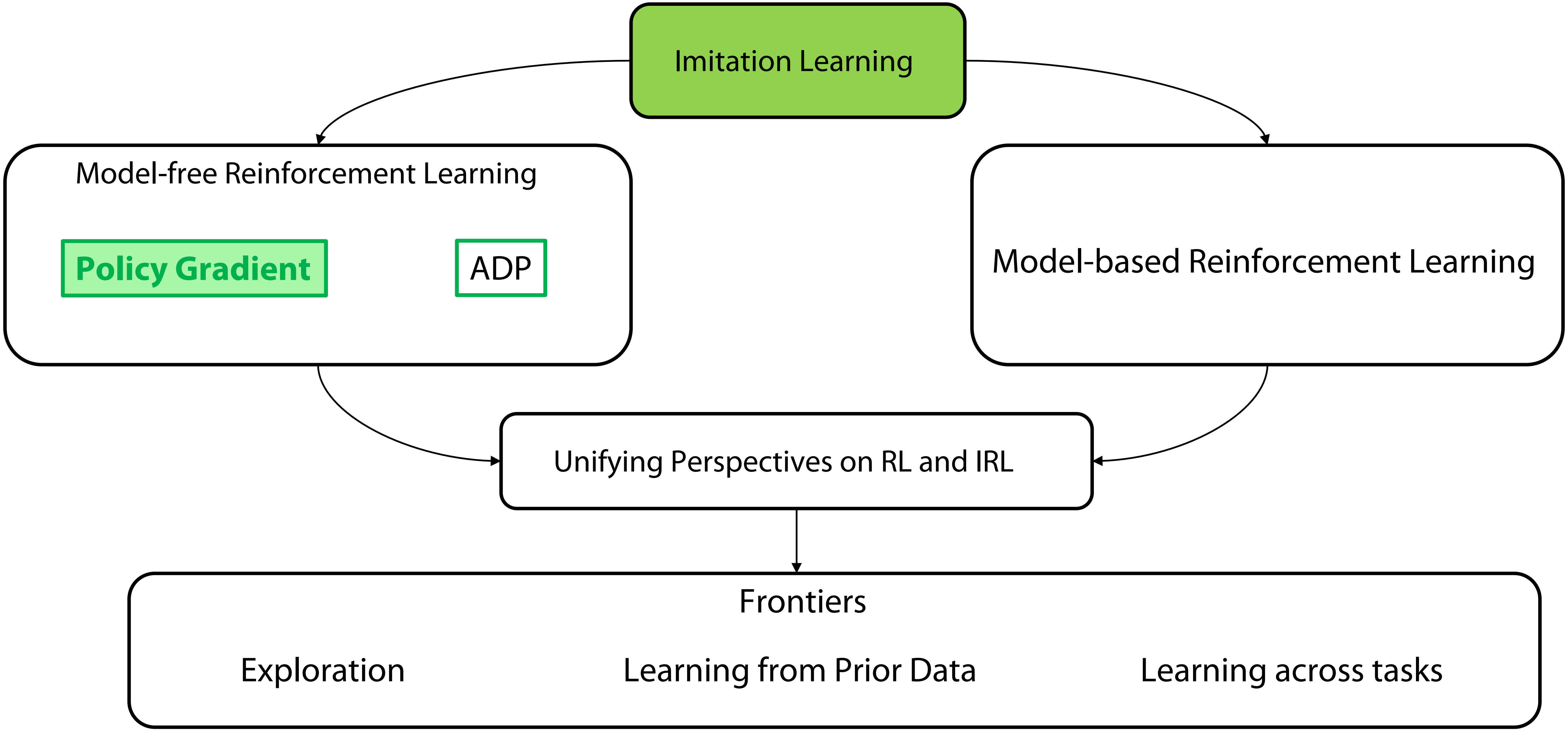
Reinforcement Learning
Spring 2024

Abhishek Gupta

TAs: Patrick Yin, Qiuyu Chen

Class Structure



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

Does this converge?

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

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

Bellman operator

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

inf-norm

Value functions

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

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

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

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

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

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

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

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

Contraction, hence converges to a fixed point

Does this converge for arbitrary function approximation?

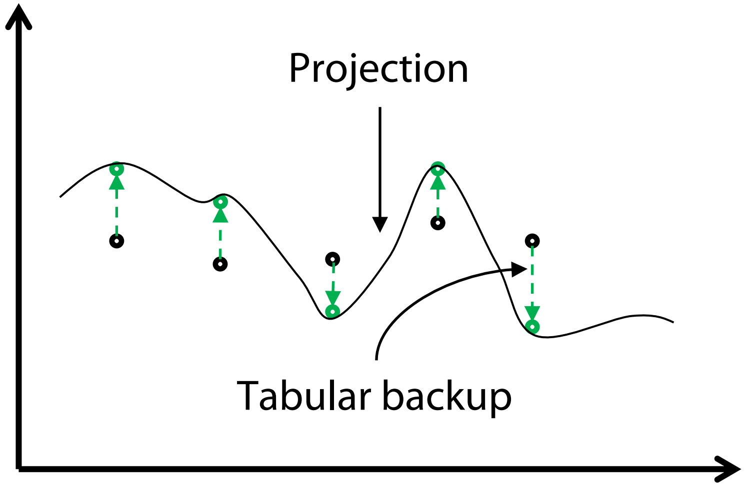
For arbitrary function approximation, it is not just a Bellman backup

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

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

We perform a Bellman backup + a projection

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



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

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

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

Attempt 2: Recursive structure in Q functions directly

Q functions have special recursive structure themselves!

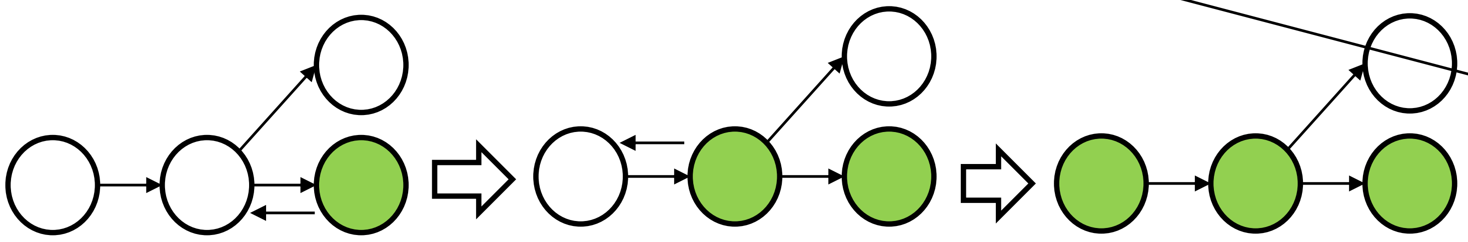
$$Q^\pi(s_t, a_t) = \mathbb{E}_{\pi_\theta}\left[\sum_{t'=t}^{T} r(s'_t, a'_t) | s_t, a_t\right]$$

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

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!

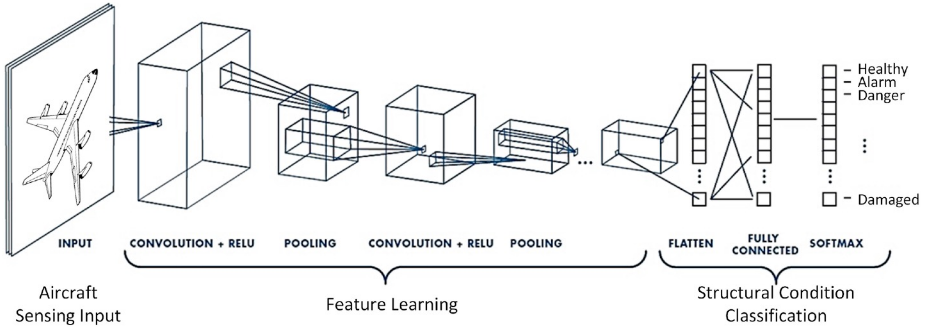
Learning Q-functions via Dynamic Programming

Policy Evaluation: Try to minimize Bellman Error (almost)

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

Same function approximator

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



Lecture outline

Working through a complete off-policy algorithm

↓

Getting Off-Policy RL to Work

↓

Frontiers of Off-Policy RL

↓

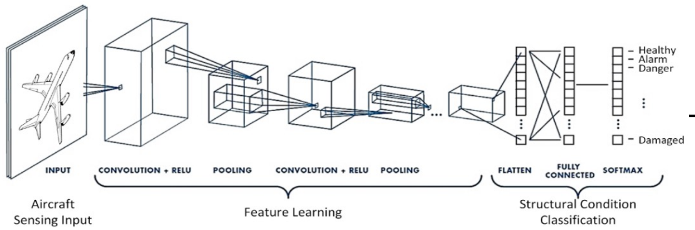
Model-Based RL - Formulation

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

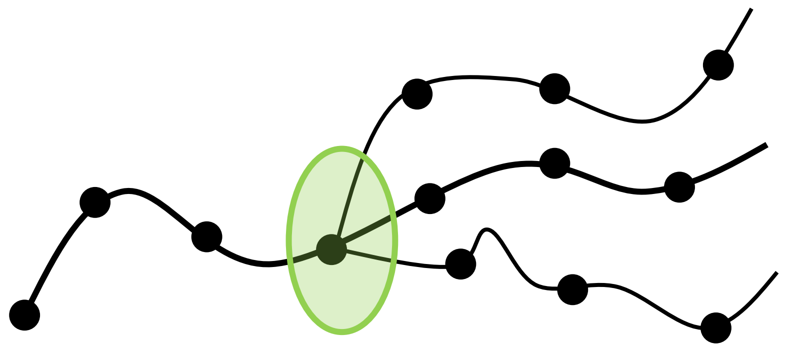
<u>Policy Improvement:</u> Improve policy with **<u>policy gradient</u>**

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

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>

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

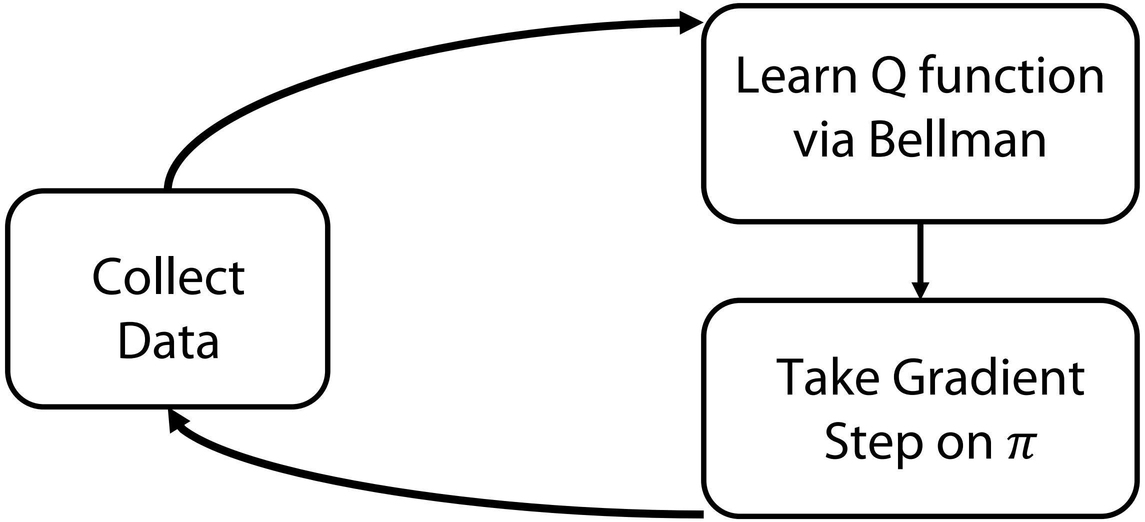
Lower variance (empirically)

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)

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

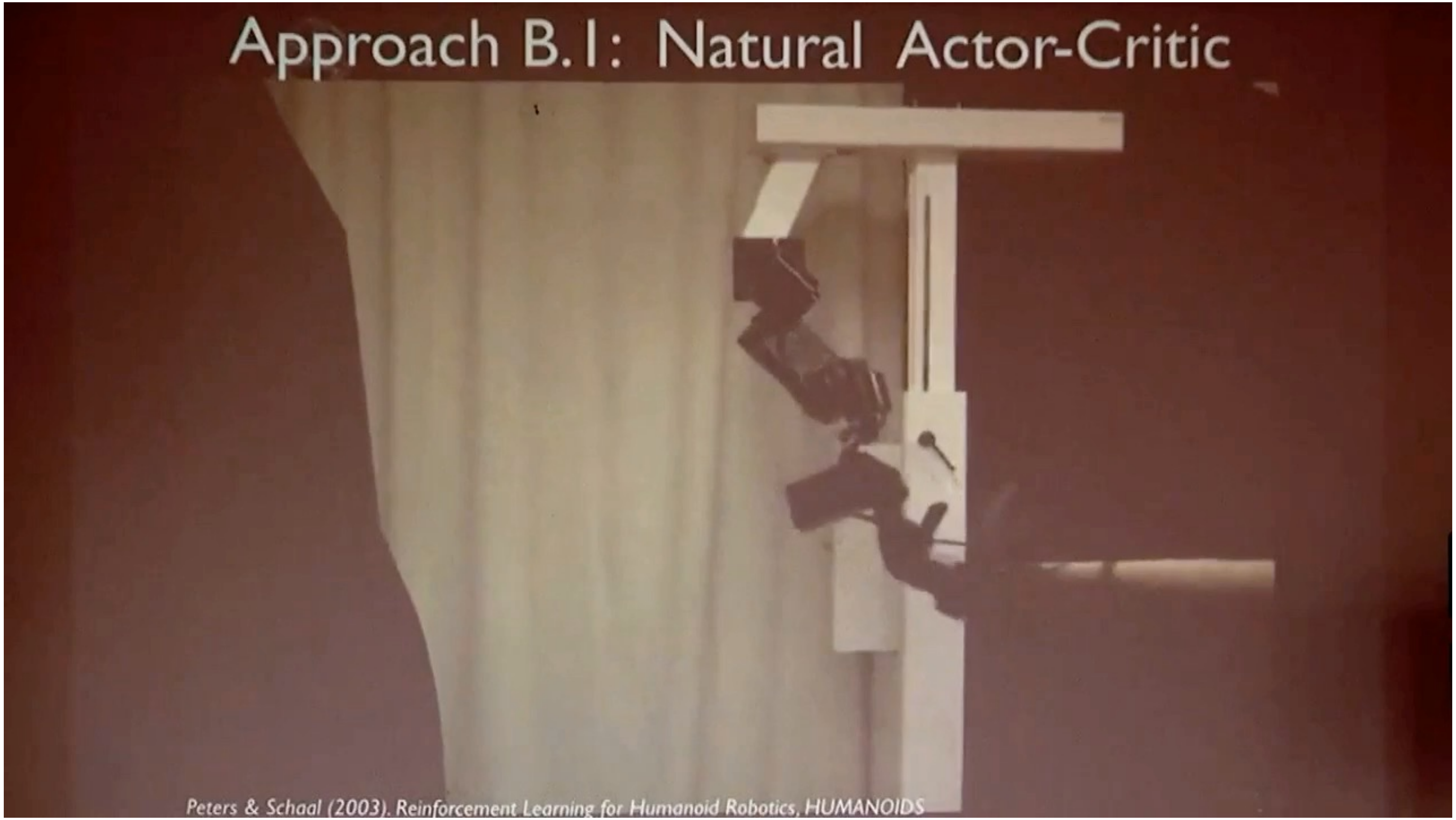


Lowers variance and is off-policy!

