

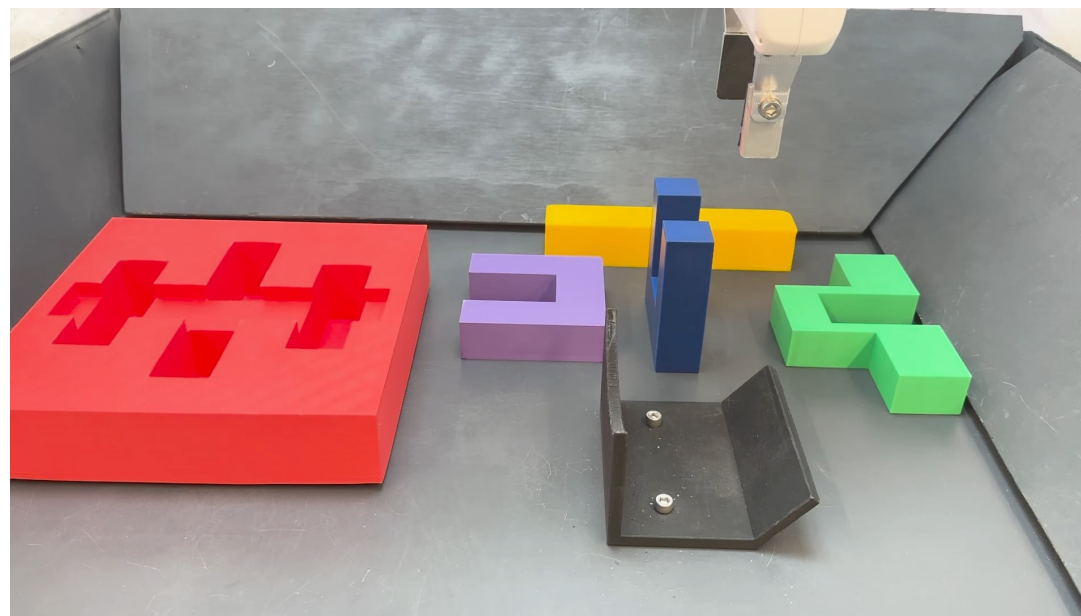


# Reinforcement Learning

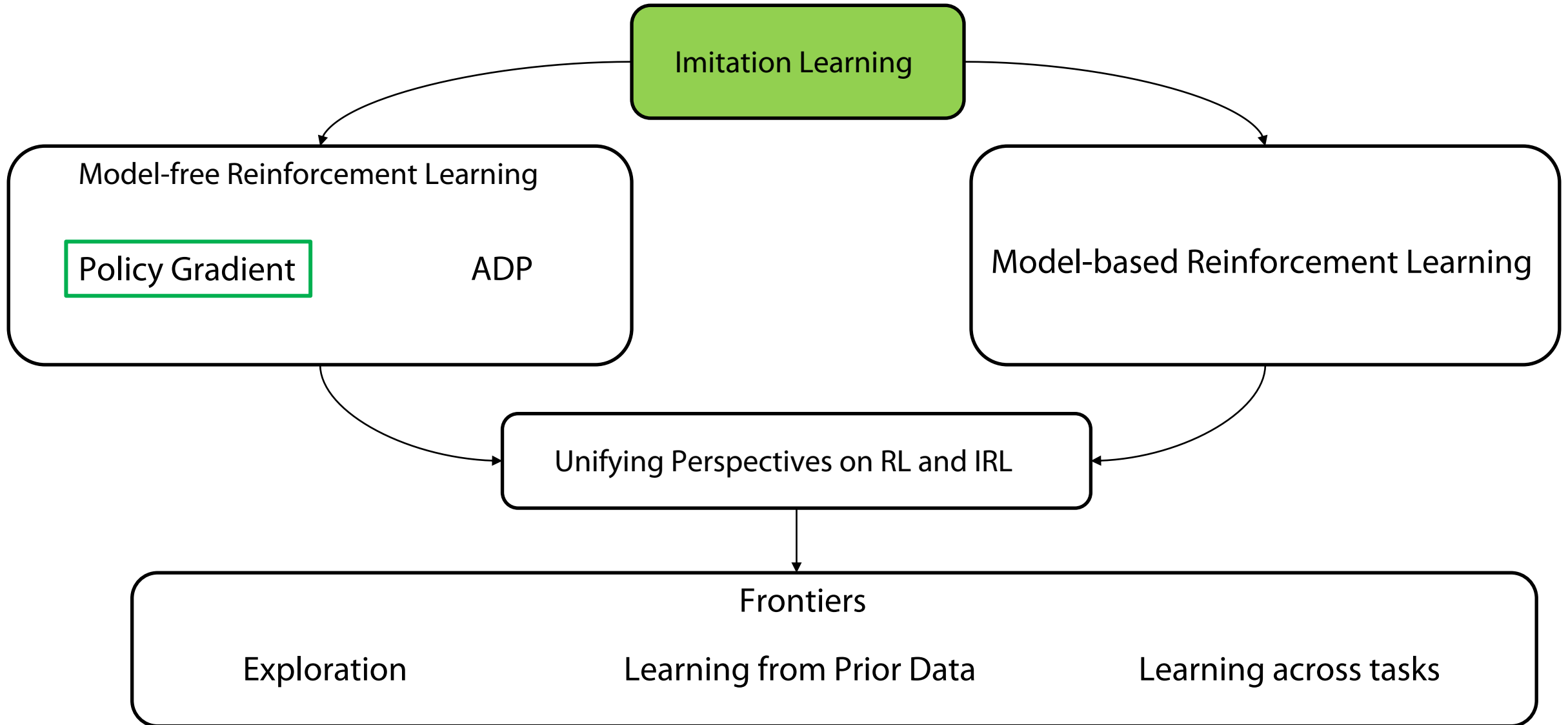
## Spring 2026

Abhishek Gupta

TA: Mateo Guaman Castro



# Class Structure



# Lecture outline

---

Recap: Policy Gradient and Variance Reduction



Natural Policy Gradients and Covariant Parameterization

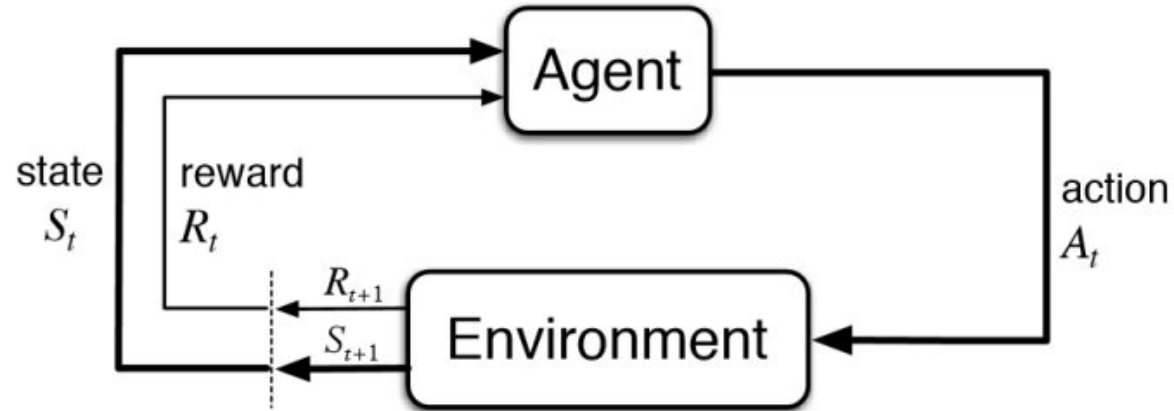


Trust Region Policy Optimization



Proximal Policy Optimization

# 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

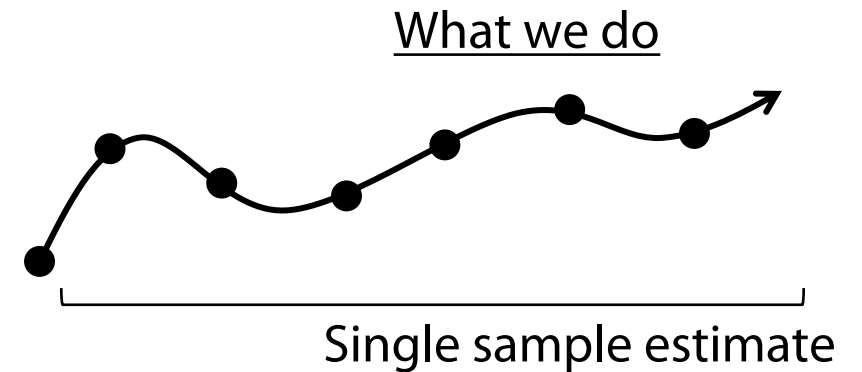
$$\begin{aligned} \nabla_{\theta} J(\theta) &= \mathbb{E}_{\tau \sim p_{\theta}(\tau)} \left[ \nabla_{\theta} \log p_{\theta}(\tau) \sum_{t=0}^T r(s_t, a_t) \right] \\ \nabla_{\theta} J(\theta) &= \mathbb{E}_{\substack{s_0 \sim p(s_0) \\ s_{t+1} \sim p(s_{t+1} | s_t, a_t) \\ a_t \sim \pi(a_t | s_t)}}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

(Monte-Carlo approximation)

# What makes policy gradient challenging?

Hard to tell what matters without many samples

$$\begin{aligned} \nabla_{\theta} J(\theta) &= \int p_{\theta}(\tau) \nabla_{\theta} \log p_{\theta}(\tau) d\tau \\ &\approx \frac{1}{N} \sum_{i=0}^N \sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t^i | s_t^i) \sum_{t'=0}^T r(s_{t'}^i, a_{t'}^i) \end{aligned}$$



For every (s, a) pair, weight by only the sum of rewards in the current trajectory

Couples together all actions

Susceptible to scale variations

Susceptible to lucky samples

Makes policy gradient unstable, requires huge numbers of samples and huge batch size

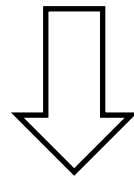
# Variance Reduction with Causality

Idea: Trajectory returns depend on past and future, but we only care about the future, since actions cannot affect the past. Instead, consider **“return-to-go”**

