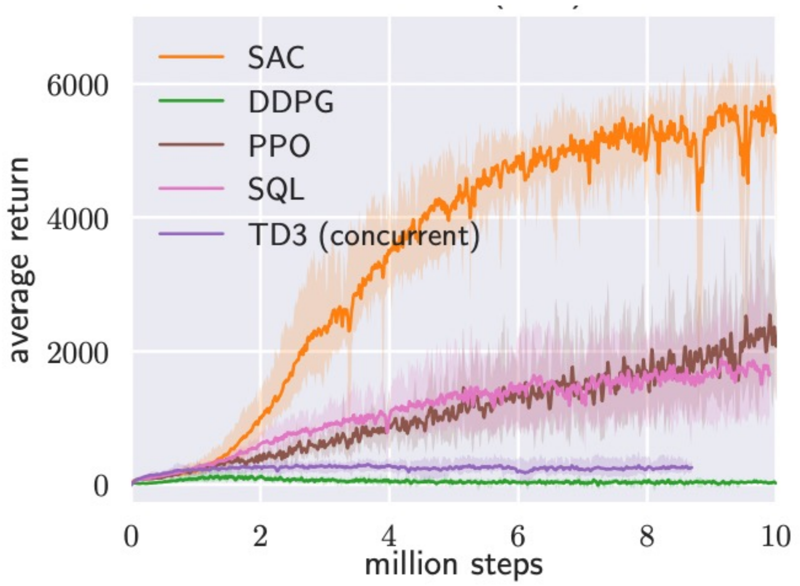
Maximum Entropy Actor-Critic Algorithms in Action











(f) Humanoid (rllab)

Lecture outline

Going from On-Policy to Off-Policy AC

↓

Getting Off-Policy RL to Work

↓

Frontiers of Off-Policy RL

Ok, so are off-policy algorithms perfect?

What makes off-policy RL hard?

Deadly triad:

1. Function Approximation
2. Bootstrapping
3. Off-policy learning

$$\min_{\phi} \mathbb{E}_{(s,a,s')\sim\mathcal{D}} \left[\left[Q^{\pi}_{\phi}(s_t, a_t) - \left(r(s_t, a_t) + \max_{a_{t+1}} \left[Q_{\bar{\phi}}(s_{t+1}, a_{t+1}) \right] \right) \right]^2 \right]$$

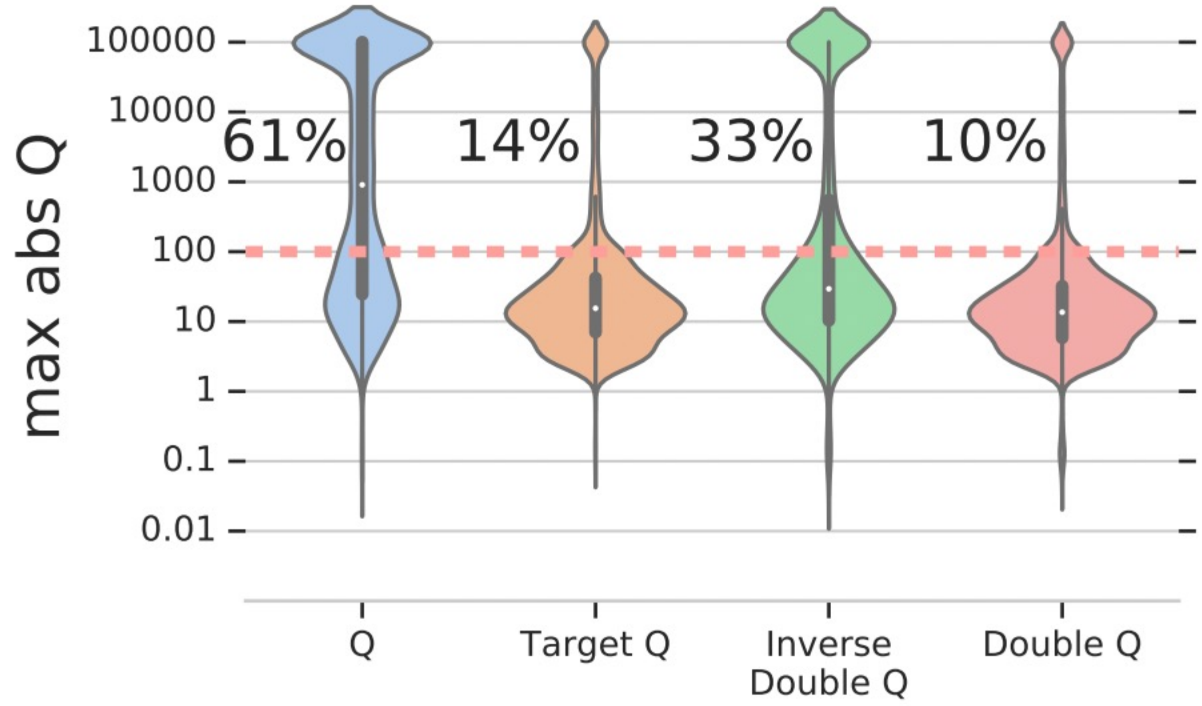
These in combination lead to many of the difficulties in stabilizing off-policy RL with function approximation

Zooming out – what makes off-policy RL hard?