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

Working through a complete off-policy algorithm

↓

Getting Off-Policy RL to Work

↓

Frontiers of Off-Policy RL

↓

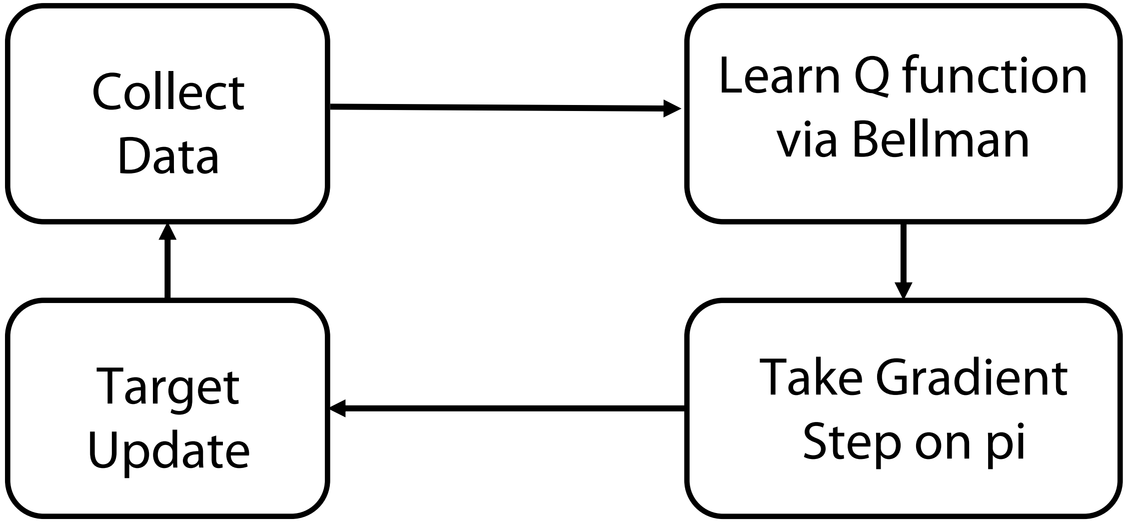
Model-Based RL - Formulation

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



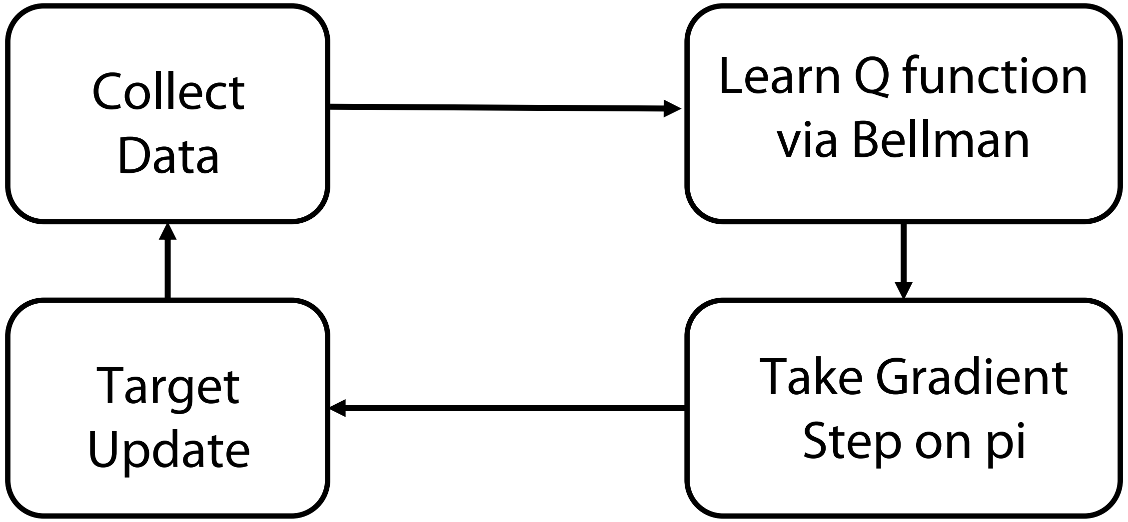
Critic: 1 gradient step on Bellman error

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

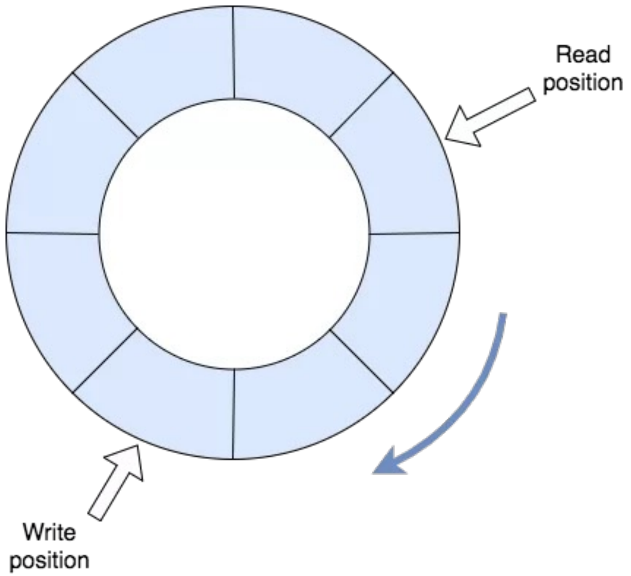
Actor: 1 gradient step on policy gradient

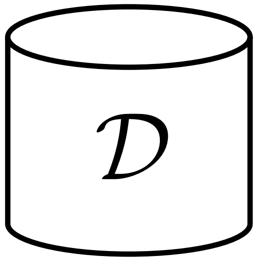
When updates are performed online, two issues persist:

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

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

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

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

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

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

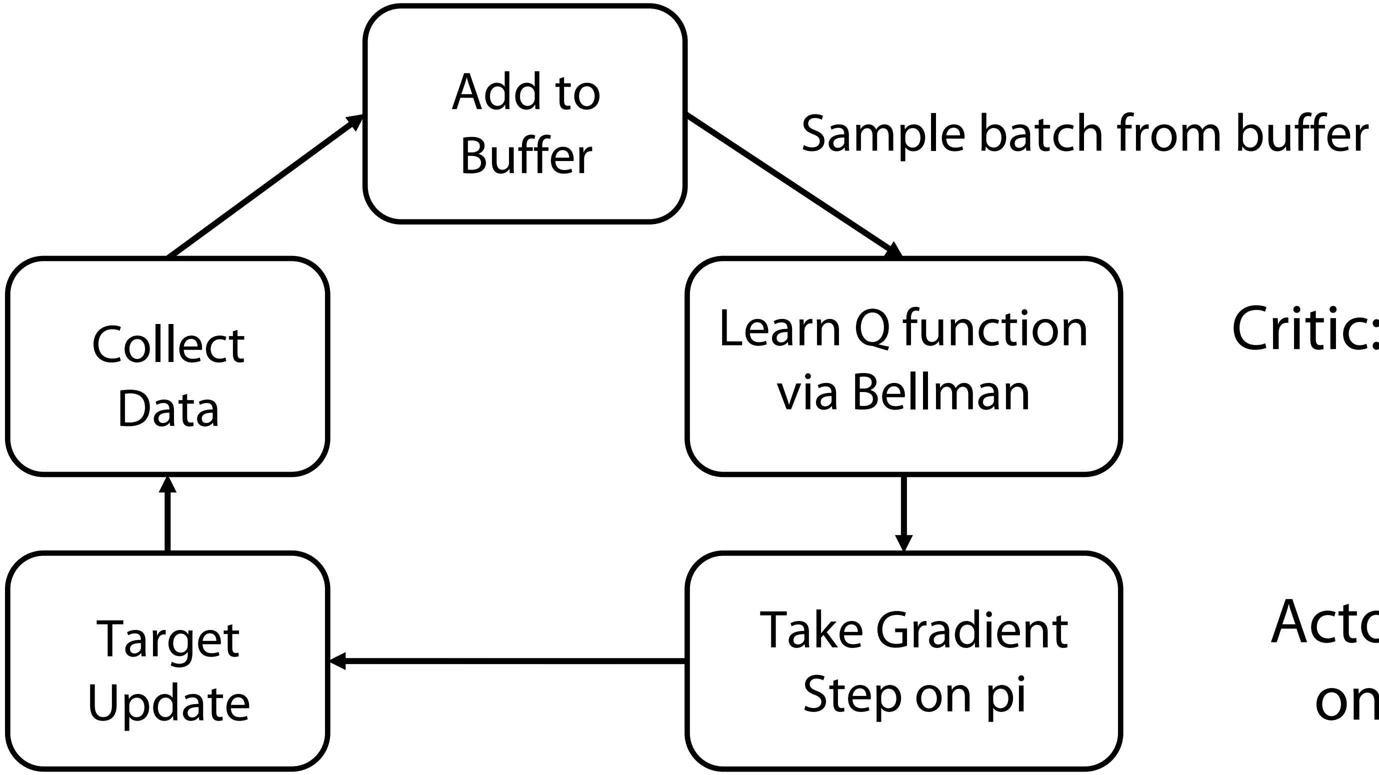
⇩

Move $\bar{\phi}$ to ϕ slowly!

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

Polyak averaging

A Practical Off-Policy RL Algorithm



1 sample

Critic: 1 gradient step on Bellman error

Polyak Averaging

Actor: 1 gradient step on policy gradient

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

Critic Update

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

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*

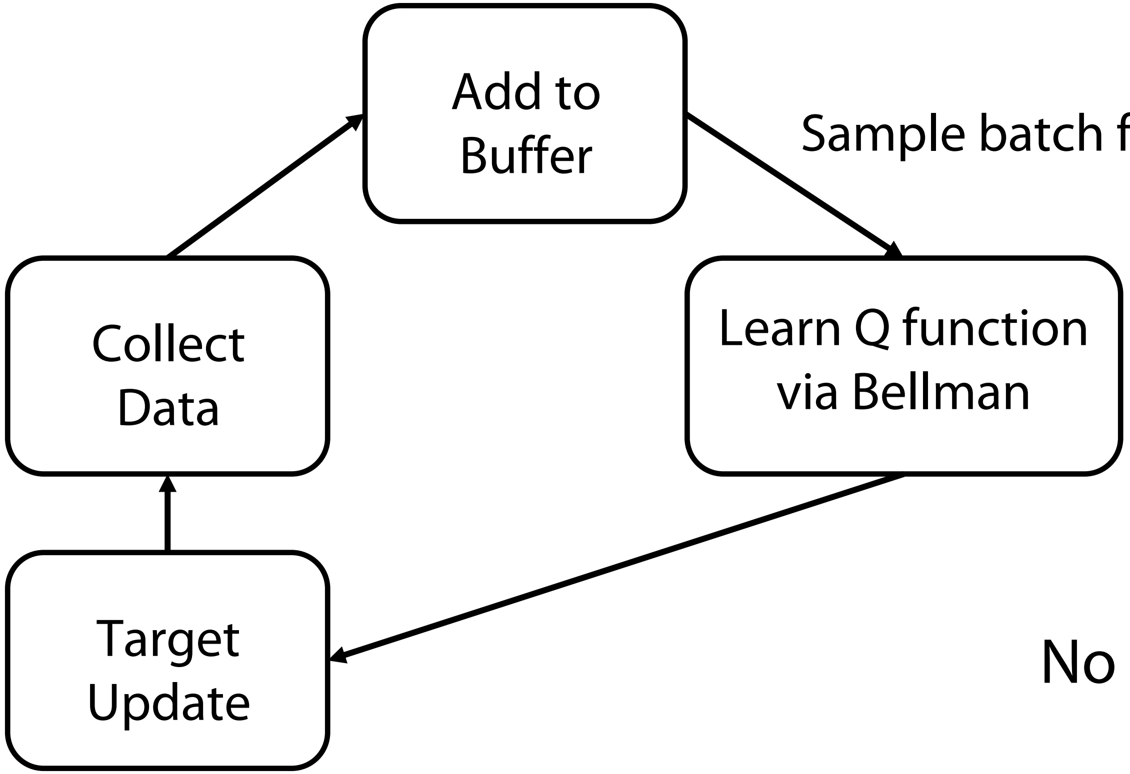
$$\min_{\phi} \mathbb{E}_{(s,a,s')\sim\mathcal{D}} \left[\left[Q_{\phi}^{\pi}(s_t, a_t) - (r(s_t, a_t) + \max_{a_{t+1}} \left[Q_{\bar{\phi}}(s_{t+1}, a_{t+1}) \right]) \right]^2 \right]$$

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

Directly do max in the Bellman update

1 sample

Polyak Averaging



Sample batch from buffer

Critic: 1 gradient step on Bellman error

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!

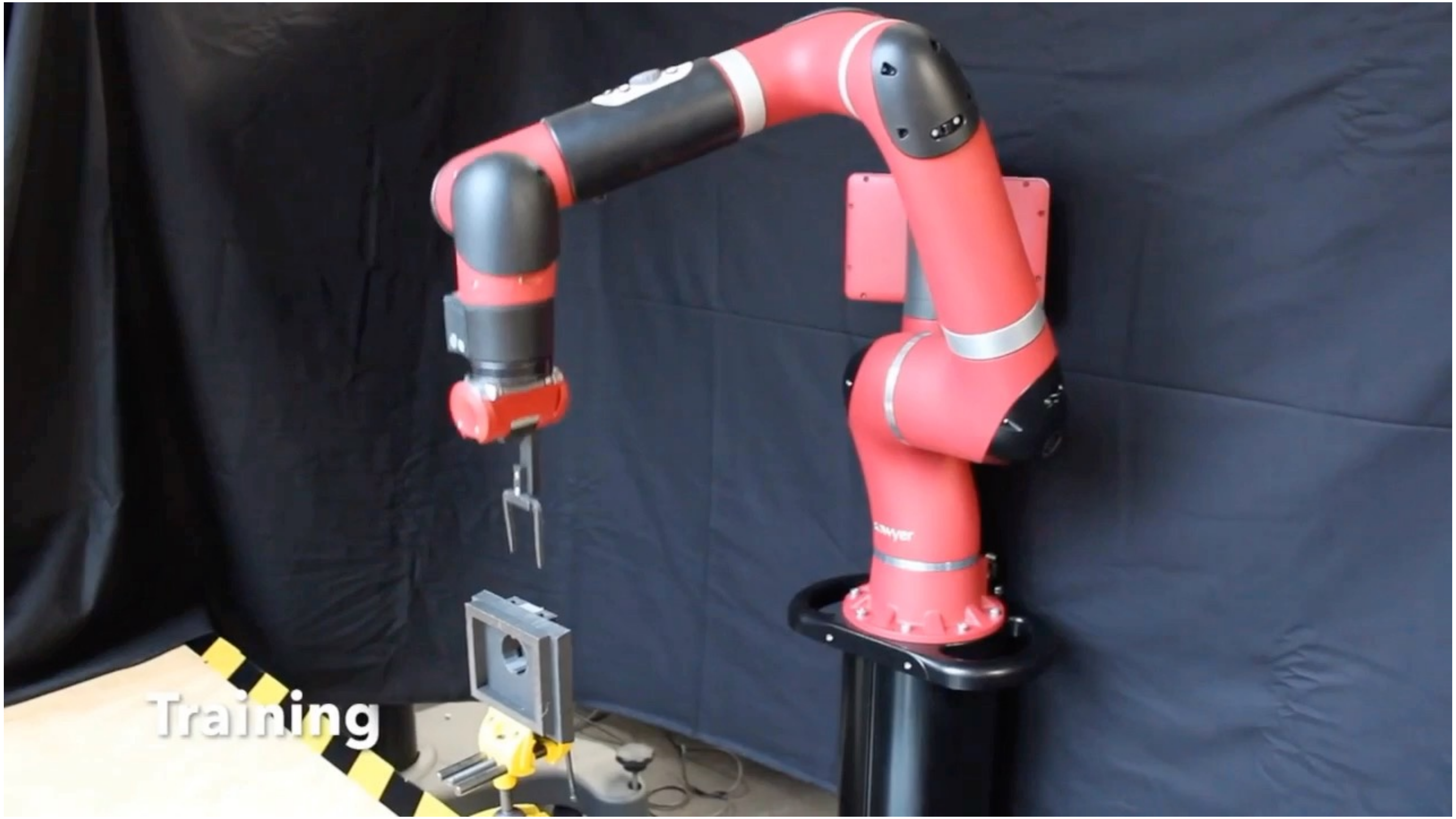
Some ideas to do maximization:

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

Practical Actor-Critic in Action

Trained using QT-Opt

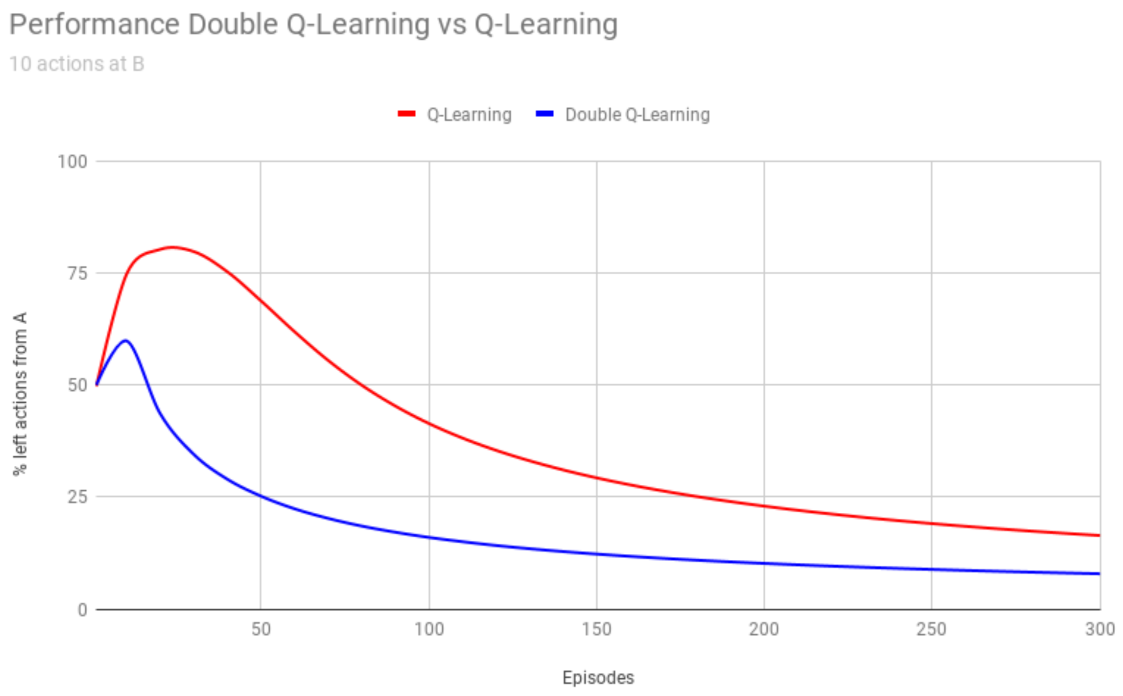
Practical Actor-Critic in Action



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!