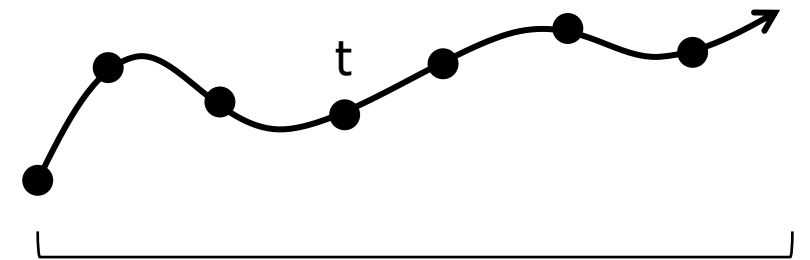
$$\approx \frac{1}{N} \sum_{i=0}^N \sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t^i | s_t^i) \underbrace{\sum_{t'=0}^T r(s_{t'}^i, a_{t'}^i)}$$

Includes  $t' < t$

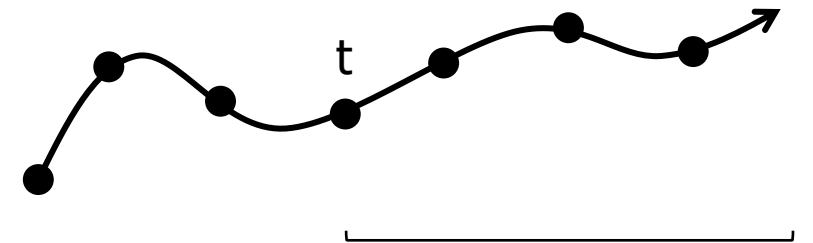
Ignore past terms



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




Full trajectory return



Return to go

# Variance Reduction with a Baseline

Idea: We can reduce variance by subtracting a current state dependent function from the policy gradient return

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


Baseline: Centers the returns, reduces variance

We proved this was unbiased!

# Why do baselines really reduce variance?

---

Let's define variance:  $\text{Var}[x] = E[x^2] - E[x]^2$        $\nabla_{\theta} J(\theta) = E_{\tau \sim p_{\theta}(\tau)}[\nabla_{\theta} \log p_{\theta}(\tau)(r(\tau) - b)]$

Whiteboard

# Lecture outline

---

## **Recap: Policy Gradient and Variance Reduction**



Natural Policy Gradients and Covariant Parameterization



Trust Region Policy Optimization



Proximal Policy Optimization

# Take a deeper look at REINFORCE

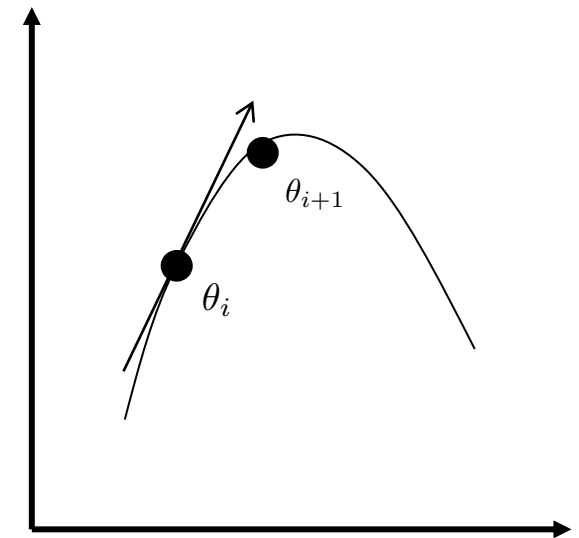
$$\nabla_{\theta} J(\theta) = \int p_{\theta}(\tau) \nabla_{\theta} \log p_{\theta}(\tau) d\tau \approx \frac{1}{N} \sum_{i=0}^N \sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t^i | s_t^i) \sum_{t'=0}^T r(s_{t'}^i, a_{t'}^i)$$

Gradient ascent is steepest ascent on linear approximation under the Euclidean metric!

$$\begin{aligned} \max \quad & J(\theta_i) + \nabla_{\theta} J(\theta)|_{\theta=\theta_i} (\theta - \theta_i) && \text{Linear approximation} \\ & (\theta - \theta_i)^T (\theta - \theta_i) \leq \epsilon && \text{Quadratic Constraint} \end{aligned}$$

