

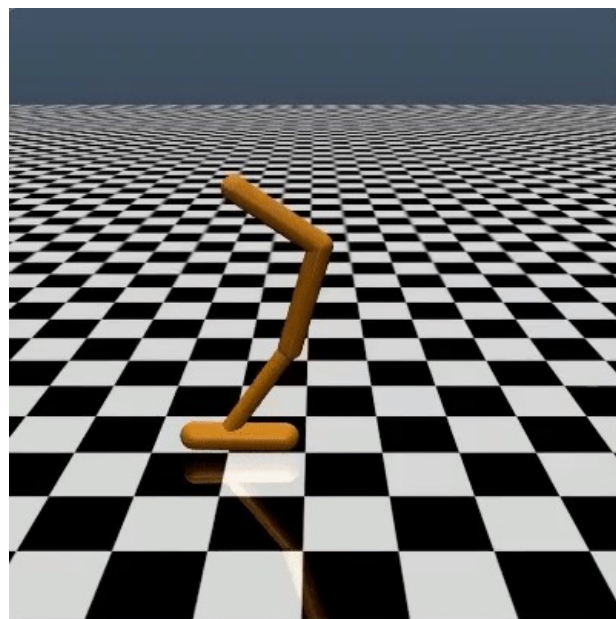


Reinforcement Learning

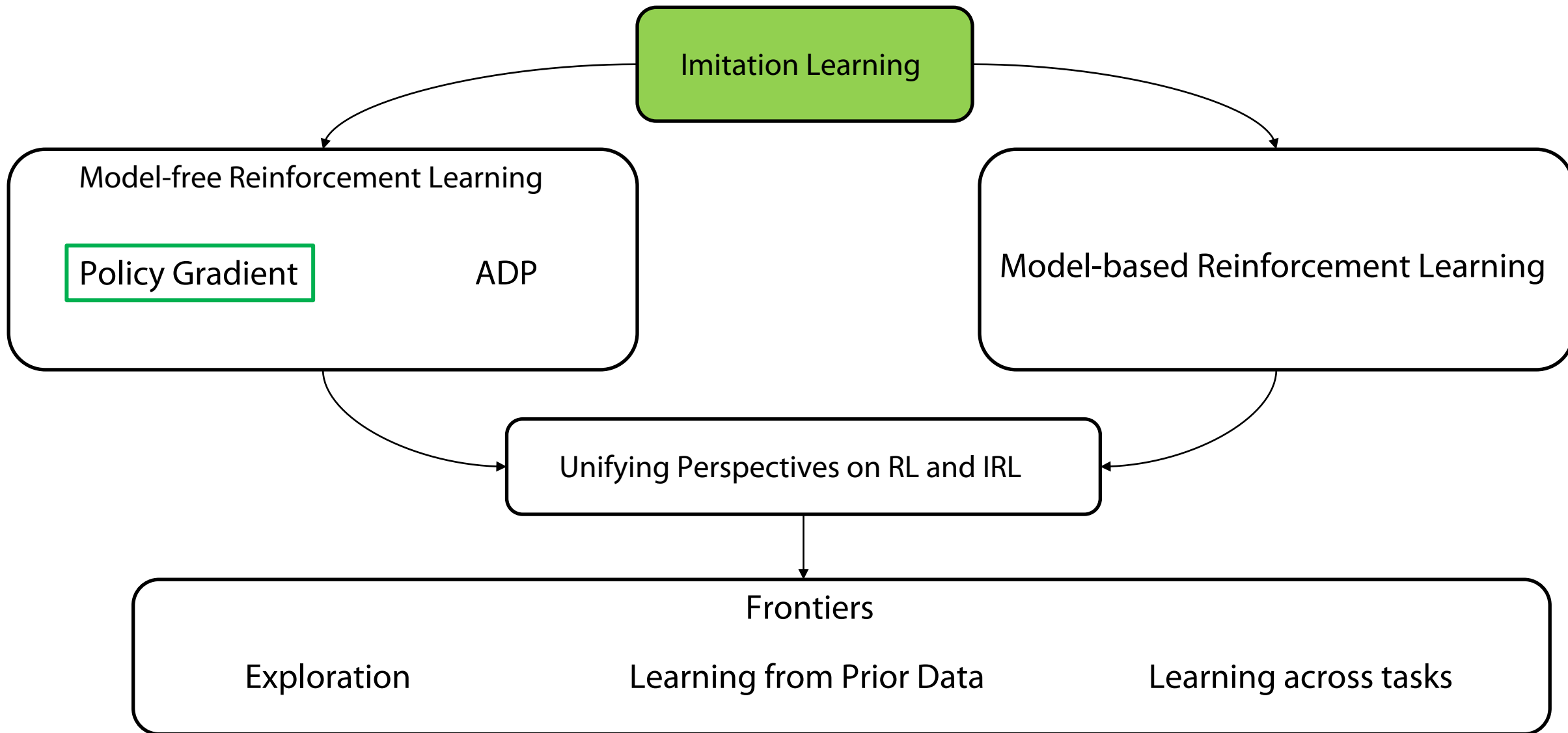
Spring 2026

Abhishek Gupta

TA: Mateo Guaman Castro



Class Structure



Lecture outline

Recap: Policy Gradients ++



Policy Gradients Finish Up + Clarifications

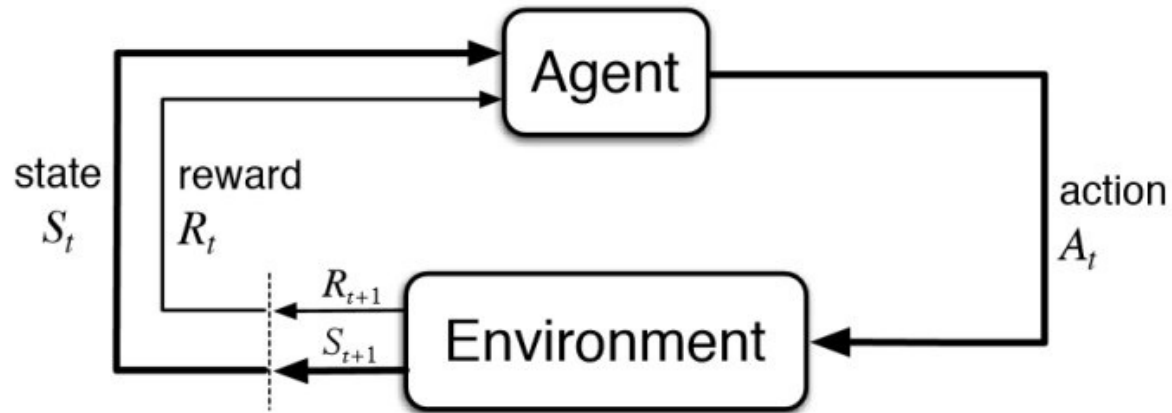


Kronecker Factorization



Off-Policy RL

Optimizing the RL Objective



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

Gradient Ascent

Dynamic Programming

Model-Based Optimization

Each method has its own +/-

Taking the gradient of return

$$\begin{aligned} \nabla_{\theta} J(\theta) &= \mathbb{E}_{\tau \sim p_{\theta}(\tau)} \left[\nabla_{\theta} \log p_{\theta}(\tau) \sum_{t=0}^T r(s_t, a_t) \right] \\ \nabla_{\theta} J(\theta) &= \mathbb{E}_{\substack{s_0 \sim p(s_0) \\ s_{t+1} \sim p(s_{t+1} | s_t, a_t) \\ a_t \sim \pi(a_t | s_t)}} \left[\sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \sum_{t'=0}^T r(s_{t'}, a_{t'}) \right] \\ &\approx \frac{1}{N} \sum_{i=0}^N \sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t^i | s_t^i) \sum_{t'=0}^T r(s_{t'}^i, a_{t'}^i) \quad (\text{approximating using samples}) \end{aligned}$$

(Monte-Carlo approximation)

Resulting “Natural” Policy Gradient

$$\max J(\theta_i) + \nabla_{\theta} J(\theta)|_{\theta=\theta_i} (\theta - \theta_i)$$

$$D_{\text{KL}}(\pi_{\theta} || \pi_{\theta_i}) \leq \epsilon$$

2nd order approximation of KL \rightarrow Fisher Information Metric

$$F = \mathbb{E}_{\pi_{\theta}} [(\nabla_{\theta} \log \pi_{\theta})(\nabla_{\theta} \log \pi_{\theta})^T]$$

$$\max J(\theta_i) + \nabla_{\theta} J(\theta)|_{\theta=\theta_i} (\theta - \theta_i)$$

$$(\theta - \theta_i)^T F (\theta - \theta_i) \leq \epsilon$$

Resulting update $\theta_{i+1} = \theta_i + \alpha F^{-1} \nabla_{\theta} J(\theta)|_{\theta=\theta_i}$ Covariant to parameterization

Trust Region Policy Optimization

3 key ideas:

1. Huge matrix inversion \rightarrow conjugate gradient method
2. Step size may be too large \rightarrow backtracking line search
3. On-policy updates \rightarrow importance sampled objective

Algorithm 3 Trust Region Policy Optimization

Input: initial policy parameters θ_0

for $k = 0, 1, 2, \dots$ **do**

Collect set of trajectories \mathcal{D}_k on policy $\pi_k = \pi(\theta_k)$

Estimate advantages $\hat{A}_t^{\pi_k}$ using any advantage estimation algorithm

Form sample estimates for

- policy gradient \hat{g}_k (using advantage estimates)
- and KL-divergence Hessian-vector product function $f(v) = \hat{H}_k v$

Use CG with n_{cg} iterations to obtain $x_k \approx \hat{H}_k^{-1} \hat{g}_k$

Estimate proposed step $\Delta_k \approx \sqrt{\frac{2\delta}{x_k^T \hat{H}_k x_k}} x_k$

Perform backtracking line search with exponential decay to obtain final update

$$\theta_{k+1} = \theta_k + \alpha^j \Delta_k$$

end for

Generalized Advantage Estimation

Sum up all the estimators in a geometric sum

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

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

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

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

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

Proximal Policy Optimization Algorithm

$$\mathcal{L}(s, a, \theta_i, \theta) = \min \left(\frac{\pi_{\theta}(a|s)}{\pi_{\theta_i}(a|s)} A(s, a), \text{clip} \left(\frac{\pi_{\theta}(a|s)}{\pi_{\theta_i}(a|s)}, 1 - \epsilon, 1 + \epsilon \right) A(s, a) \right)$$

- ✓ Multiple minibatch gradient steps
- ✓ No second order optimization
- ✓ Simple and stable, without huge updates

Lecture outline

Recap: Policy Gradients ++



Policy Gradients Finish Up + Clarifications



Kronecker Factorization



Off-Policy RL

So should we just use PPO for everything?

IMPLEMENTATION MATTERS IN DEEP POLICY GRADIENTS: A CASE STUDY ON PPO AND TRPO

Logan Engstrom^{1*}, Andrew Ilyas^{1*}, Shibani Santurkar¹, Dimitris Tsipras¹,
Firdaus Janoos², Larry Rudolph^{1,2}, and Aleksander Mądry¹

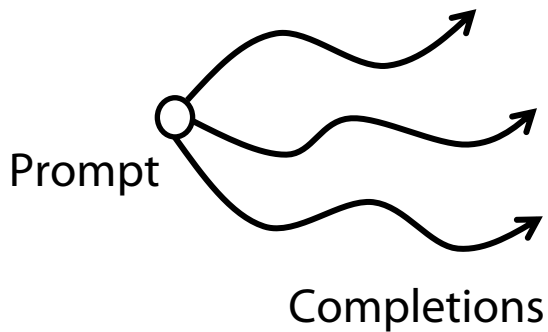
Open question!

STEP	MUJoCo TASK		
	WALKER2D-V2	HOPPER-V2	HUMANOID-V2
PPO	3292 [3157, 3426]	2513 [2391, 2632]	806 [785, 827]
PPO-M	2735 [2602, 2866]	2142 [2008, 2279]	674 [656, 695]
TRPO	2791 [2709, 2873]	2043 [1948, 2136]	586 [576, 596]
TRPO+	3050 [2976, 3126]	2466 [2381, 2549]	1030 [979, 1083]
AAI	242	99	224
ACLI	557	421	444

Code optimizations may make a bigger impact than algorithmic ones
→ needs more investigation!

Ok but what is GRPO?

Critics are annoying to learn, GAE can be unstable – what happens if we remove the critic?



$$A(s, a) = r + \gamma V(s') - V(s)$$

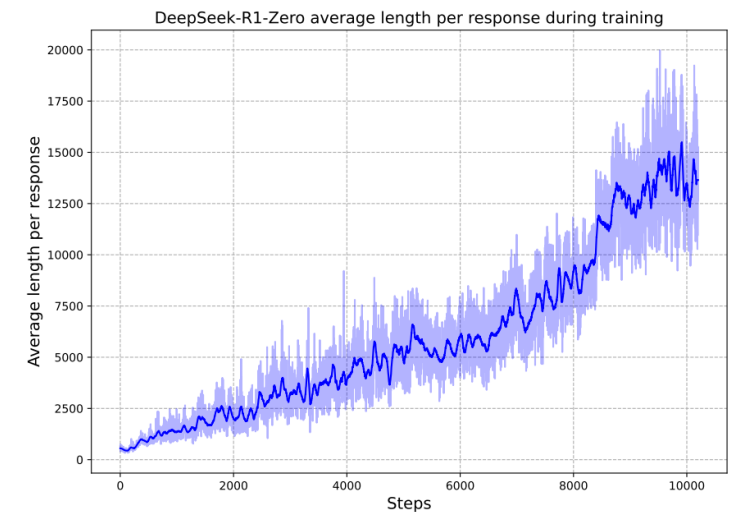
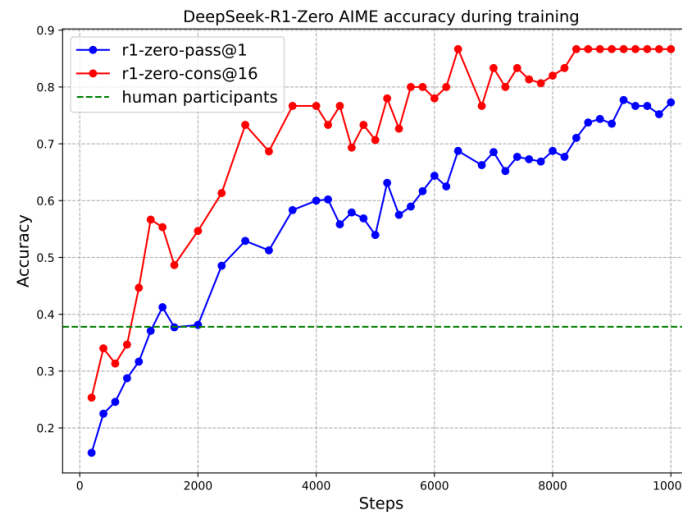


$$A(s, a) = \frac{r(s, a) - \text{mean}(r_1, r_2, \dots, r_G)}{\text{std}(r_1, r_2, \dots, r_G)}$$

Compare only with other completions for same prompt

Emphasizes the difference between choices at a state

Ignores magnitude differences across states



Some additional tricks that PG uses

Common to regularize training with entropy or KL regularization

Entropy maximization

$$H(\pi_{\theta}(\cdot|s_t)) = - \sum_{a \in \mathcal{A}} \pi_{\theta}(a|s_t) \log \pi_{\theta}(a|s_t)$$

KL regularization to a base policy

$$D_{KL}(\pi_{\theta_{old}}(\cdot|s_t) \parallel \pi_{\theta}(\cdot|s_t)) = \sum_{a \in \mathcal{A}} \pi_{\theta_{old}}(a|s_t) \log \frac{\pi_{\theta_{old}}(a|s_t)}{\pi_{\theta}(a|s_t)}$$

Encourage exploration

Prevent collapse/straying from base

PC-PG: Policy Cover Directed Exploration for Provable Policy Gradient Learning

Prevents flatness of optimization landscape

Alekh Agarwal¹, Mikael Henaff², Sham Kakade^{3,1}, and Wen Sun^{*4}

Some Clarifications

What does the Fisher Information Matrix mean?

Sensitivity matrix – identifies which parameter directions matter for changing the prediction

$$\delta\theta^T F \delta\theta = \mathbb{E} \left[(\delta\theta^T \nabla_{\theta} \log p_{\theta}(x))^2 \right]$$

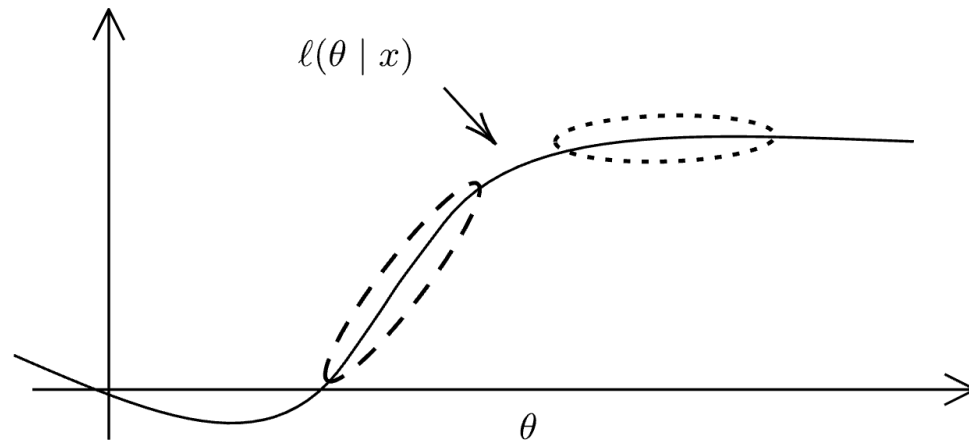


Figure 2: An example log-likelihood function $\ell(\theta | x)$. The section of the log-likelihood surrounded by the dashed line changes rapidly as a function of θ . This would likely correspond to a region of high Fisher information. The region surrounded by the dotted line changes slowly as a function of θ . This would likely correspond to a region of low Fisher information.

Top eigenvalues give most sensitive direction

Tells us how sensitive the model is to moving in a direction

Covariance matrix of grad log-probabilities

Is the FIM always invertible?

The Fisher Information Matrix is Positive **Semi**definite, not Positive definite \rightarrow can have 0 eigenvalues

$$v^\top F v = \mathbb{E}[(v^\top \nabla_\theta \log p_\theta(z))^2] \geq 0.$$

Use the pseudoinverse

$$F = Q \Lambda Q^\top,$$

$$\Lambda = \text{diag}(\lambda_1, \dots, \lambda_n).$$

$$\Lambda^\dagger = \text{diag}\left(\frac{1}{\lambda_i} \text{ for } \lambda_i > 0, 0 \text{ for } \lambda_i = 0\right).$$

$$F^\dagger = Q \Lambda^\dagger Q^\top,$$

Use a damped objective

$$\delta\theta = -(F + \lambda I)^{-1} g.$$

Why is the 1-step GAE estimator biased?

1-step estimator $\hat{A}_t^{(1)} = r_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t)$

True advantage $A^\pi(s_t, a_t) = r_t + \gamma \mathbb{E} [V^\pi(s_{t+1}) | s_t, a_t] - V^\pi(s_t)$

$$\mathbb{E} [\hat{A}_t^{(1)} | s_t, a_t] = r_t + \gamma \mathbb{E} [V_\phi(s_{t+1}) | s_t, a_t] - V_\phi(s_t)$$

$$\text{Bias}(s_t, a_t) = \mathbb{E} [\hat{A}_t^{(1)} | s_t, a_t] - A^\pi(s_t, a_t)$$

Bias Computation

$$\text{Bias}(s_t, a_t) = \gamma \mathbb{E} [V_\phi(s_{t+1}) - V^\pi(s_{t+1}) | s_t, a_t] - (V_\phi(s_t) - V^\pi(s_t))$$

$$e(s) = V_\phi(s) - V^\pi(s)$$

$$\text{Bias}(s_t, a_t) = \gamma \mathbb{E} [e(s_{t+1}) | s_t, a_t] - e(s_t)$$

Non-zero unless $e(s) = 0$

Lecture outline

Recap: Policy Gradients ++



Policy Gradients Finish Up + Clarifications



Kronecker Factorization



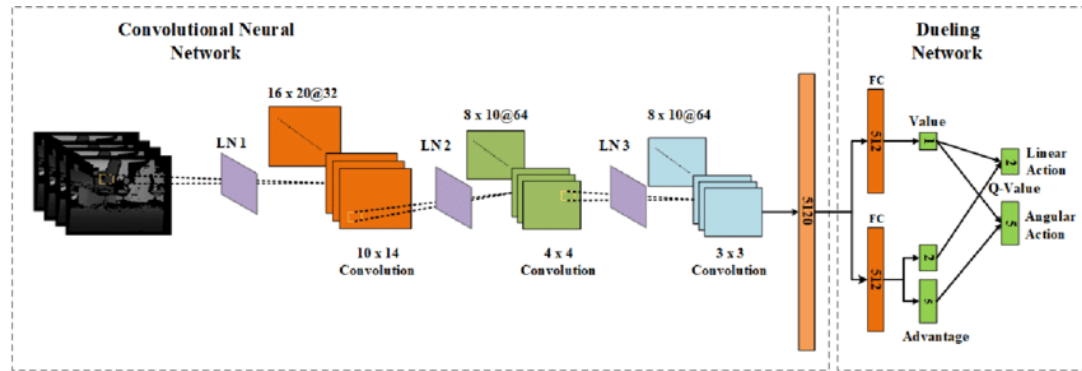
Off-Policy RL

A Fun Aside

Approximations to effectively invert FIM

Instead of solving with conjugate gradient, what if we approximated FIM

Major issue with FIM is huge dimensionality



Dimensionality of FIM – huge!
Very challenging to invert

Kronecker factorization (K-FAC) makes it tractable with 2 approx:

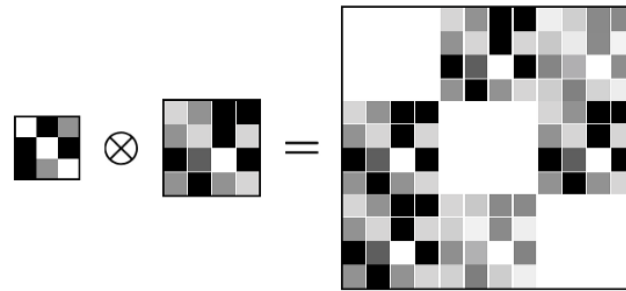
1. Block diagonalize FIM by layer
2. Represent each block diagonal as a Kronecker product (easy inversion)

What is a Kronecker product?