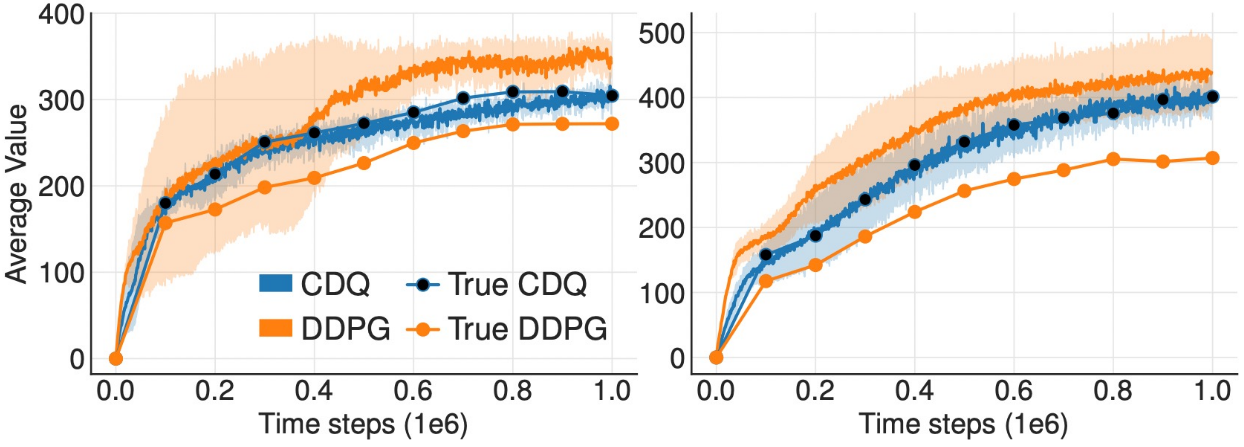
Overestimation Bias in Actor-Critic

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

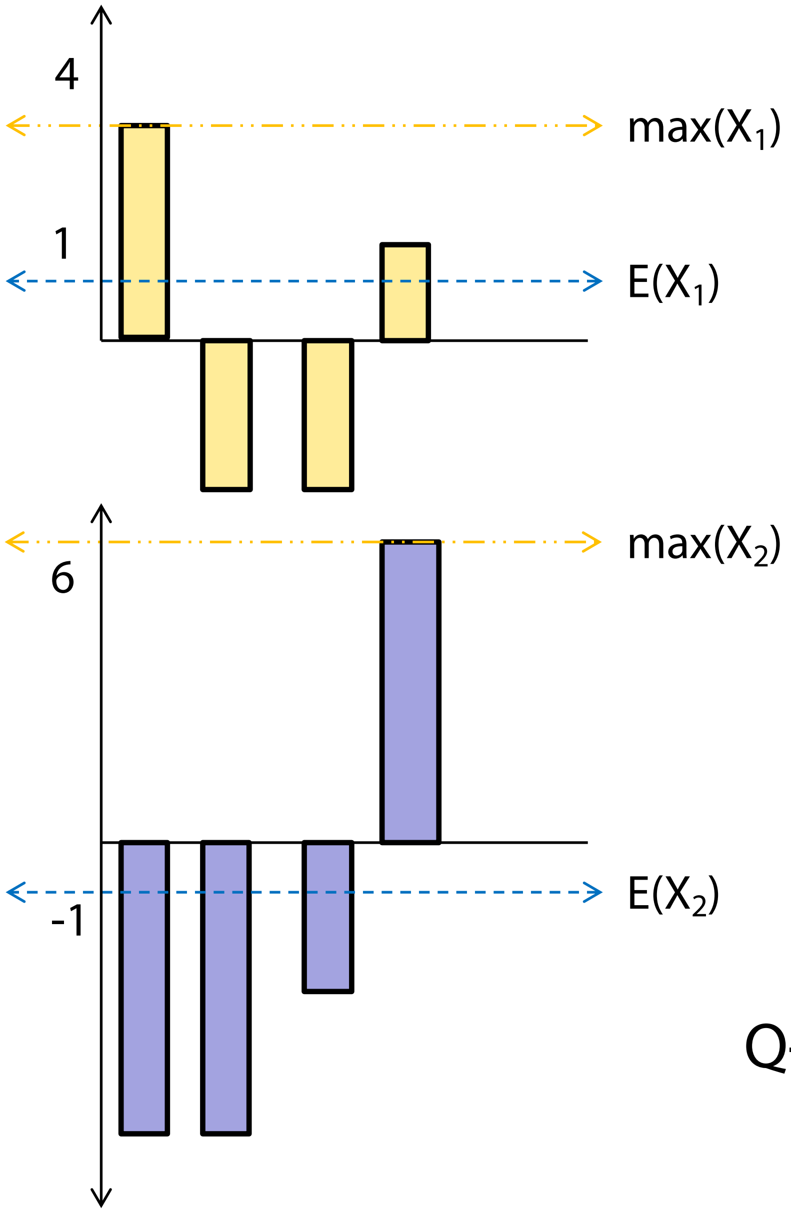
→ actually a random variable because of limited data/stochasticity

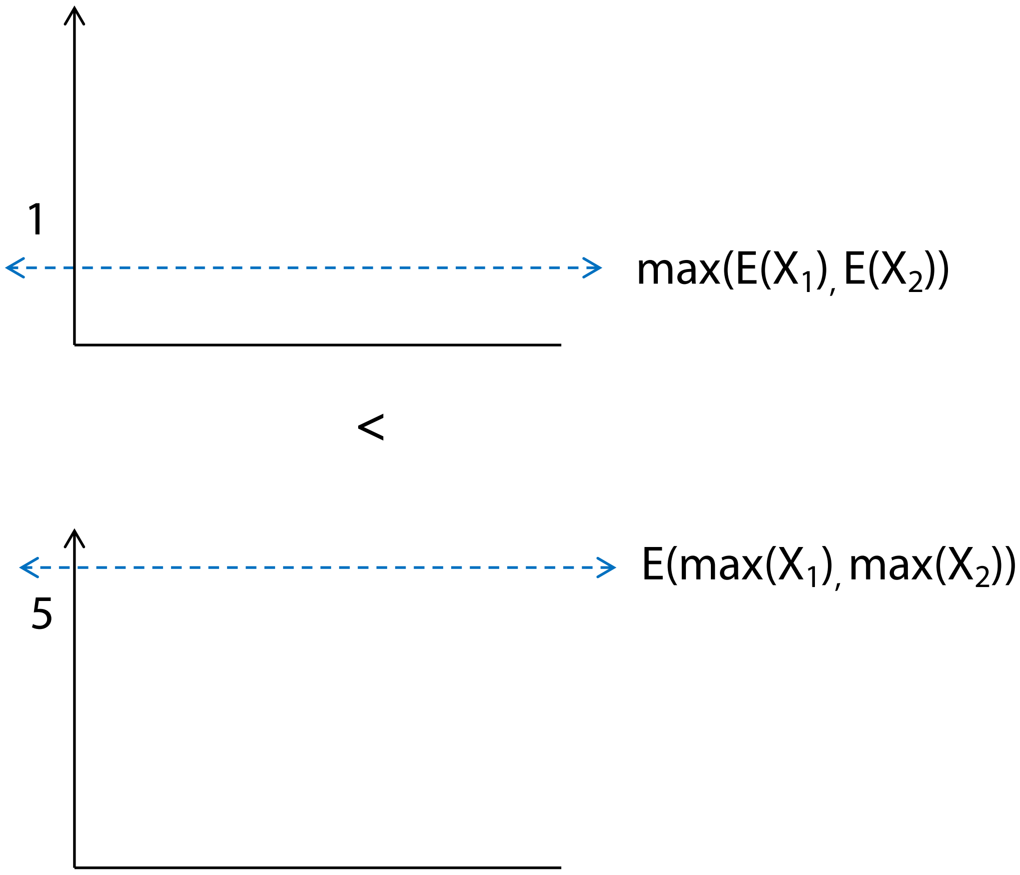


(a) Hopper-v1 (b) Walker2d-v1

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

Overestimation Bias in Actor-Critic

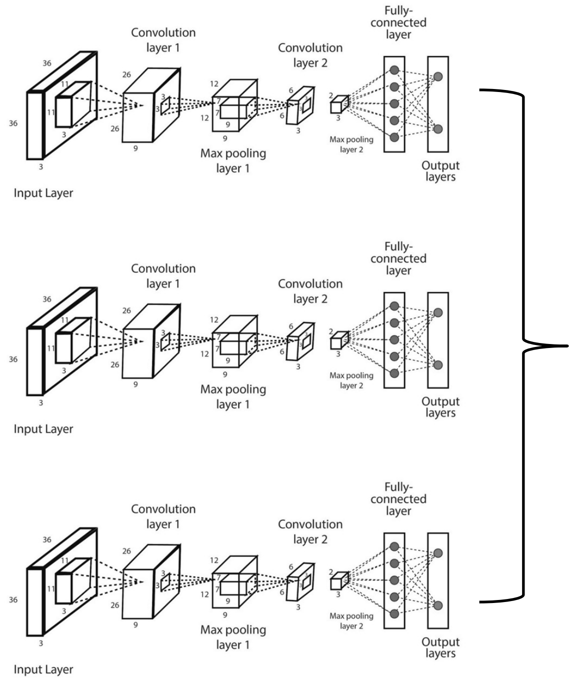




Q-learning can overestimate when values are imperfect
(even when unbiased)

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

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

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

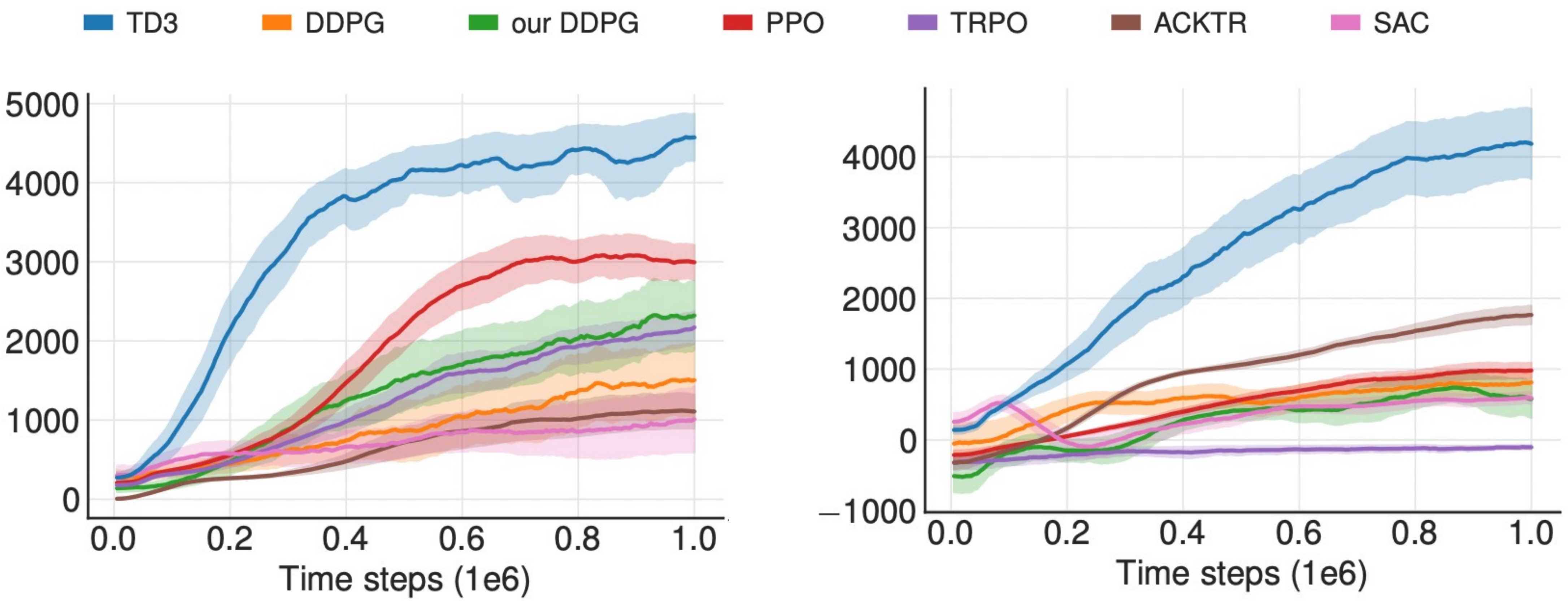
Actor

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

Significantly improves overestimation and in turn sample efficiency!

Overestimation Bias in Actor-Critic

Significantly improves overestimation and in turn sample efficiency!

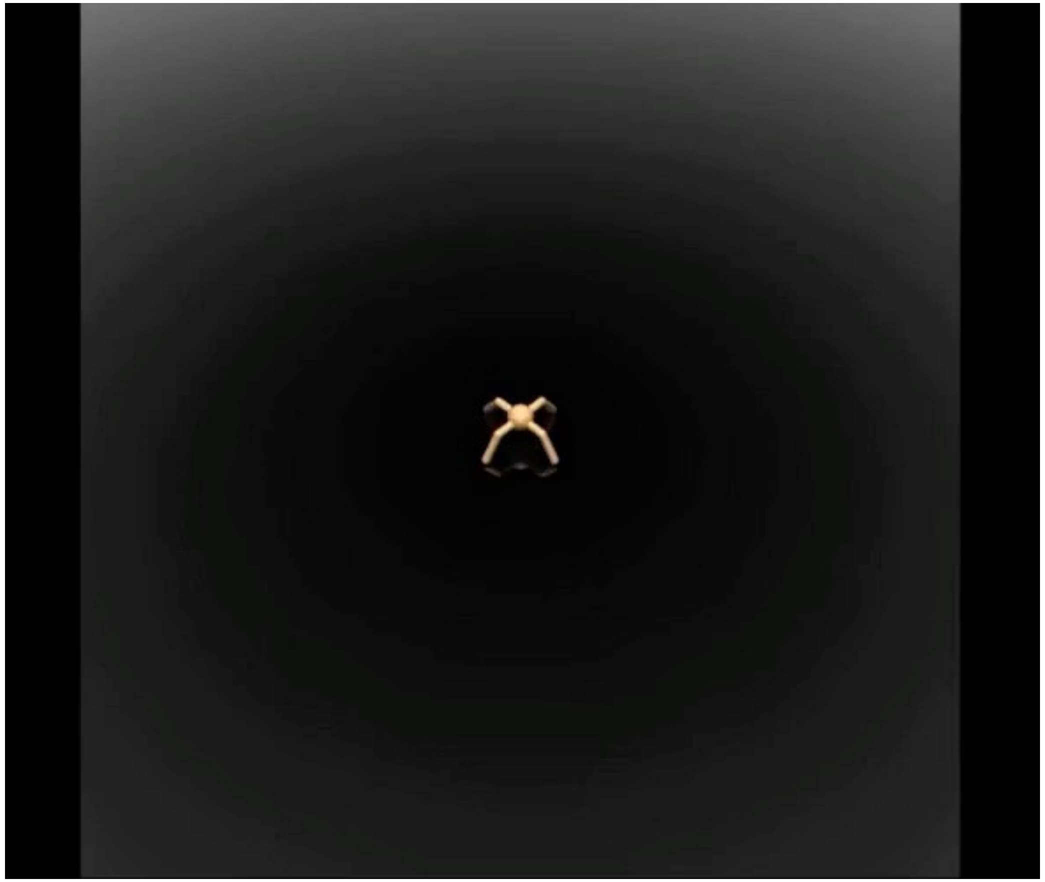


(c) Walker2d-v1 (d) Ant-v1

Double Actor Critic in Action

Double Actor Critic in Action

Where does this fail?