Deadly triad:

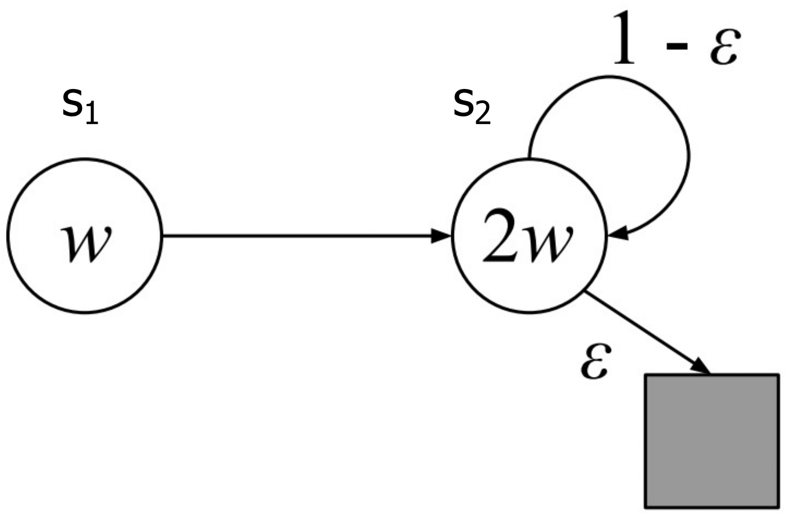
1. Function Approximation
2. Bootstrapping
3. Off-policy learning

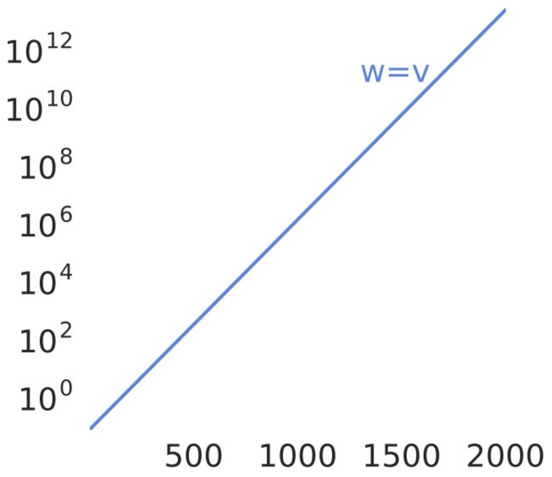
61% of runs show divergence of Q-values



Diverges even with linear function approximation, when off-policy + bootstrapping

Zooming out – what makes off-policy RL hard?





(b) $v(s) = w\phi(s)$ diverges.

Let's go to the whiteboard!

What should I work on?

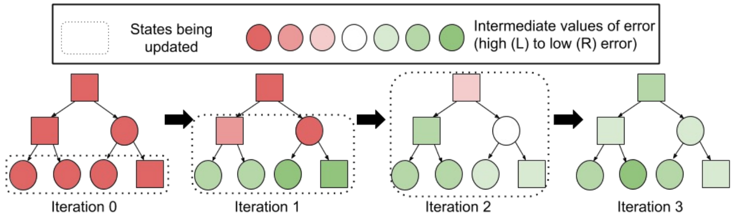
Where does the frontier of off-policy RL lie?

Off-policy is an extremely promising tool, but not quite plug and play like PG methods

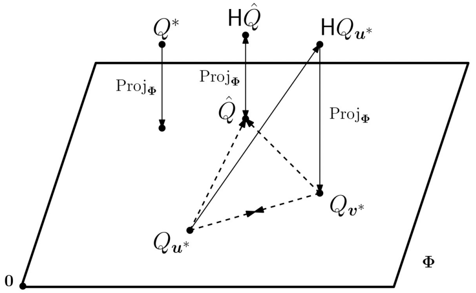
- Low variance, off-policy, avoids reconstruction, performs dynamic programming
- Has the potential to be **performant** and **sample efficient**

But in practice is often unstable, inefficient with high dimensional observations

Sampling



Theory



Exploration

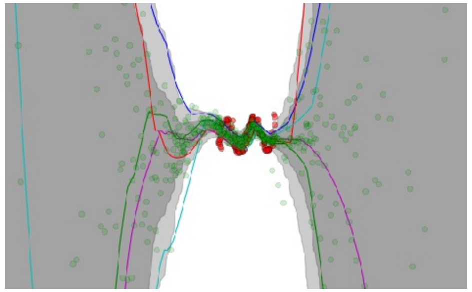
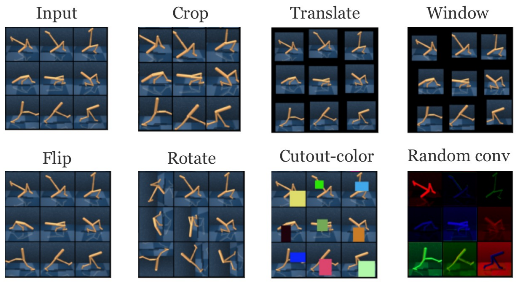
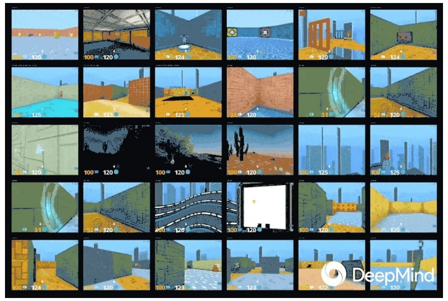