Generalization of the tensor product

$$\mathbf{A} \otimes \mathbf{B} := \begin{pmatrix} [\mathbf{A}]_{1,1} \mathbf{B} & \cdots & [\mathbf{A}]_{1,n} \mathbf{B} \\ \vdots & \ddots & \vdots \\ [\mathbf{A}]_{m,1} \mathbf{B} & \cdots & [\mathbf{A}]_{m,n} \mathbf{B} \end{pmatrix} \in \mathbb{R}^{ma \times nb}$$

$\mathbf{A} \in \mathbb{R}^{m \times n}, \mathbf{B} \in \mathbb{R}^{a \times b}$: Kronecker factors

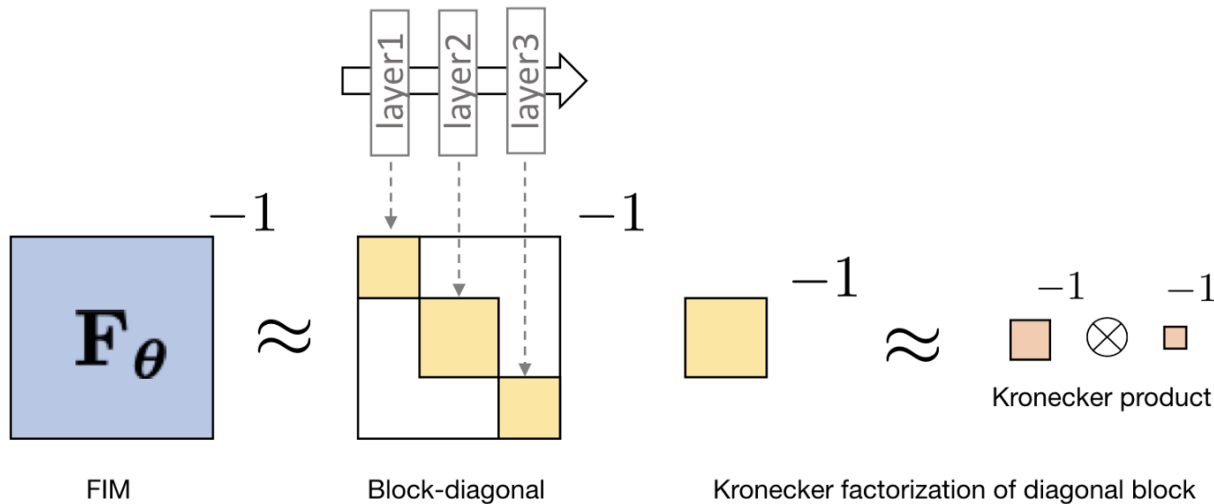


Identities

$$\text{vec}(uv^T) = u \otimes v$$

$$(a \otimes b)(c \otimes d) = (ac \otimes bd)$$

$$(\mathbf{A} \otimes \mathbf{B})^{-1} = \mathbf{A}^{-1} \otimes \mathbf{B}^{-1}$$



We will represent the (per layer) fisher information metric as a Kronecker product of two smaller matrices

K-FAC approximation for NPG

Kronecker factorization (K-FAC) makes it tractable with 2 approx:

1. Block diagonalize FIM by layer
2. Represent each block diagonal as a Kronecker product (easy inversion)

Weight defs

$$s = Wa$$
$$\nabla_W L = (\nabla_s L) a^\top$$

Matrix Identities

$$\text{vec}(uv^\top) = u \otimes v$$
$$(a \otimes b)(c \otimes d) = (ac \otimes bd)$$

K-FAC reparam

$$F_\ell = \mathbb{E}[\text{vec}\{\nabla_W L\}\text{vec}\{\nabla_W L\}^\top] = \mathbb{E}[aa^\top \otimes \nabla_s L(\nabla_s L)^\top]$$
$$\approx \mathbb{E}[aa^\top] \otimes \mathbb{E}[\nabla_s L(\nabla_s L)^\top] := A \otimes S := \hat{F}_\ell,$$

Easy to compute and invert (order of magnitude smaller matrices)

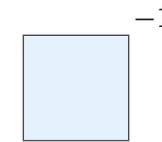
How much does this help?

All layers in AlexNet

60,000,000 parameters

Fisher information matrix

$$\mathbf{F}_\theta \in \mathbb{R}^{60,000,000 \times 60,000,000}$$



Final layer of AlexNet

Input dimension: 4,096

Output dimension: 1,000

4,096,000 parameters



Fisher block

$$\mathbf{F}_i \in \mathbb{R}^{4,096,000 \times 4,096,000}$$



Kronecker factors

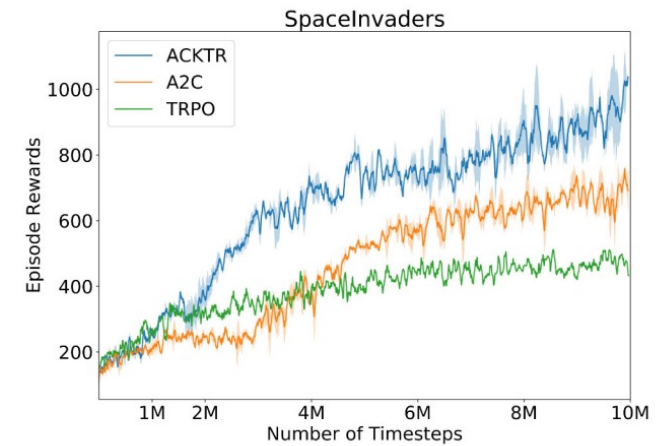
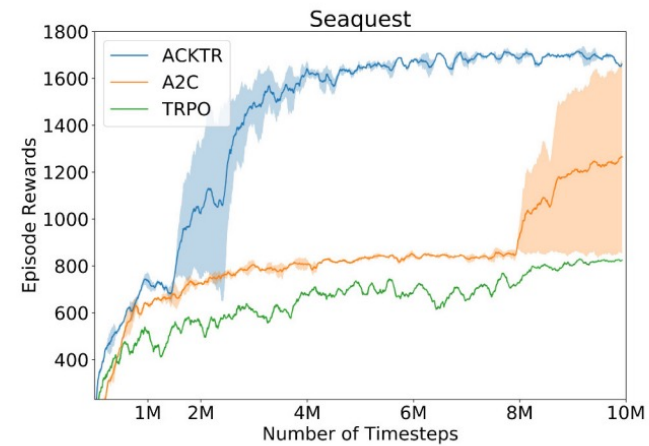
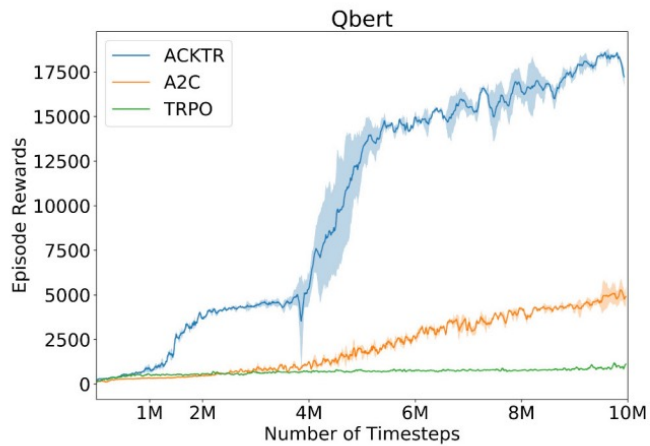
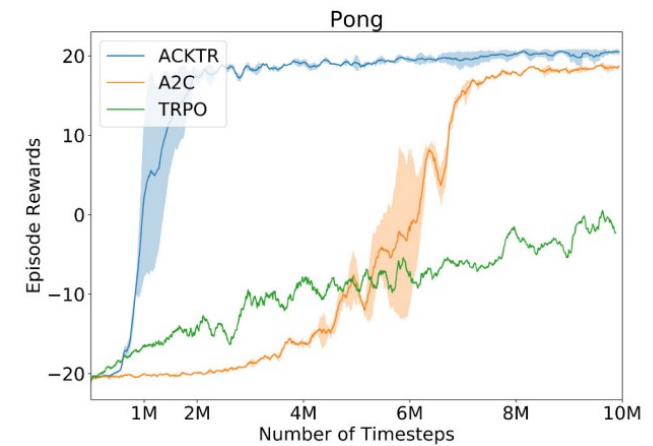
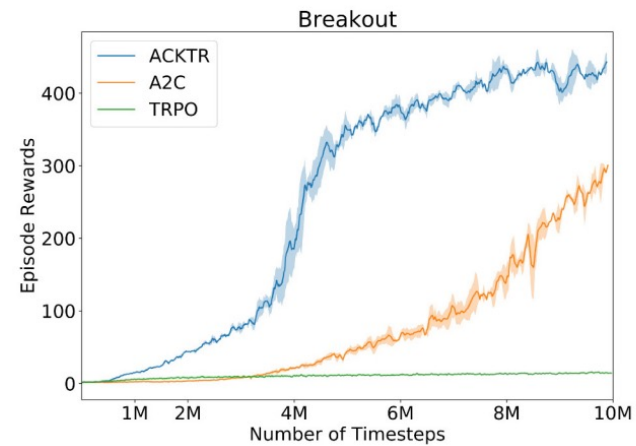
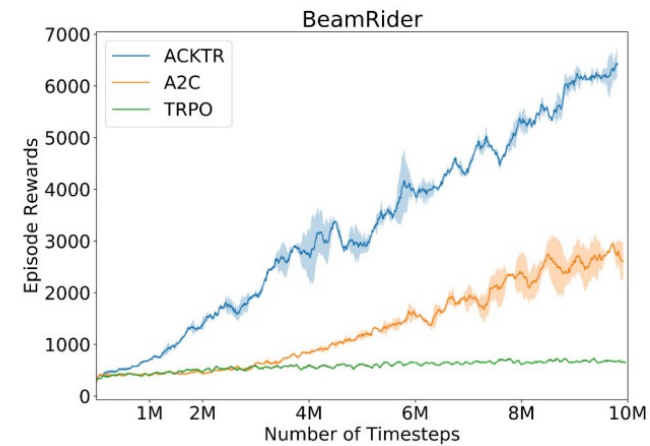
$$\mathbf{A}_{i-1} \in \mathbb{R}^{4,096 \times 4,096}$$

$$\mathbf{G}_i \in \mathbb{R}^{1,000 \times 1,000}$$



How much does this help?

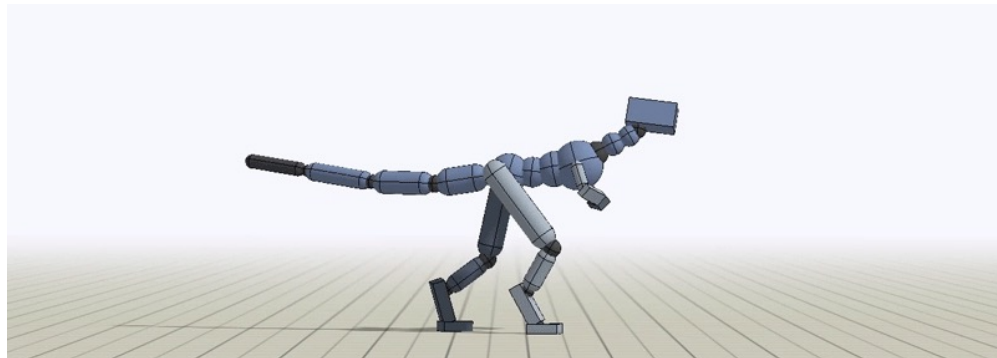
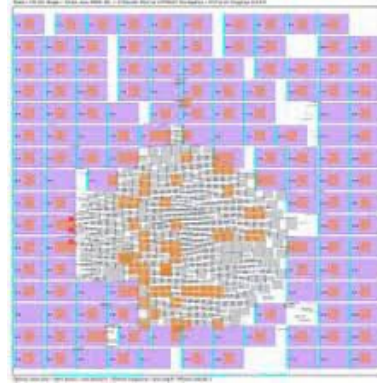
Significant sample complexity gains



Pros/Cons of Policy Gradient Methods

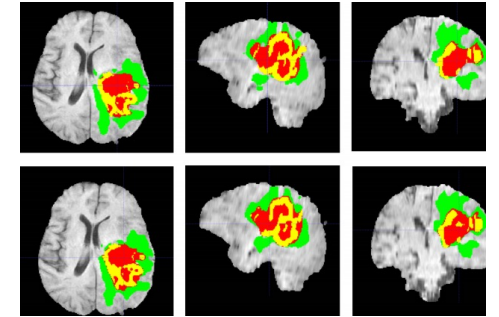
Pros

- Conceptually simple, easy to implement
- Stable, good asymptotic performance
- Compatible with deep models
- Require minimal modeling



Cons

- Sample inefficient
- Unable to reuse prior data effectively → on-policy
- Blackbox, can be hard to debug



Frontiers of Policy Gradient Research

Major open challenges in policy gradient research:

Convergence guarantees

Asynchronous/Parallel Methods

Better Variance Reduction

Learning from high-dimensional inputs

Bootstrapping from prior data

Multi-agent Policy Gradient

Finetuning generative models/policies with RL

Frontiers of Policy Gradient Research

Convergence guarantees and empirical investigations

Globally Convergent in LQR/LQG Case

- *Gradient descent case: For an appropriate (constant) setting of the stepsize η ,*

$$\eta = \text{poly} \left(\frac{\mu \sigma_{\min}(Q)}{C(K_0)}, \frac{1}{\|A\|}, \frac{1}{\|B\|}, \frac{1}{\|R\|}, \sigma_{\min}(R) \right)$$

and for

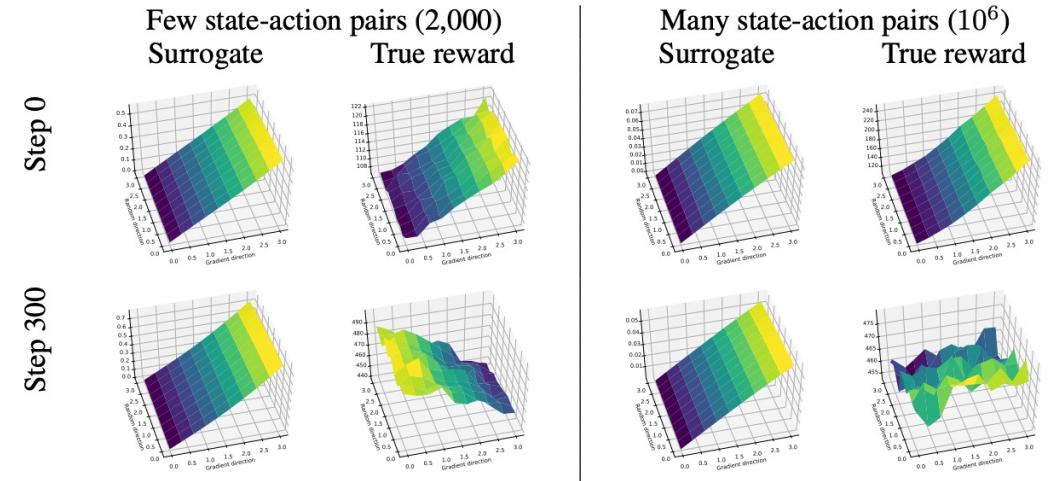
$$N \geq \frac{\|\Sigma_{K^*}\|}{\mu} \log \frac{C(K_0) - C(K^*)}{\varepsilon} \\ \times \text{poly} \left(\frac{C(K_0)}{\mu \sigma_{\min}(Q)}, \|A\|, \|B\|, \|R\|, \frac{1}{\sigma_{\min}(R)} \right),$$

then, with high probability, gradient descent (Equation 8) enjoys the following performance bound:

$$C(K_N) - C(K^*) \leq \varepsilon.$$

Global Convergence of Policy Gradient Methods for the Linear Quadratic Regulator, Fazel et al '19
Global Convergence of Policy Gradient Methods to (Almost) Locally Optimal Policies, Zhang et al, '19
Globally convergent policy search over dynamic filters for output estimation, Umenberger '21

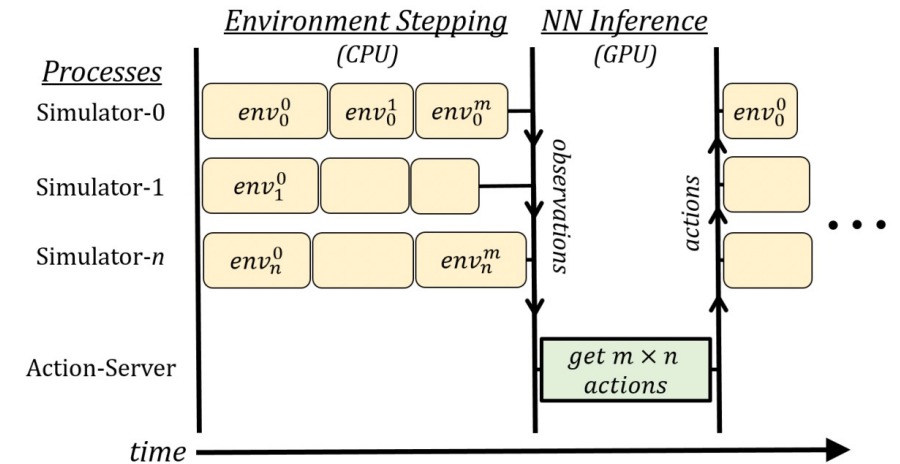
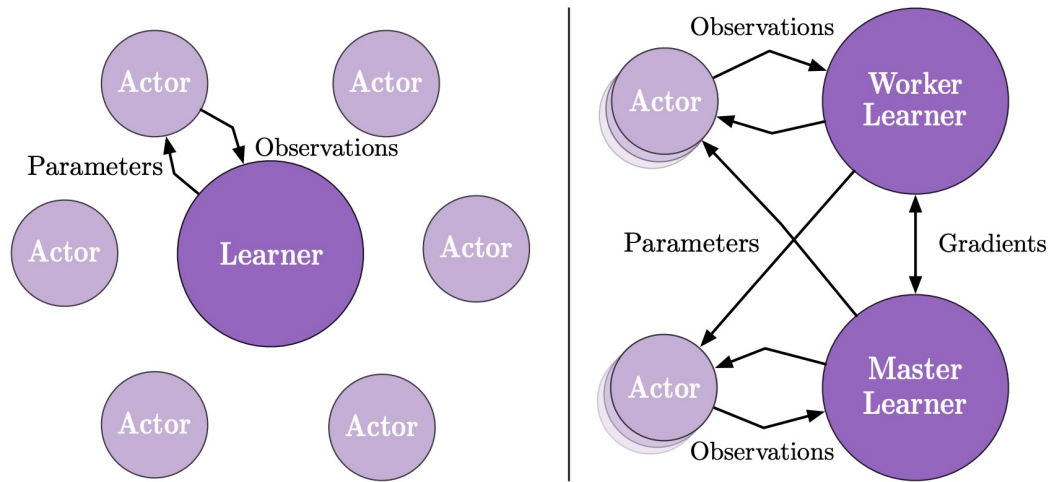
Practical Algorithms Deviate from Theory



Is the Policy Gradient a Gradient?, Nota et al, '19
A Closer Look at Deep Policy Gradients, Ilyas et al '19
An Empirical Analysis of Proximal Policy Optimization with Kronecker-factored Natural Gradients, Song et al '18
What Matters In On-Policy Reinforcement Learning? A Large-Scale Empirical Study, Andrychowicz et al '20

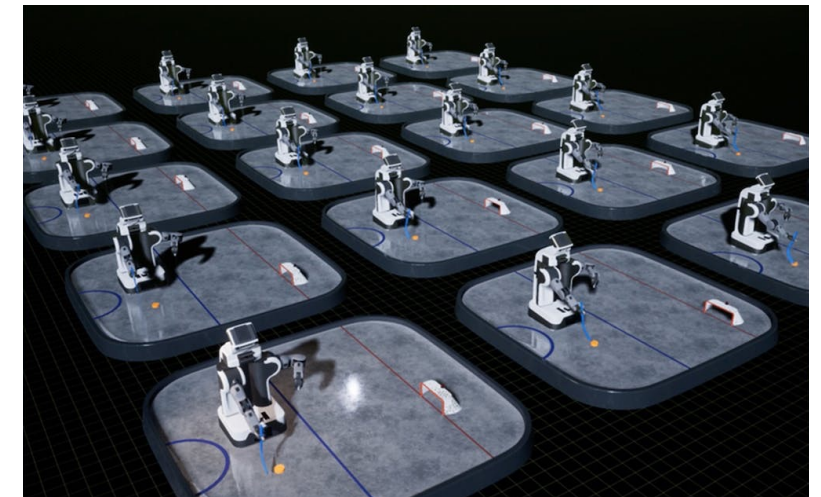
Frontiers of Policy Gradient Research

Asynchronous methods for large scale speedup



IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures, Espeholt '18

Accelerated Methods for Deep Reinforcement Learning, Stooke et al '19



Frontiers of Policy Gradient Research

Better Variance Reduction Methods

Action dependent baselines

$$\pi_{\theta}(a_t|s_t) = \prod_{i=1}^m \pi_{\theta}(a_t^i|s_t)$$

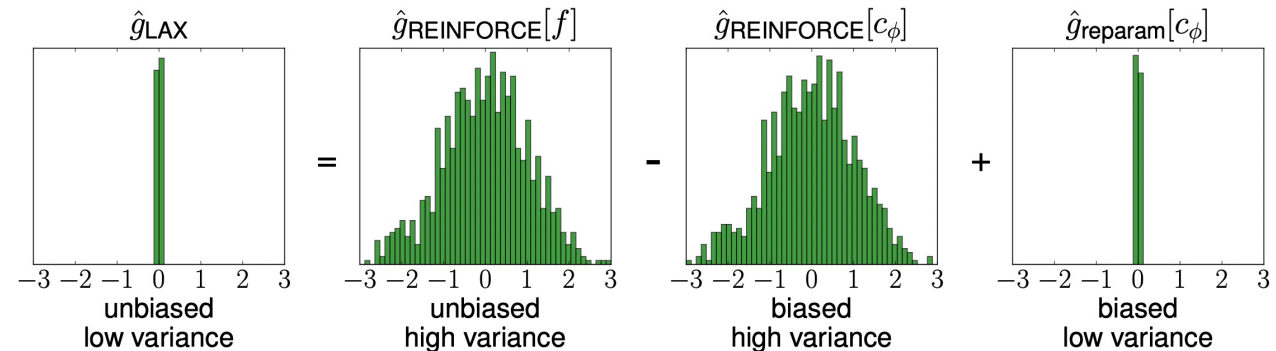
$$\nabla_{\theta} \eta(\pi_{\theta}) = \mathbb{E}_{\rho_{\pi}, \pi} \left[\sum_{i=1}^m \nabla_{\theta} \log \pi_{\theta}(a_t^i|s_t) \left(\hat{Q}(s_t, a_t) - b_i(s_t, a_t^{-i}) \right) \right]$$

For factorized spaces, baselines can depend on independent action factors

The Mirage of Action-Dependent Baselines in Reinforcement Learning, Tucker et al '18

Variance Reduction for Policy Gradient with Action-Dependent Factorized Baselines, Wu et al '18