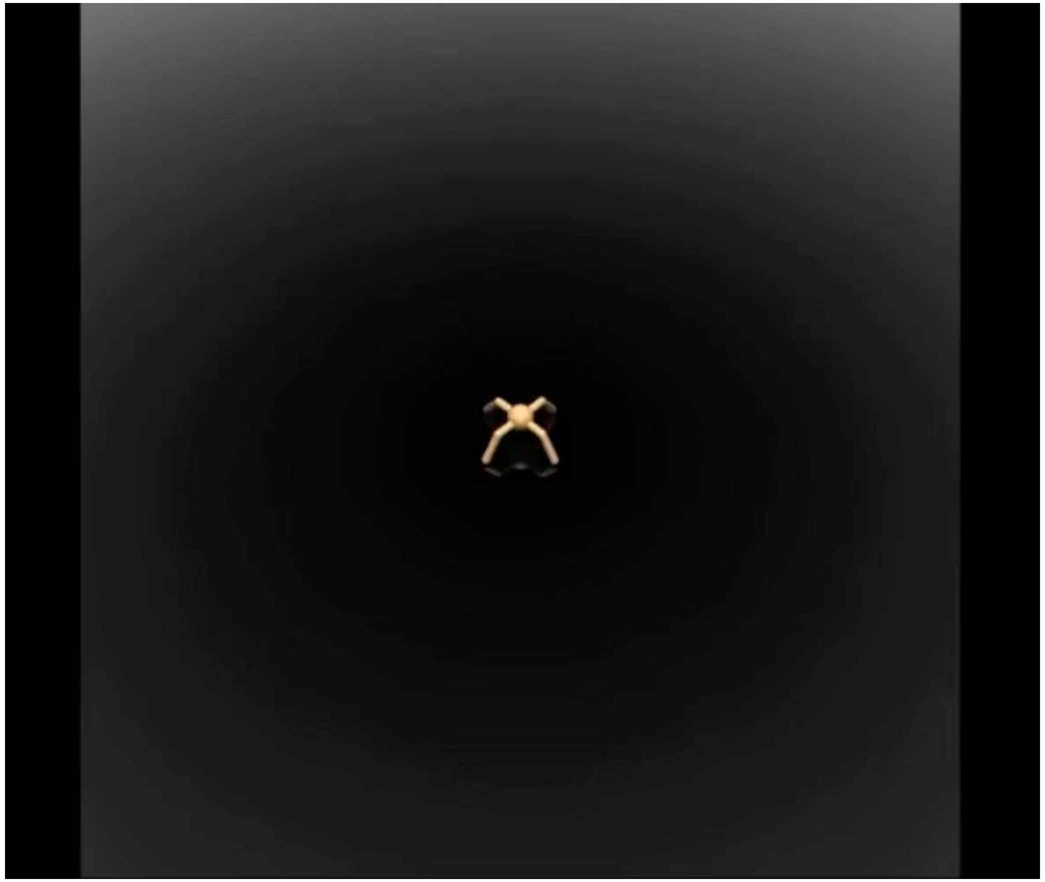
Some issues remain:

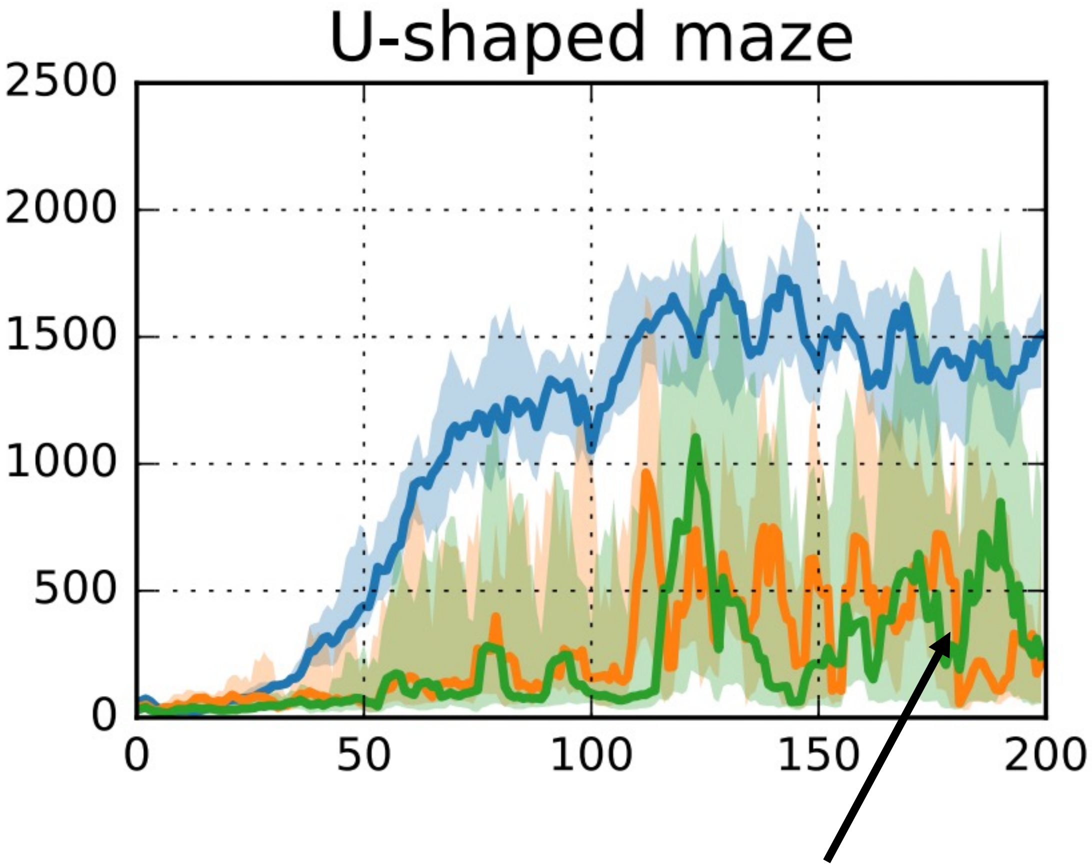
1. Overestimation bias
2. **Insufficient exploration**

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

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

$$\Downarrow$$

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

Simple change in on-policy RL

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

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

Actor – Policy Improvement

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

Soft Bellman Equation from Max-Ent RL

Optimize a ”soft” Bellman equation

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

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

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

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

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

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

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!

Ok, but how do I choose **α**?

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

↑

Often hard to set as a constant

Can simply formulate a constrained optimization problem → entropy above some value

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

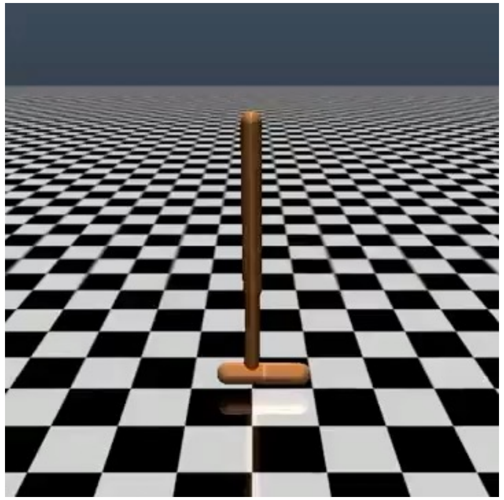
$$\mathbb{E}_{s\sim d^{\pi}(s)}\left[\mathcal{H}(\pi(\cdot|s_t))\right] \geq \epsilon$$

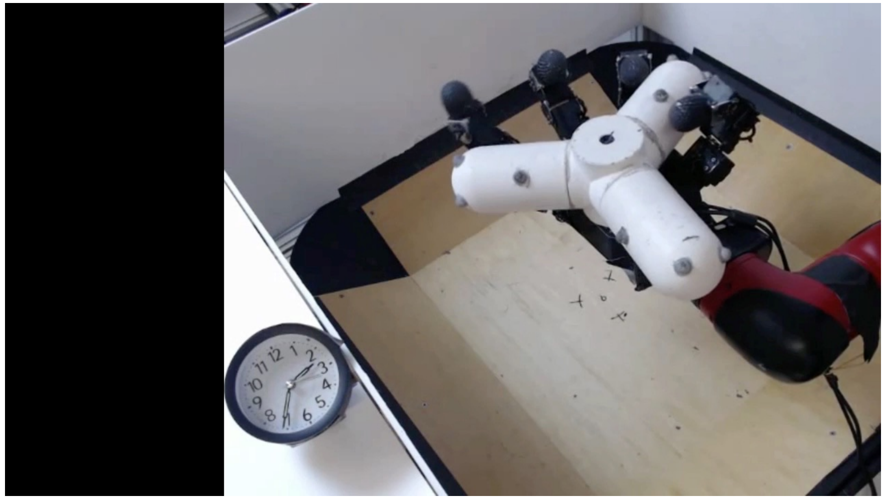
⇨

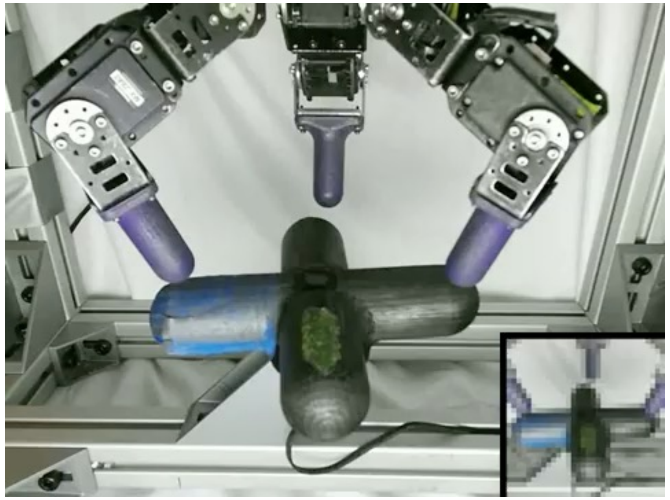
$$\max_{\pi} \min_{\alpha} \mathbb{E}_{\pi}\left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t) - \alpha(\mathcal{H}(\pi(\cdot|s_t) - \epsilon))\right]$$

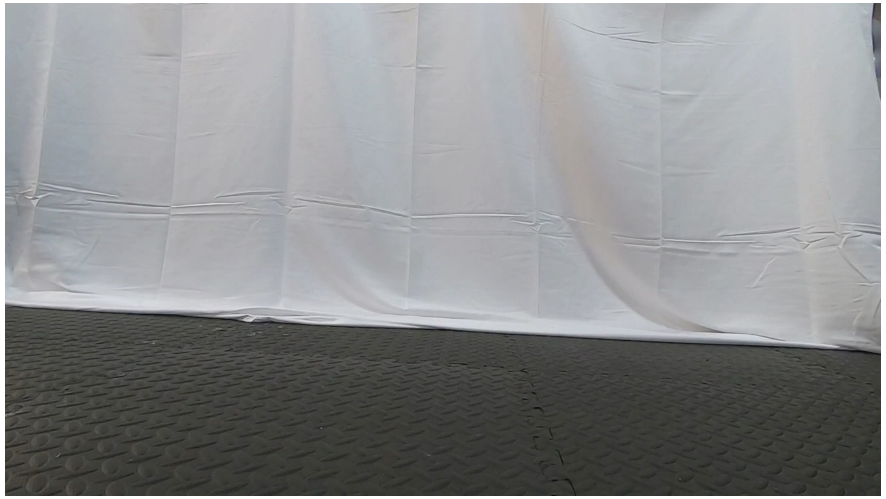
Alternate between gradient steps on π, α

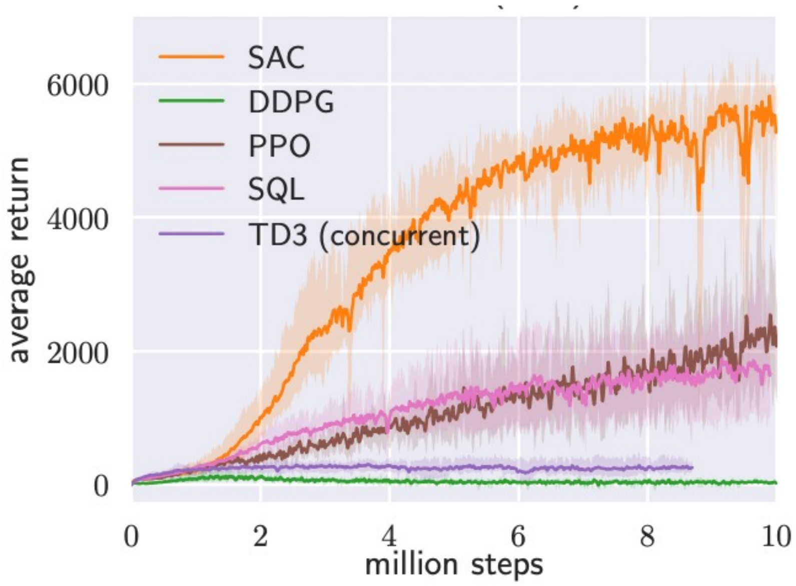
Maximum Entropy Actor-Critic Algorithms in Action











(f) Humanoid (rllab)

Lecture outline

Working through a complete off-policy algorithm

↓

Getting Off-Policy RL to Work

↓

Frontiers of Off-Policy RL

↓

Model-Based RL - Formulation

Ok, so are off-policy algorithms perfect?

What makes off-policy RL hard?

Deadly triad:

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

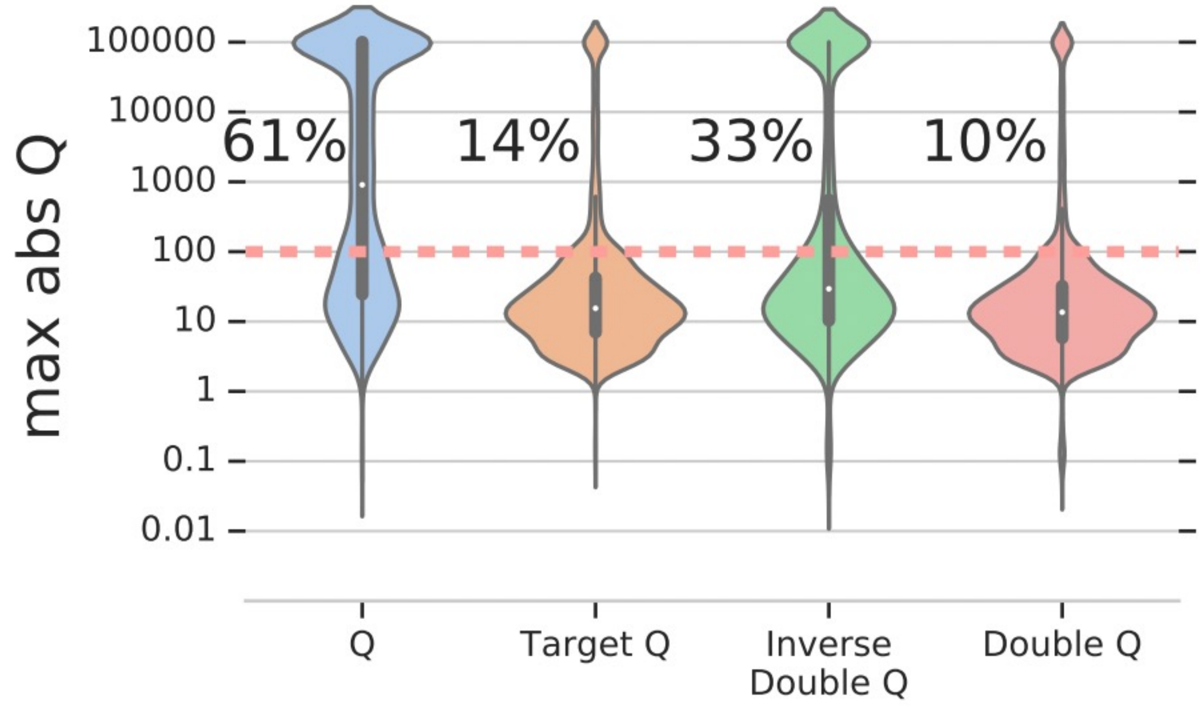
$$\min_{\phi} \mathbb{E}_{(s,a,s')\sim\mathcal{D}} \left[\left[Q^{\pi}_{\phi}(s_t, a_t) - \left(r(s_t, a_t) + \max_{a_{t+1}} \left[Q_{\bar{\phi}}(s_{t+1}, a_{t+1}) \right] \right) \right]^2 \right]$$

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?

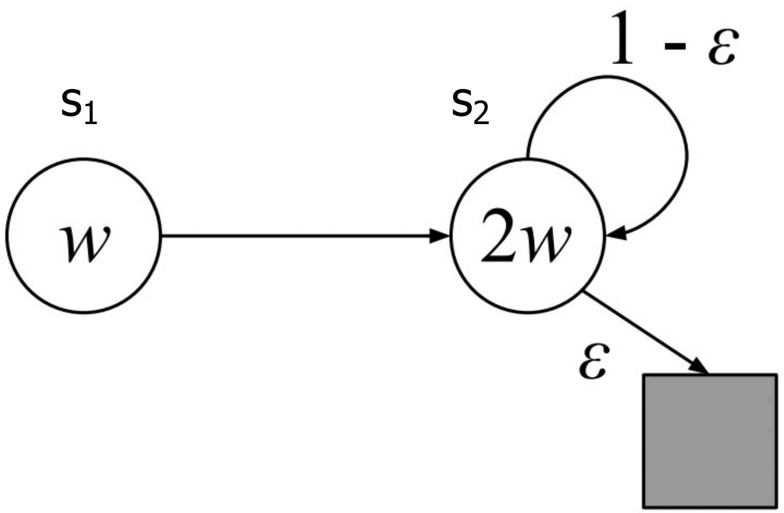
Deadly triad:
1. Function Approximation
2. Bootstrapping
3. Off-policy learning

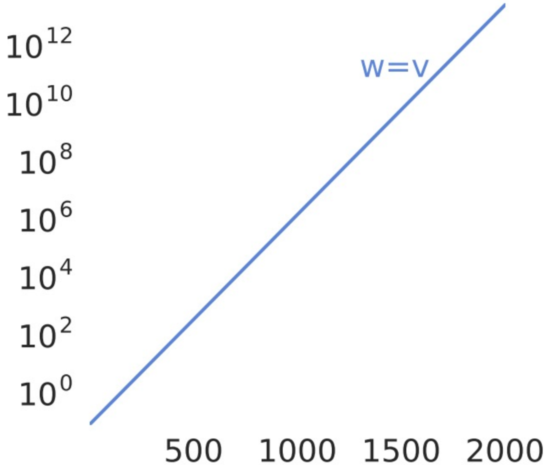
61% of runs show divergence of Q-values



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?

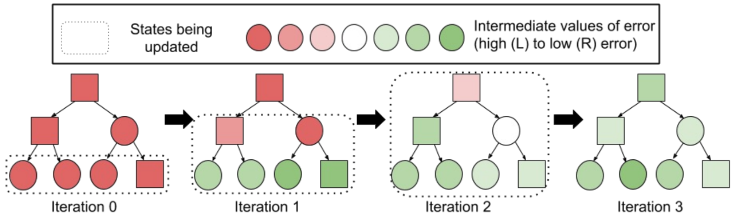
Where does the frontier of off-policy RL lie?