Image-based RL

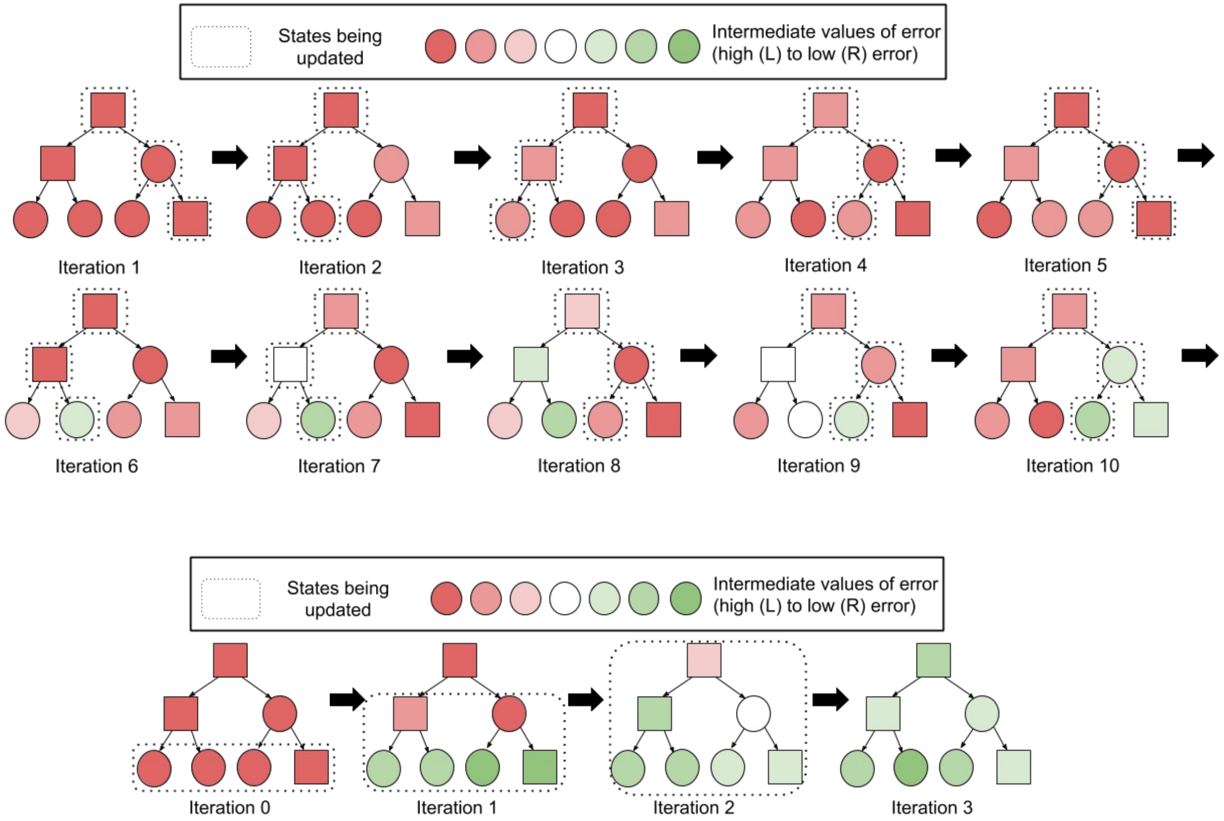


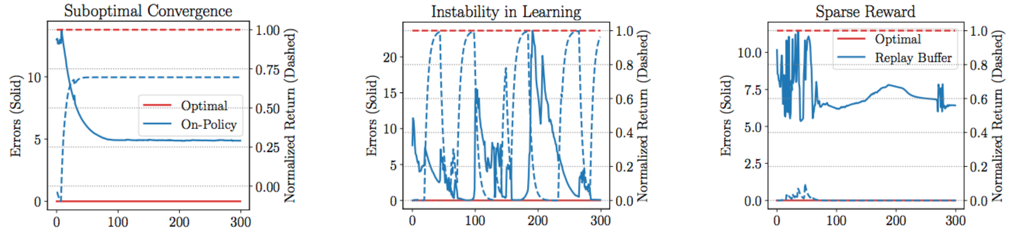
Partial Observability



Prioritizing Experience

Performing uniform buffer TD updates can be catastrophically bad





(a) Sub-optimal convergence for on-policy distributions: return (dashed) and value error (solid). Note that value error decreases rapidly at the start and finally converges to a nonzero value, leading to sub-optimal return.

(b) Instability for replay buffer distributions: return (dashed) and value error (solid) over training iterations. Note the rapid increase in value error at multiple points, which co-occurs with instabilities in returns.

(c) Error (left) and returns (right) for sparse reward MDP with replay buffer distributions. Note the inability to learn, low return, and highly unstable value error $\mathcal{E}_k$, often increasing sharply, destabilizing the learning process.

Need to prioritize updates to propagate good values

Theory/Convergence with Function Approximation

Significant body of work on learning dynamics with function approximation

Delusional Bias

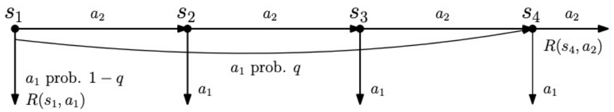


Figure 1: A simple MDP that illustrates delusional bias (see text for details).

Implicit regularization

$$\overline{\mathcal{R}}_{\exp}(\theta) = \sum_{i \in \mathcal{D}} \phi(\mathbf{s}_i, \mathbf{a}_i)^\top \phi(\mathbf{s}'_i, \mathbf{a}'_i).$$

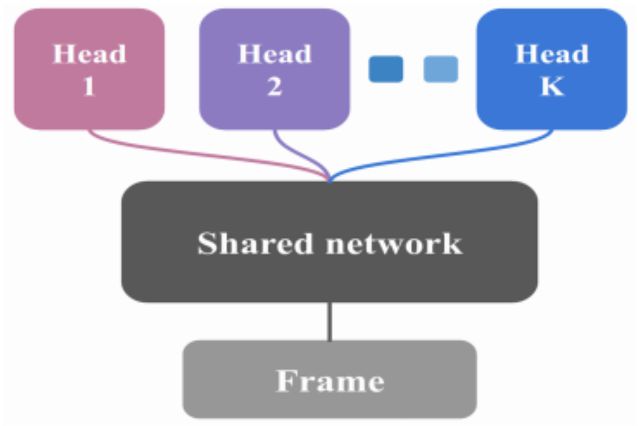
Bilinear classes

Framework	B-Rank	B-Complete	W-Rank	Bilinear Class (this work)
B-Rank	✓	✗	✓	✓
B-Complete	✗	✓	✗	✓
W-Rank	✗	✗	✓	✓
Bilinear Class (this work)	✗	✗	✗	✓