Alternative Estimators



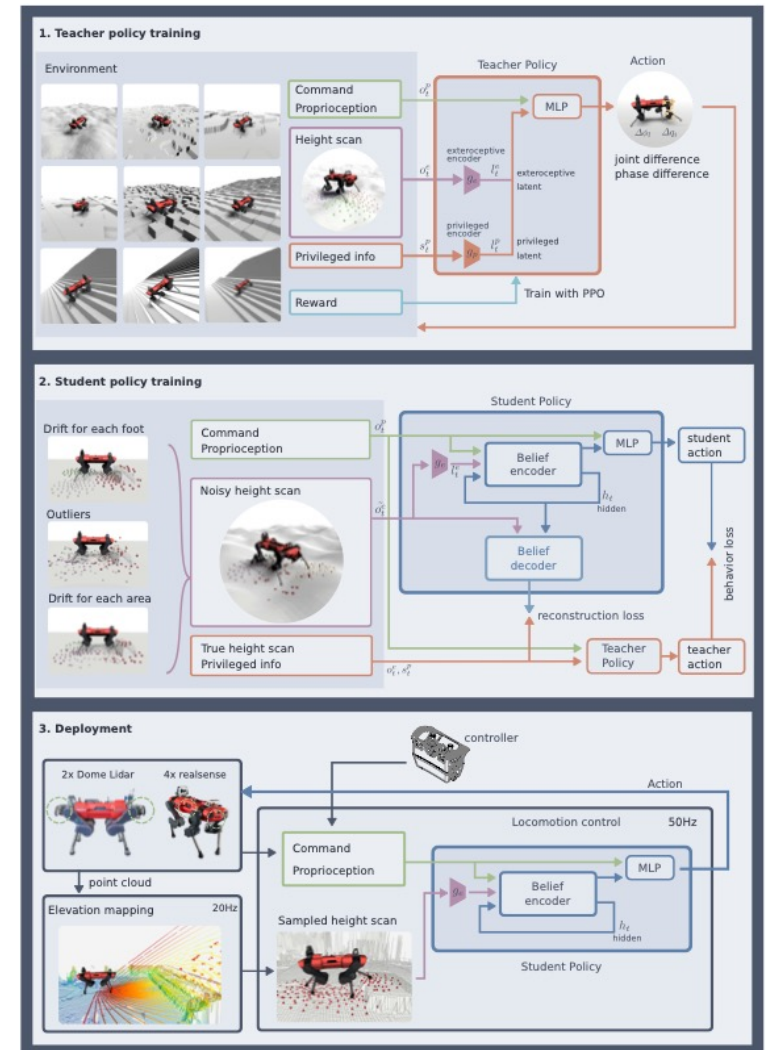
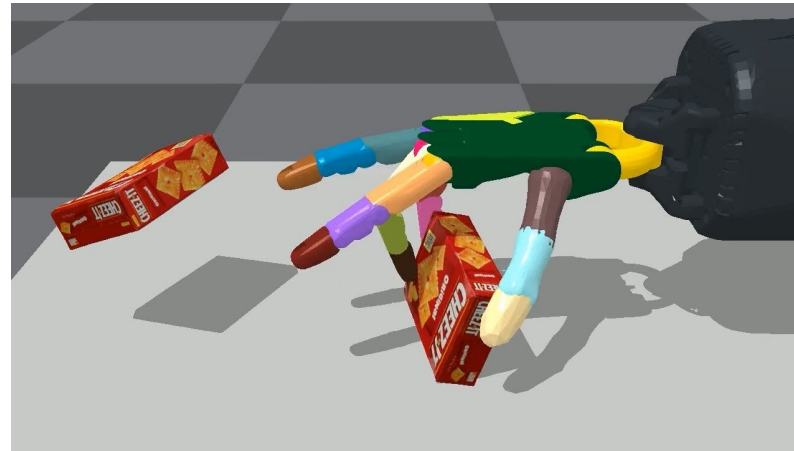
Q-Prop: Sample-Efficient Policy Gradient with An Off-Policy Critic, Gu et al '16

Backpropagation through the Void: Optimizing control variates for black-box gradient estimation, Grathwohl et al '17

Categorical Reparameterization with Gumbel-Softmax, Jang et al '16

Frontiers of Policy Gradient Research

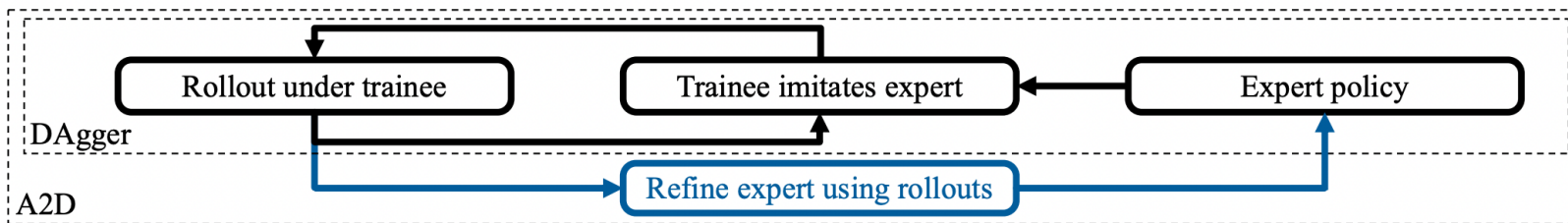
Learning from High Dimensional Observations



Learning Quadrupedal Locomotion over Challenging Terrain, Lee et al '20

A System for General In-Hand Reorientation, Chen et al '21

Challenging to provide guarantees in partially observed settings!



Robust Asymmetric Learning in POMDPs, Warrington et al '20

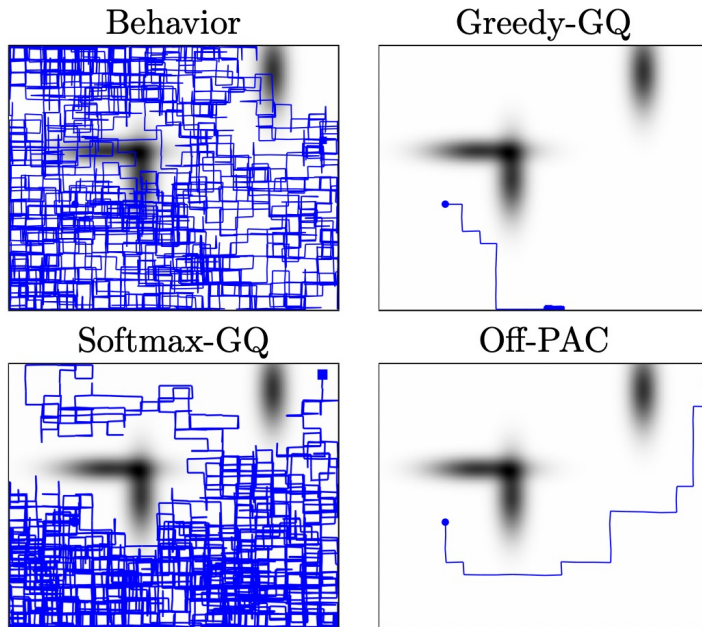
Frontiers of Policy Gradient Research

Bootstrapping from Prior/Off-Policy Data

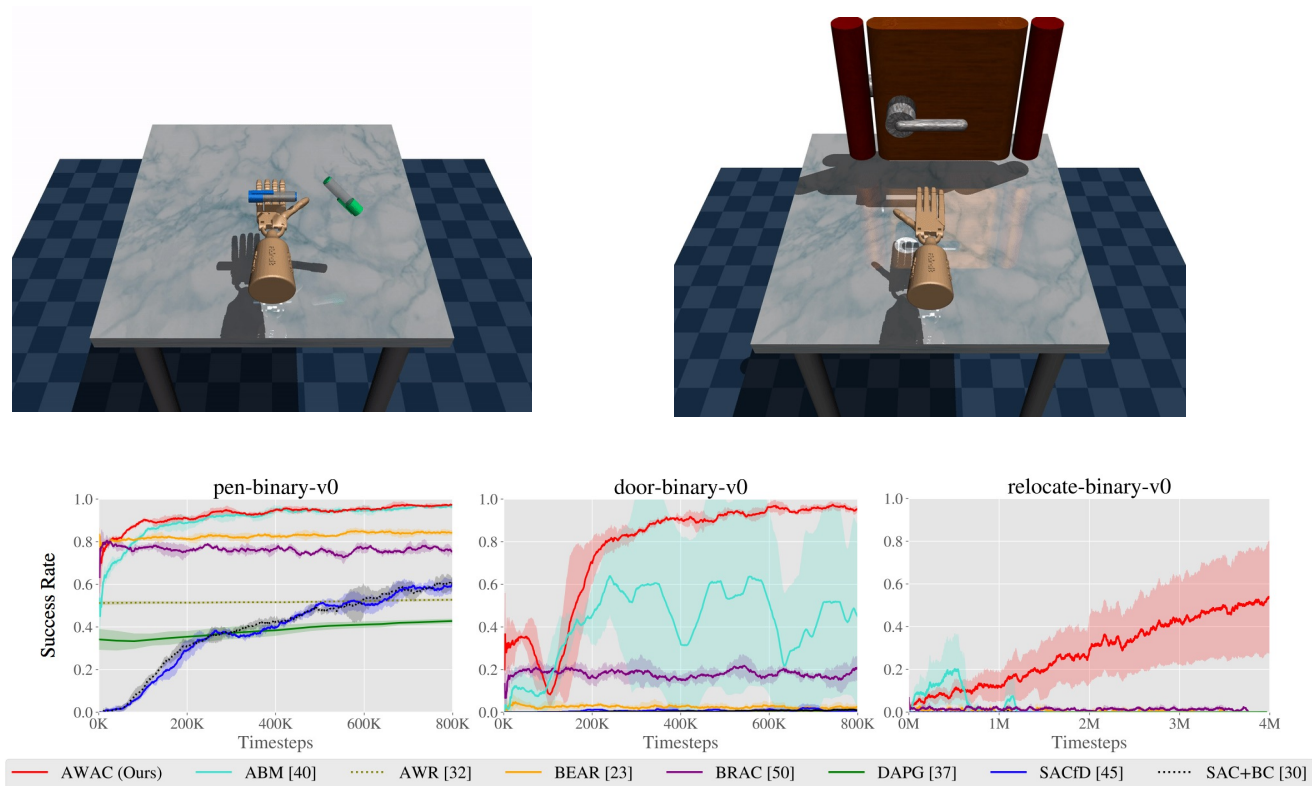
Off-policy policy gradient

$$\mathbb{E}_{\beta} \left[\frac{\pi_{\theta}(a|s)}{\beta(a|s)} Q^{\pi}(s, a) \nabla_{\theta} \ln \pi_{\theta}(a|s) \right]$$

Learning from Prior Data



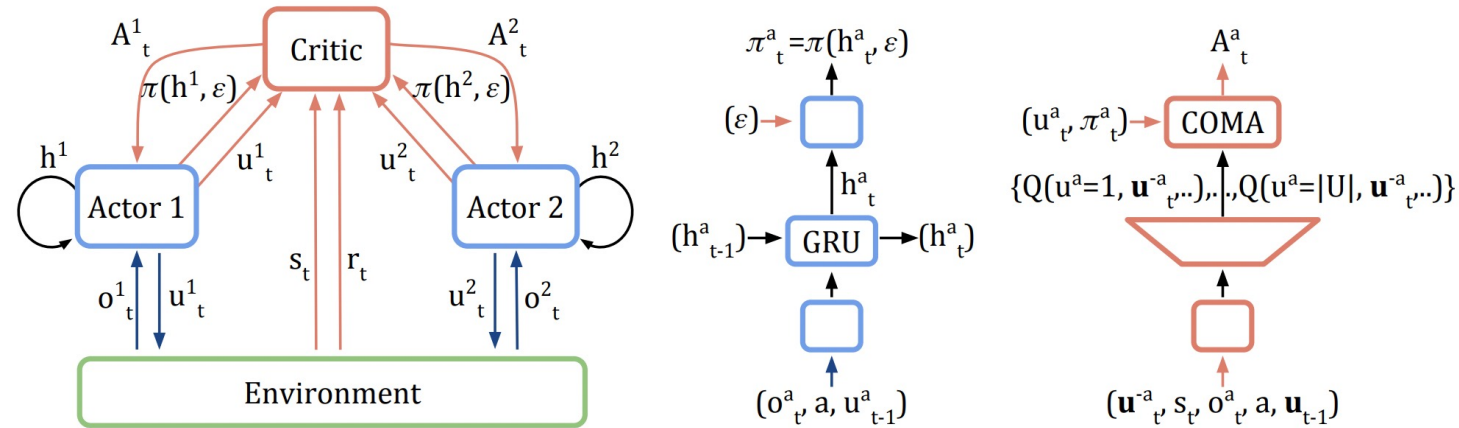
Off-Policy Actor-Critic, Degrís et al '13



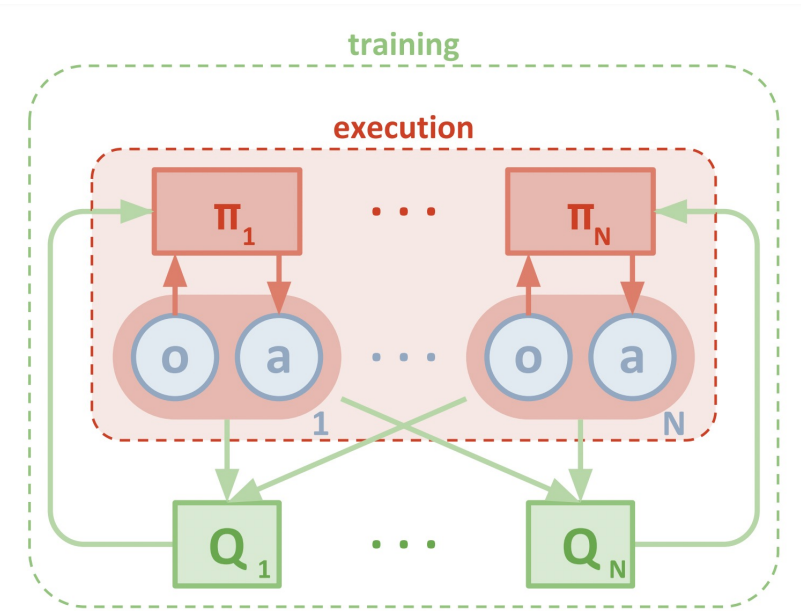
Advantage Weighted Actor Critic, Nair et al '20
DDPGfD, Vecerik '17
DAPG, Rajeswaran '17

Frontiers of Policy Gradient Research

Multi-agent policy gradient



Counterfactual Multi-Agent Policy Gradients, Foerster et al '17



Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments, Lowe et al '17

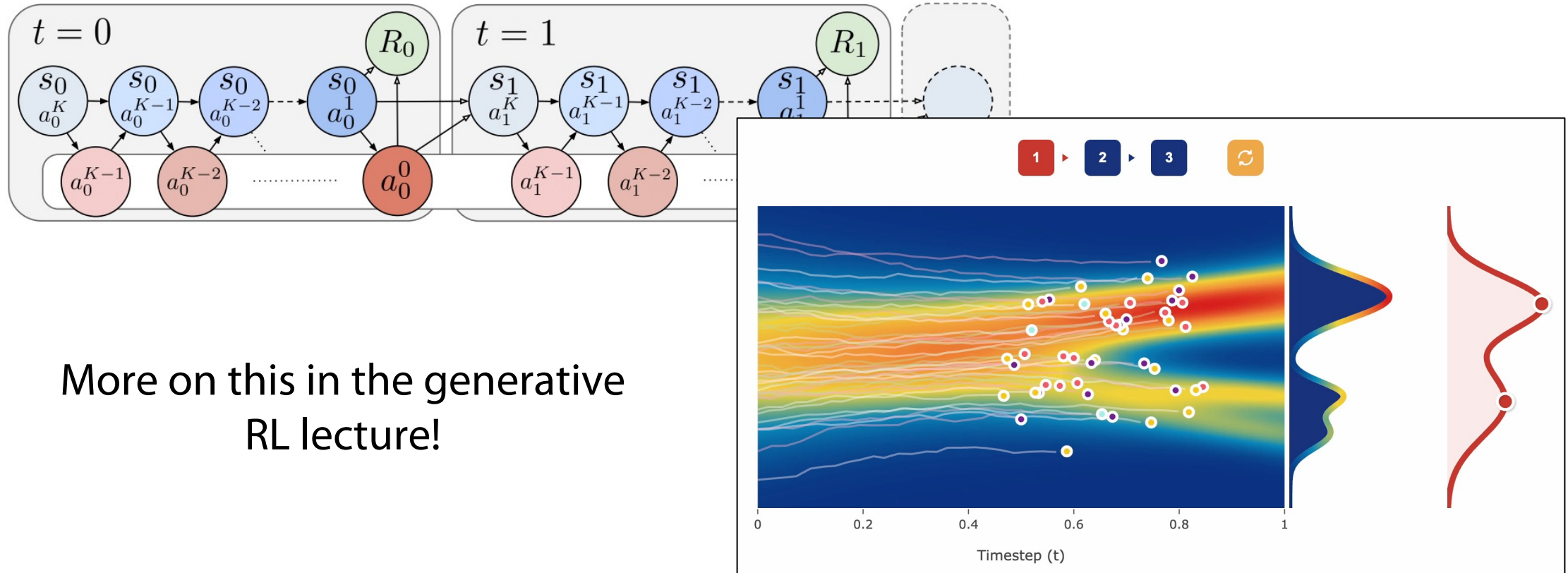
Primary challenges:

1. Non-stationarity
2. Data-efficiency
3. Communication

Frontiers of Policy Gradient Research

Finetuning generative models like diffusion with RL

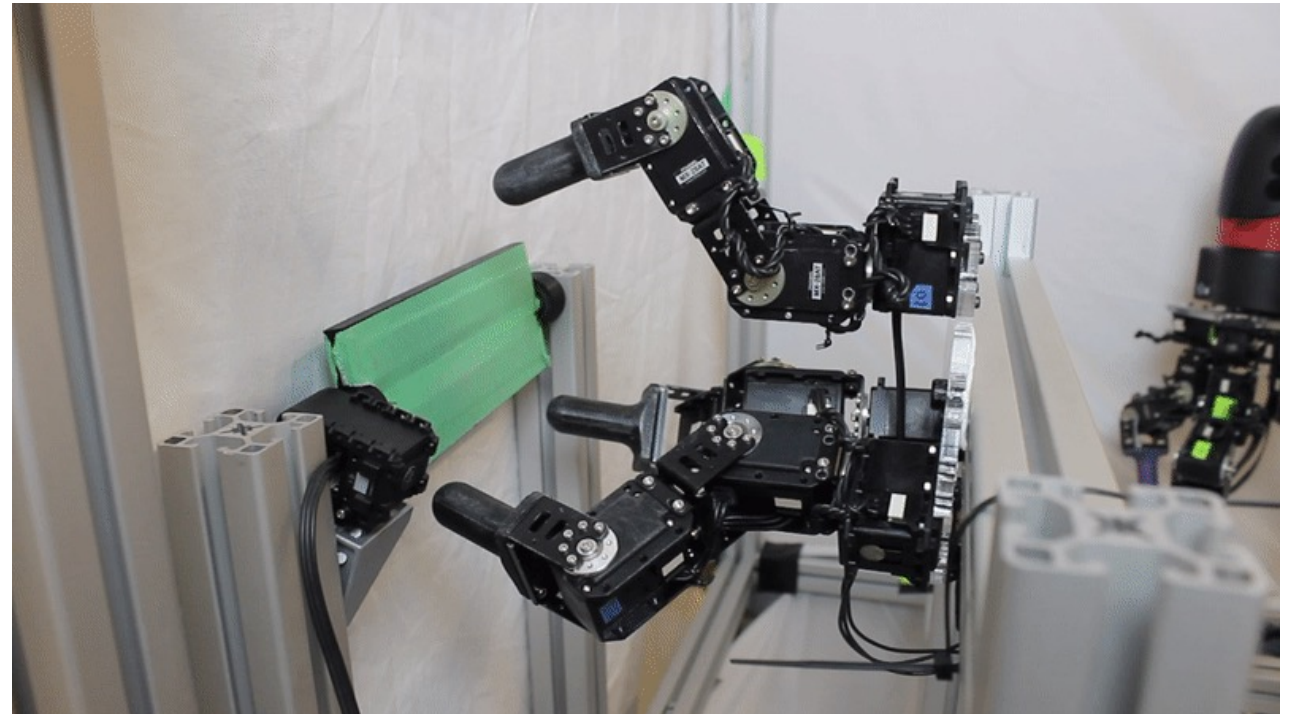
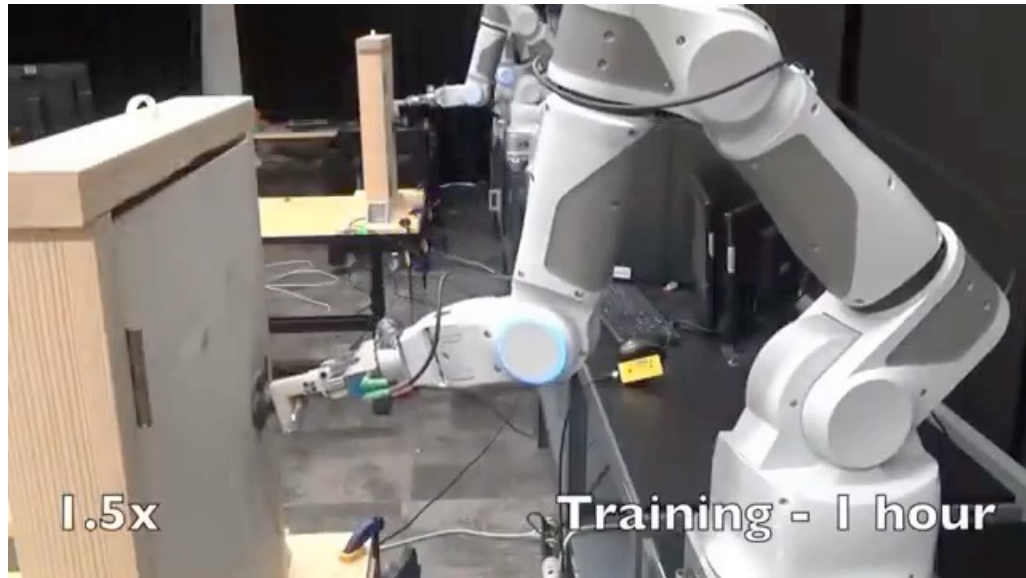
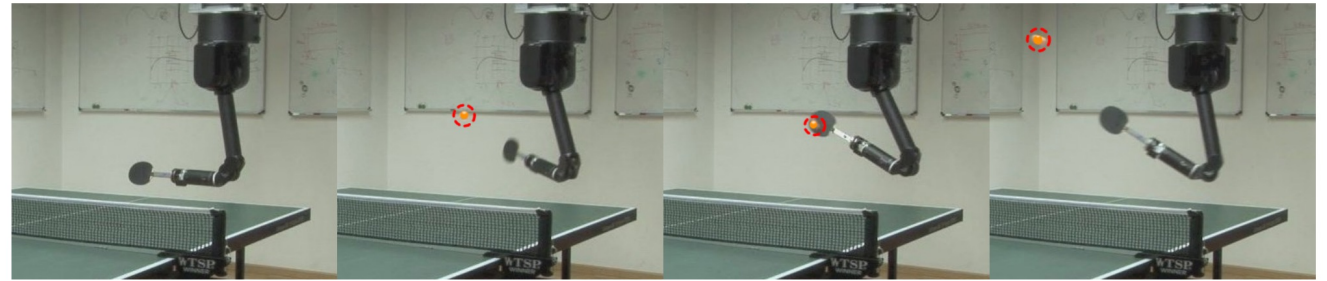
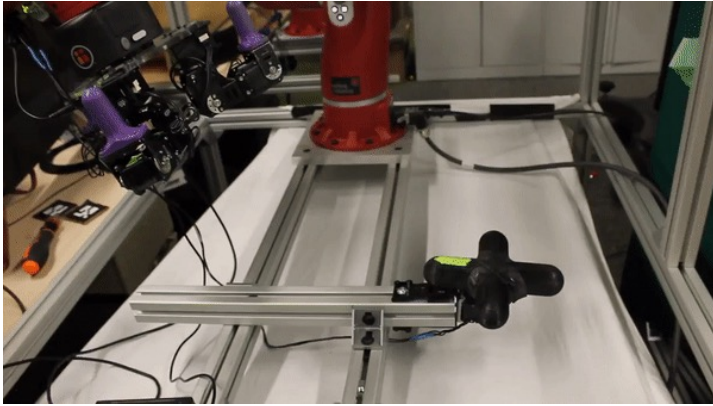
Challenge: hard to compute and update likelihoods



More on this in the generative
RL lecture!