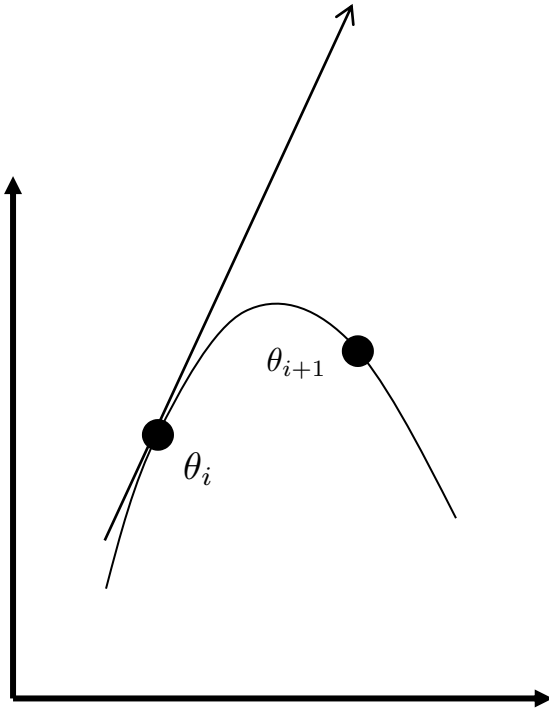


$$\theta = \theta_i + \alpha \nabla_{\theta} J(\theta)|_{\theta=\theta_i}$$



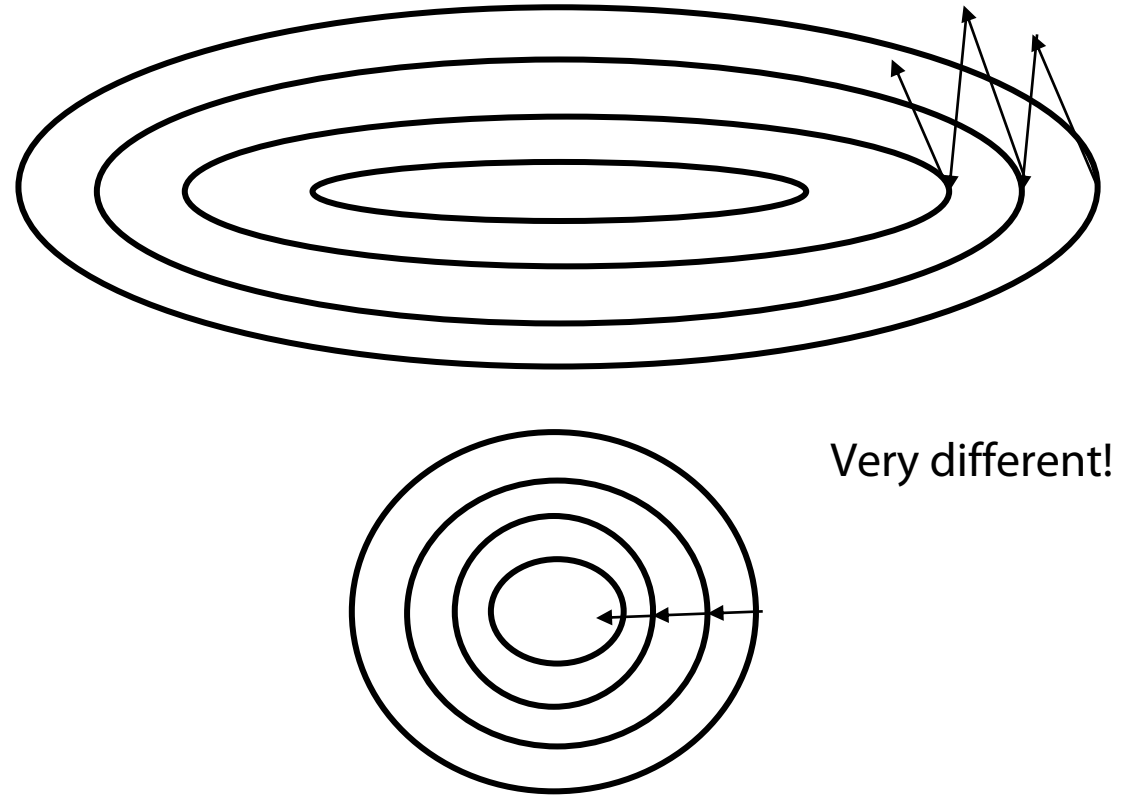
# When might this fail?

Large step sizes may cause collapse



Must use very small step sizes, slow!

Sensitive to Policy Parameterization



Can struggle for a deep neural network!

# Parameterization dependence of PG

## Sensitive to Policy Parameterization

$$L(\theta) = \theta_1 + \theta_2$$

$$\nabla_{\theta_1} L = 1$$

$$\nabla_{\theta_2} L = 1$$

$$L(\phi) = \phi_1^{0.5} + \phi_2^{-1}$$

$$\phi_1 = \theta_1^2$$

$$\phi_2 = \theta_2^{-1}$$

$$\nabla_{\phi_1} L = 0.5\phi_1^{-0.5} = 0.5\theta_1^{-1}$$

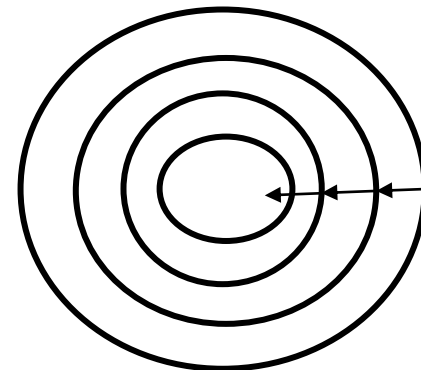
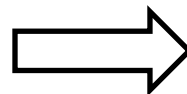
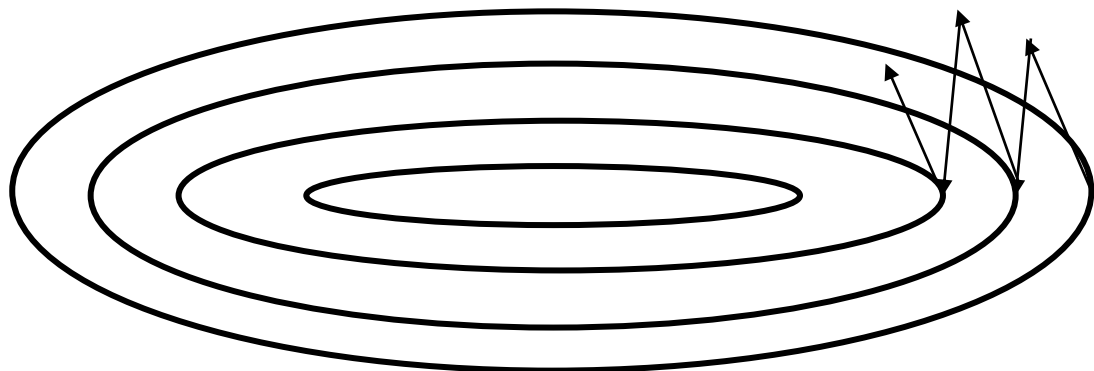
$$\nabla_{\phi_2} L = -\phi_2^{-2} = -\theta_2^2$$

←————→  
Not covariant!