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

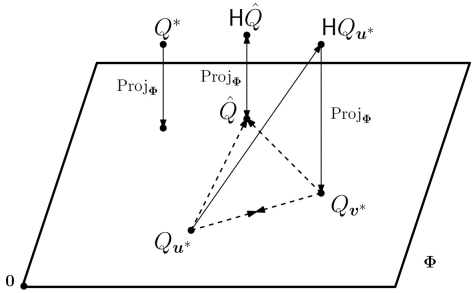
- Low variance, off-policy, avoids reconstruction, performs dynamic programming
- Has the potential to be **performant** and **sample efficient**

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

Sampling



Theory



Exploration

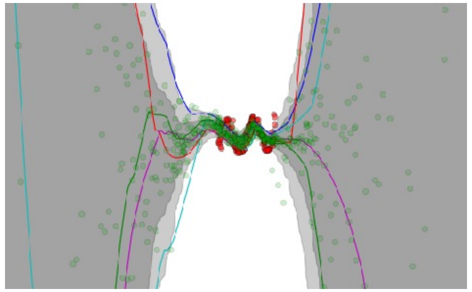
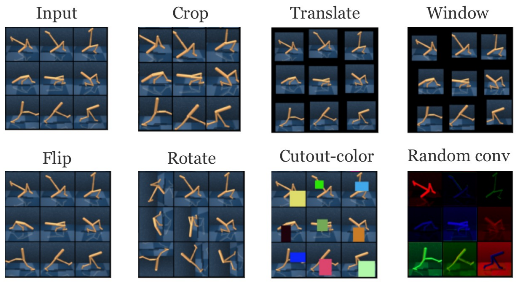
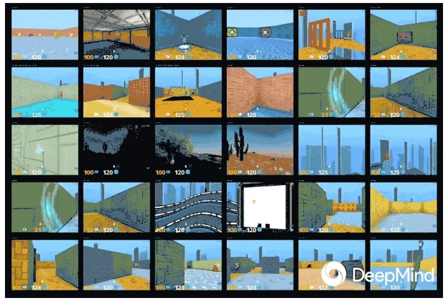


Image-based RL

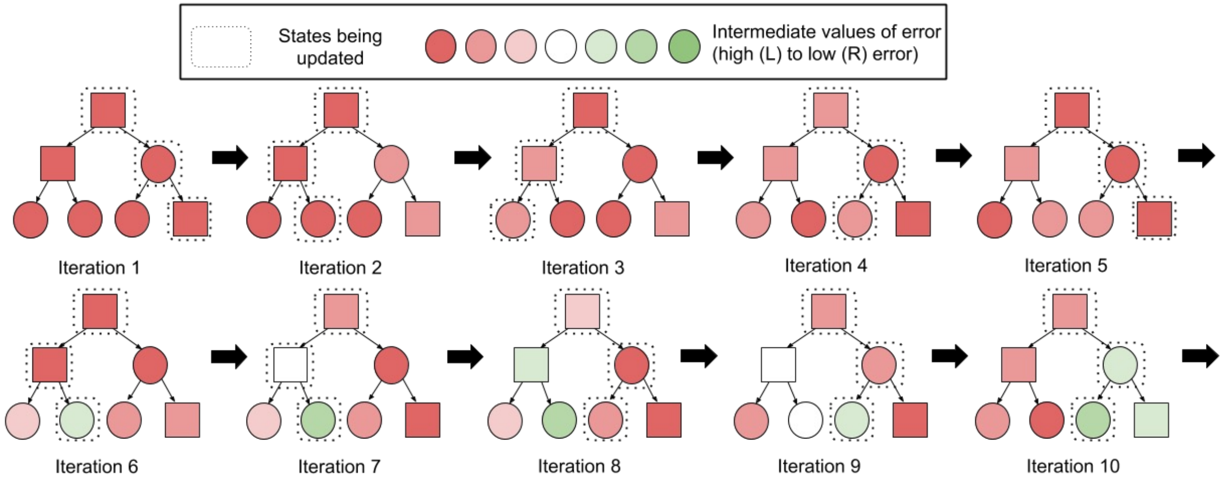


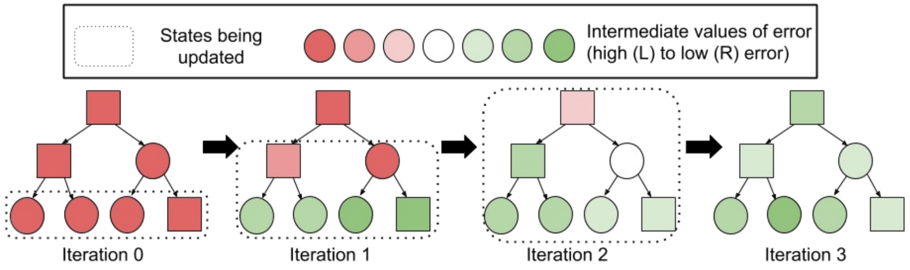
Partial Observability

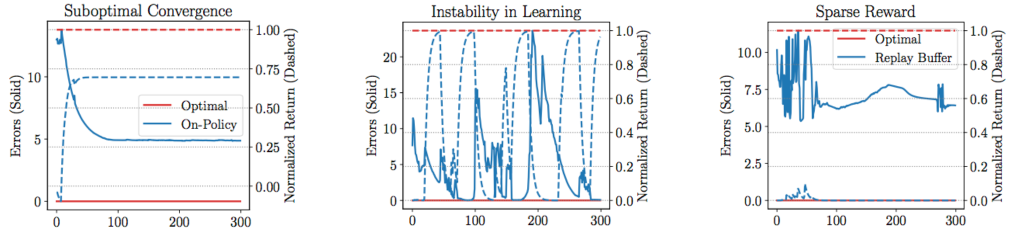


Prioritizing Experience

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

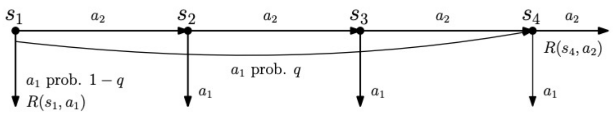


Figure 1: A simple MDP that illustrates delusional bias (see text for details).

Implicit regularization

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

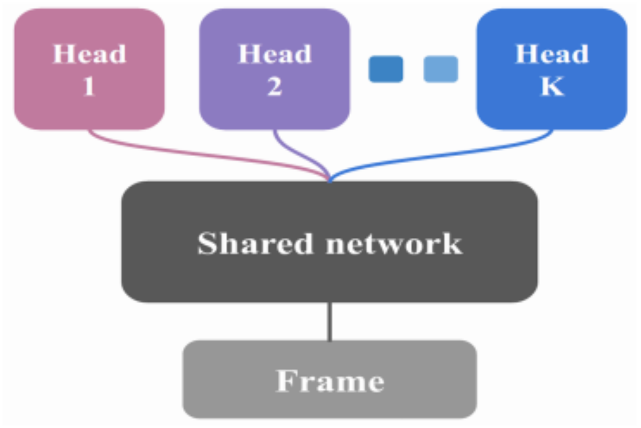
Bilinear classes

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

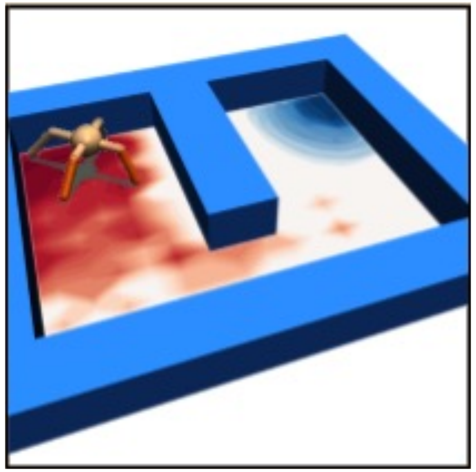
Exploration in Off-Policy RL

Better exploration methods

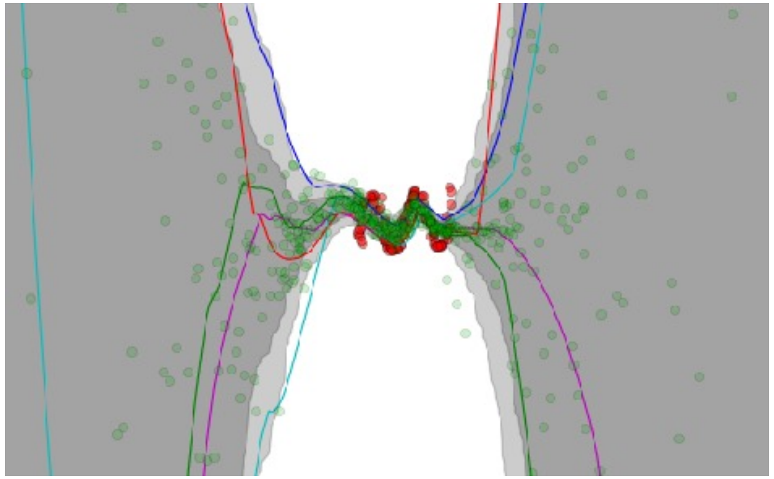
Uncertainty based methods



Count-based methods



Information gain methods

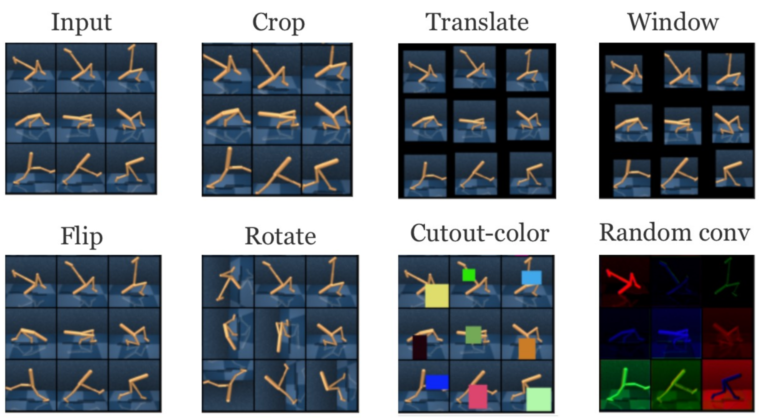


Often critical for getting algorithms to work!

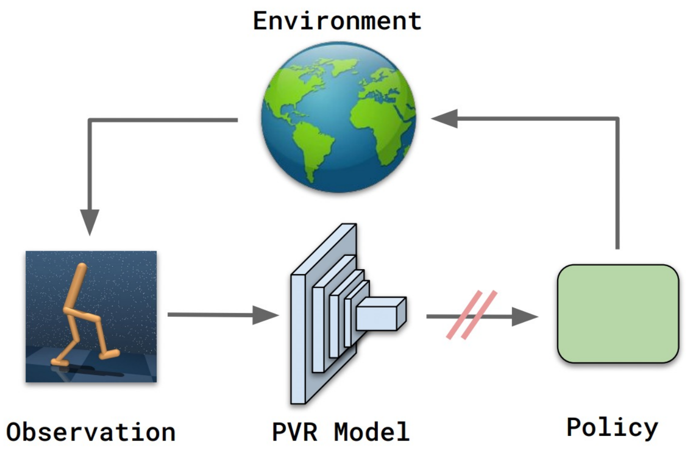
Image-based Off-Policy RL

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

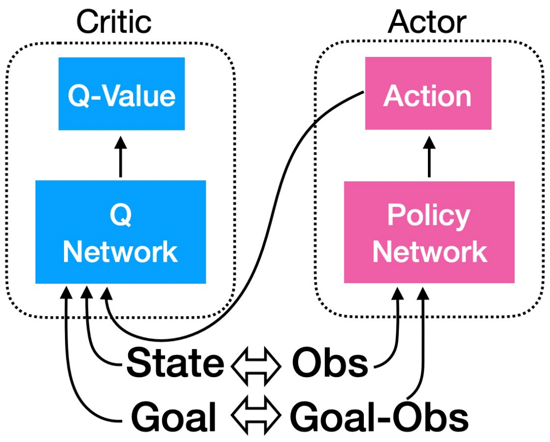
Data augmentations



Pre-trained representations



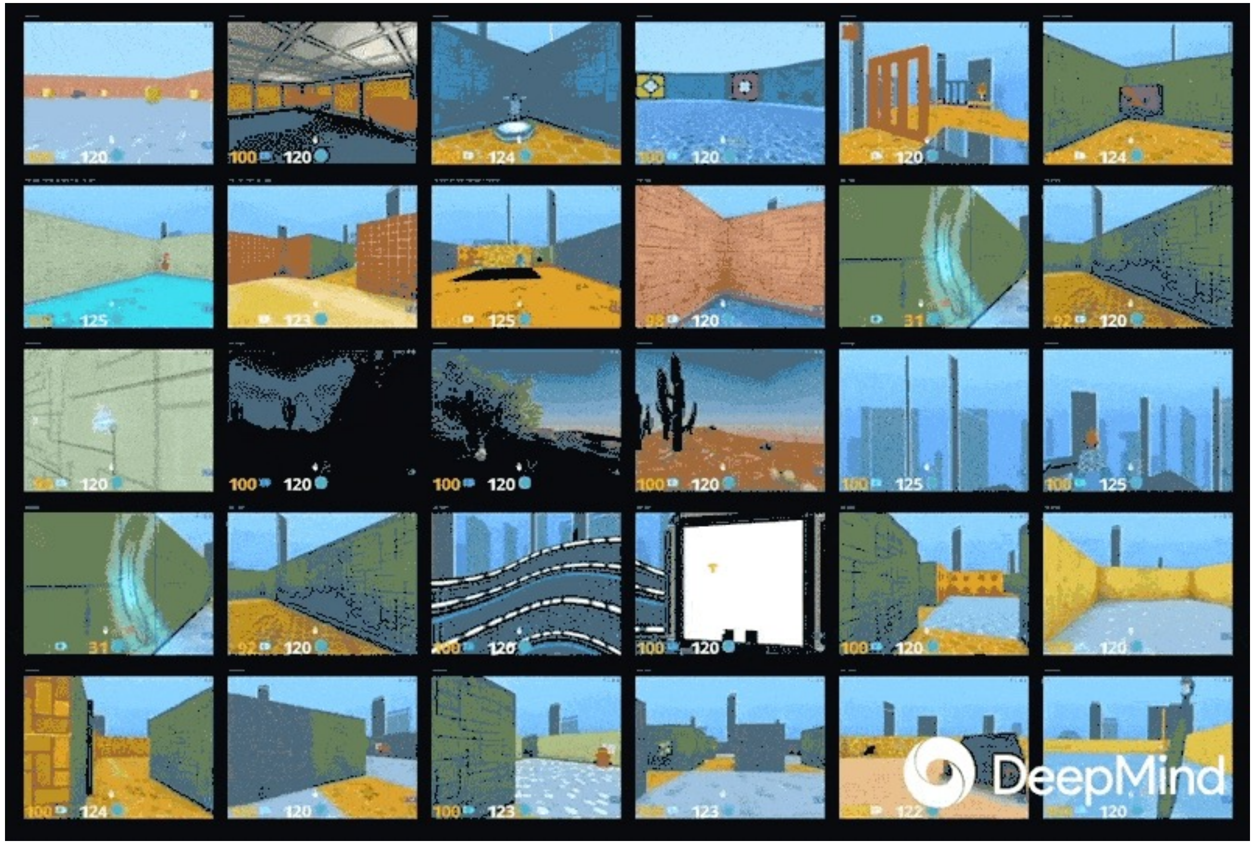
Student-teacher

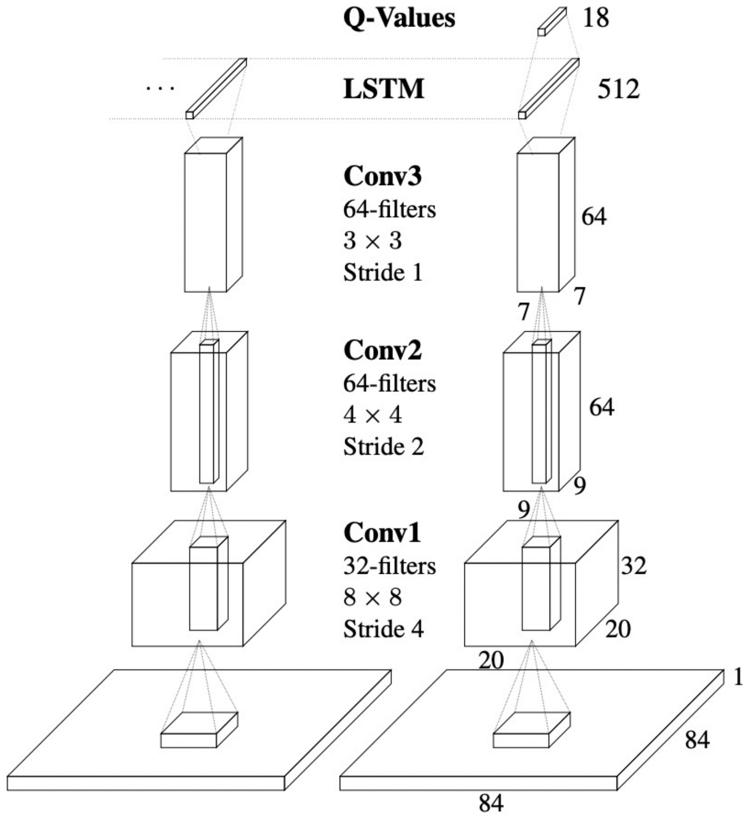


Still very unstable, lot of open research problems!