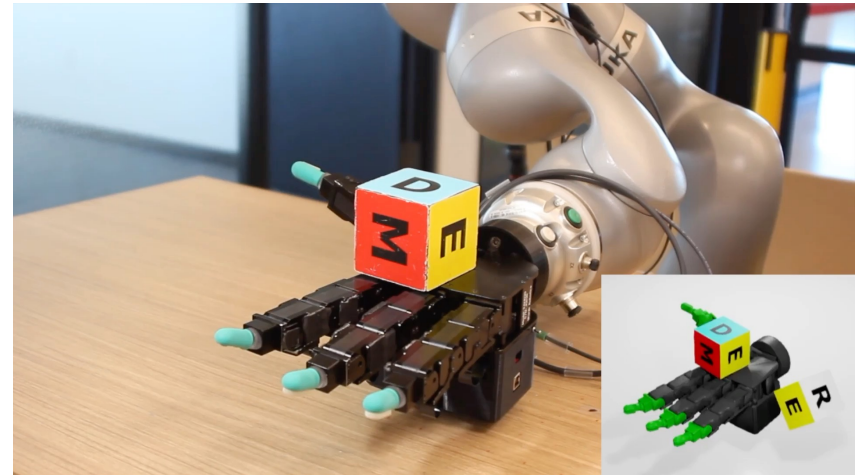
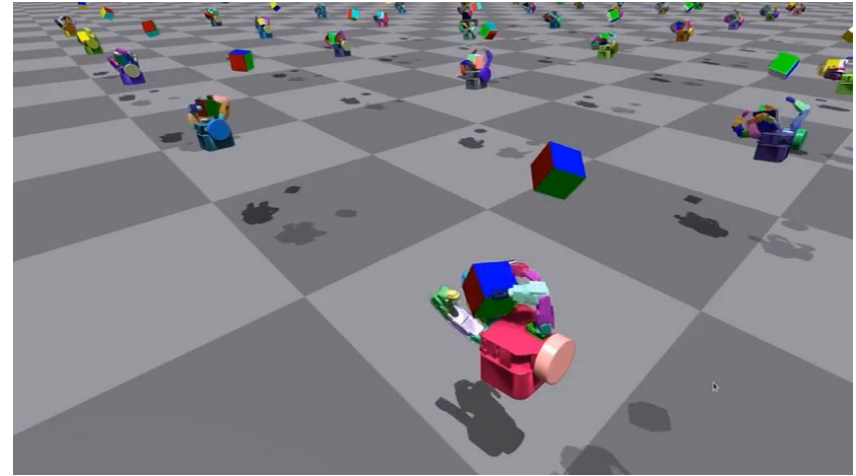
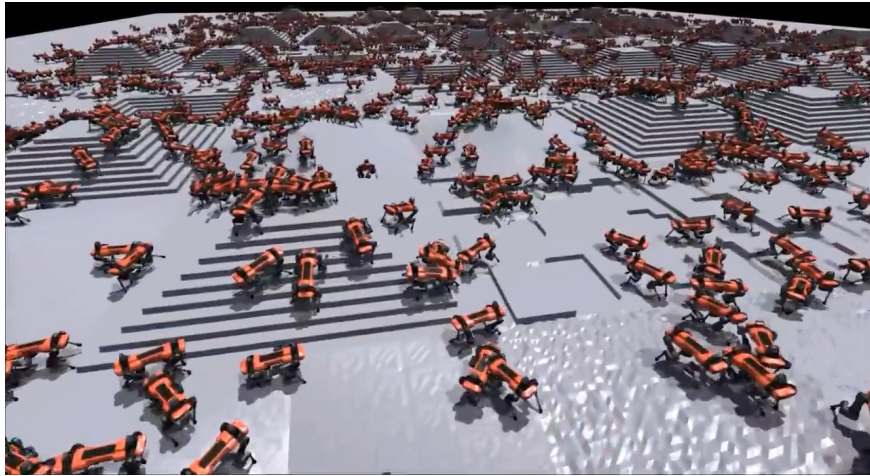
How is this useful for robotics?

Can be used to train robots in the real world but only in limited settings



How is this useful for robotics?

Largely useful for pretraining in simulation

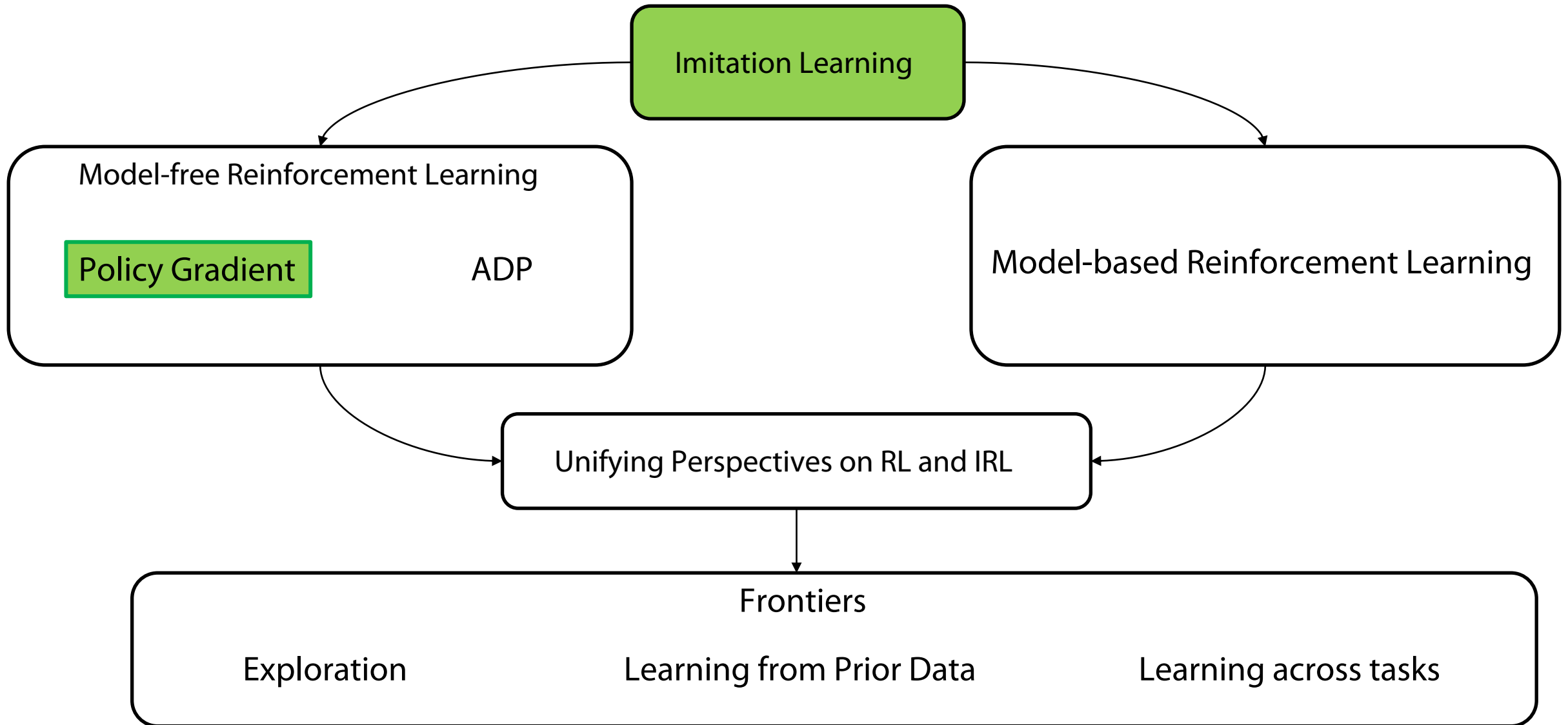


More in the sim2real lecture!

Summary

- Policy gradient methods form an effective solution technique to the RL problem
- Techniques range from vanilla policy gradient to NPG to PPO, each with its own pros and cons
- PG can be very adept at solving black-box optimization asymptotically, but can be very slow
- Several open frontiers still exist for research into PG methods
- Most promising use of PG methods in robotics is through simulation to reality transfer

Class Structure



Lecture outline

Recap: Policy Gradients ++



Policy Gradients Finish Up + Clarifications



Kronecker Factorization



Off-Policy RL

Off-Policy RL outline

Off-Policy RL Motivation



Getting Actor Critic to Work in Practice



A Closer Look at Convergence



Frontiers of Off-Policy RL

What can we do to make PG suitable for robots?

Why is Policy Gradient sample inefficient?

$$\begin{aligned}\nabla_{\theta} J(\theta) &= \int p_{\theta}(\tau) \nabla_{\theta} \log p_{\theta}(\tau) d\tau \\ &\approx \frac{1}{N} \sum_{i=1}^N \sum_t \nabla_{\theta} \log \pi_{\theta}(a_t^i | s_t^i) \sum_{t'=t}^T r(s_{t'}^i, a_{t'}^i)\end{aligned}$$

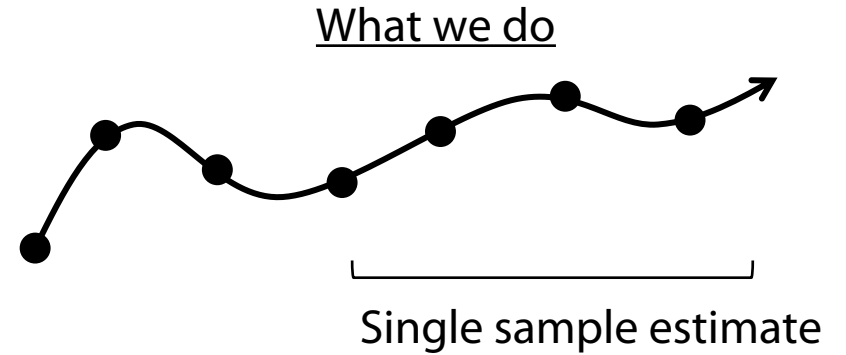
On-policy, unable to effectively use past data

High Variance Estimator

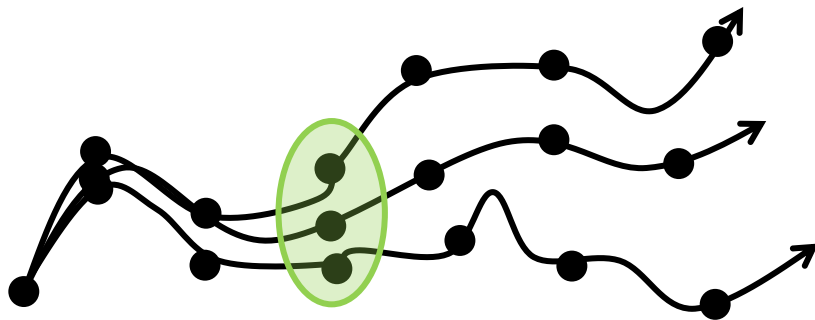
Can we develop a **low variance off-policy** RL algorithm that can bootstrap from prior data?

What can we do to lower variance?

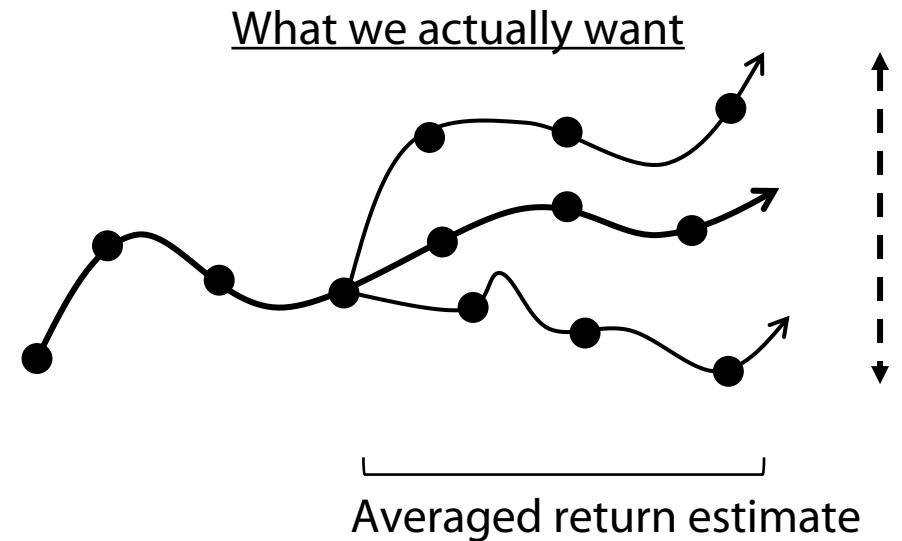
$$\begin{aligned}\nabla_{\theta} J(\theta) &= \int p_{\theta}(\tau) \nabla_{\theta} \log p_{\theta}(\tau) d\tau \\ &\approx \frac{1}{N} \sum_{i=1}^N \sum_t \nabla_{\theta} \log \pi_{\theta}(a_t^i | s_t^i) \underbrace{\sum_{t'=t}^T r(s_{t'}^i, a_{t'}^i)}\end{aligned}$$



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

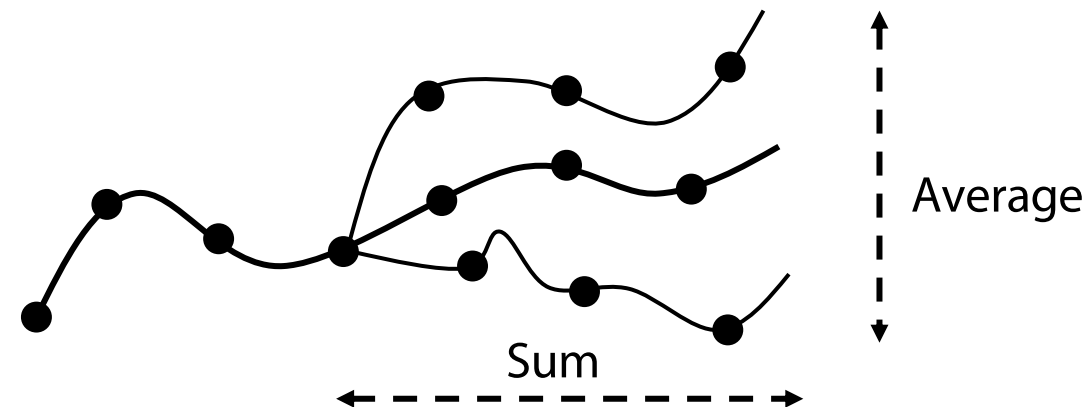


Function approximator bundles return estimates across states



Notation: Q 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

$$Q^{\pi}(s_t, a_t) = \mathbb{E}_{\pi_{\theta}} \left[\sum_{t'=t}^T r(s_{t'}', a_{t'}') \mid s_t, a_t \right] \quad \text{Bundles estimates across } (s, a)$$

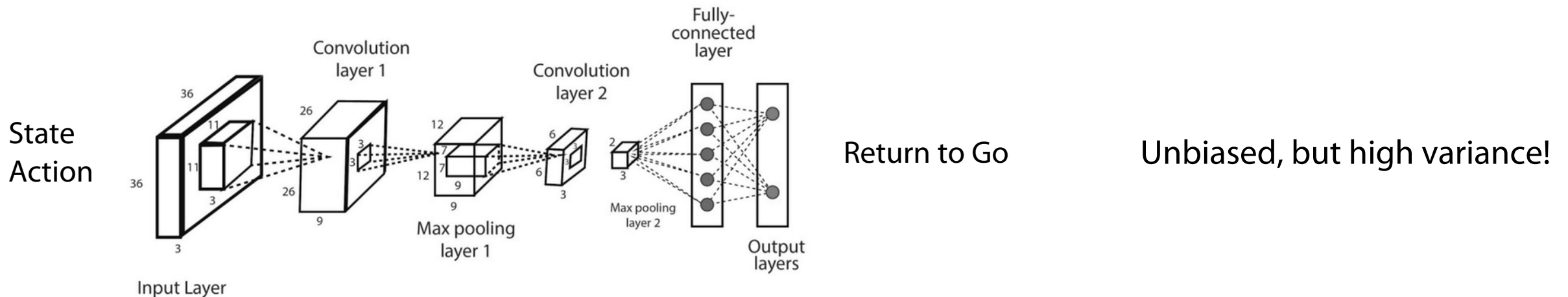
Use the magic of (deep) function approximation

Attempt 0: Monte-Carlo Estimation of Q-Functions

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

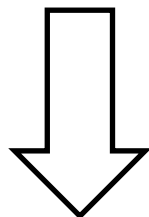
$$Q^{\pi}(s_t, a_t) = \mathbb{E}_{\pi_{\theta}} \left[\sum_{t'=t}^T r(s_{t'}, a_{t'}) | s_t, a_t \right] \leftarrow \text{Monte-carlo approximation}$$

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



Can we do better?

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



Much lower variance if estimated well

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

Can be learned off-policy!

Has special structure we can exploit!!

Attempt 1: Using Recursive Structure

$$\frac{1}{N} \sum_{i=0}^N \sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t^i | s_t^i) Q^{\pi}(s_t^i, a_t^i) \quad Q^{\pi}(s_t, a_t) = \mathbb{E}_{\pi_{\theta}} \left[\sum_{t'=t}^T r(s_{t'}, a_{t'}) | s_t, a_t \right]$$

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

Average Q-function over actions sampled from policy

Value functions are recursive

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

Q-values via 1-step lookahead $Q^{\pi}(s_t, a_t) = \mathbb{E}_{s_{t+1} \sim p(\cdot | s_t, a_t)} [r(s_t, a_t) + V^{\pi}(s_{t+1}) | s_t = s_t, a_t = a_t]$

Attempt 1: Using Recursive Structure

$$\frac{1}{N} \sum_{i=0}^N \sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t^i | s_t^i) Q^{\pi}(s_t^i, a_t^i) \quad Q^{\pi}(s_t, a_t) = \mathbb{E}_{\pi_{\theta}} \left[\sum_{t'=t}^T r(s_{t'}, a_{t'}) | s_t, a_t \right]$$

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



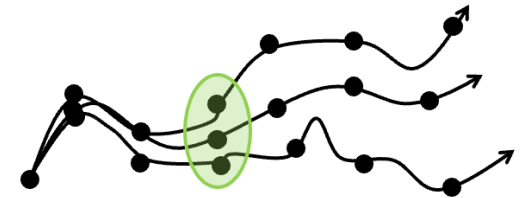
Recipe for policy gradient

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



Value Bellman equation

$$\nabla_{\theta} J(\theta) = \frac{1}{N} \sum_{i=0}^N \sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t^i | s_t^i) \underbrace{(r(s_t, a_t) + V(s_{t+1}) - V(s_t))}_{\text{1-step lookahead - better estimate of future return}}$$



1-step lookahead - better estimate of future return

Attempt 1: Using Recursive Structure

$$\nabla_{\theta} J(\theta) = \frac{1}{N} \sum_{i=0}^N \sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t^i | s_t^i) (r(s_t, a_t) + V(s_{t+1}) - V(s_t)) \quad V^{\pi}(s_t) = \mathbb{E}_{\pi_{\theta}} \left[\sum_{t'=t}^T r(s_{t'}, a_{t'}) | s_t \right]$$

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

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

Iteratively update y with latest V

Fit a value function on on-policy data

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

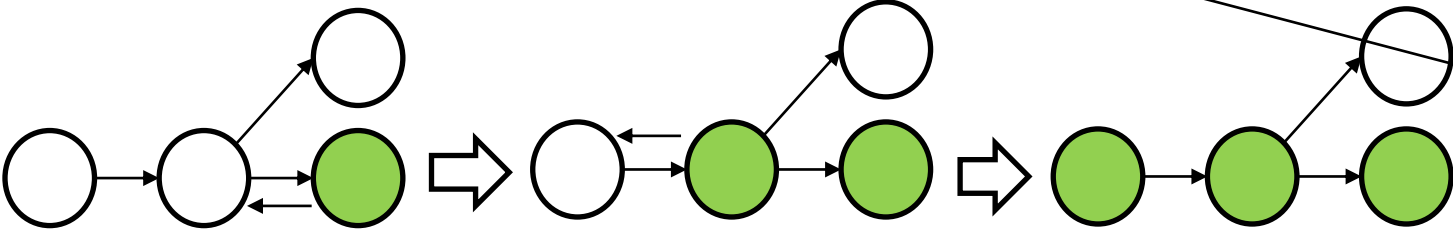
Attempt 2: Recursive structure in Q functions directly

Q functions have special recursive structure themselves!

$$\begin{aligned}
 Q^\pi(s_t, a_t) &= \mathbb{E}_{\pi_\theta} \left[\sum_{t'=t}^T r(s_{t'}, a_{t'}) \mid s_t, a_t \right] \\
 &= r(s_t, a_t) + \mathbb{E}_\pi \left[\sum_{t'=t+1} r(s_{t'}, a_{t'}) \mid s_{t+1}, a_{t+1} \sim \pi(\cdot \mid s_{t+1}) \right]
 \end{aligned}$$

Bellman equation

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



Can be from different policies

Decompose temporally via dynamic programming

Off-policy!

Learning Q-functions via Dynamic Programming

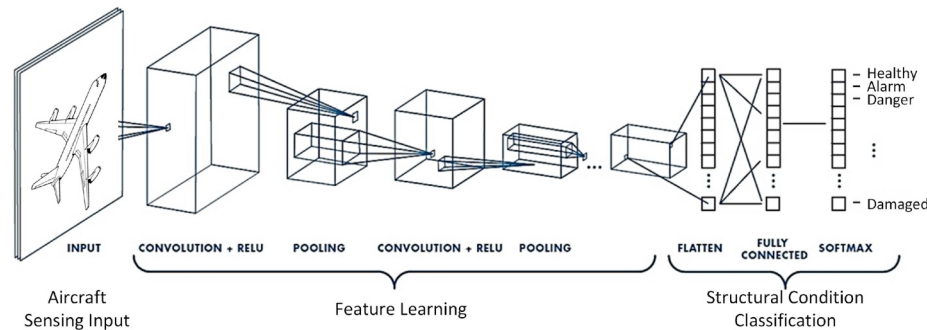
Policy Evaluation: Try to minimize Bellman Error (almost)

Bellman equation

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

Same function approximator

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

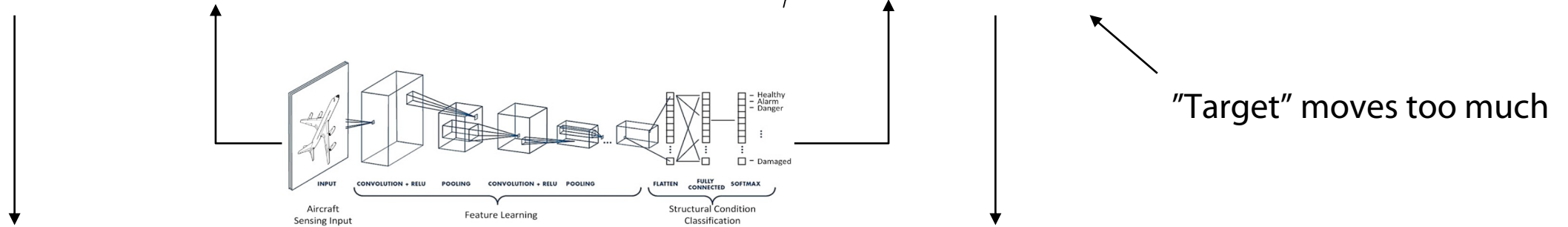


Why is this not just the gradient of the Bellman Error?