# Modified Constraint on Policy Gradient

$$\begin{aligned} \max \quad & J(\theta_i) + \nabla_{\theta} J(\theta)|_{\theta=\theta_i}(\theta - \theta_i) \\ & (\theta - \theta_i)^T (\theta - \theta_i) \leq \epsilon \end{aligned}$$

$$\begin{aligned} \max \quad & J(\theta_i) + \nabla_{\theta} J(\theta)|_{\theta=\theta_i}(\theta - \theta_i) \\ & (\theta - \theta_i)^T G(\theta - \theta_i) \leq \epsilon \end{aligned}$$



$$\theta_{i+1} = \theta_i + \alpha G^{-1} \nabla_{\theta} J(\theta)|_{\theta=\theta_i}$$

Rescales according to  $G^{-1}$

Adaptive choice of  $G$  can avoid sensitivity to policy parameterization!

# Covariant Policy Gradient Updates

---

$$\begin{aligned} \max \quad & J(\theta_i) + \nabla_{\theta} J(\theta)|_{\theta=\theta_i} (\theta - \theta_i) \\ & (\theta - \theta_i)^T G (\theta - \theta_i) \leq \epsilon \end{aligned}$$

What should G be?

$$\begin{aligned} \max \quad & J(\theta_i) + \nabla_{\theta} J(\theta)|_{\theta=\theta_i} (\theta - \theta_i) \\ & D_{\text{KL}}(\pi_{\theta} || \pi_{\theta_i}) \leq \epsilon \end{aligned}$$

Let us use the constraint as  
KL divergence on the policy  
(2<sup>nd</sup> order Taylor expansion)

Measures functional distance, not parameter distance

# Resulting “Natural” Policy Gradient

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

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

2<sup>nd</sup> order approximation of KL  $\rightarrow$  Fisher Information Metric

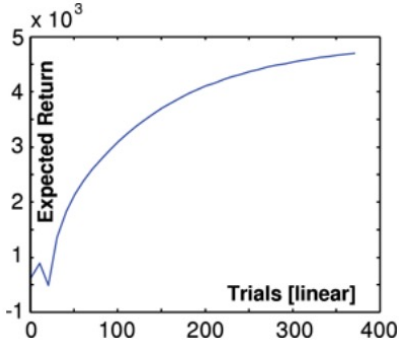
$$F = \mathbb{E}_{\pi_{\theta}} [(\nabla_{\theta} \log \pi_{\theta})(\nabla_{\theta} \log \pi_{\theta})^T]$$

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

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

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

# Natural Policy Gradient in Action



(a) Performance.



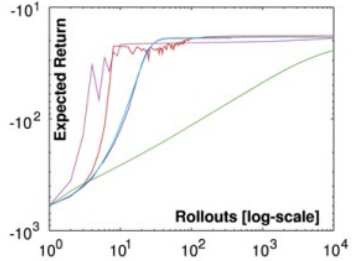
(b) Imitation learning.



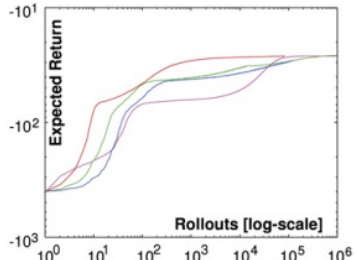
(c) Initial reproduction.



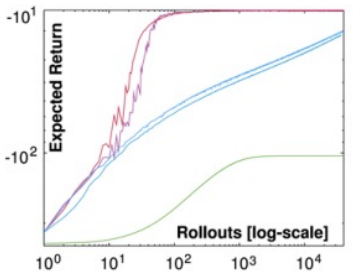
(d) After reinforcement learning.



(b) Minimum motor command with motor primitives



(c) Passing through a point with splines



(d) Passing through a point with motor primitives

- Finite Difference Gradient
- Vanilla Policy Gradient with constant baseline
- Vanilla Policy Gradient with time-variant baseline
- Episodic Natural Actor-Critic with single offset basis functions
- Episodic Natural Actor-Critic with time-variant offset basis functions

# Lecture outline

---

**Recap: Policy Gradient and Variance Reduction**



**Natural Policy Gradients and Covariant Parameterization**



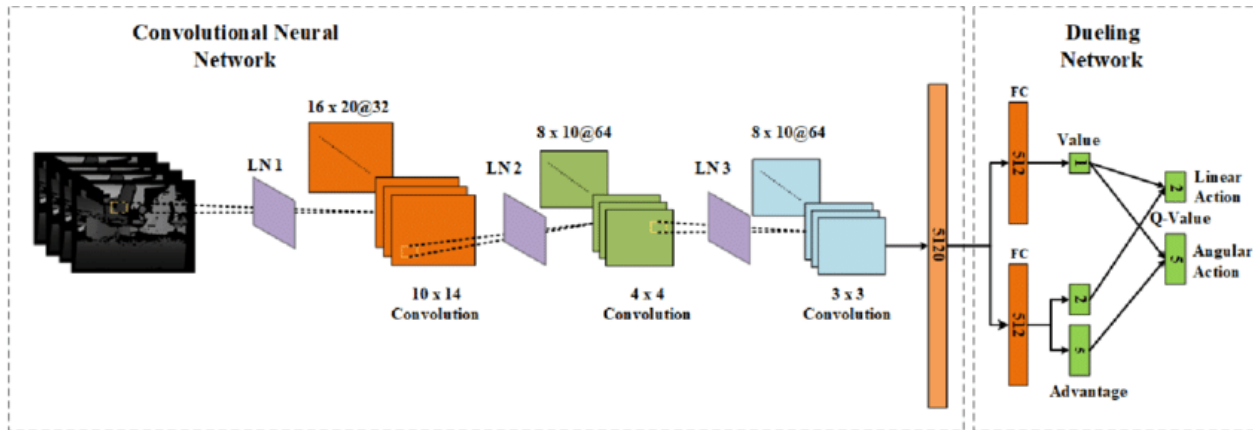
Trust Region Policy Optimization



Proximal Policy Optimization

# Natural Policy Gradient - is it enough?

## Huge matrix inversion

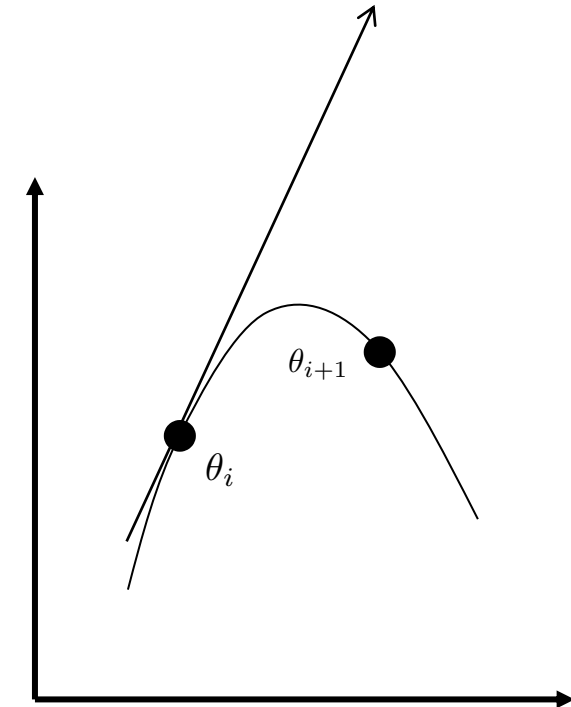


$$F - R^{d \times d}$$

For a standard convnet – d is in the millions

Hessian is way out of memory / hard to invert!

## Step-size?



Can easily overstep and collapse performance

Also, only a single gradient step at a time before recollecting data!