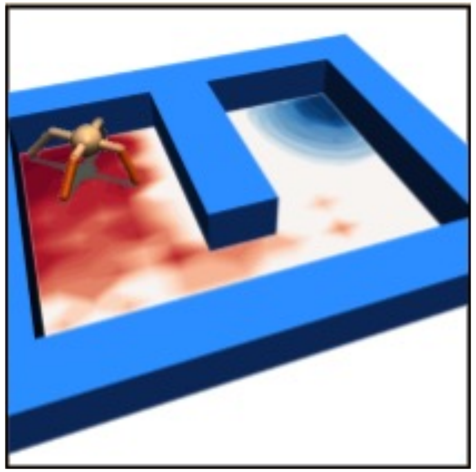
Exploration in Off-Policy RL

Better exploration methods

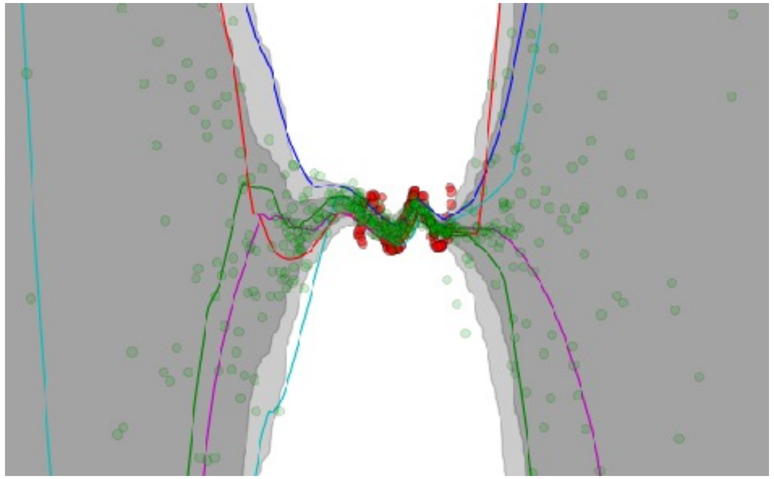
Uncertainty based methods



Count-based methods



Information gain methods

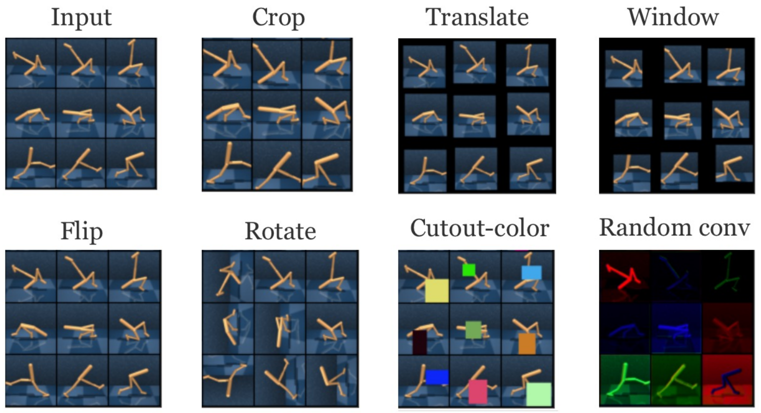


Often critical for getting algorithms to work!