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

Approximate using stochastic optimization

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



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

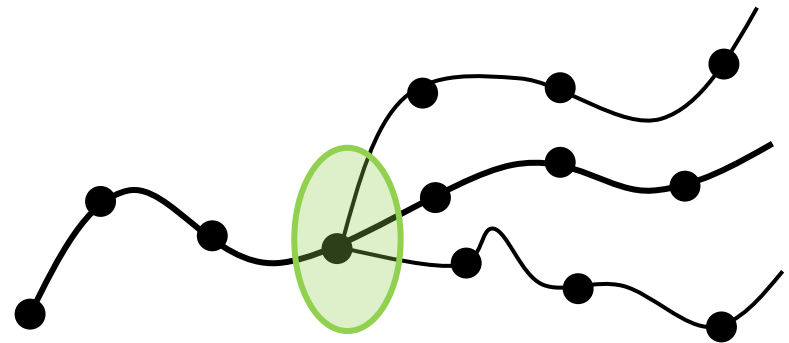
Policy Improvement: Improve policy with **policy gradient**

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

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

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

Likelihood Ratio/Score Function

Pathwise derivative/Reparameterization

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

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

Easier to Apply to Broad Policy Class

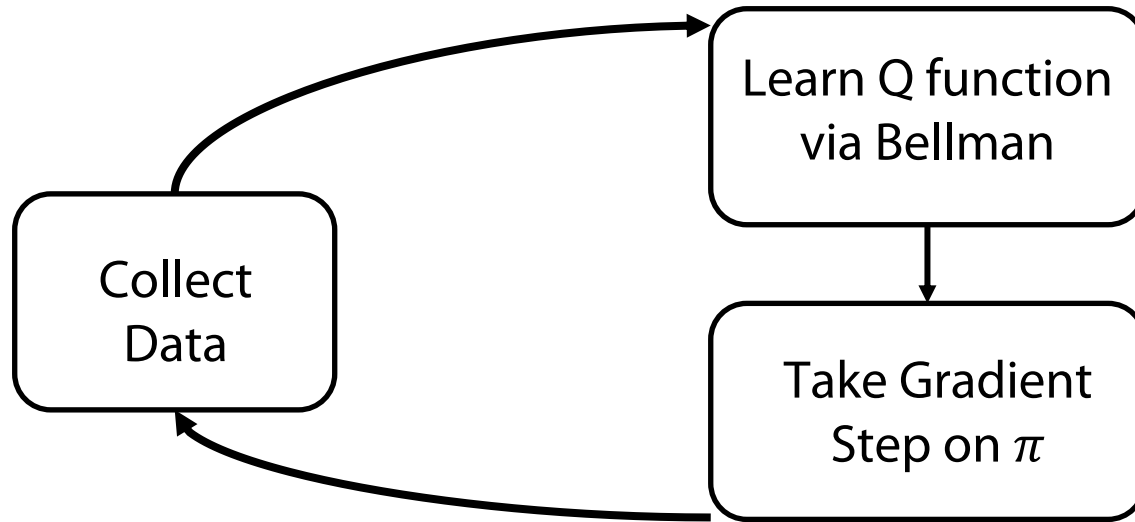
Lower variance (empirically)

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)

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



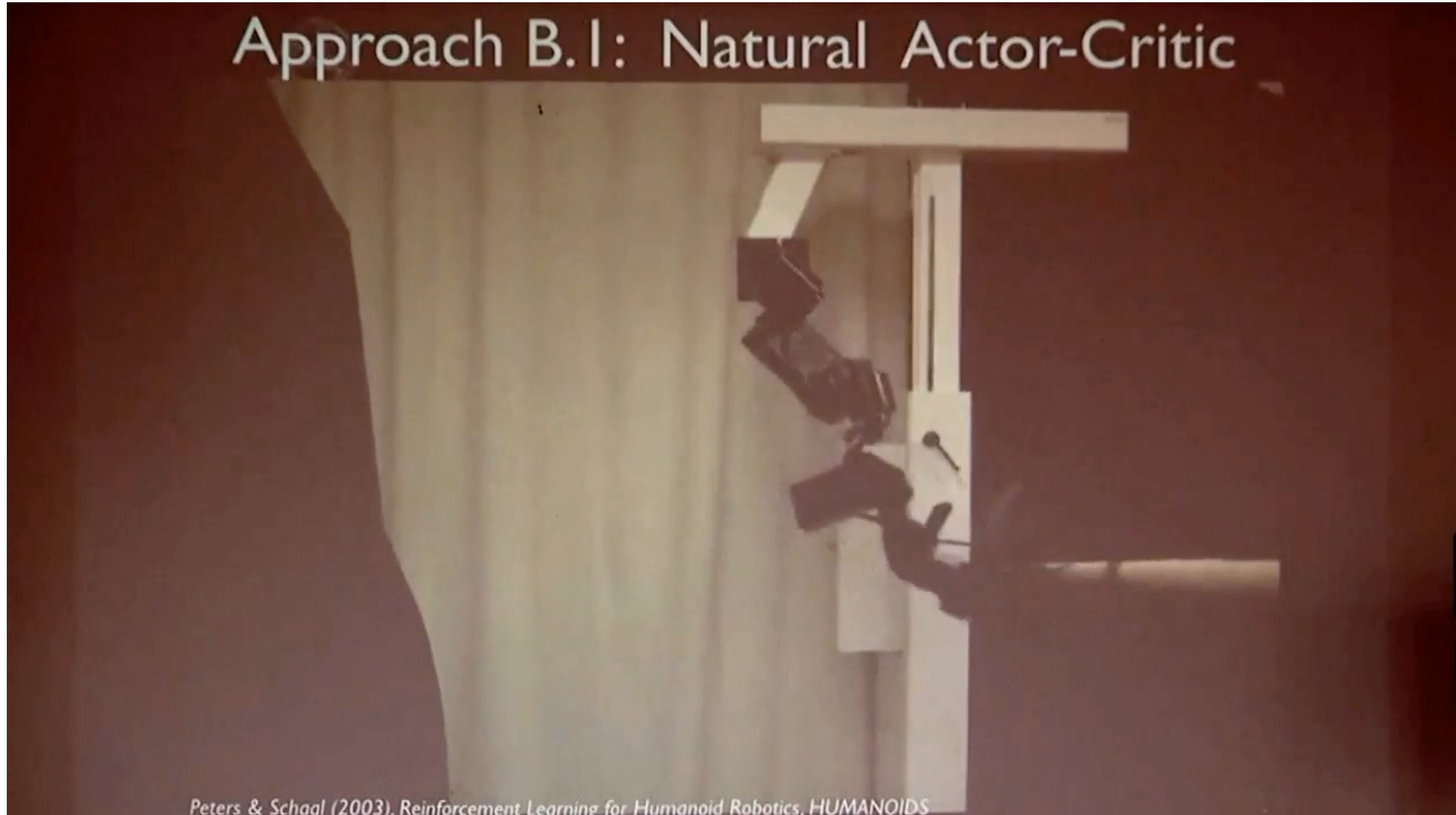
Lowers variance and is off-policy!

Actor: updated using learned critic (Policy Improvement)

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

Actor-Critic in Action

Approach B.1: Natural Actor-Critic



Peters & Schaal (2003). Reinforcement Learning for Humanoid Robotics, HUMANOIDS

Lecture outline

Off-Policy RL Motivation



Getting Actor Critic to Work in Practice



A Closer Look at Convergence

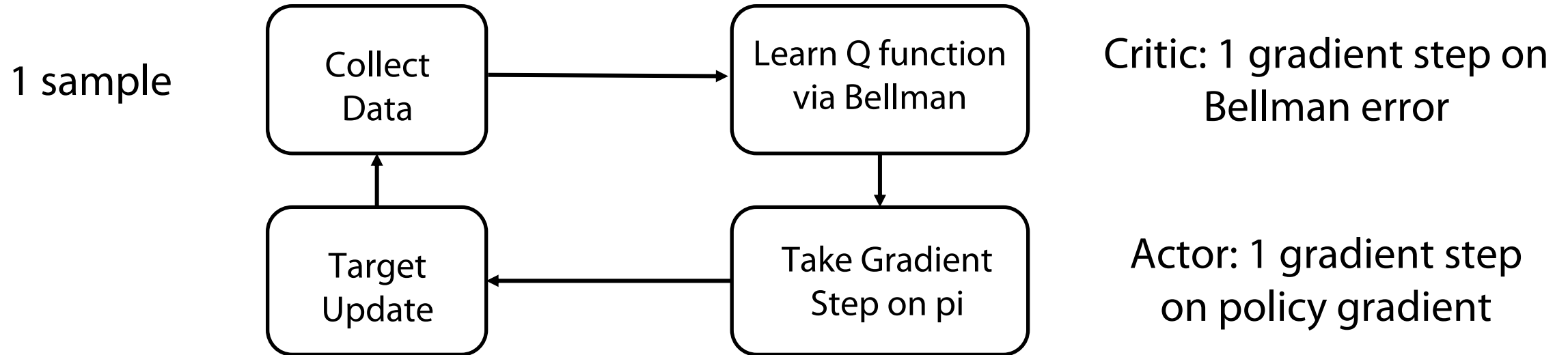


Frontiers of Off-Policy RL

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

Going from Batch Updates to Online Updates

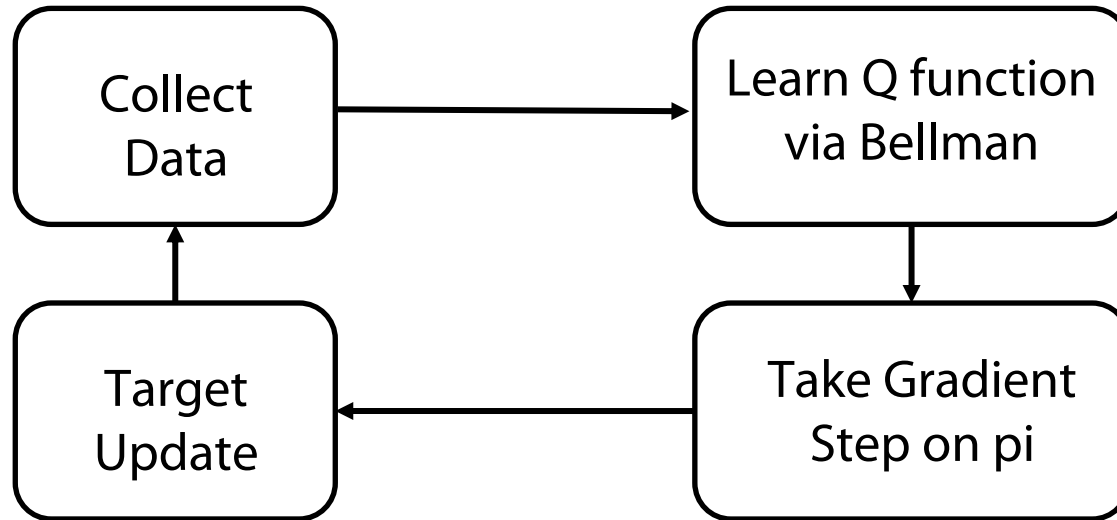
This algorithm can go from full batch mode to fully online updates



Allows for much more immediate updates

Challenges of doing online updates

1 sample



Critic: 1 gradient step on Bellman error

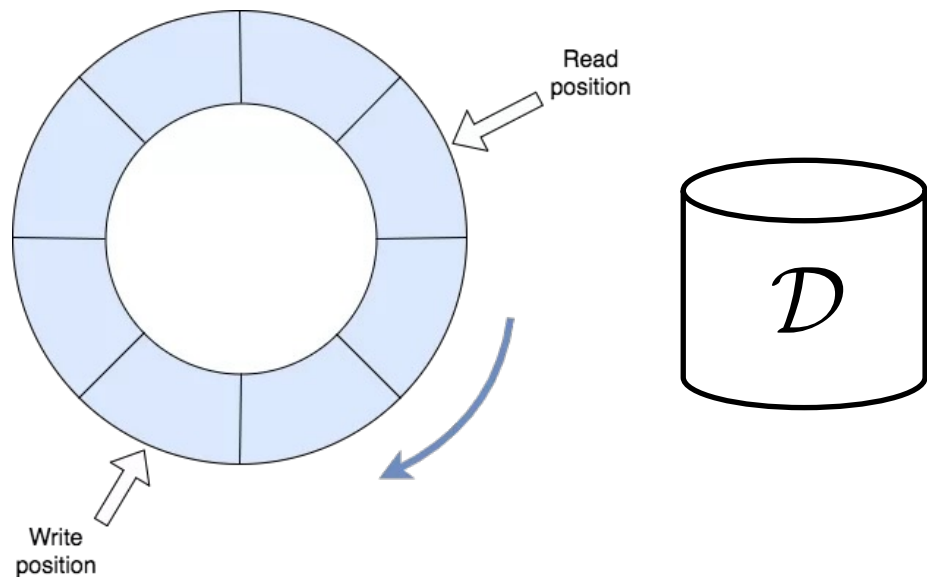
Actor: 1 gradient step on policy gradient

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

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

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

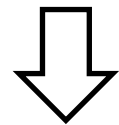
1. Sample uniformly
2. Prioritize by TD-error
3. Prioritize by target error
4. ... open area of research!

Slowing moving targets with target networks

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

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

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

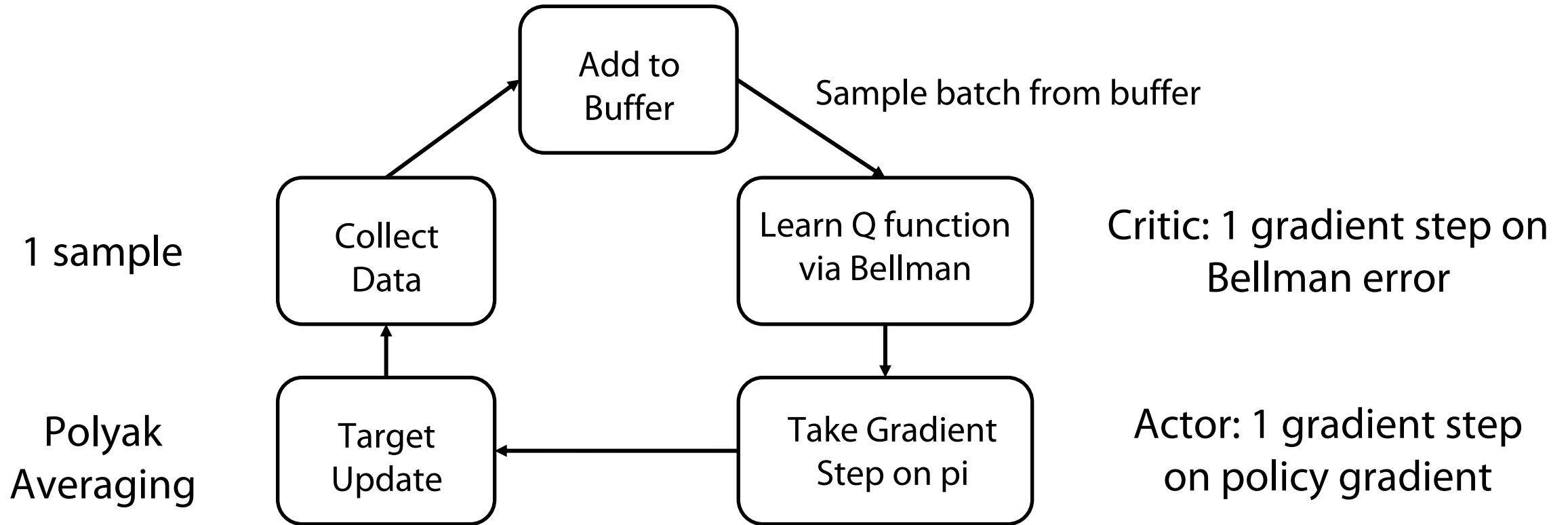


Move $\bar{\phi}$ to ϕ slowly!

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

Polyak averaging

A Practical Off-Policy RL Algorithm



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

Critic Update

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

Actor Update

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

What if we removed this explicit actor completely?

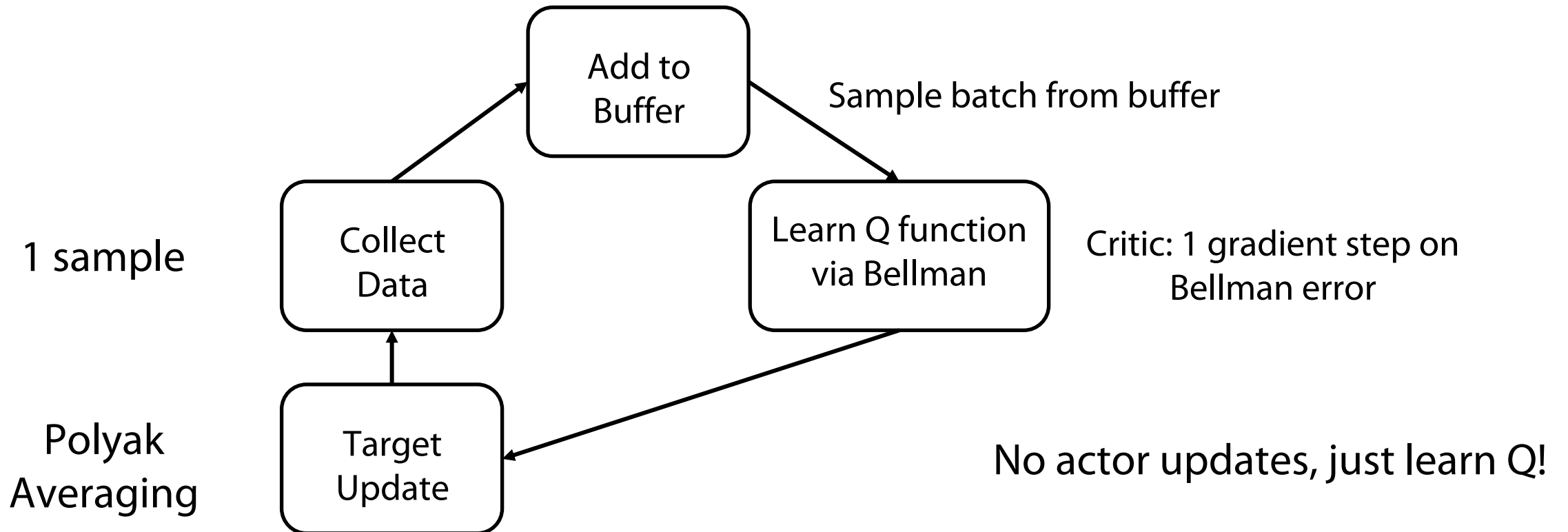


Directly Learning Q*


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

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

Directly do max in the Bellman update



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:

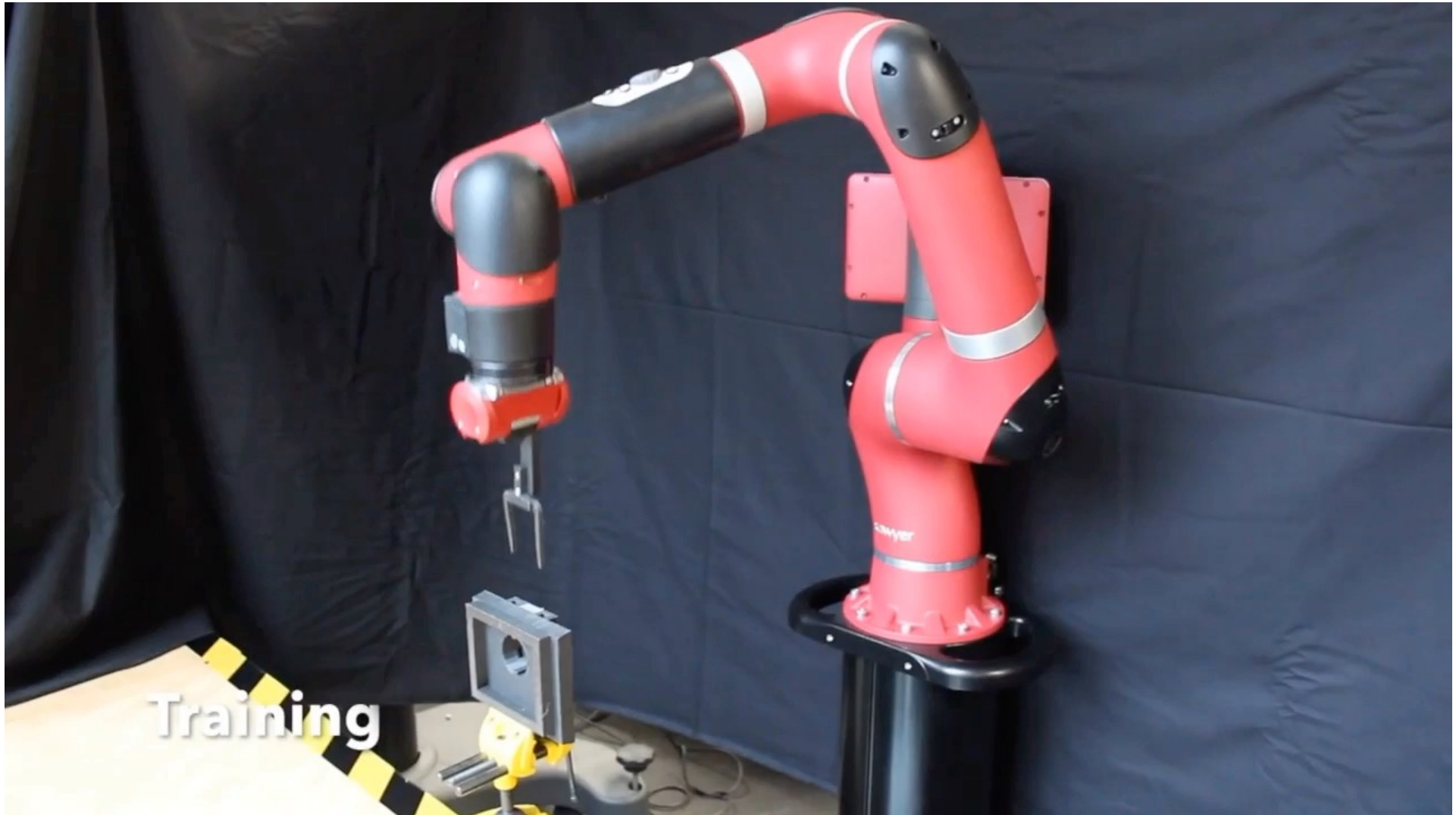
1. Sampling based (QT-opt (Kalashnikov et al))
2. Optimization based (NAF, Gu et al)

Practical Actor-Critic in Action



Trained using QT-Opt

Practical Actor-Critic in Action



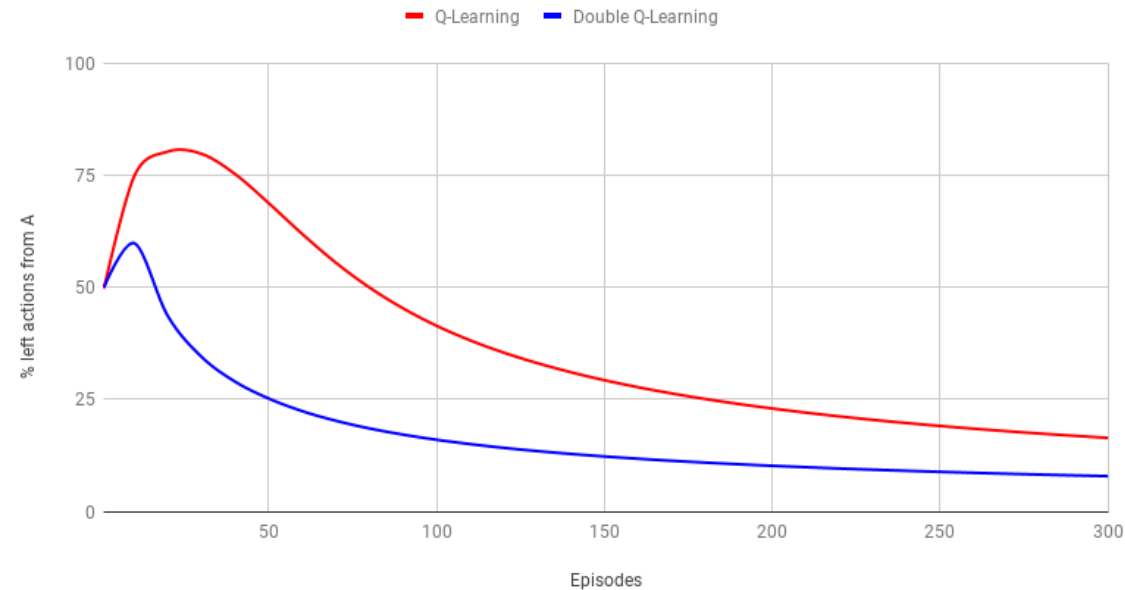
Trained using DDPG

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

Where does this fail?

Performance Double Q-Learning vs Q-Learning

10 actions at B



Some issues remain:

- 1. Overestimation bias**
2. Insufficient exploration

Let's try and understand these!