# Trust Region Policy Optimization

3 key ideas:

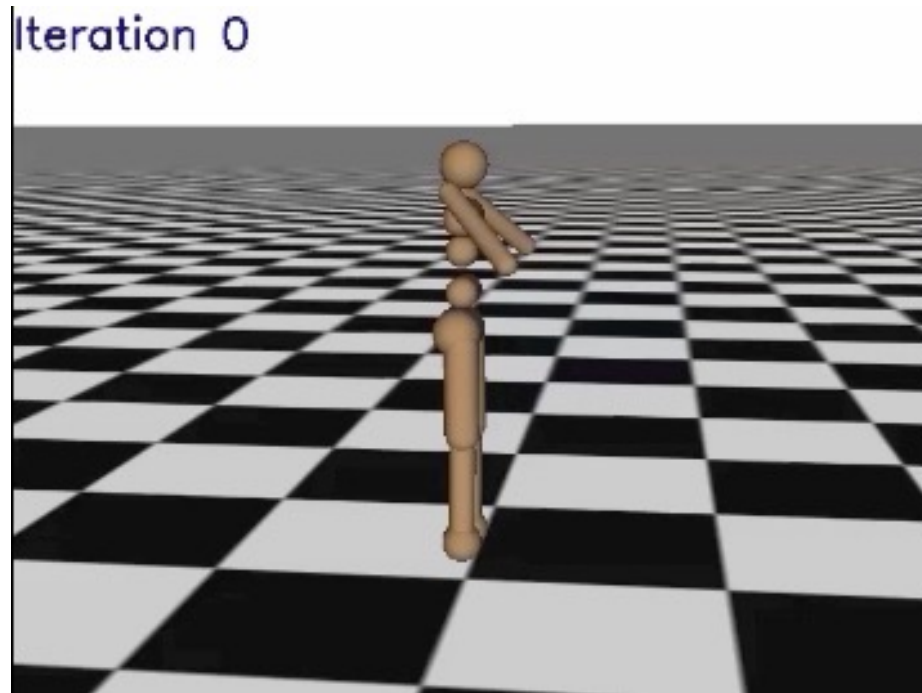
1. Huge matrix inversion  $\rightarrow$  conjugate gradient method
2. On-policy updates  $\rightarrow$  importance sampled objective
3. Step size may be too large  $\rightarrow$  backtracking line search



# Trust Region Policy Optimization

3 key ideas:

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



# Trust Region Policy Optimization – Conjugate Gradient

---

Challenging to compute  $F^{-1}$  and then get  $F^{-1}g$



Convert into an iterative minimization problem!

Solution to

$$Fx = g$$

same as

Solution to

$$\min_x \frac{1}{2} x^T F x - x^T g + c$$

# Trust Region Policy Optimization – Conjugate Gradient

Challenging to compute  $F^{-1}$  and then get  $F^{-1}g$

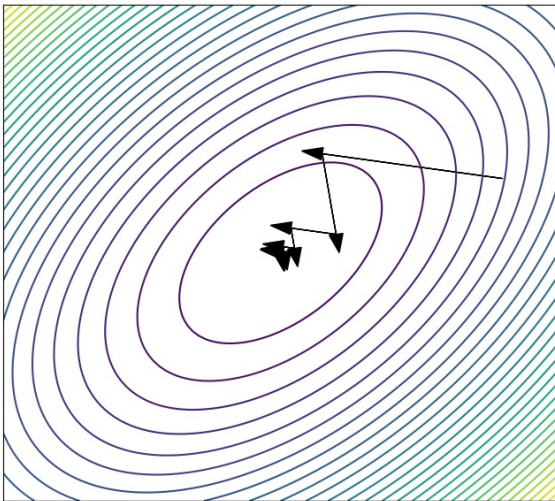


Convert into an iterative minimization problem!

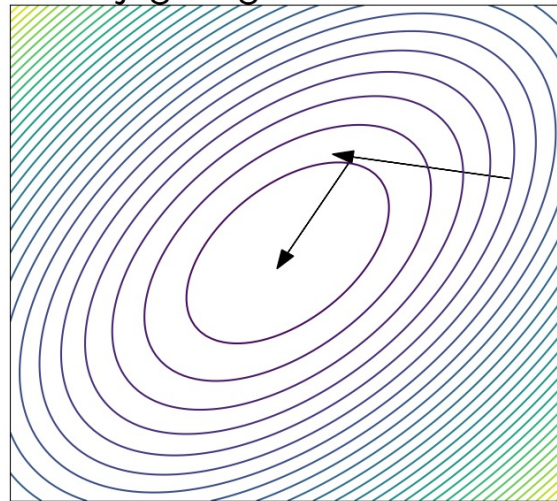


Solve with conjugate gradient

Gradient descent

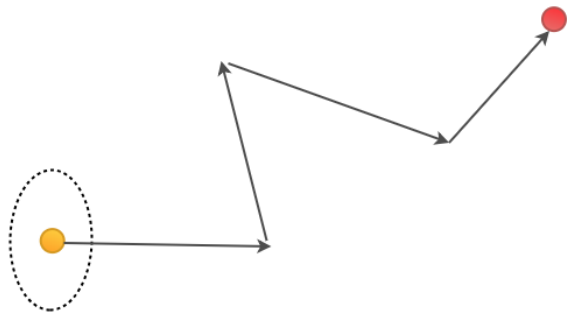


Conjugate gradient descent



Do coordinate descent in geometry aligned orthogonal directions

# Trust Region Policy Optimization – Conjugate Gradient



Gradient ascent

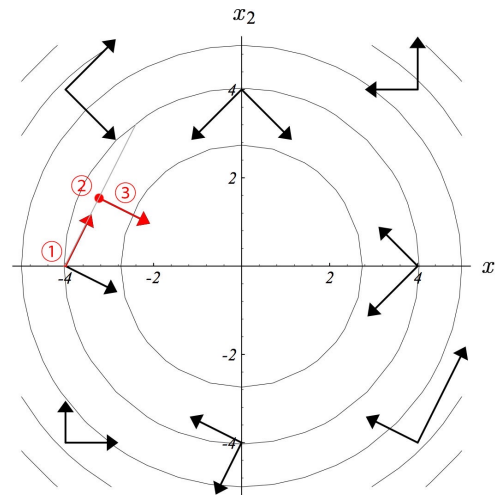
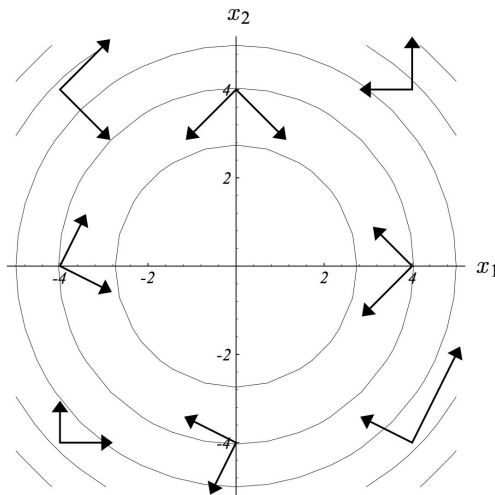