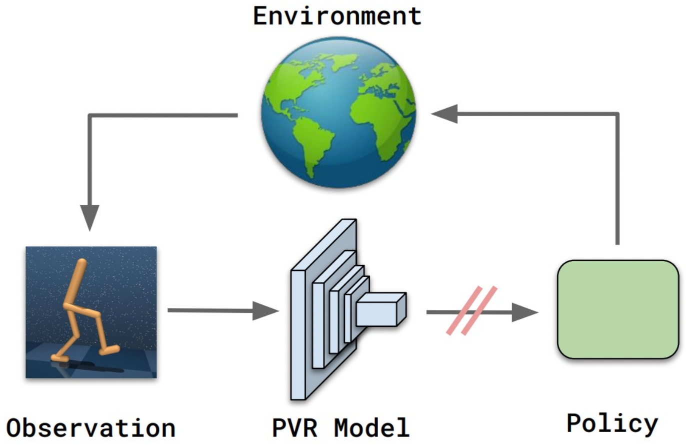
Image-based Off-Policy RL

Learning from high dimensional observations is unstable – images/point clouds

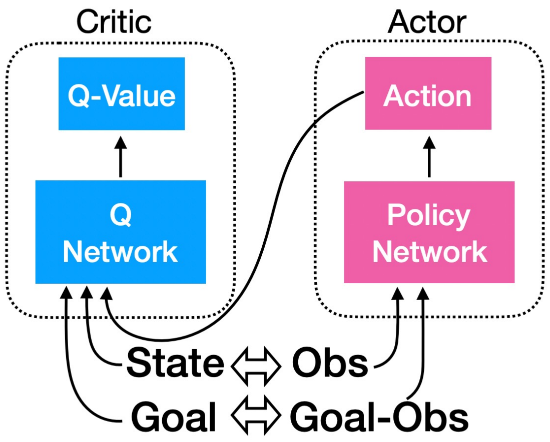
Data augmentations



Pre-trained representations



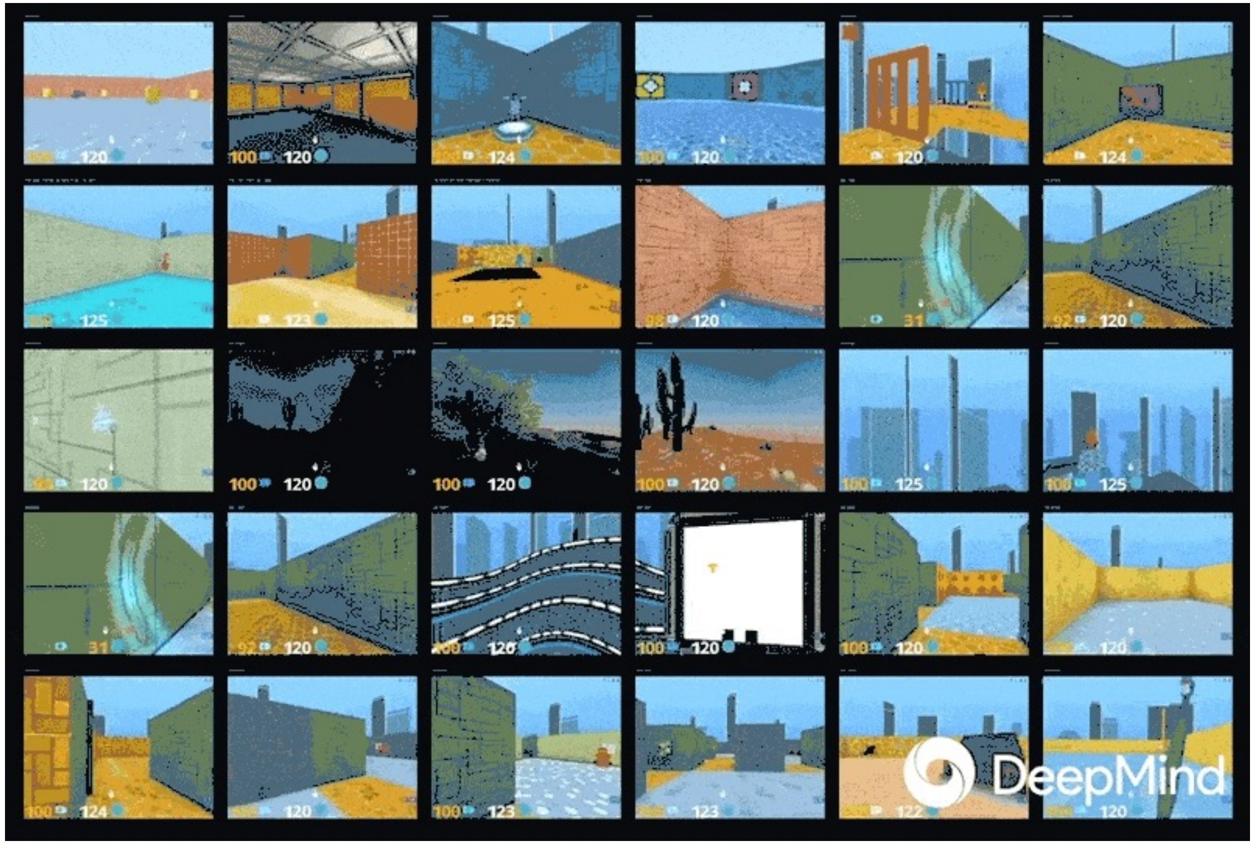
Student-teacher

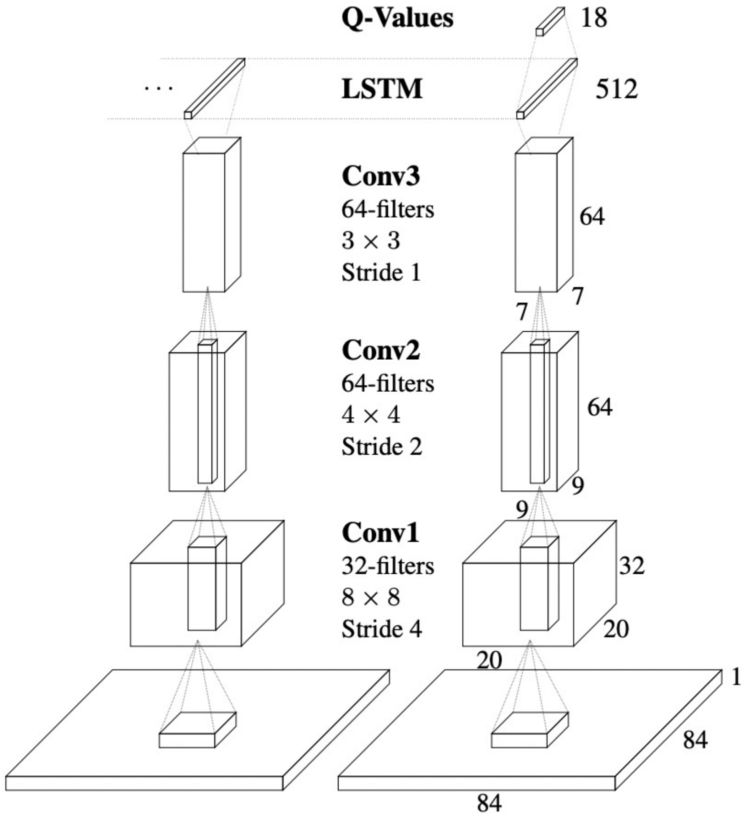


Still very unstable, lot of open research problems!