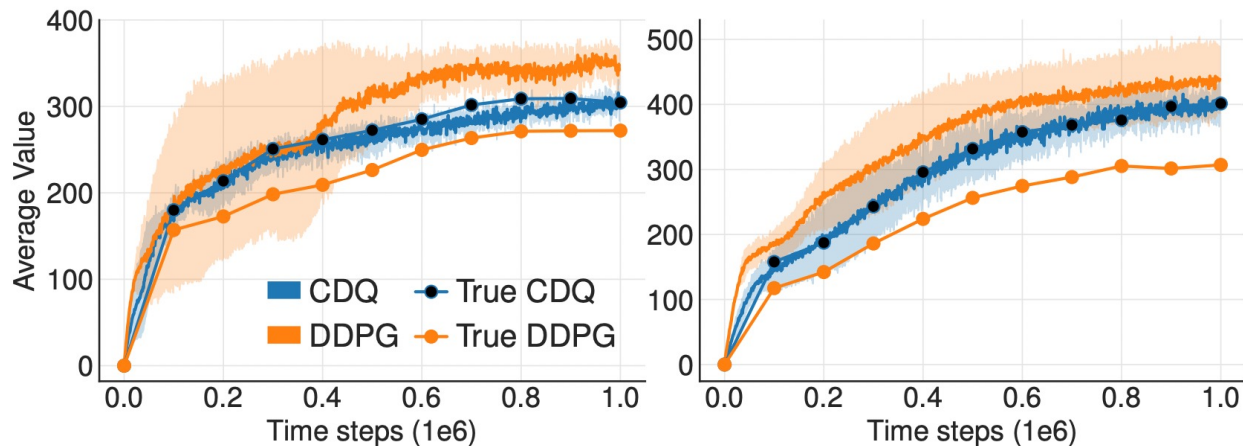
Overestimation Bias in Actor-Critic

Optimized Q's are often overly optimistic

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

Q is meant to be an expectation

→ actually a random variable because of limited data/stochasticity

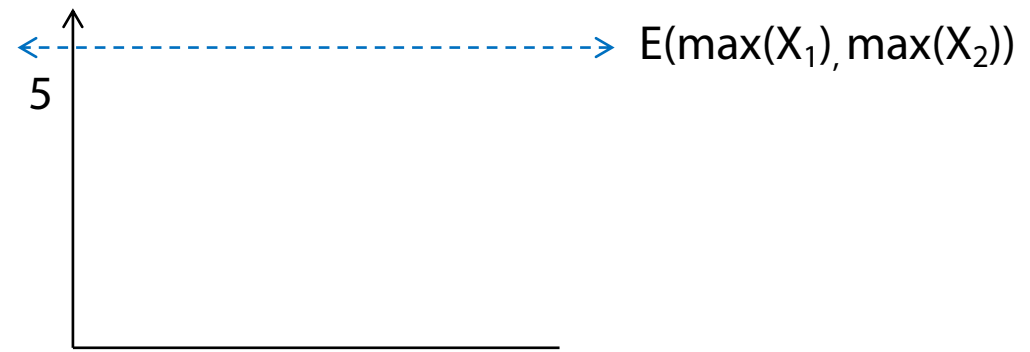
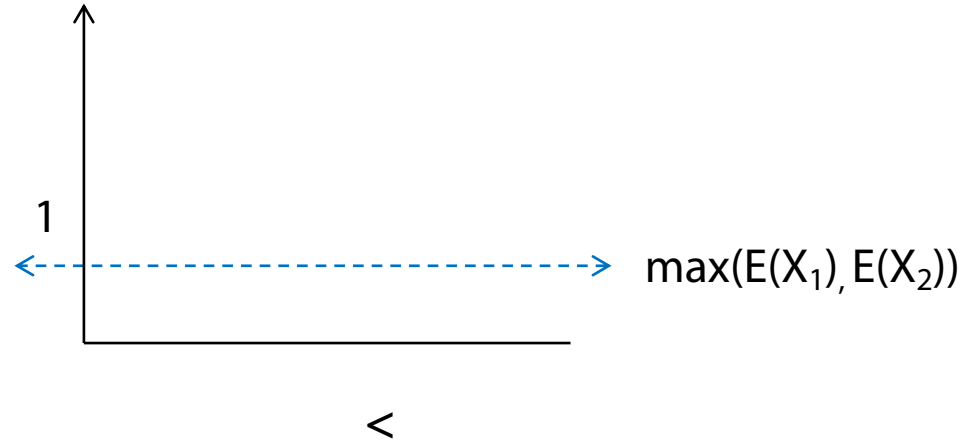
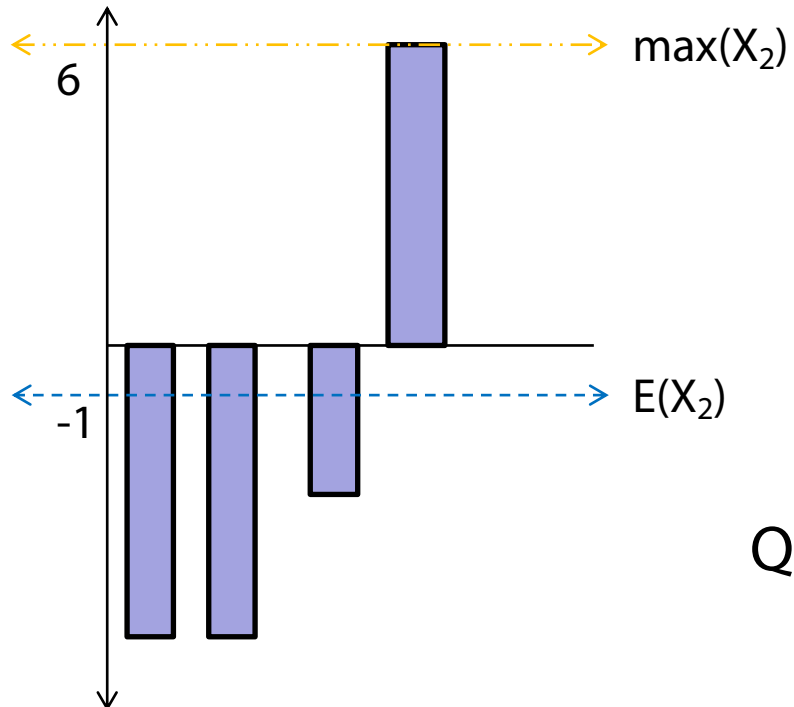
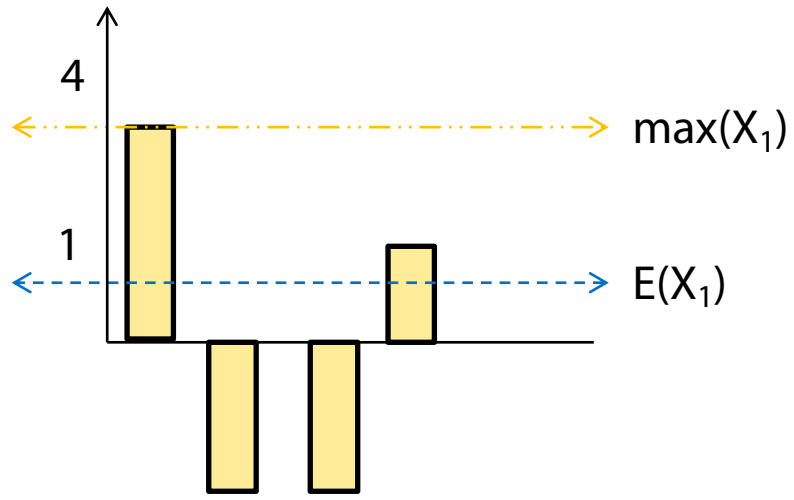


(a) Hopper-v1

(b) Walker2d-v1

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

Overestimation Bias in Actor-Critic

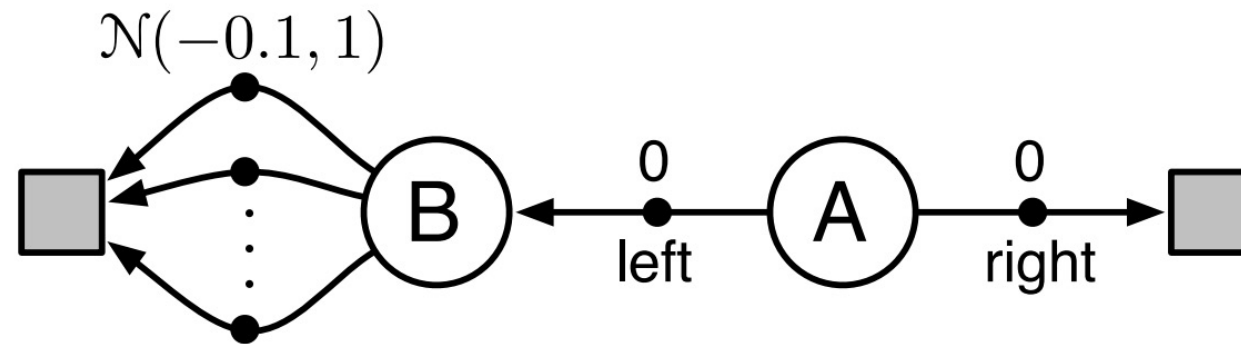


Q-learning can overestimate when values are imperfect (even when unbiased)

Overestimation Bias in Actor-Critic

Steps	X1	μ_1		X2	μ_2	Max(μ)
1	3	3.00		-8	-8.00	3.00
2	-2	0.50		-5	-6.50	0.50
3	4	1.67		-8	-7.00	1.67
4	-1	1.00		-10	-7.75	1.00
5	-5	-0.20		5	-5.20	-0.20
6	2	0.17		-7	-5.50	0.17
7	-6	-0.71		-3	-5.14	-0.71
8	-3	-1.00		3	-4.13	-1.00
9	8	0.00		8	-2.78	0.00
10	-6	-0.60		9	-1.60	-0.60
11	-9	-1.36		2	-1.27	-1.27
12	9	-0.50		-1	-1.25	-0.50
13	1	-0.38		2	-1.00	-0.38
14	-1	-0.43		8	-0.36	-0.36
E(X)	-0.43			-0.36	E(Max(μ))	0.09
Max E(X)	-0.36					

Overestimation Bias in Actor-Critic



Action 1	Action 2	Action 3
-1	1	-0.1
0.9	-0.5	-0.2

-0.25

0.5

-0.15

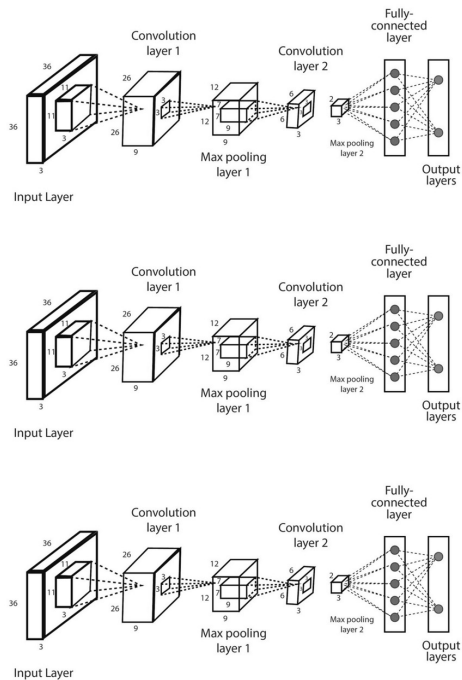
Max = 0.5

Chooses left, even though **average** is worse

Also true for actor-critic methods!

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

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

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

Actor

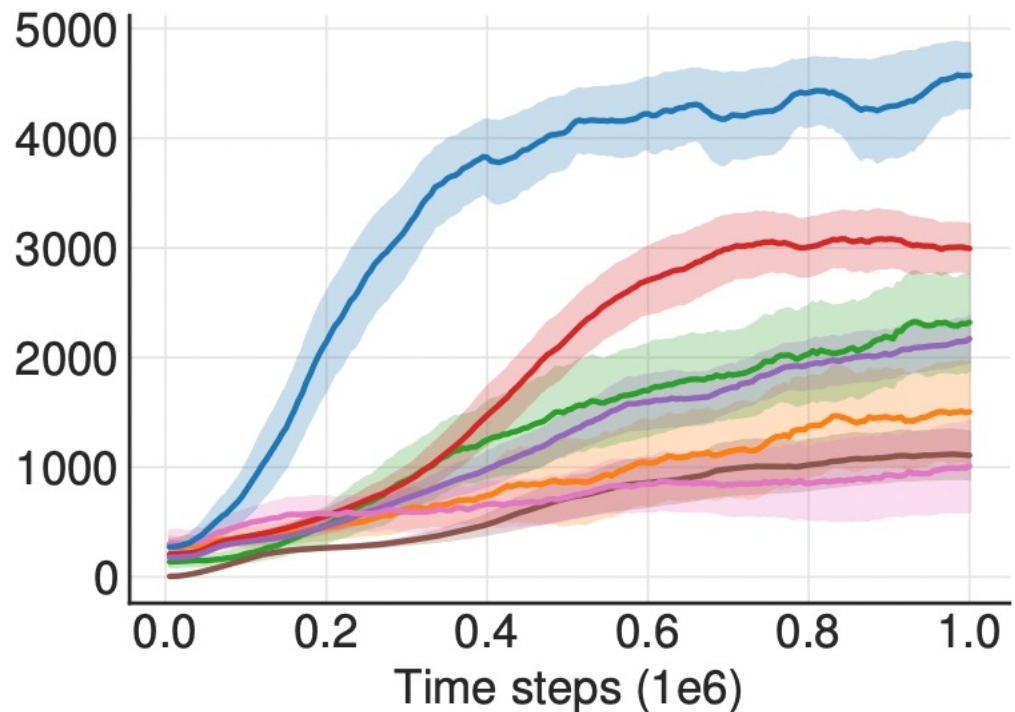
$$\max_{\theta_j} \mathbb{E}_{s \sim \mathcal{D}} \mathbb{E}_{a \sim \pi_{\theta_j}} [Q_{\phi_j}(s, a)]$$

Significantly improves overestimation and in turn sample efficiency!

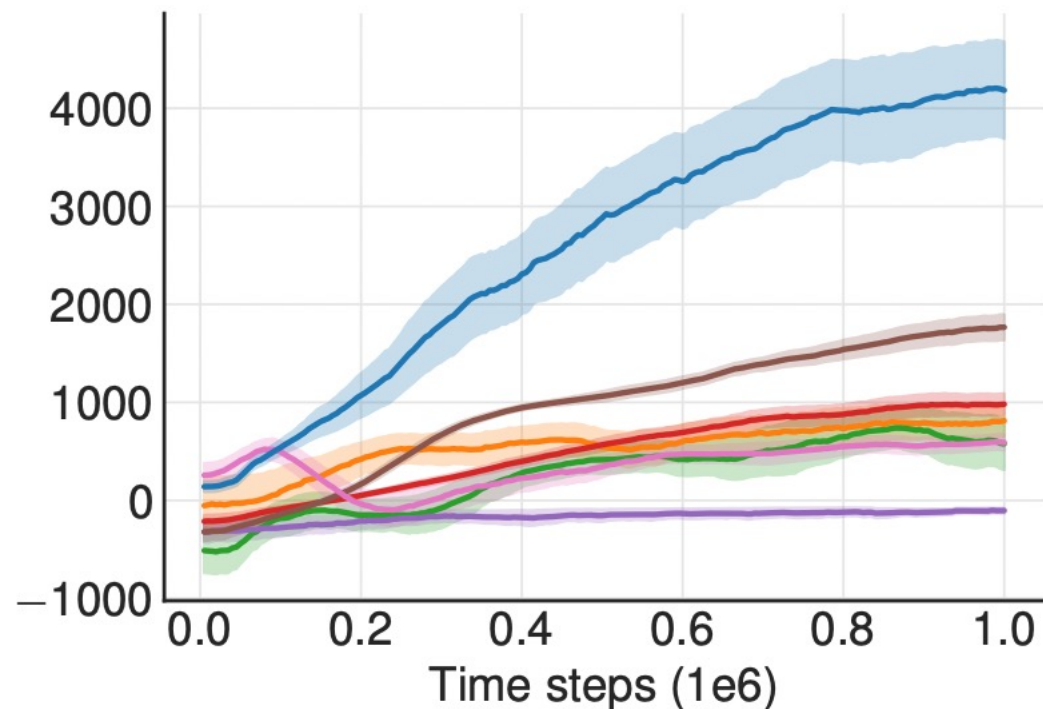
Overestimation Bias in Actor-Critic

Significantly improves overestimation and in turn sample efficiency!

■ TD3 ■ DDPG ■ our DDPG ■ PPO ■ TRPO ■ ACKTR ■ SAC



(c) Walker2d-v1



(d) Ant-v1

Double Actor Critic in Action



Double Actor Critic in Action



Where does this fail?



Some issues remain:

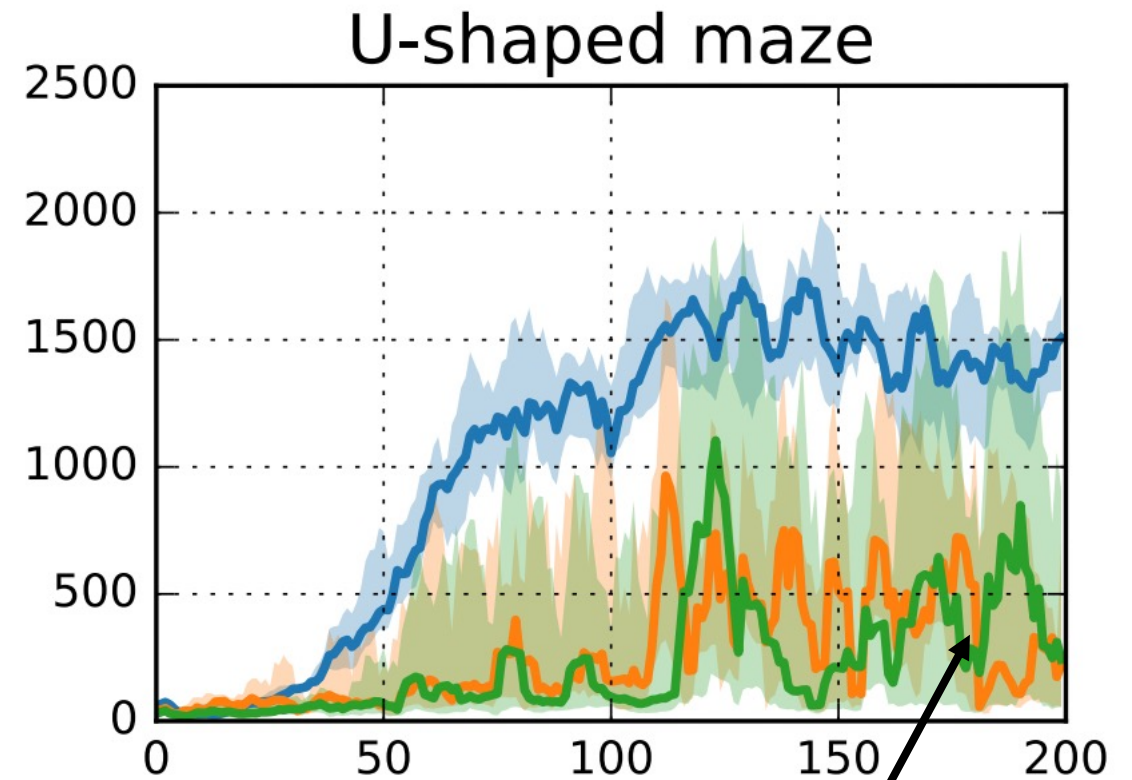
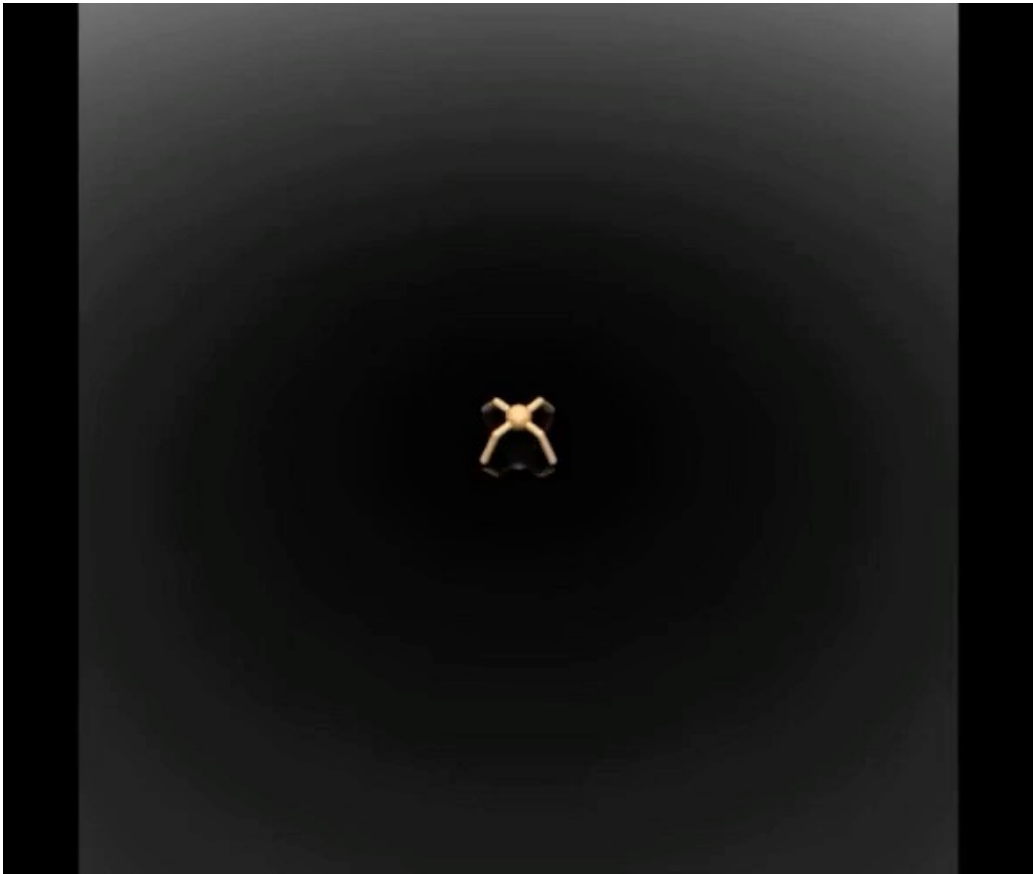
1. Overestimation bias