Conjugate gradient

$$\min_x \frac{1}{2} x^T F x - x^T g + c$$

Find search directions at every step that are F-orthogonal with previous directions

$$d_{(i)}^T F d_{(i)} = 0$$



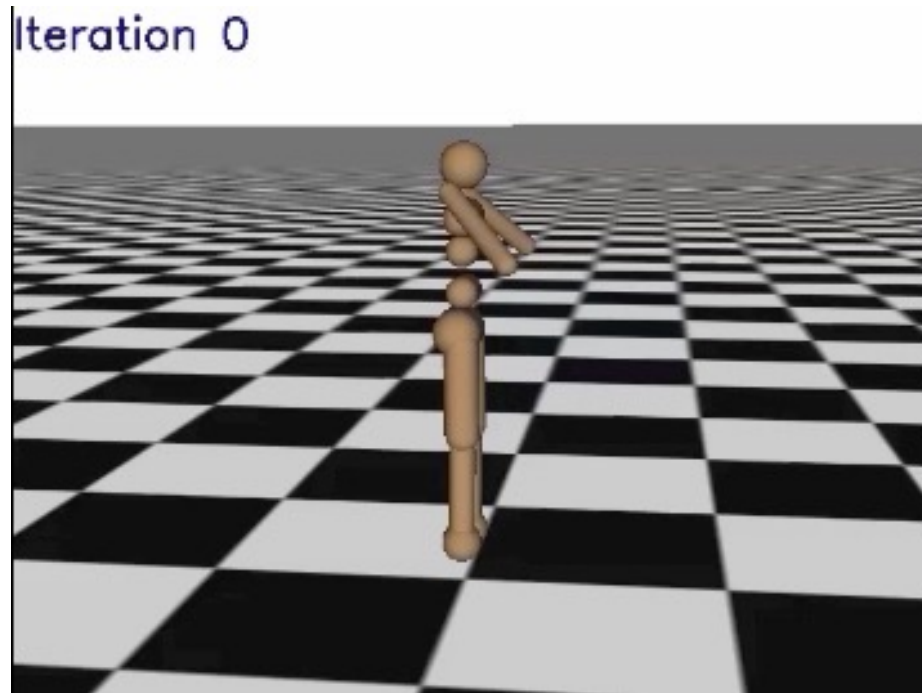
Converges in approx N steps!

Only requires matrix-vector product

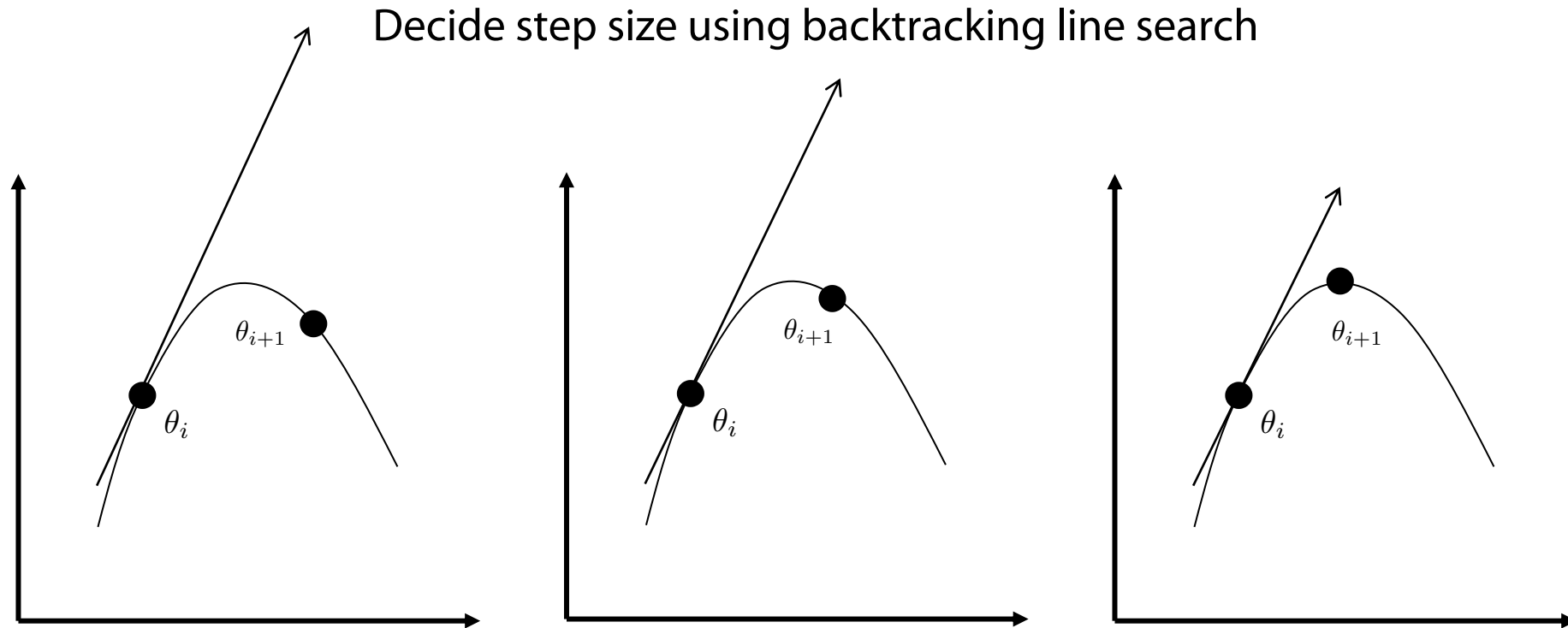
# Trust Region Policy Optimization

3 key ideas:

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



# Backtracking line search requires off-policy evaluation



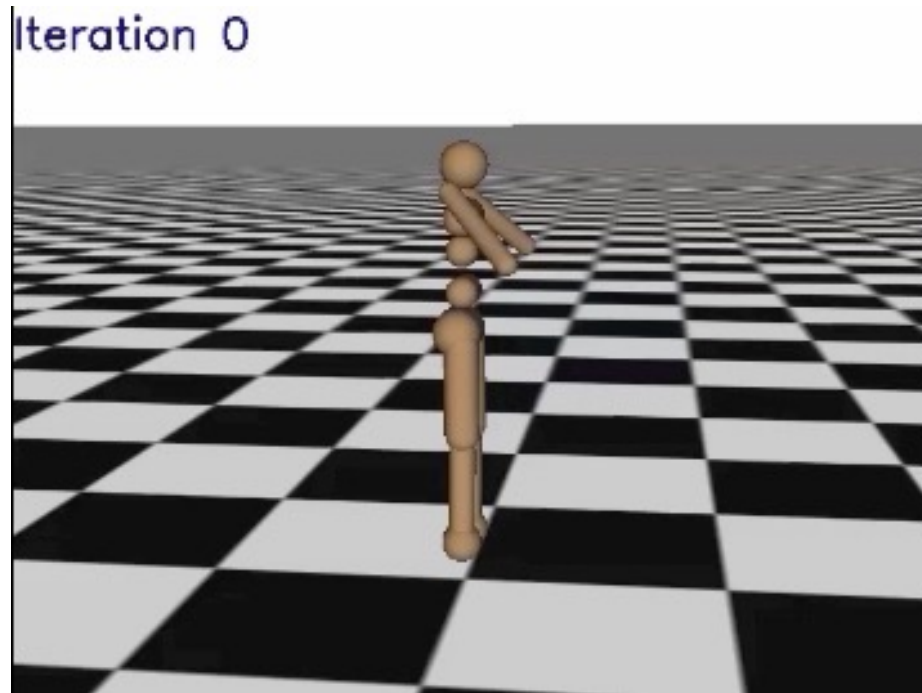
1. Choose parameter  $\beta \in (0, 1)$ , given search direction  $s = F^{-1}g$
2. Compute maximal step size such that constraint is satisfied -  $\frac{1}{2}(ts)^T F(ts) = \epsilon \rightarrow t = \sqrt{\frac{2\epsilon}{s^T F s}}$
3. While  $J(\theta_i + ts) < J(\theta_i)$ , set  $t = \beta t$