Partial Observability in Off-Policy RL

Off-policy methods critically depend on the Markov assumption



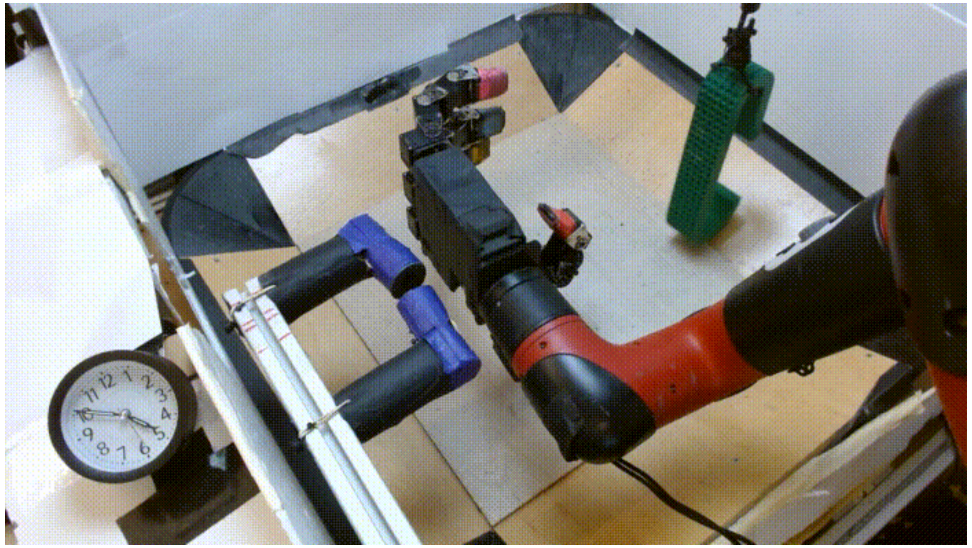


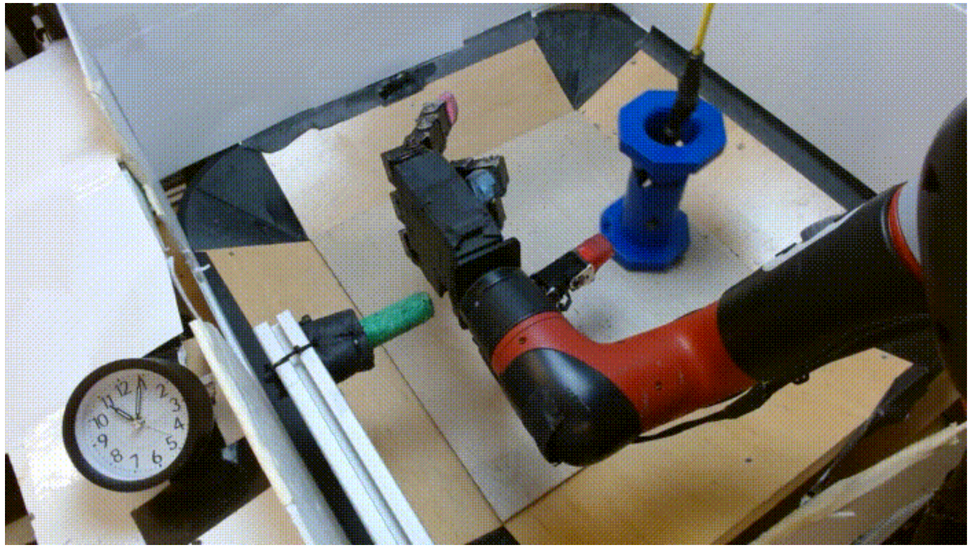
Learning history conditioned/recurrent Q-functions is an open area!

How has off-policy RL manifested in robotics?

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

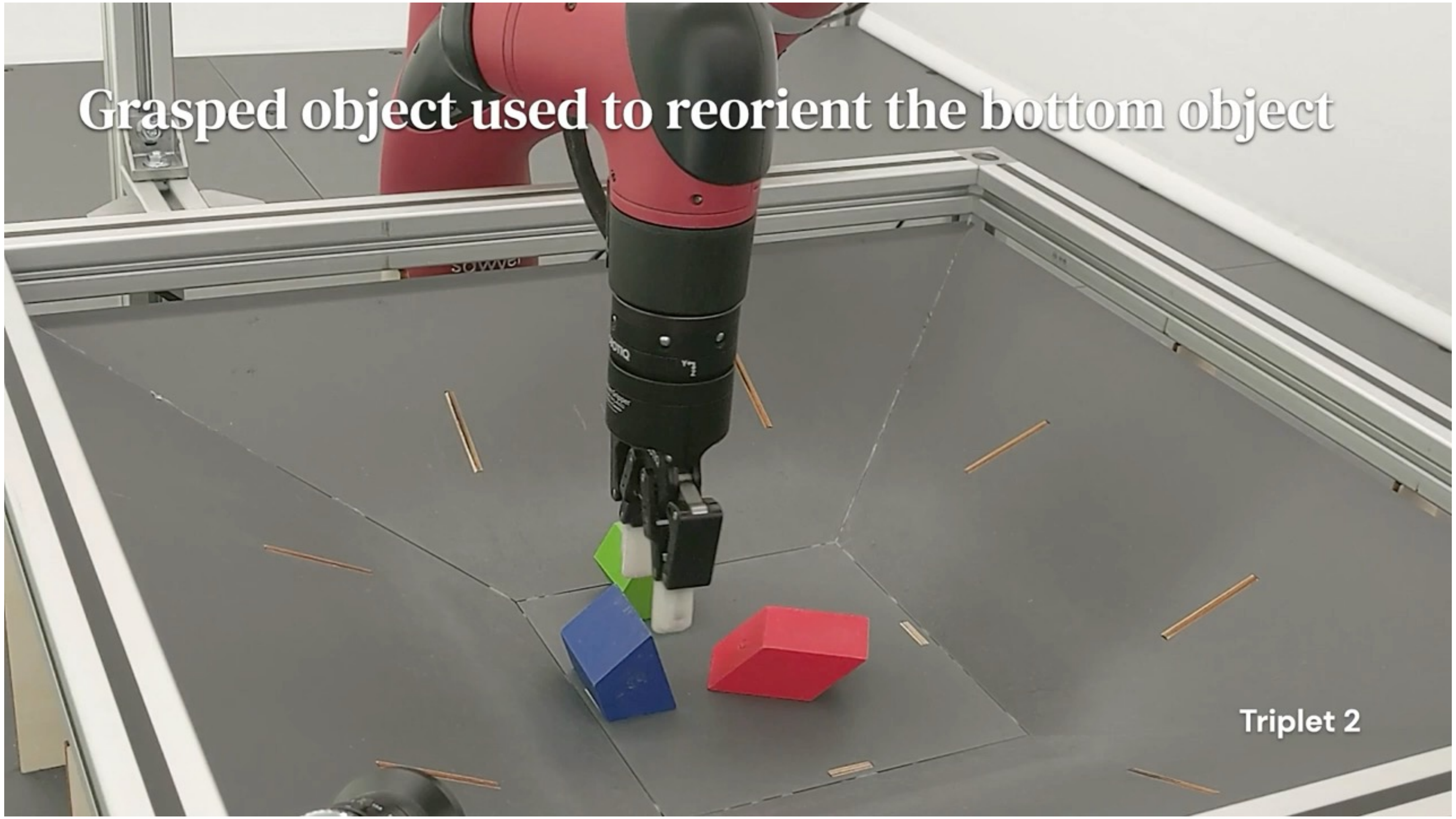
How has off-policy RL manifested in robotics?





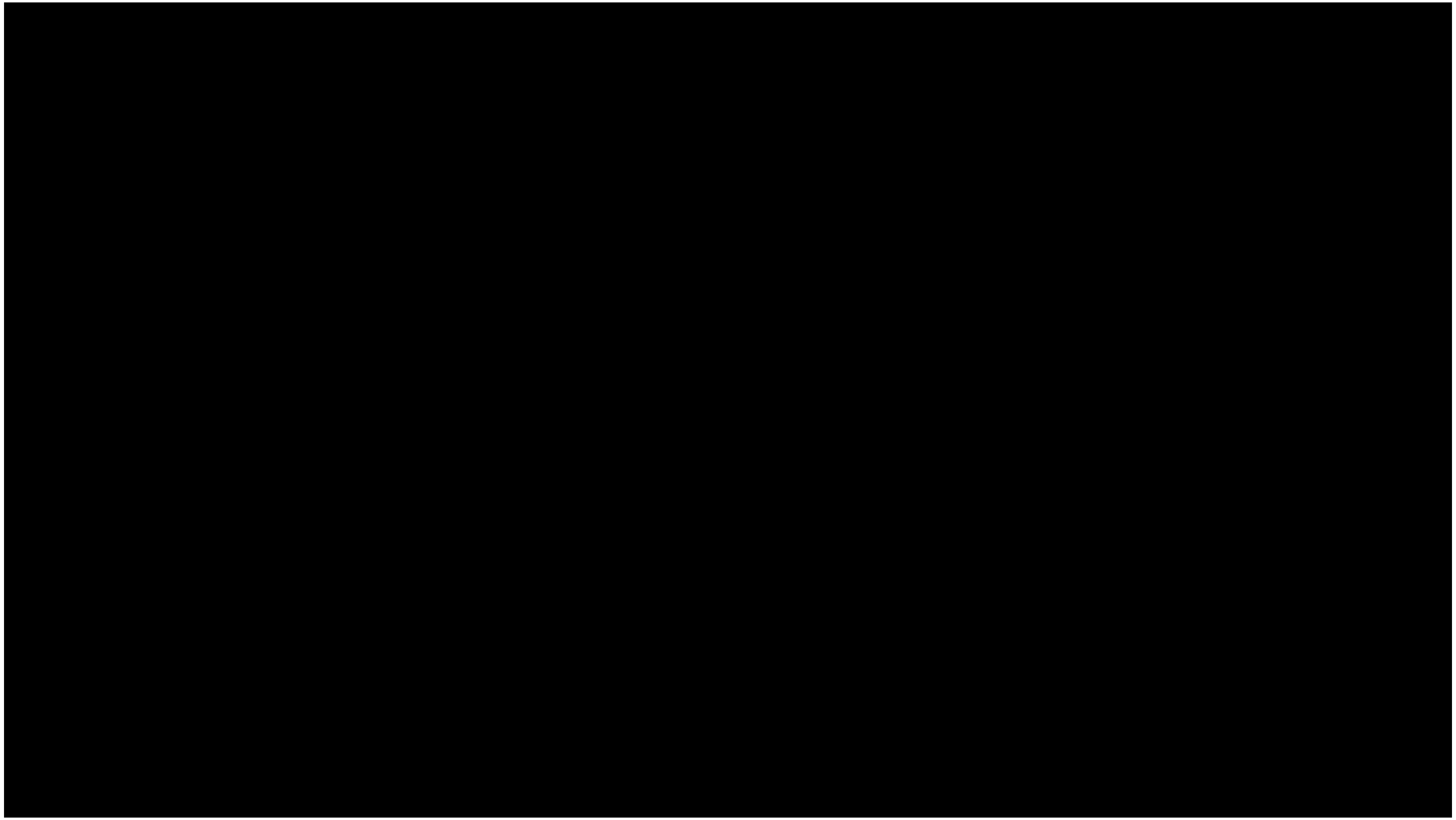
How has off-policy RL manifested in robotics?

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

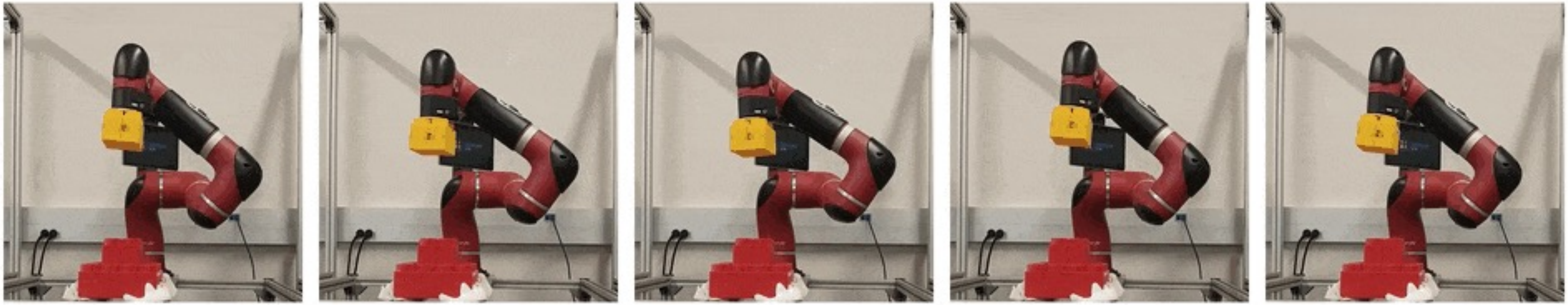


How has off-policy RL manifested in robotics?

Bootstrapped with a few demonstrations



How has off-policy RL manifested in robotics?