Partial Observability in Off-Policy RL

Off-policy methods critically depend on the Markov assumption



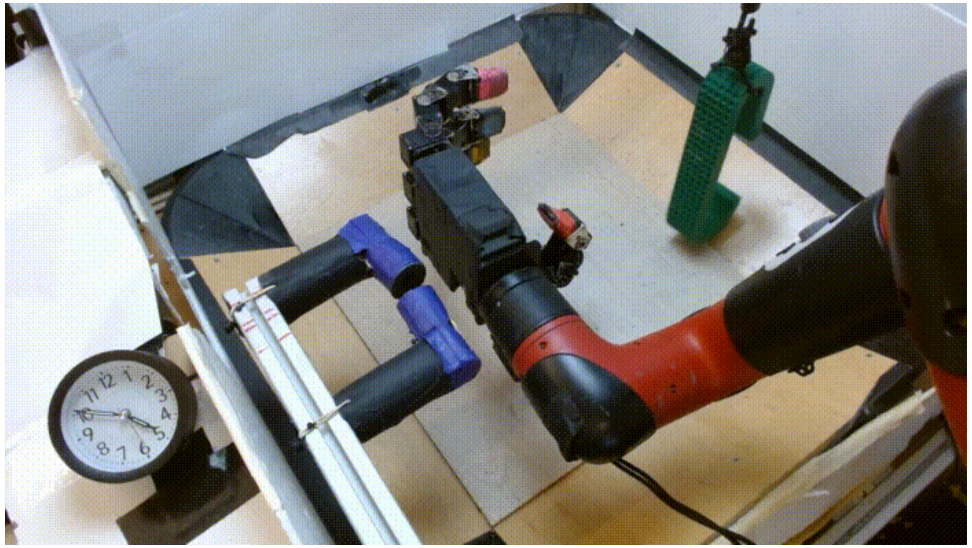


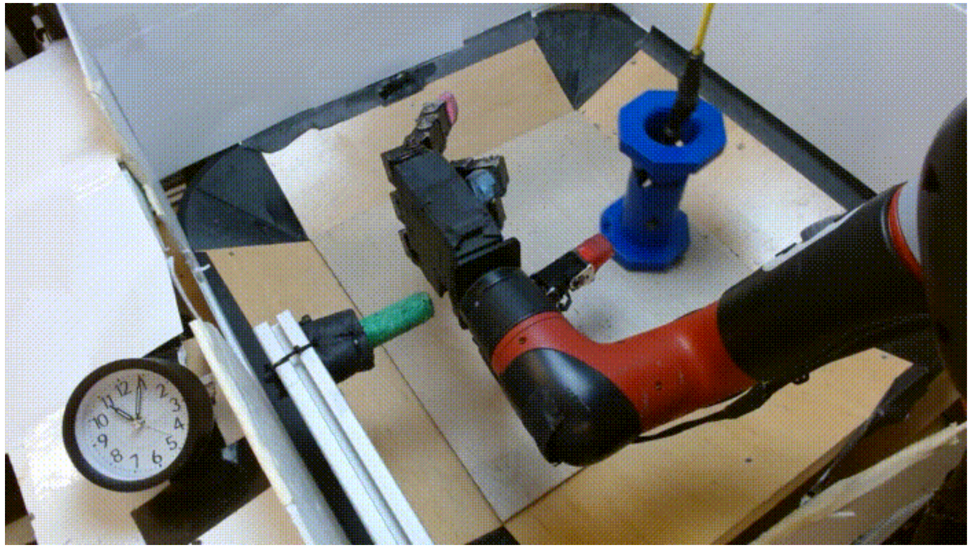
Learning history conditioned/recurrent Q-functions is an open area!

How has off-policy RL manifested in robotics?

Small changes – larger number of ensembles, more minibatch steps allow for training in < 20 mins

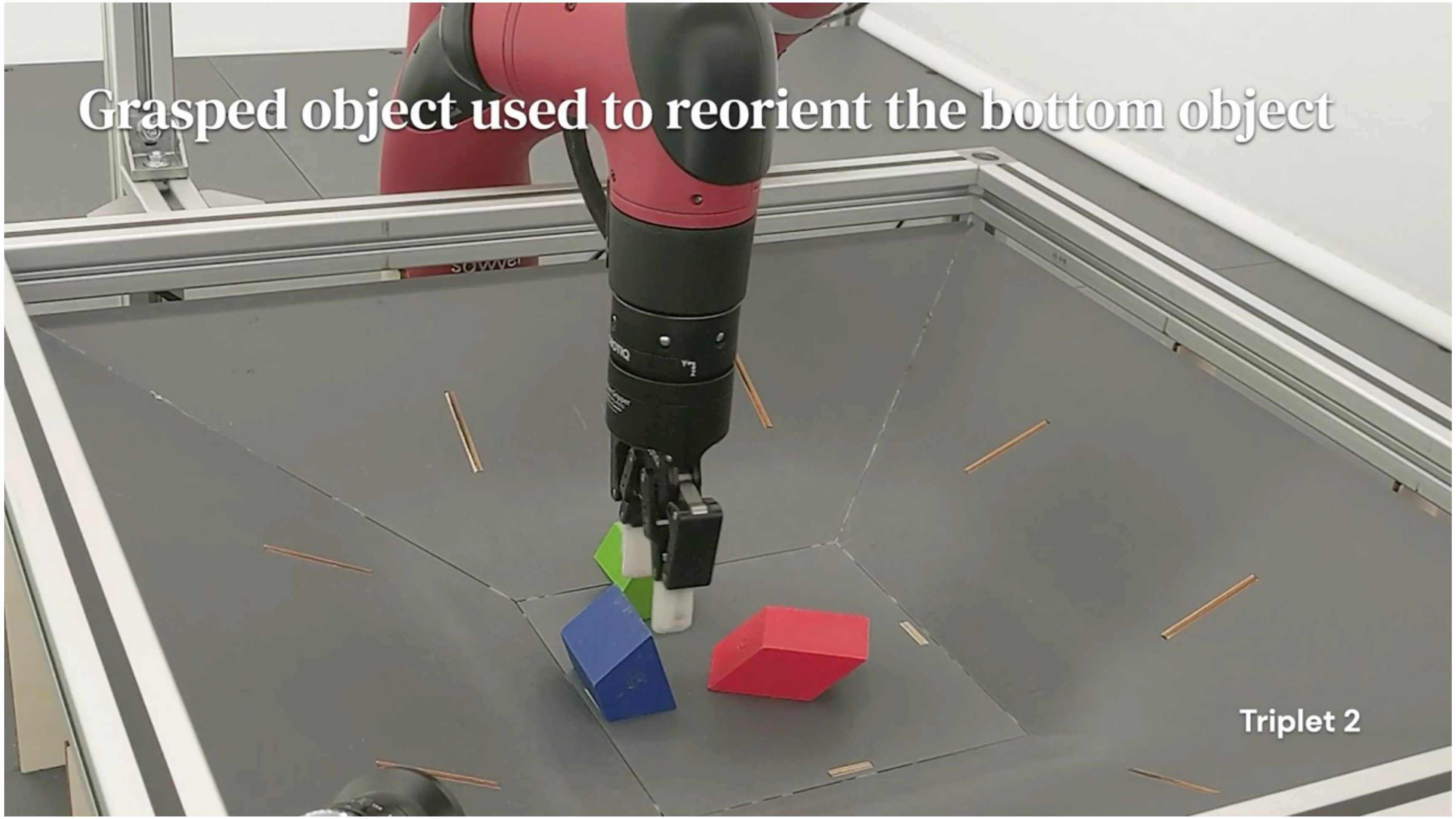
How has off-policy RL manifested in robotics?