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

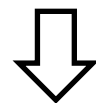


Very unstable learning

Addressing Policy Collapse in Off-Policy RL

Adding entropy to the RL objective can help significantly

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



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

Simple change in on-policy RL

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

Max-Ent Off-Policy RL

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

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

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

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

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

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

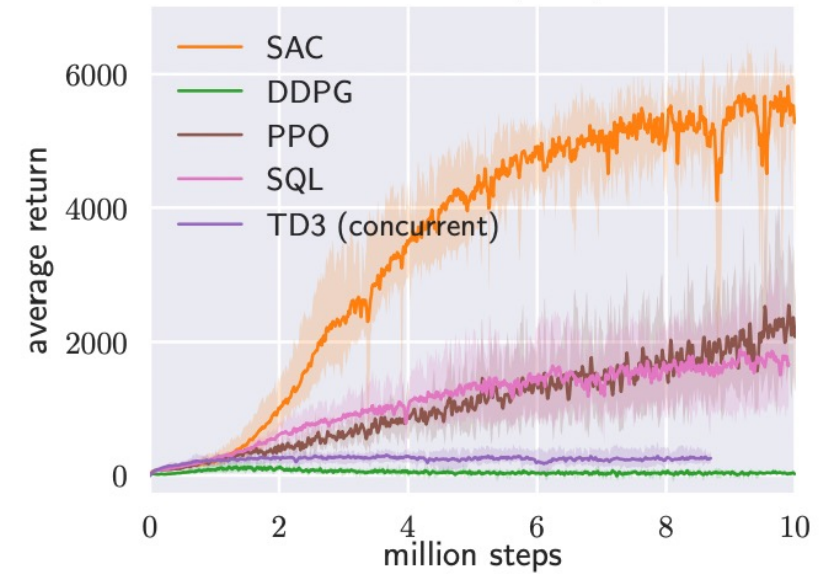
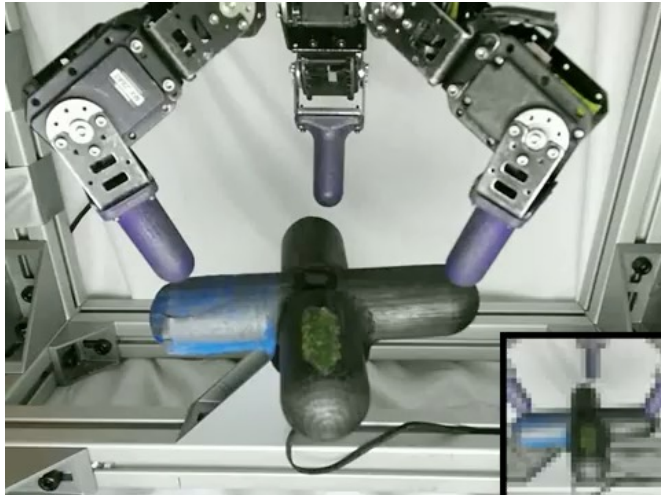
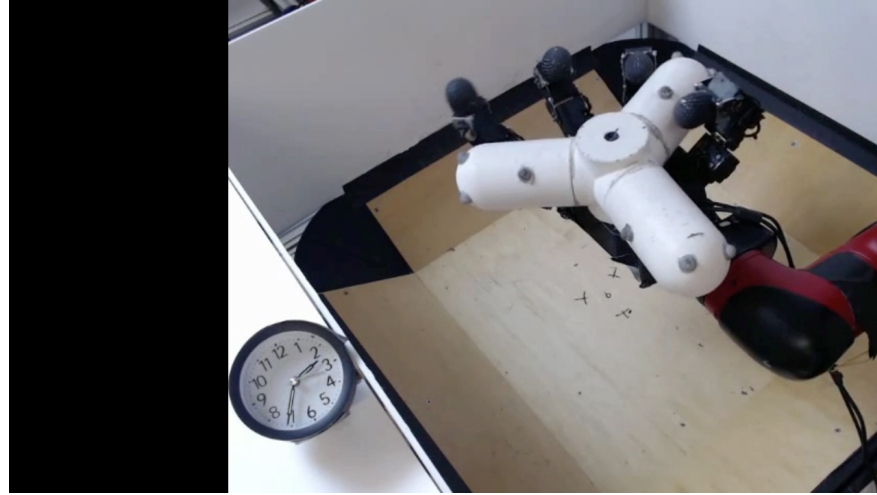
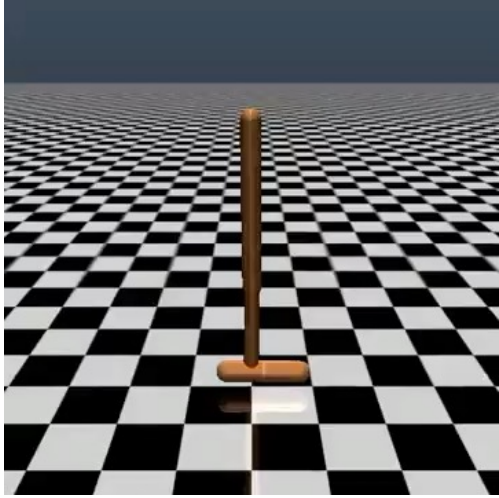
$$\pi(a|s_t) \leftarrow \arg \max_a Q(s_t, a)$$

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

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

Off-Policy RL Motivation



Getting Actor Critic to Work in Practice



A Closer Look at Convergence



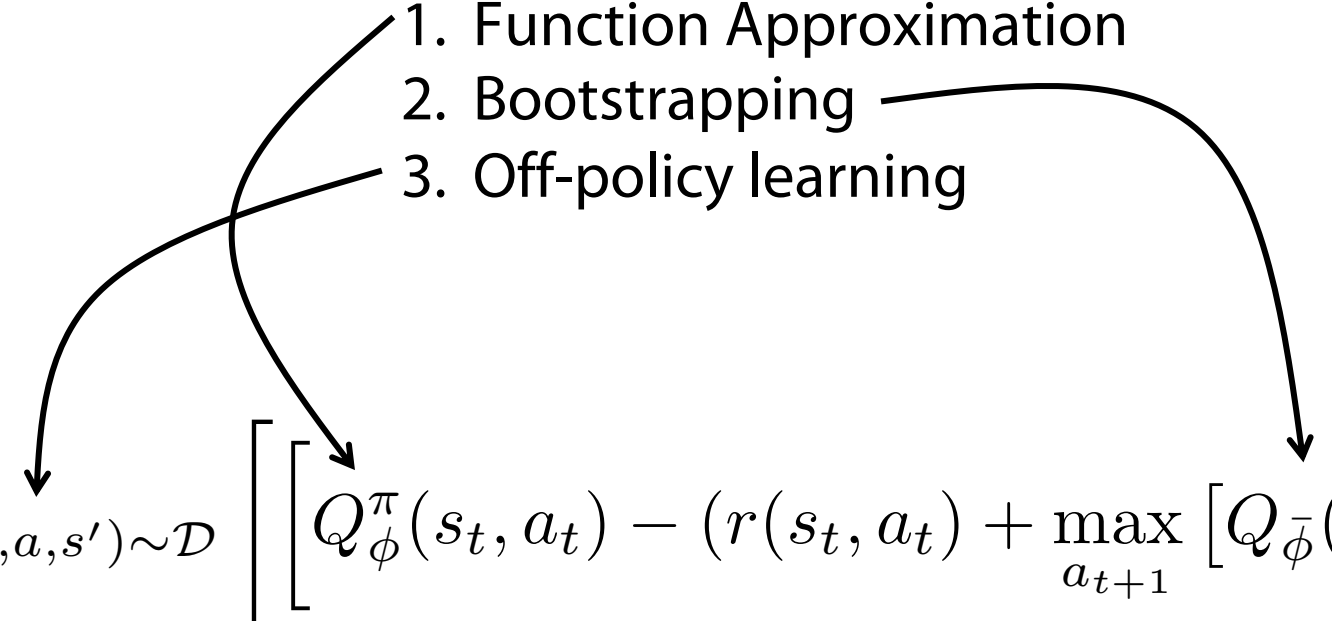
Frontiers of Off-Policy RL

Ok, so are off-policy algorithms perfect?

What makes off-policy RL hard?

Deadly triad:

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

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


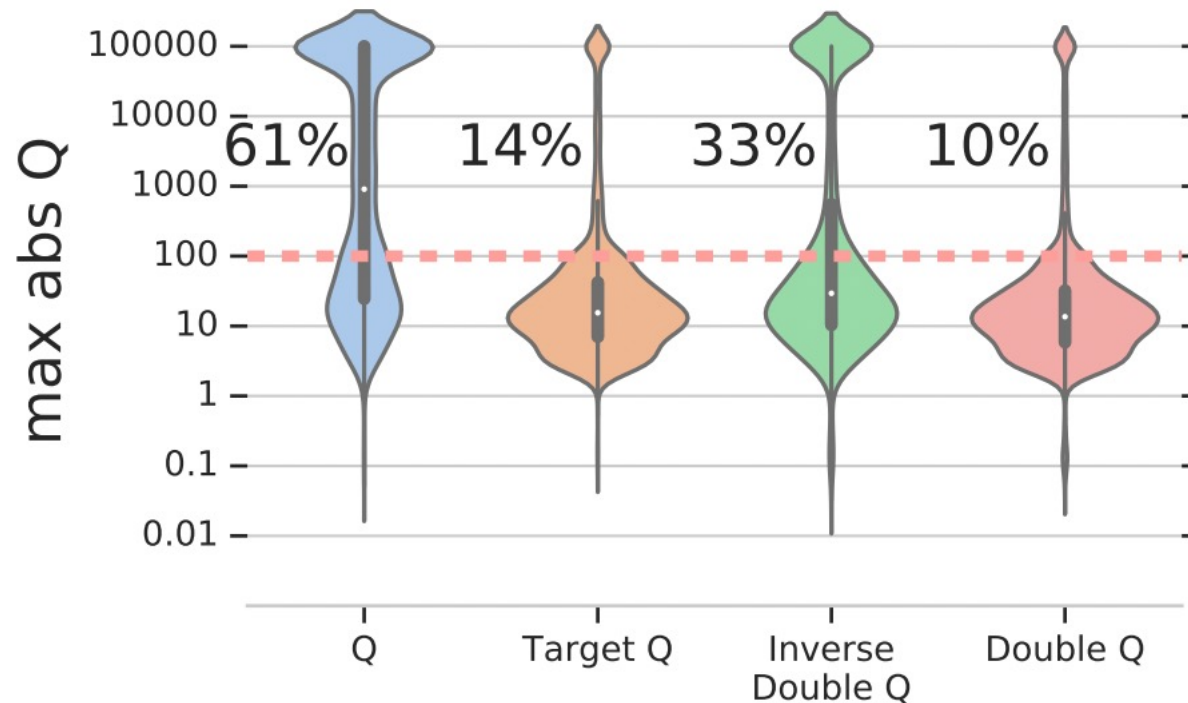
These in combination lead to many of the difficulties in stabilizing off-policy RL with function approximation

Zooming out – what makes off-policy RL hard?

Deadly triad:

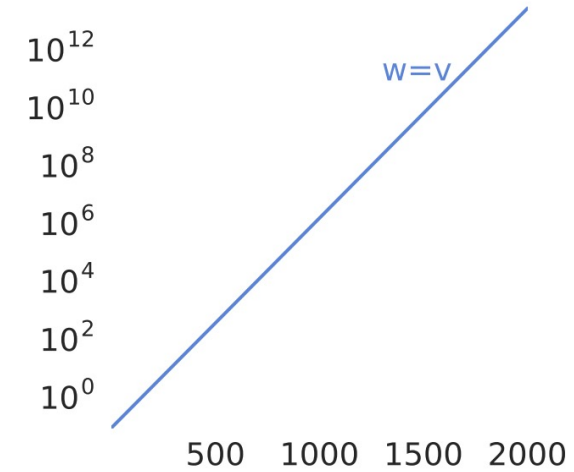
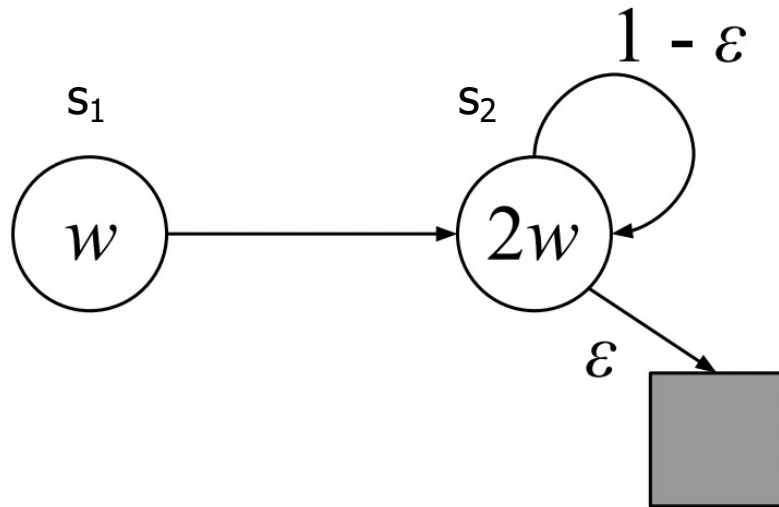
1. Function Approximation
2. Bootstrapping
3. Off-policy learning

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?

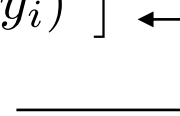


(b) $v(s) = w\phi(s)$ diverges.

Let's go to the whiteboard!

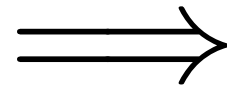
A Closer Look at the Value Bellman Equation

Bellman Update

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


Fixed-point iteration algorithm

$$x_{n+1} = f(x_n)$$



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

$$x^* = f(x^*)$$

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?

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

Banachs fixed point theorem

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

Let $f : M \rightarrow M$ be a **contraction**.

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

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

Then there exists a unique **fixed point** of f .

Let's consider a simple version of this algorithm

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

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



Prove this is a contraction

Does this converge?

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

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

Bellman operator



To prove:

$$d(f(x), f(y)) \leq \gamma d(x, y)$$



inf-norm

Value functions

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

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

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

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

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

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

$$= \gamma \max_s \left| \left(\int \pi(a|s) \max_x (U_i(x) - V_i(x)) da \right) \right|$$

$$= \gamma \max_x |U_i(x) - V_i(x)| = \gamma |U_i - V_i|_\infty \quad \text{Contraction, hence converges to a fixed point}$$

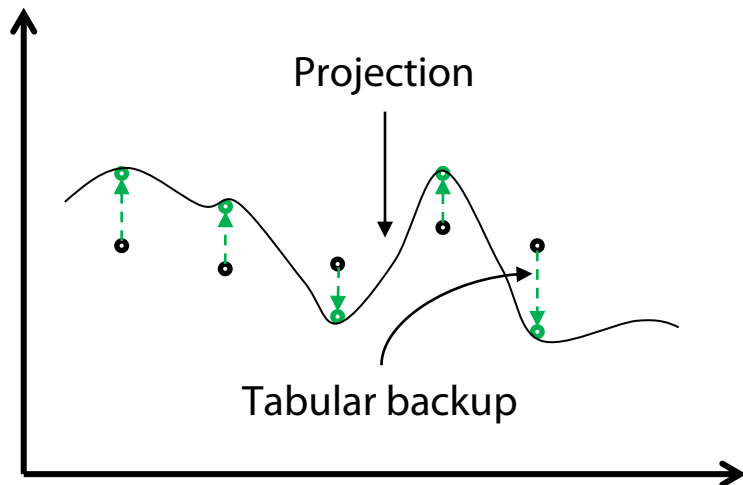
Does this converge for arbitrary function approximation?

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

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

We perform a Bellman backup + a projection

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



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

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

Lecture outline

Off-Policy RL Motivation



Getting Actor Critic to Work in Practice



A Closer Look at Convergence



Frontiers of Off-Policy RL

What should I work on?

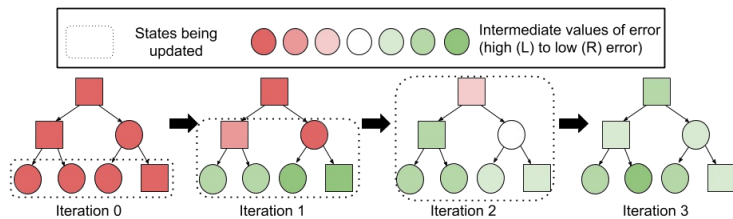
Where does the frontier of off-policy RL lie?

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

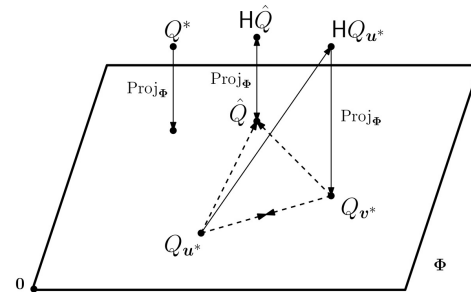
- Low variance, off-policy, avoids reconstruction, performs dynamic programming
- Has the potential to be **performant** and **sample efficient**

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

Sampling



Theory



Exploration

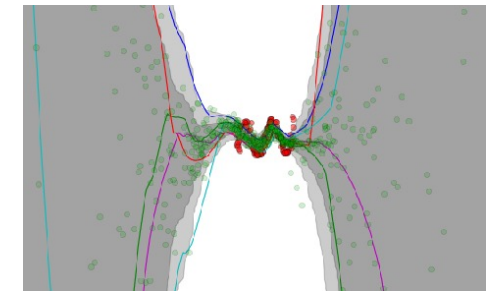
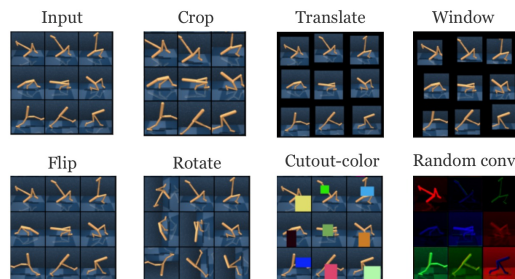


Image-based RL

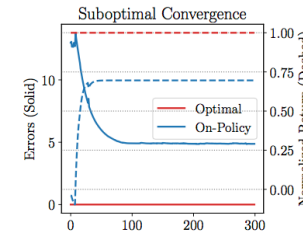
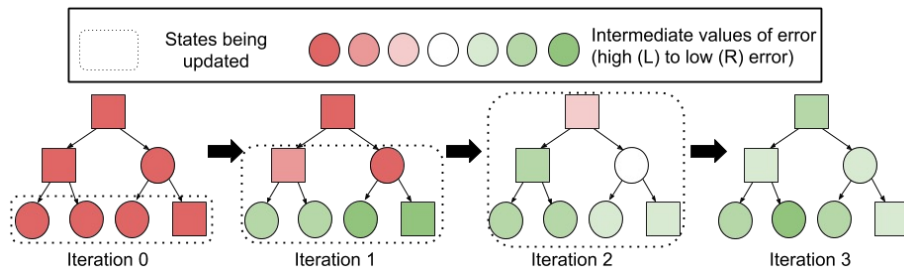
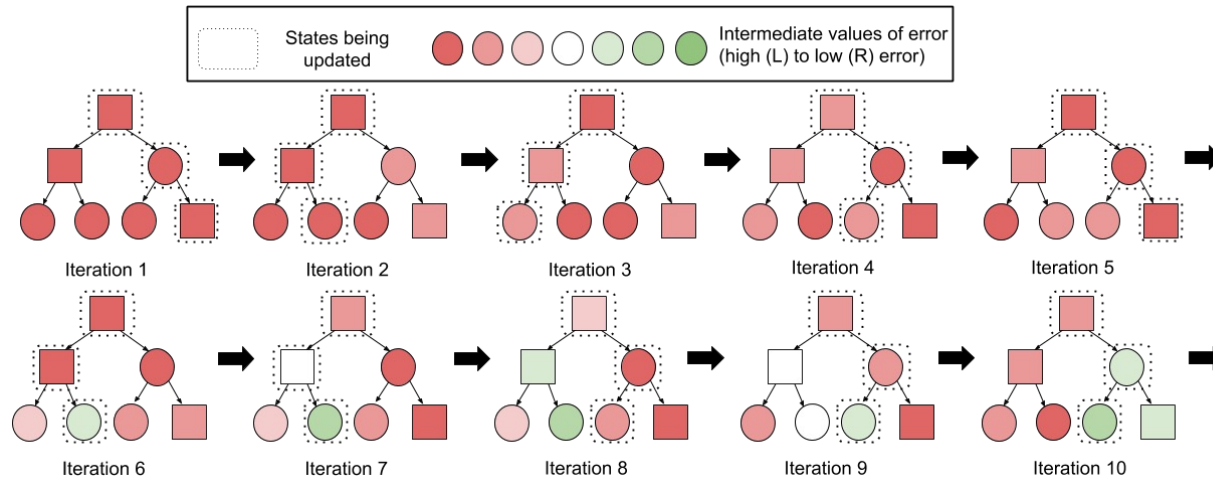


Partial Observability

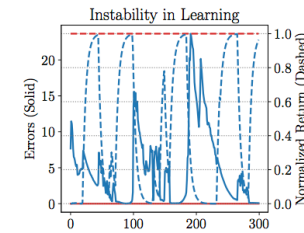


Prioritizing Experience

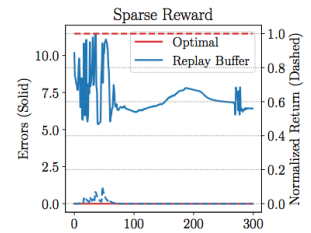
Performing uniform buffer TD updates can be catastrophically bad



(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

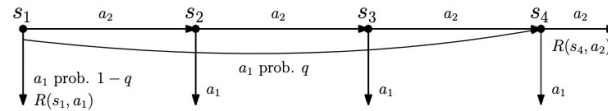


Figure 1: A simple MDP that illustrates delusional bias (see text for details).

Implicit regularization

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

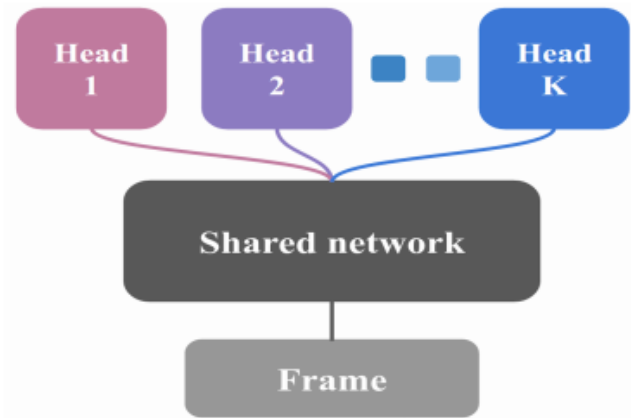
Bilinear classes

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

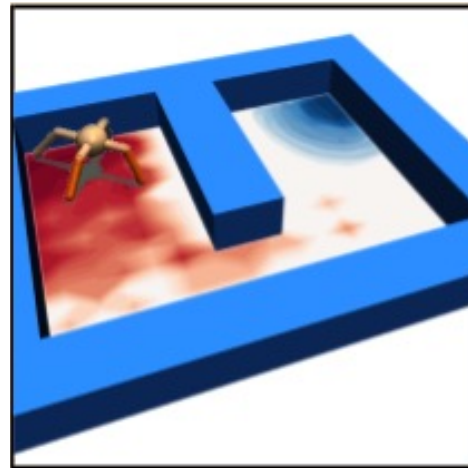
Exploration in Off-Policy RL

Better exploration methods

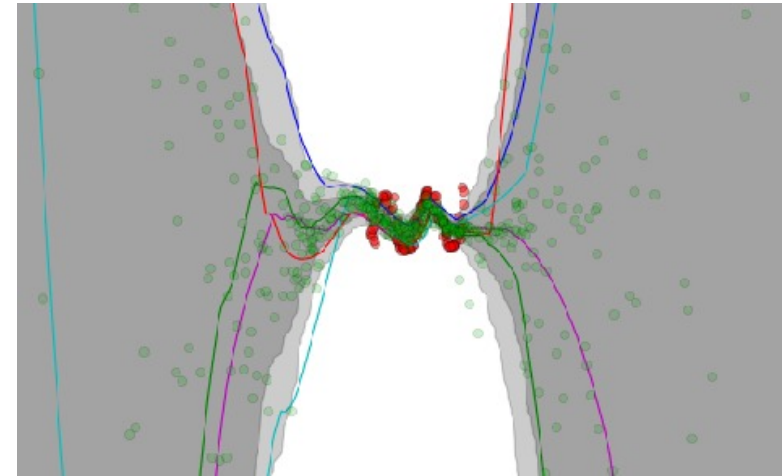
Uncertainty based methods



Count-based methods



Information gain methods

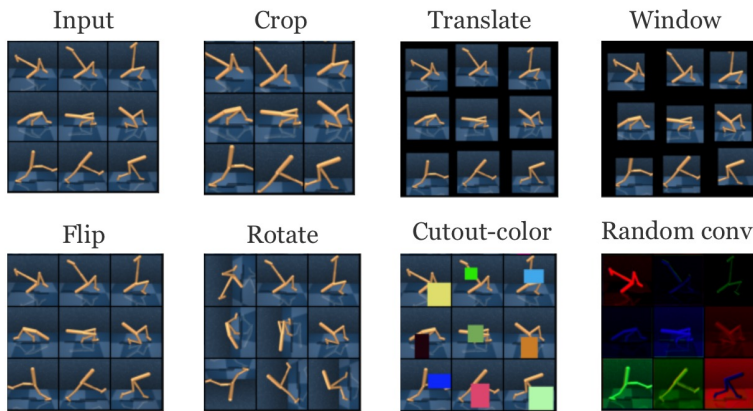


Often critical for getting algorithms to work!