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

Working through a complete off-policy algorithm

↓

Getting Off-Policy RL to Work

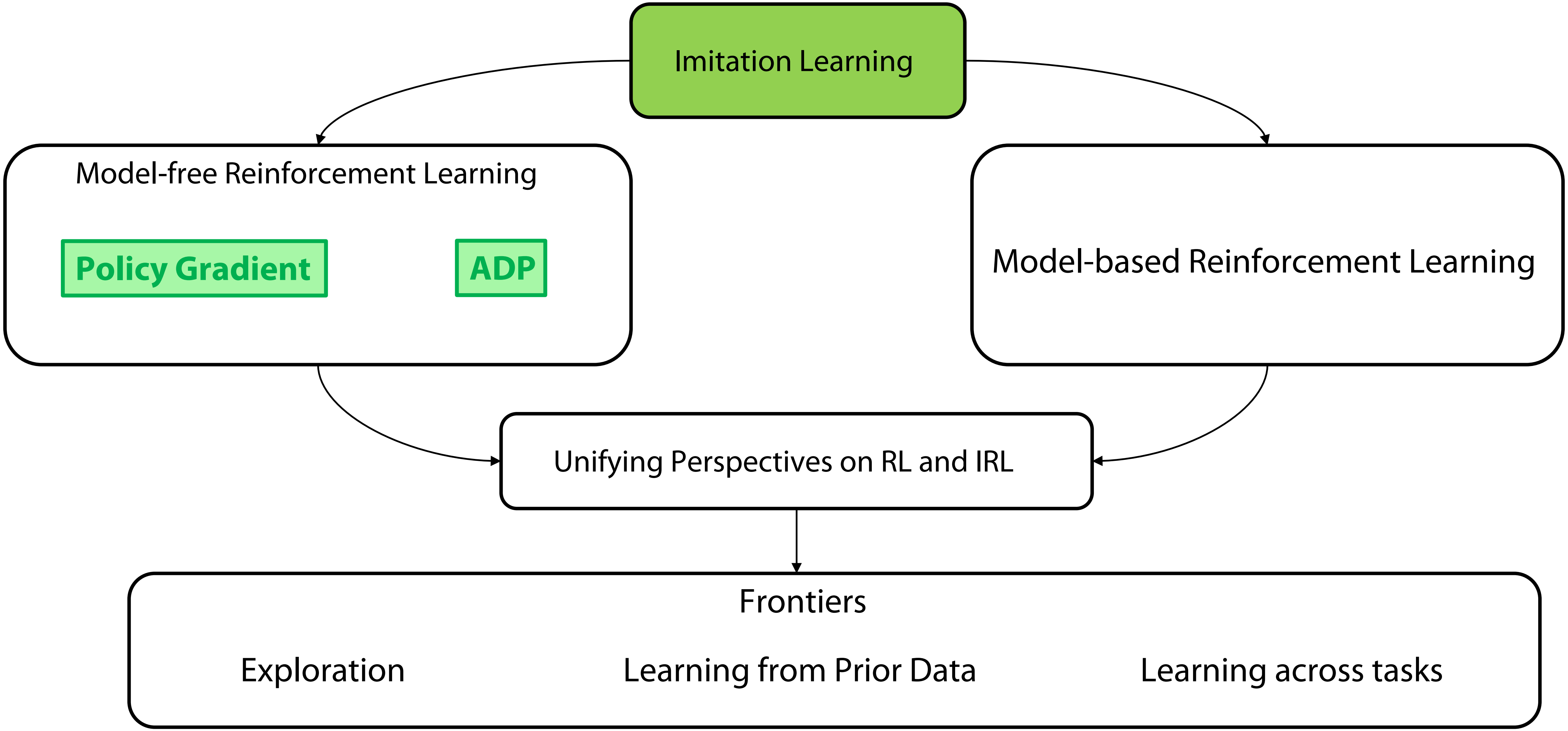
↓

Frontiers of Off-Policy RL

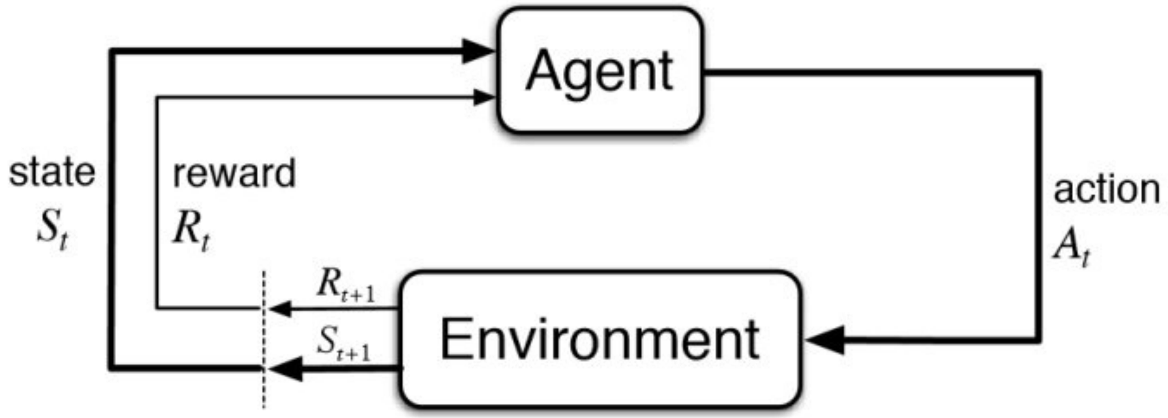
↓

Model-Based RL - Formulation

Class Structure



Landscape of Reinforcement Learning Algorithms



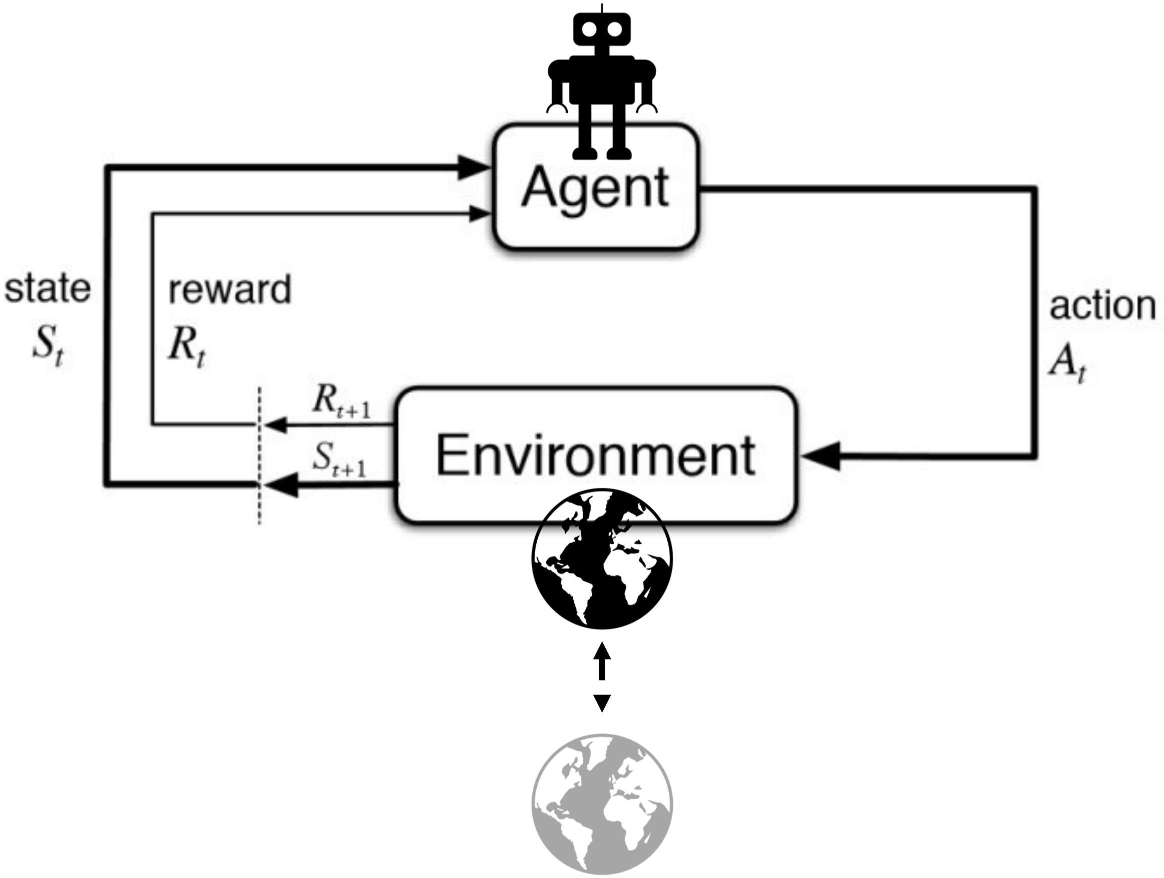
$$\max_\theta \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} r(s_t, a_t) \right]$$

Gradient Ascent

Dynamic Programming

Model-Based Optimization

What if we just learned how the world worked?



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

1. Learn a surrogate model of the transition dynamics from arbitrary off-policy data
2. Do reward maximization against this model

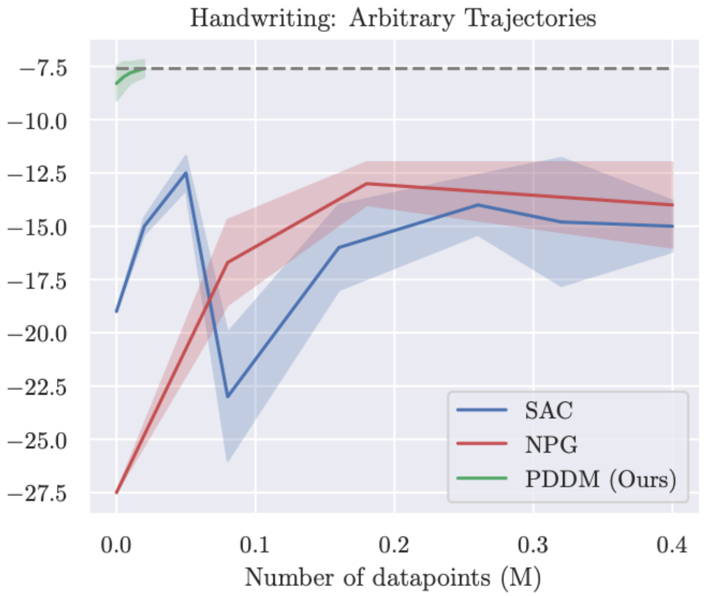
Intuitive: learn how the world works first and then plan in that model

Why do model-based RL?

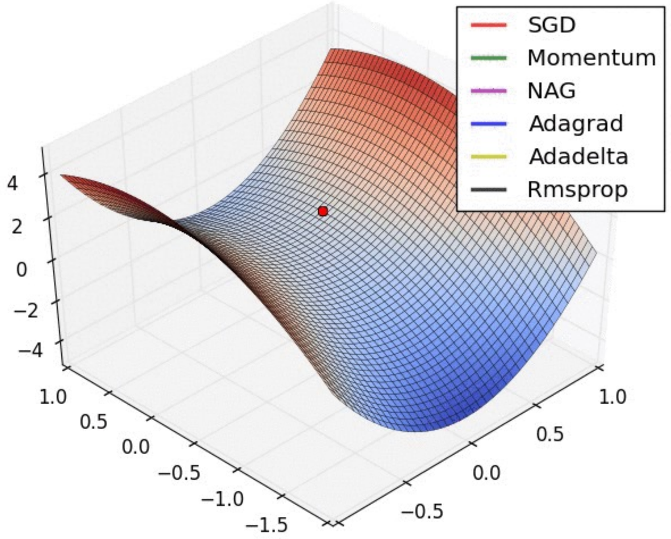
Why would we do this?

Transfer/Adaptive

Efficiency



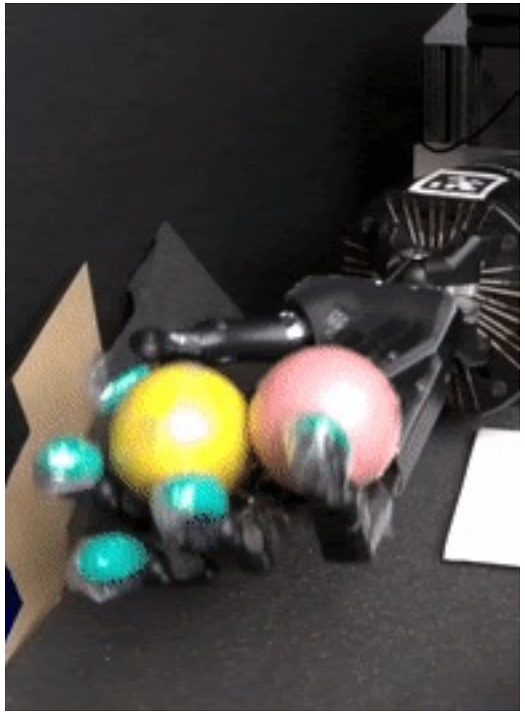
Simplicity



Naturally off-policy!

Why do model-based RL?

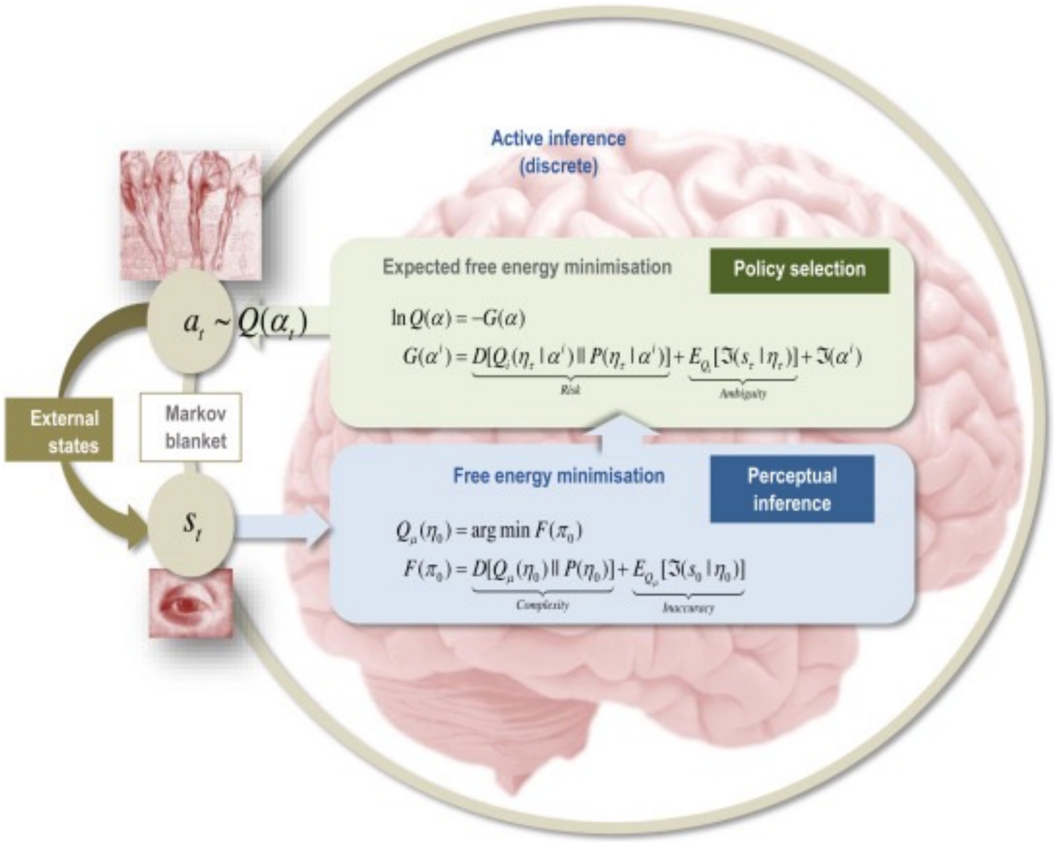


Just 2 hours of real robot training

Connections to Cognitive Science

Significant evidence for mechanisms for prediction of outcomes in neuro/cognitive science

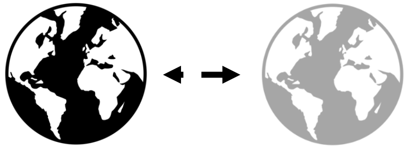


Reinforcement learning in the brain

Yael Niv
Psychology Department & Princeton Neuroscience Institute, Princeton University

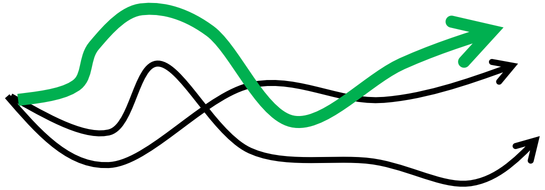
Model Based RL – Problem Statement

Model Learning



$$\hat{p}_\theta \leftarrow \arg\min_{\hat{p}_\theta} \mathcal{L}(\mathcal{D}, \hat{p}_\theta)$$

Planning



$$\arg\max_{\pi} \mathbb{E}_{\hat{p},\pi}\left[\sum_t r(s_t, a_t)\right]$$

Can also just be a single trajectory

How should we instantiate these?

What will we not cover today?

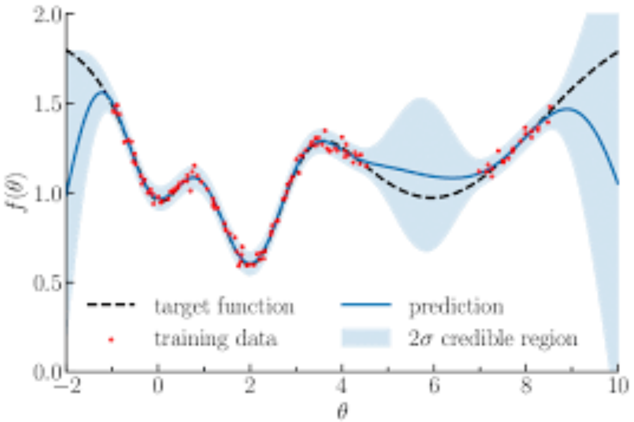
<u>iLQR/iLQG</u>

$$\min_{\mathbf{u}_1,\ldots,\mathbf{u}_T} \sum_{t=1}^{T} c(\mathbf{x}_t, \mathbf{u}_t) \text{ s.t. } \mathbf{x}_t = f(\mathbf{x}_{t-1}, \mathbf{u}_{t-1})$$

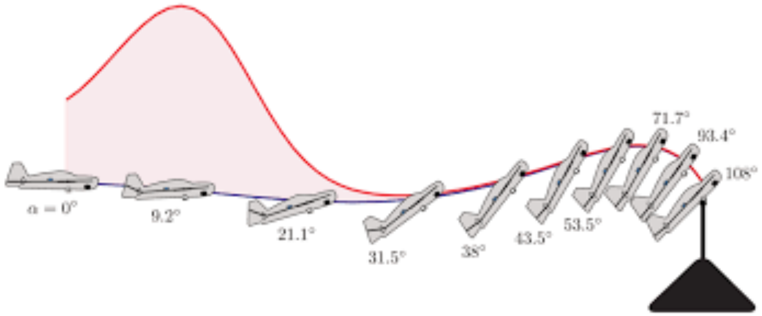
$$f(\mathbf{x}_t, \mathbf{u}_t) = \mathbf{F}_t \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix} + \mathbf{f}_t$$

$$c(\mathbf{x}_t, \mathbf{u}_t) = \frac{1}{2} \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix}^T \mathbf{C}_t \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix} + \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix}^T \mathbf{c}_t$$

<u>MBRL with GPs/Non-Parametrics</u>



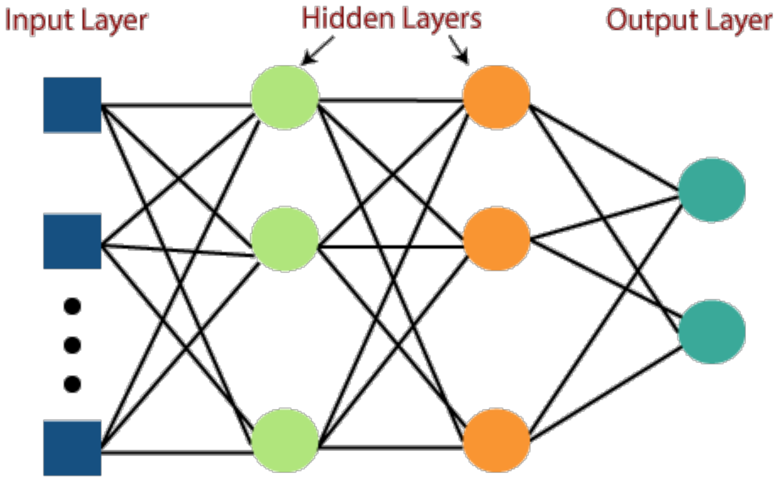
<u>Non-linear TrajOpt</u>



Byron’s lectures do a wonderful job, do go watch them!