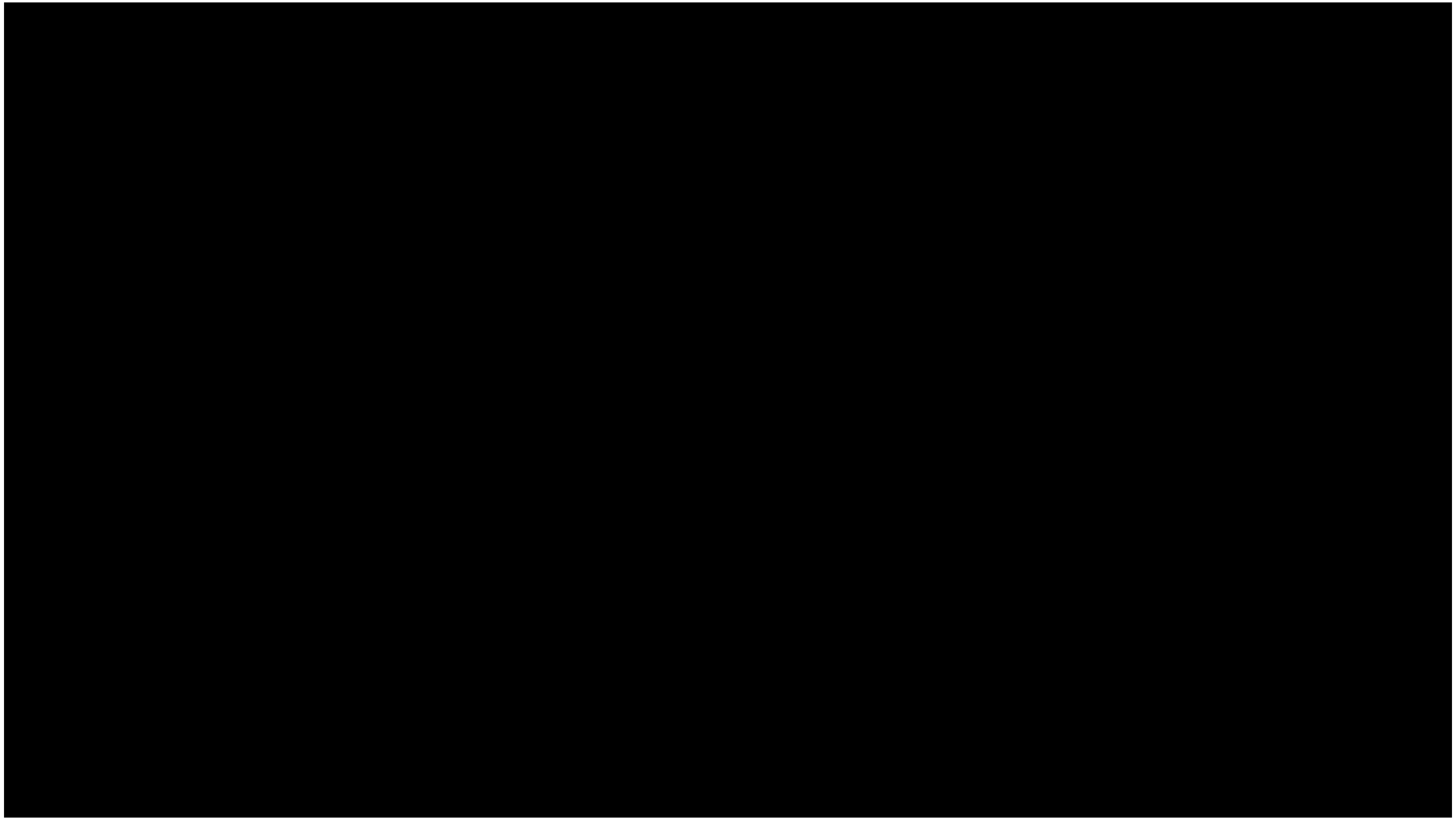
How has off-policy RL manifested in robotics?