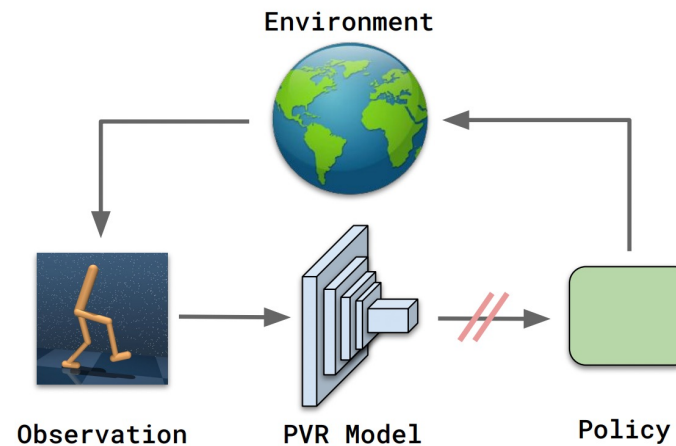
Image-based Off-Policy RL

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

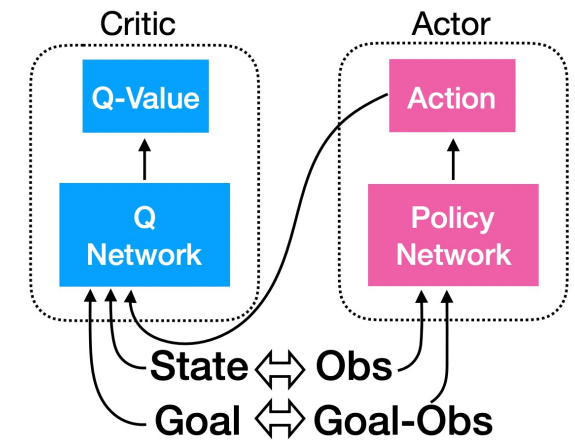
Data augmentations



Pre-trained representations



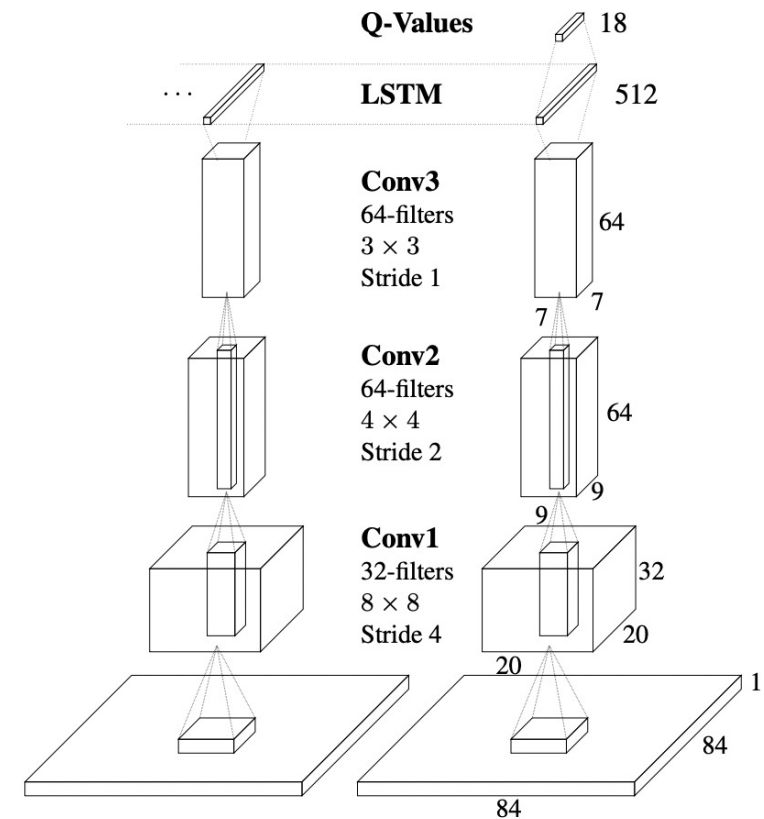
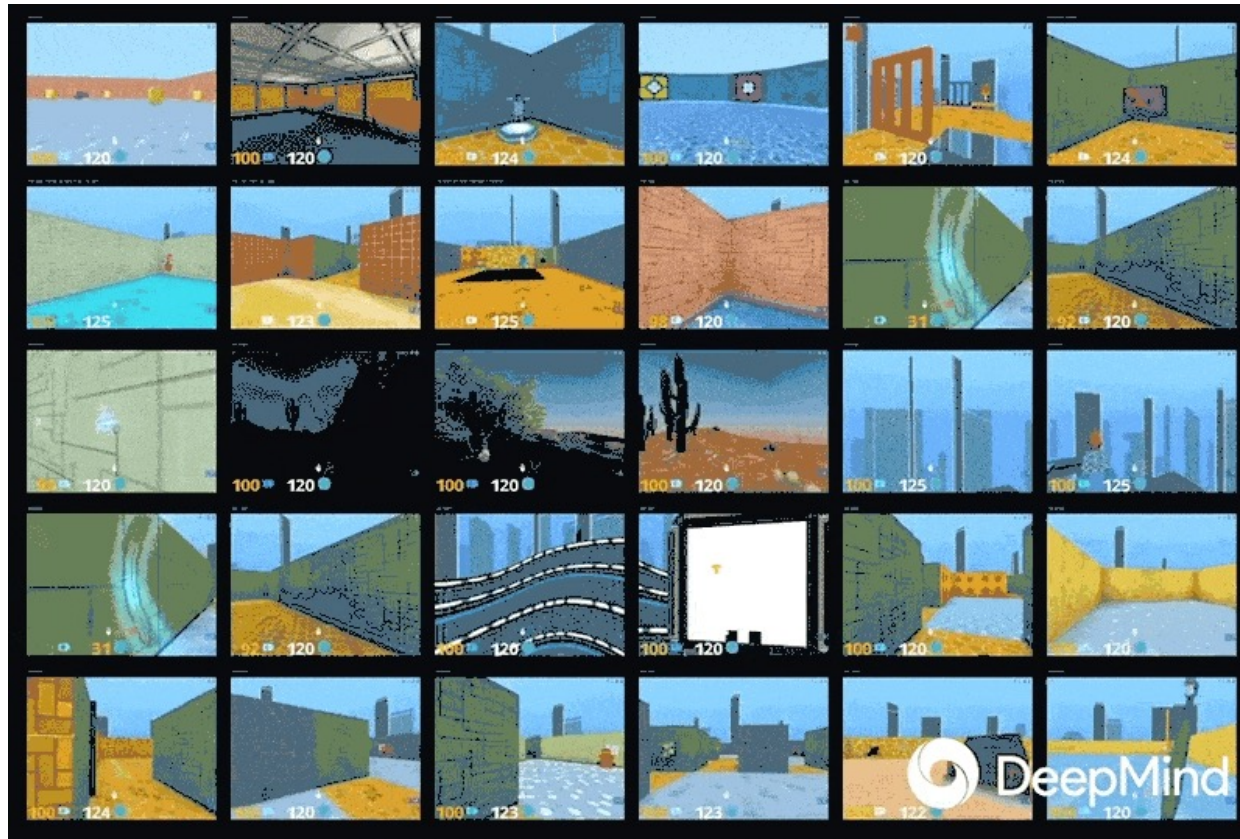
Student-teacher



Still very unstable, lot of open research problems!

Partial Observability in Off-Policy RL

Off-policy methods critically depend on the Markov assumption



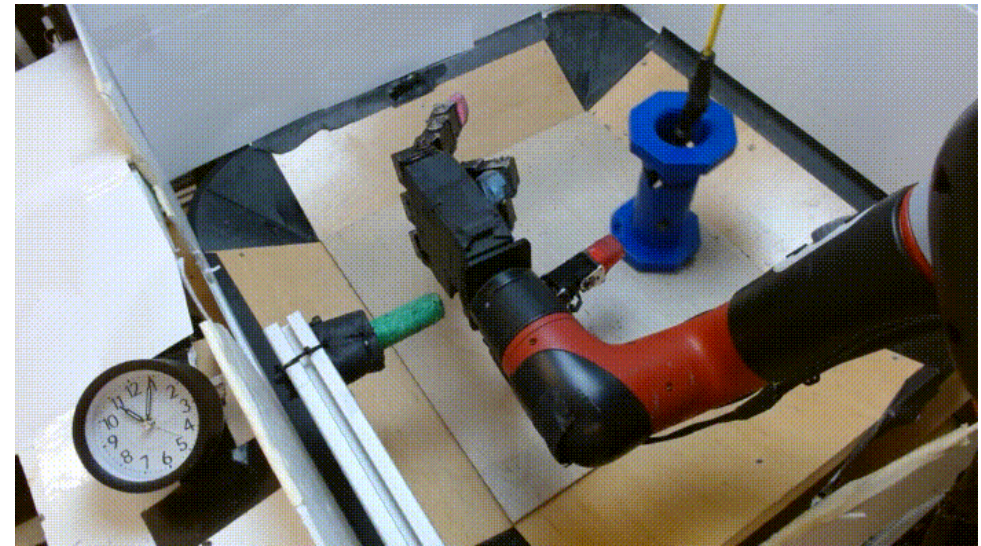
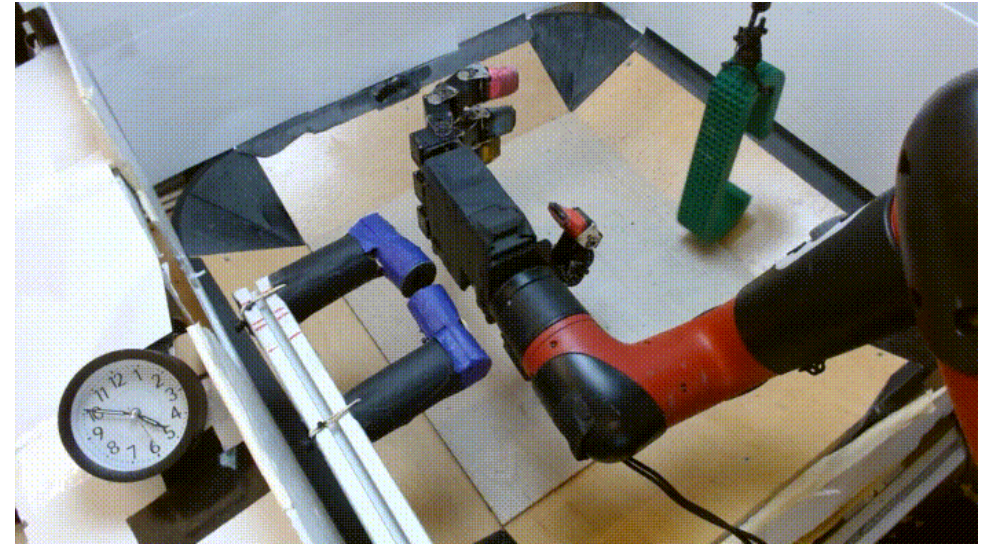
Learning history conditioned/recurrent Q-functions is an open area!

How has off-policy RL manifested in robotics?

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

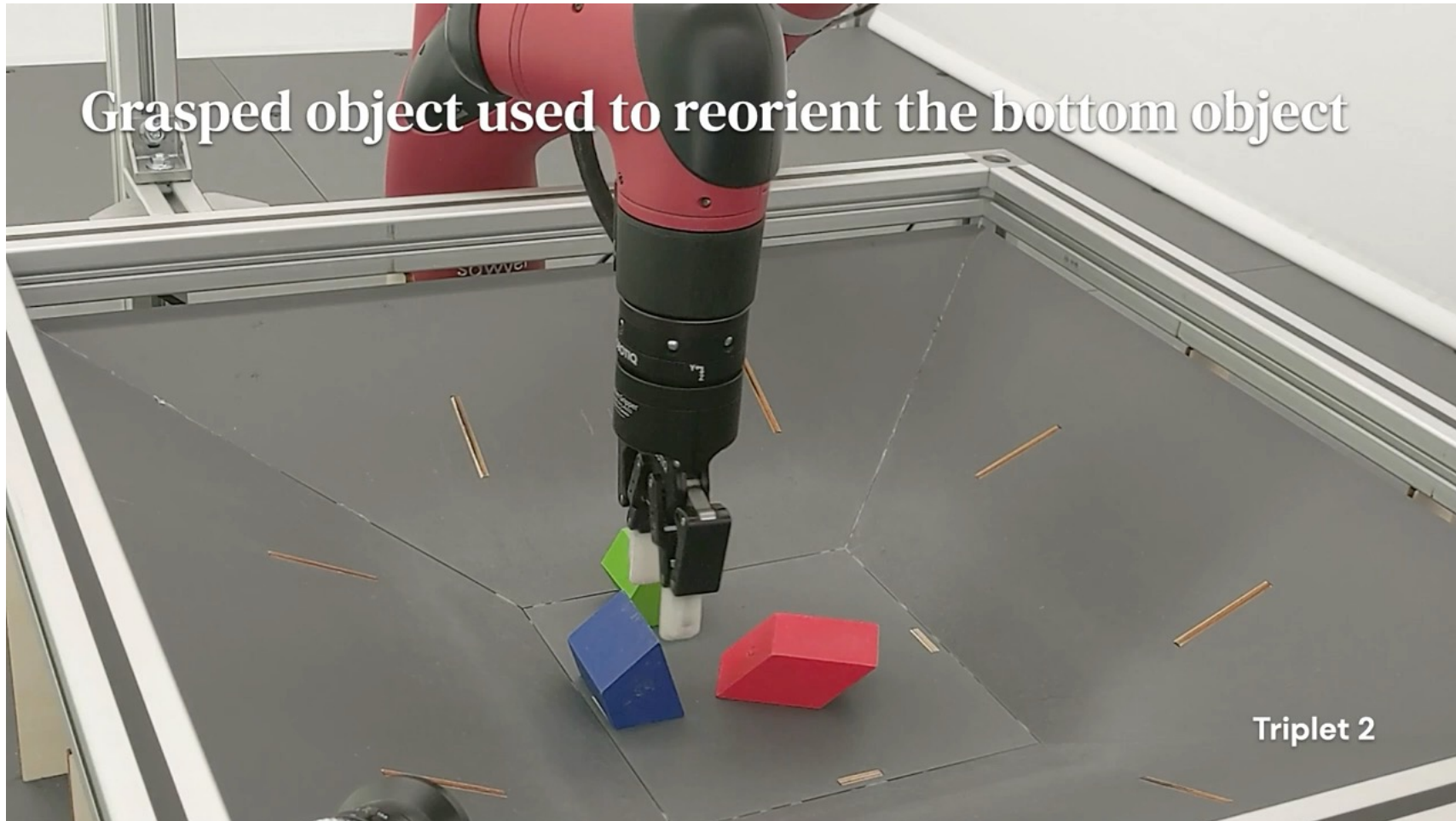


How has off-policy RL manifested in robotics?



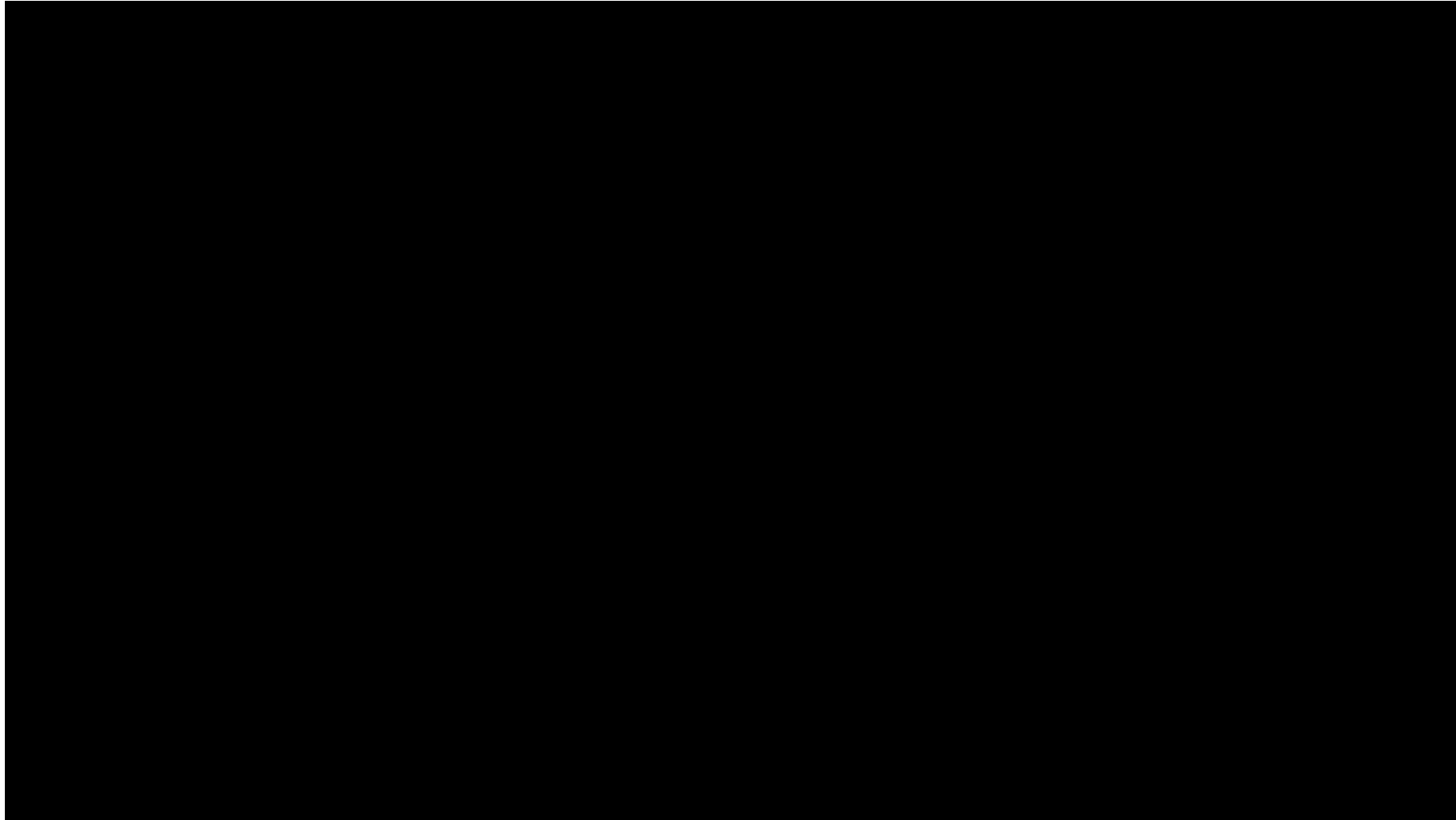
How has off-policy RL manifested in robotics?

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

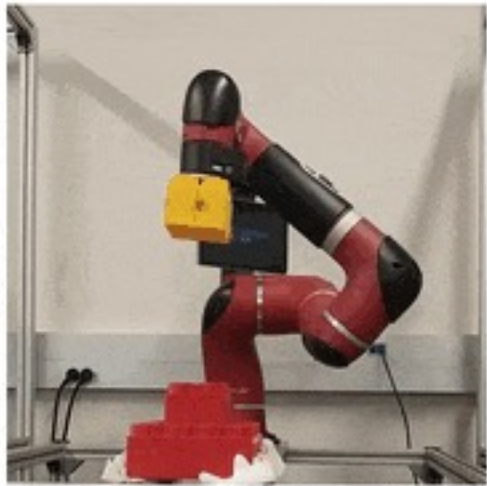


How has off-policy RL manifested in robotics?

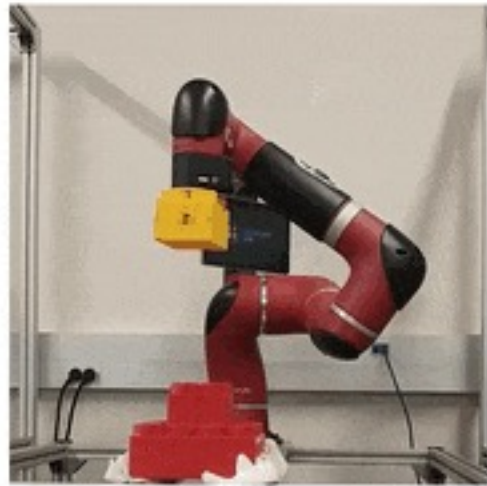
Bootstrapped with a few demonstrations



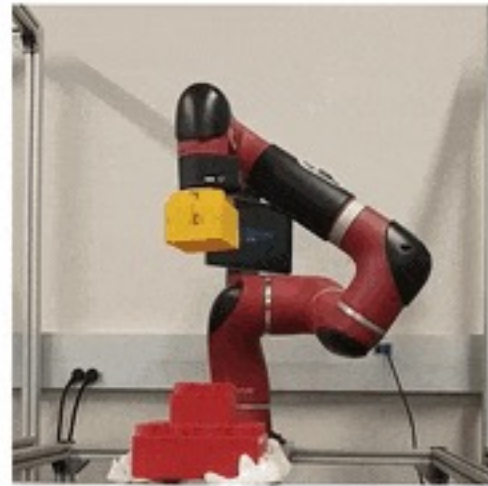
How has off-policy RL manifested in robotics?



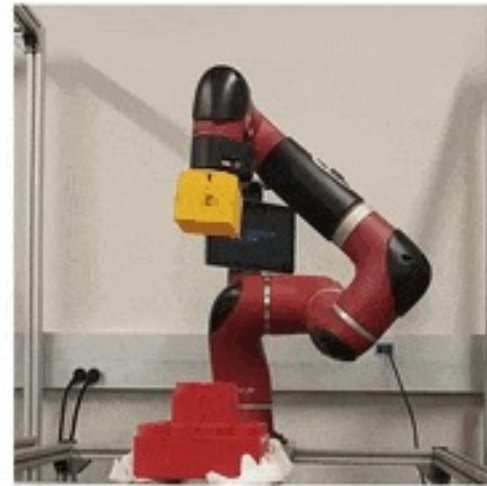
untrained



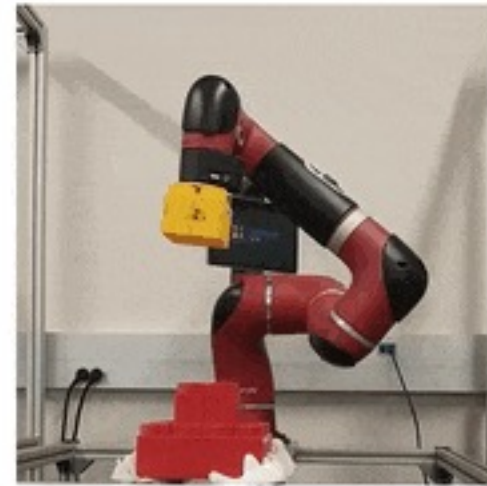
12 min later



30 min later



1 hour later



2 hours later

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

Off-Policy RL Motivation



Getting Actor Critic to Work in Practice



A Closer Look at Convergence



Frontiers of Off-Policy RL