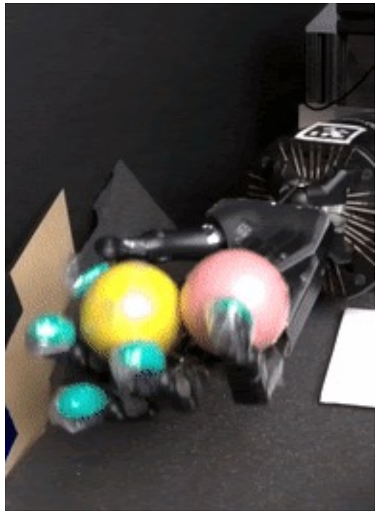
What will we cover today?

Use neural networks as our model!

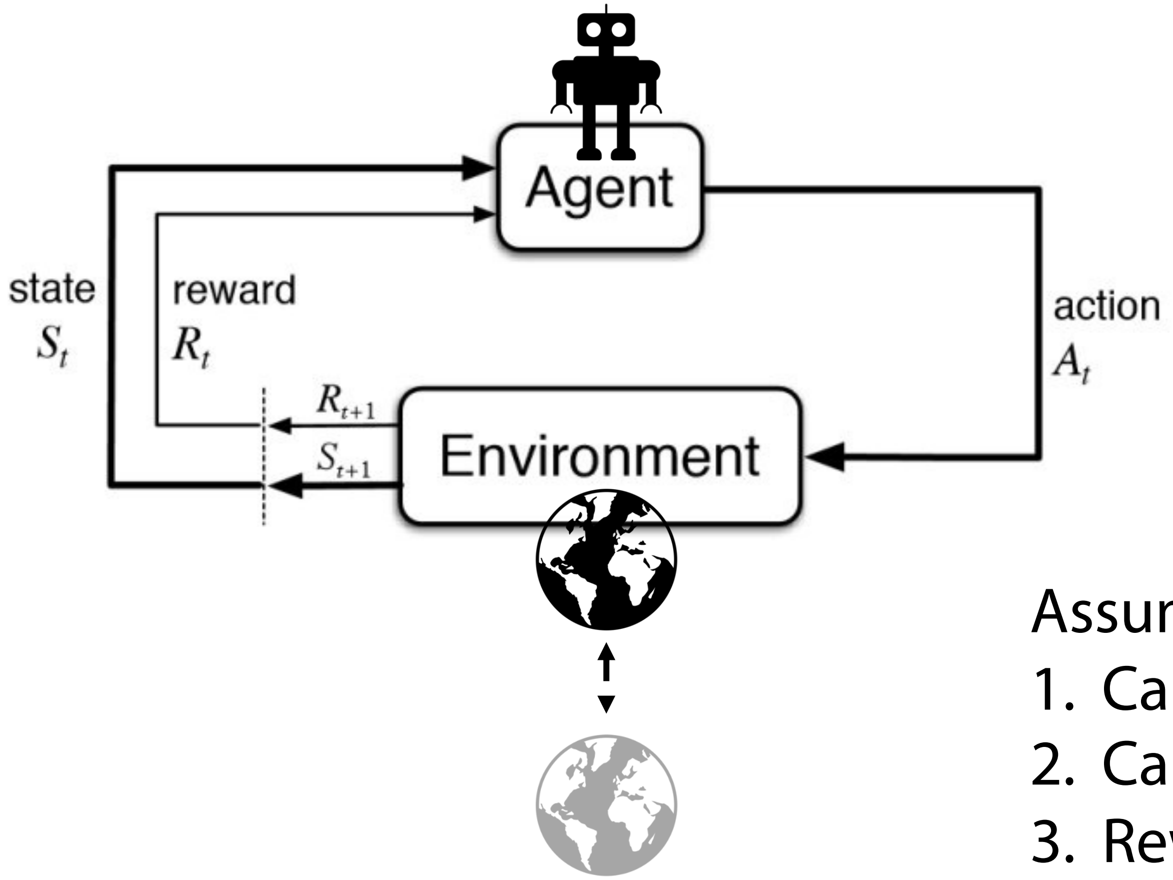


$$\hat{p}_\theta \leftarrow \arg\min_{\hat{p}_\theta} \mathcal{L}(\mathcal{D}, \hat{p}_\theta)$$

$$\arg\max_{\pi} \mathbb{E}_{\hat{p},\pi}\left[\sum_t r(s_t, a_t)\right]$$



Model Based RL – Assumptions



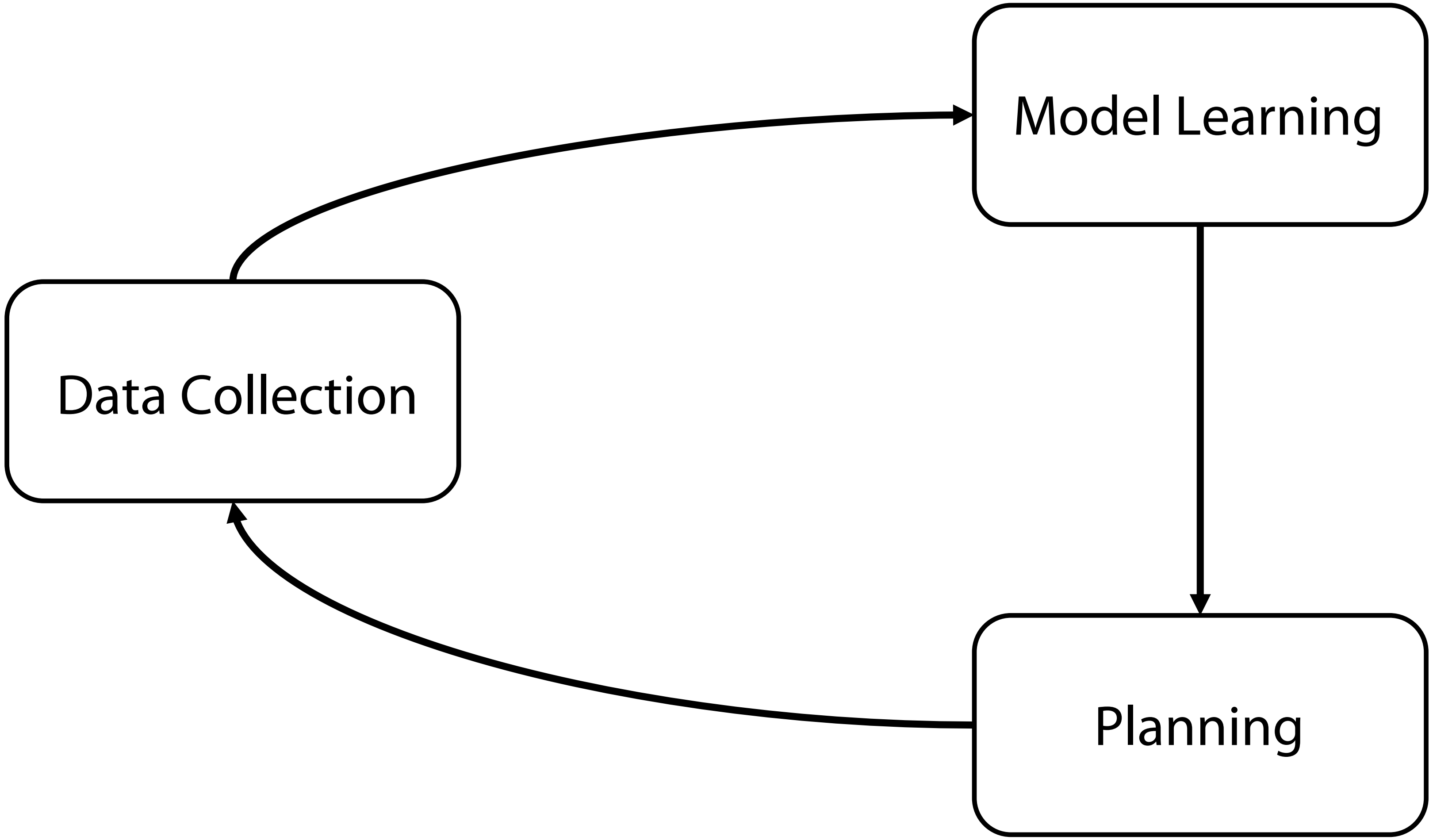
$$\max_{\theta} \mathbb{E}_{\tau \sim \pi_{\theta}} \left[\sum_{t=0}^{T} r(s_t, a_t) \right]$$

Assumptions:

1. Can only **sample** from dynamics
2. Can **reset** the environment
3. Reward function is **known**

We will get into this in a later lecture!

Model Based RL – A template



Lecture Outline

⇨ Model based RL v0 → random shooting + MPC

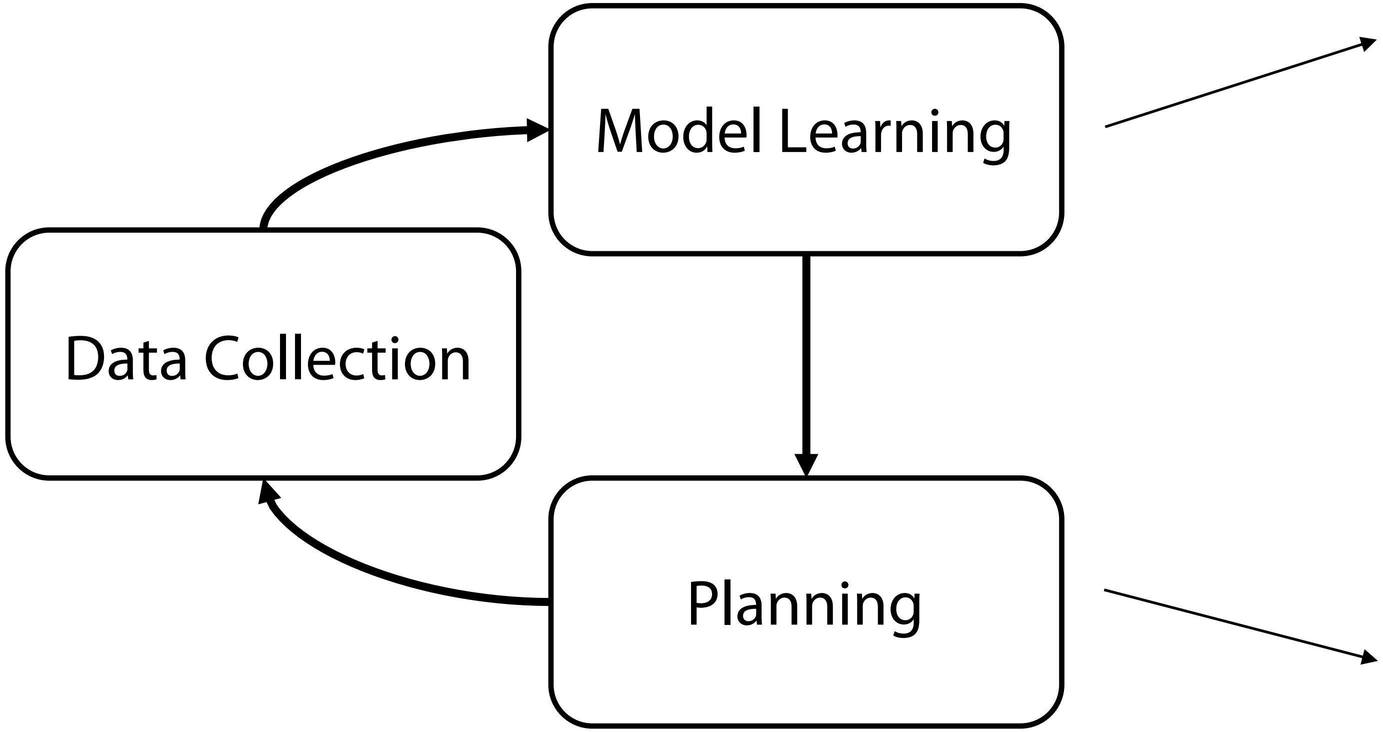
Model based RL v1 → MPPI + MPC

Model based RL v2 → uncertainty based models

Model based RL v3 → policy optimization with models

Model based RL v4 → latent space models with images

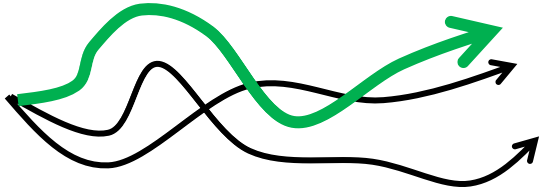
Model Based RL – Naïve Algorithm (v0)



Maximum likelihood supervised Learning

$$\max_{\theta} \mathbb{E}_{(s,a,s')\sim\mathcal{D}} \left[\log \hat{p}_{\theta}(s'|s,a)\right]$$

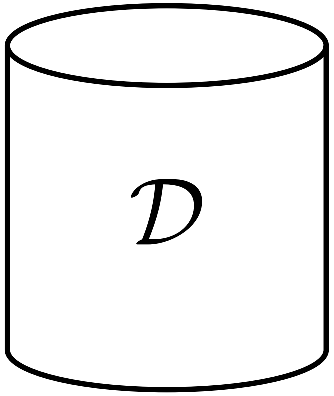
Random Search

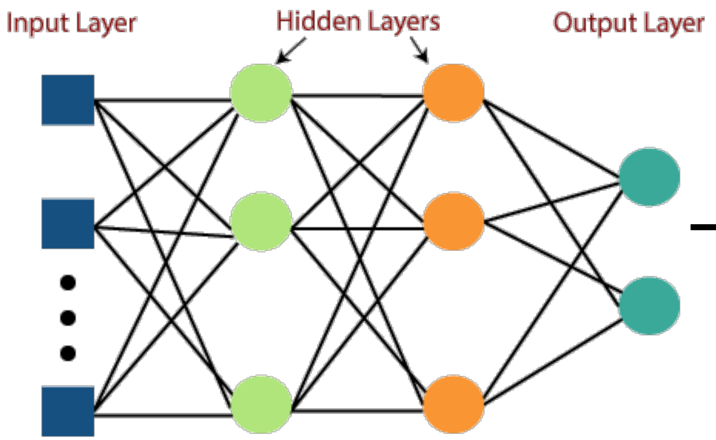


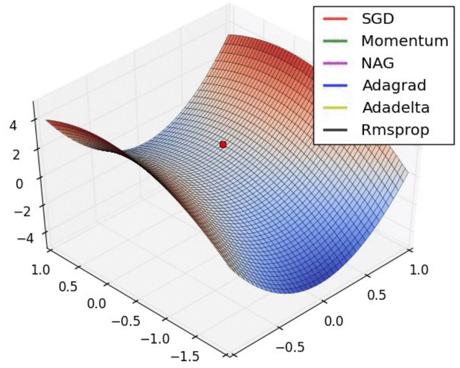
Model Based RL – Naïve Algorithm (Model Learning) (v0)

$$\max_{\theta} \mathbb{E}_{(s,a,s')\sim\mathcal{D}} \left[\log \hat{p}_\theta(s'|s,a)\right]$$

Fit 1-step models







More expressive may be better, at the risk of overfitting

Choice of $\hat{p}_\theta$ distribution determines the loss function:

1. Gaussian → L_2
2. Energy Based Model → Contrastive Divergence
3. Diffusion Model → Score Matching

Trick: Model Residual's (s' –s)

Model Based RL – Naïve Algorithm (Planning)

Planning

$$\max_{a_0, a_1, \ldots, a_T} \sum_{t=0}^{T} r(\hat{s}_t, a_t)$$

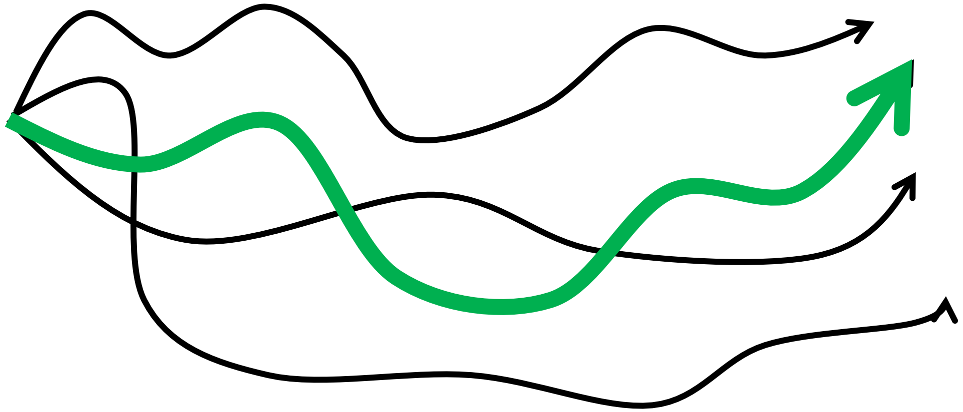
$$\hat{s}_{t+1} \sim \hat{p}_\theta(s_{t+1} | \hat{s}_t, a_t)$$

$$\hat{s}_1 \sim \hat{p}_\theta(s_{t+1} | s_0, a_0)$$

Just do random search!

$$\arg \max_{a_0^j, a_1^j, \ldots, a_T^j} \sum_{t=0}^{T} r(\hat{s}_t^j, a_t^j)$$

$$\hat{s}_{t+1}^j \sim \hat{p}_\theta(.|\hat{s}_t^j, a_t^j)$$



Just execute actions open loop!

Can soften by taking softmax rather than argmax

Lecture outline

Working through a complete off-policy algorithm

↓

Getting Off-Policy RL to Work

↓

Frontiers of Off-Policy RL

↓

Model-Based RL - Formulation

Fin.