Backtracking

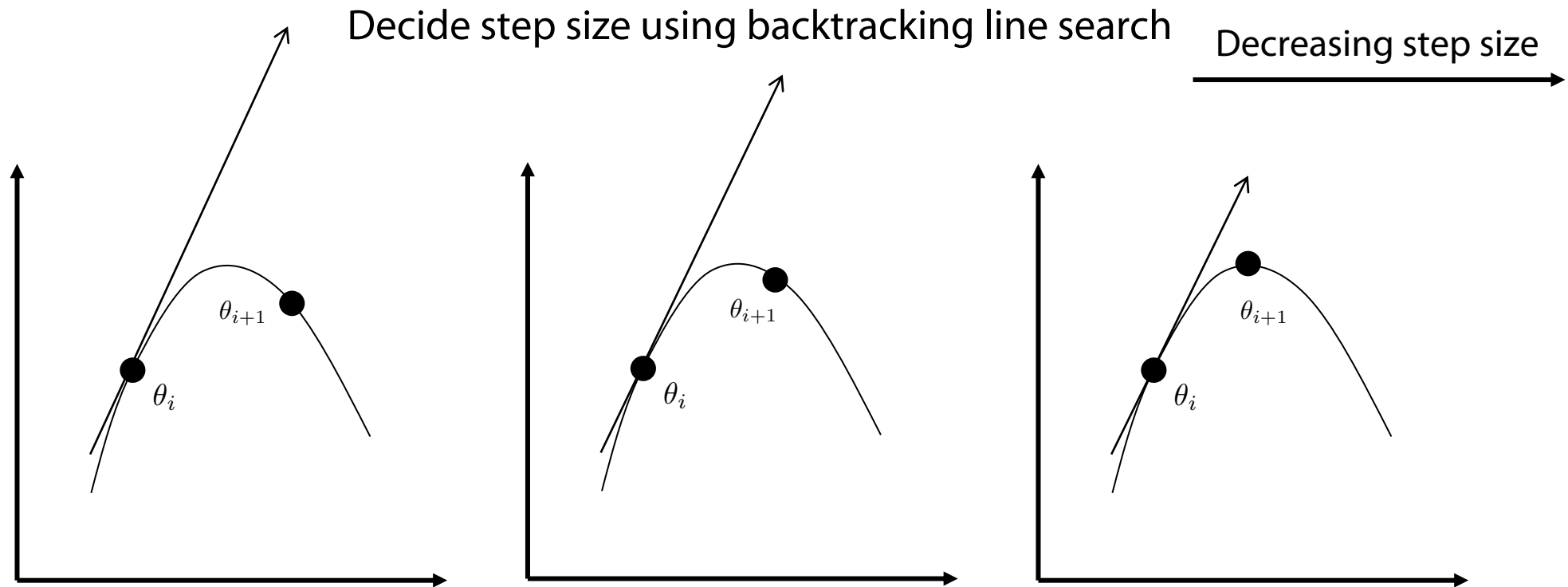
# Trust Region Policy Optimization

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



# Trust Region Policy Optimization – Backtracking line search



1. Choose parameter  $\beta \in (0, 1)$ , given search direction  $s = F^{-1}g$
2. Compute maximal step size such that constraint is satisfied
3. While  $J(\theta_i + ts) < J(\theta_i)$ , set  $t = \beta t$

Need to have samples from  $\theta_i + ts$

# Off Policy Policy Gradient – Importance Sampling

How do we evaluate  $J(\theta)$  without resampling in the environment?

$$J(\theta) = \mathbb{E}_{\tau \sim p_{\theta}(\tau)} [R(\tau)]$$

Can we better re-use old data?

$$J(\theta) = \int p_{\theta}(\tau) R(\tau)$$

Can rederive policy gradient from this perspective

$$J(\theta) = \int \frac{q(\tau)}{q(\tau)} p_{\theta}(\tau) R(\tau)$$

Has numerical challenges

$$J(\theta) = \mathbb{E}_{\tau \sim q(\tau)} \left[ \frac{p_{\theta}(\tau)}{q(\tau)} R(\tau) \right]$$

# Expanding the Importance Sampling Objective

$$J(\theta) = \int p_{\theta}(\tau) R(\tau)$$

$$J(\theta) = \int \frac{q(\tau)}{q(\tau)} p_{\theta}(\tau) R(\tau)$$

$$J(\theta) = \mathbb{E}_{\tau \sim q(\tau)} \left[ \frac{p_{\theta}(\tau)}{q(\tau)} R(\tau) \right]$$

$$\frac{p_{\theta}(\tau) = \cancel{p(s_0)} \prod_{t=1}^T \cancel{p(s_{t+1}|s_t, a_t)} \pi_{\theta}(a_t|s_t)}{q(\tau) = \cancel{p(s_0)} \prod_{t=1}^T \cancel{p(s_{t+1}|s_t, a_t)} q(a_t|s_t)}$$

$$J(\theta) = \mathbb{E}_{\tau \sim q(\tau)} \left[ \left( \prod_{t=1}^T \frac{\pi_{\theta}(a_t|s_t)}{q(a_t|s_t)} \right) \left( R(\tau) \right) \right]$$

Numerically unstable, high variance!

# Low Variance Surrogate Objective

Cannot evaluate without resampling

Original Objective

$$J(\theta) = \mathbb{E}_{\tau \sim p_{\theta}(\tau)} [R(\tau)]$$

$$J(\theta) = \mathbb{E}_{s \sim p_{\theta}(s), a \sim \pi_{\theta}(a|s)} [Q(s, a)]$$

Importance Sampling (ish)

$$J(\theta) = \mathbb{E}_{s \sim p_{\theta_i}(s), a \sim \pi_{\theta_i}(a|s)} \left[ \frac{\cancel{p_{\theta}(s)} \pi_{\theta}(a|s)}{\cancel{p_{\theta_i}(s)} \pi_{\theta_i}(a|s)} Q(s, a) \right]$$

(Surrogate Objective)

$$\approx \mathbb{E}_{s \sim p_{\theta_i}(s), a \sim \pi_{\theta_i}(a|s)} \left[ \frac{\pi_{\theta}(a|s)}{\pi_{\theta_i}(a|s)} Q(s, a) \right]$$

↑  
If policies are close, we can show that this is not so bad!

↑  
Often replaced by  $A = Q - V$

# 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  $\theta_0$

**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**

# TRPO in action

---

Trust Region Policy Optimization

# Why might TRPO not be enough?

---

## Algorithm 3 Trust Region Policy Optimization

---

Input: initial policy parameters  $\theta_0$

**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**

---

Advantage estimation is too high variance

Optimization expensive/unstable

# Better Advantage Estimation - Generalized Advantage Estimation

Advantage estimator

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

High variance!