Uses MPO – a variant of actor critic with a supervised learning style actor update

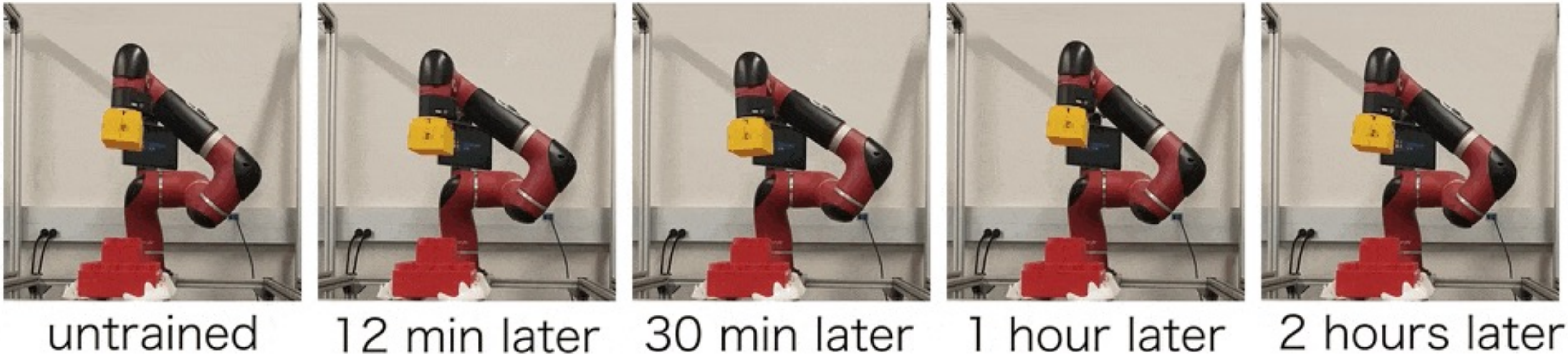


How has off-policy RL manifested in robotics?

Bootstrapped with a few demonstrations



How has off-policy RL manifested in robotics?



Pros/Cons of Off-Policy Methods in Robotics

Pros:

1. Sample-efficient enough for real world
2. Can learn from images with suitable design choices
3. Off-policy, can incorporate prior data

Cons

1. Often unstable
2. Can achieve lower asymptotic performance
3. Requires significant storage

Lecture outline

Going from On-Policy to Off-Policy AC

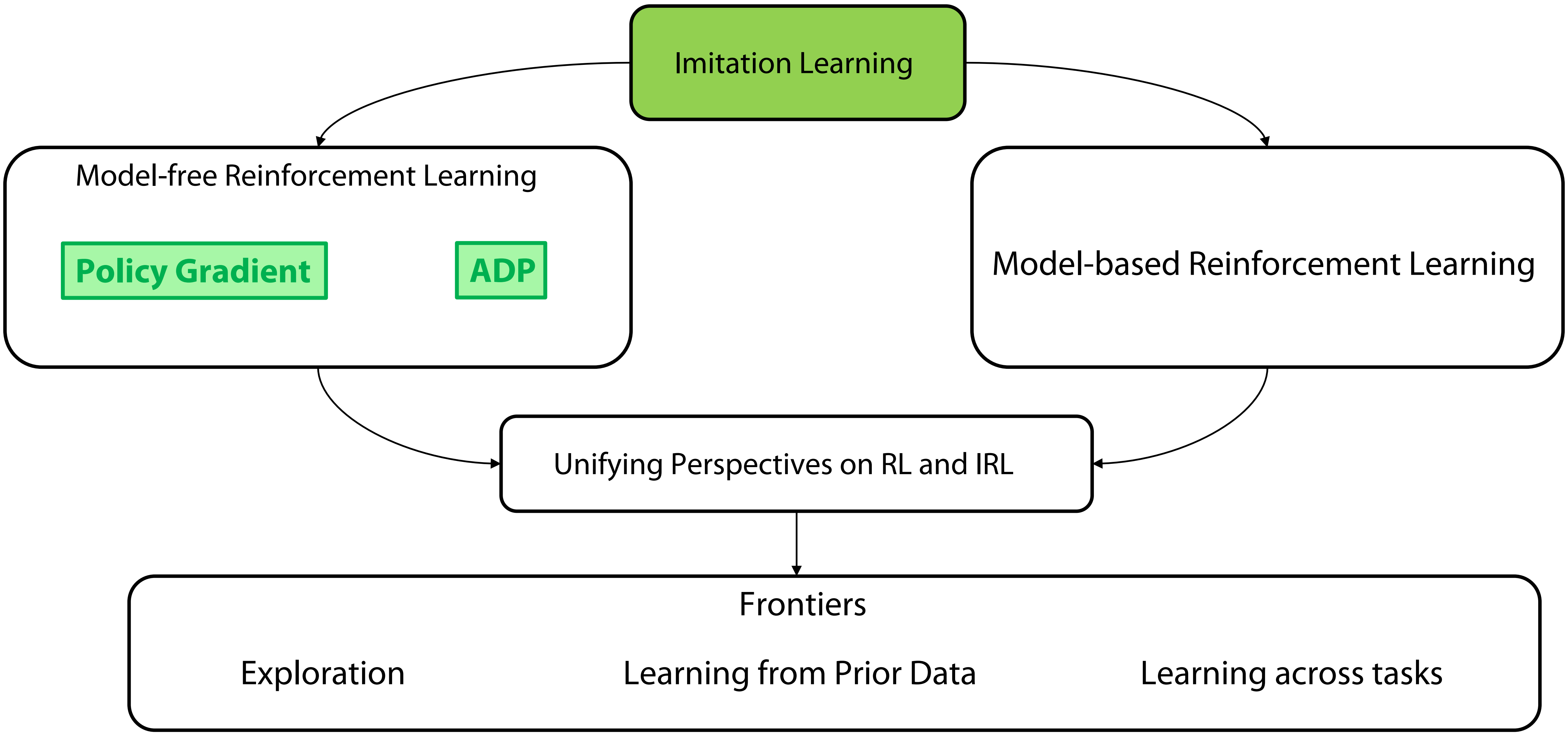
↓

Getting Off-Policy RL to Work

↓

Frontiers of Off-Policy RL

Class Structure



Fin.