N step advantage estimator

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

N-1 step advantage estimator

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

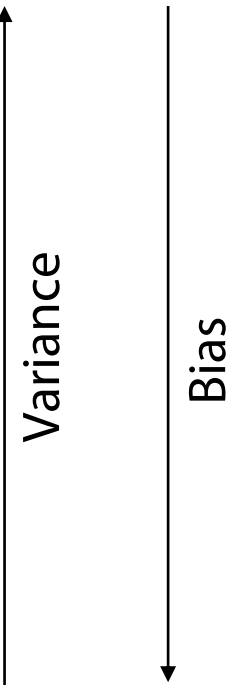
⋮

2 step advantage estimator

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

1 step advantage estimator

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



# Generalized Advantage Estimation

Sum up all the estimators in a geometric sum

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

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

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

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

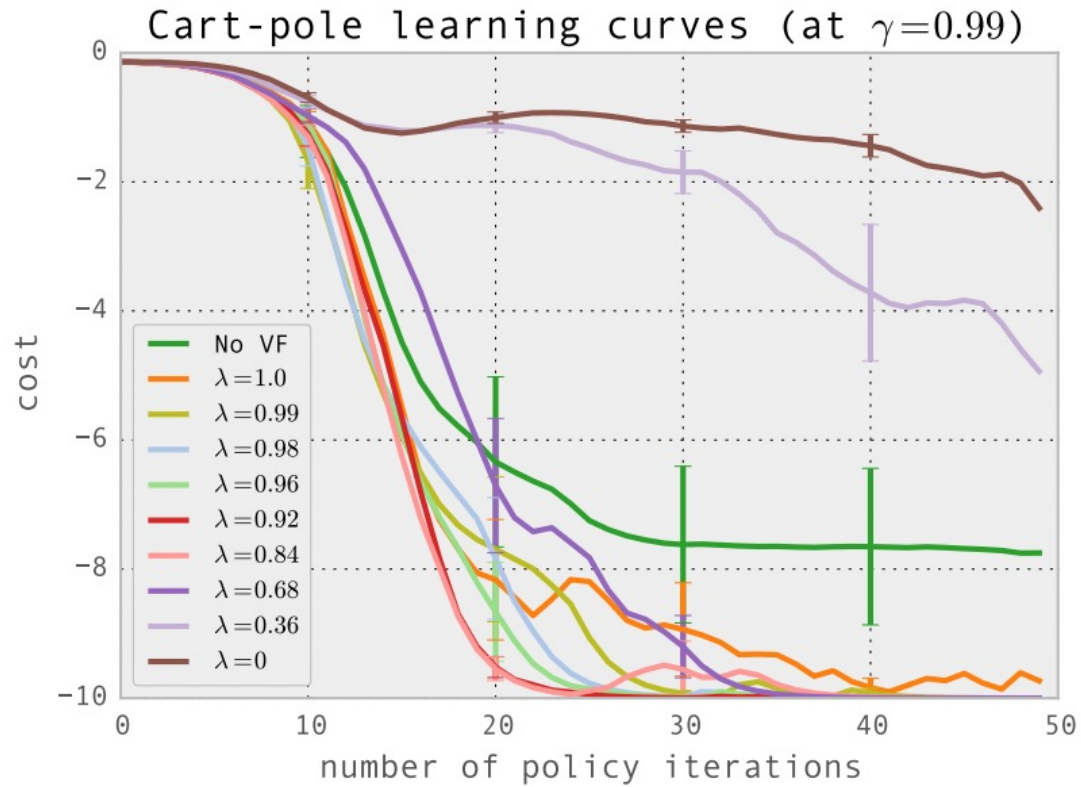
Geometric sum

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

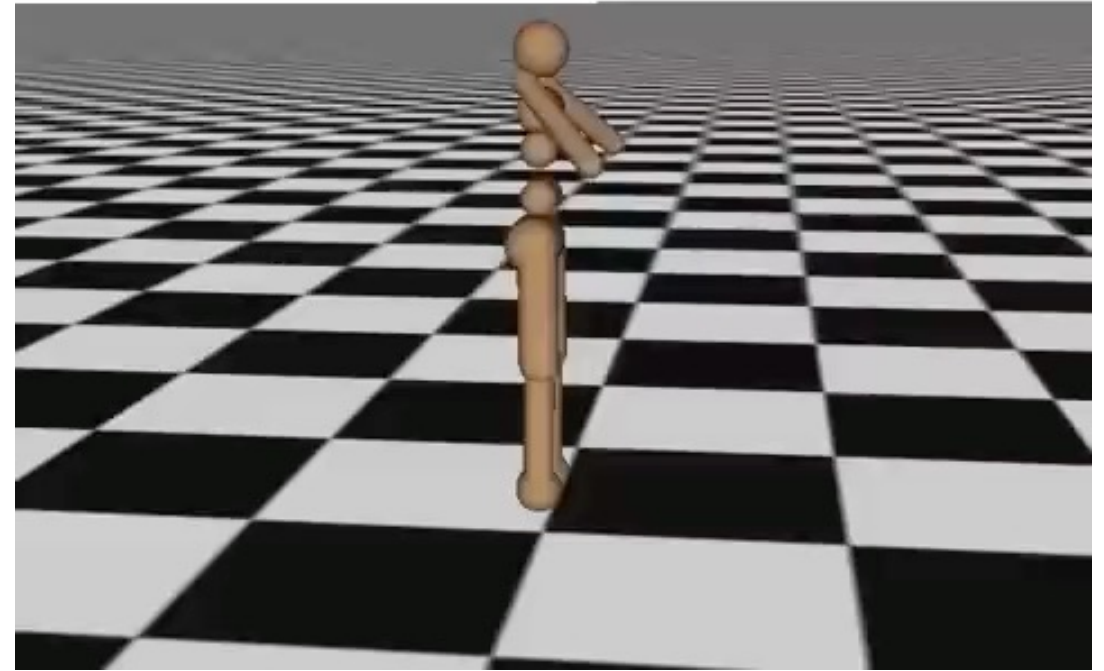
$\lambda$  controls bias-variance tradeoff

Best of both worlds – very similar idea to eligibility traces

# Generalized Advantage Estimation in Action



Iteration 0



# Lecture outline

---

**Recap: Policy Gradient and Variance Reduction**



**Natural Policy Gradients and Covariant Parameterization**



**Trust Region Policy Optimization**



Proximal Policy Optimization

# Avoiding Second Order Optimization

---

**Algorithm 3** Trust Region Policy Optimization

---

Input: initial policy parameters  $\theta_0$

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

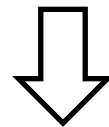
Perform backtracking line search with exponential decay to obtain final update

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

**end for**

---

Expensive second order optimization, can we avoid?

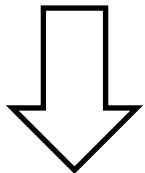


What if we just restricted how much the policy changes directly!

# Proximal Policy Optimization Update

## Trust Region Policy Optimization

$$\max J(\theta_i) + \nabla_{\theta} J(\theta)|_{\theta=\theta_i} (\theta - \theta_i)$$
$$D_{\text{KL}}(\pi_{\theta} || \pi_{\theta_i}) \leq \epsilon$$



## Proximal Policy Optimization

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

Restrict the amount the policy moves

# 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

# Whats up with this min term?

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

Why take the minimum here?

Pessimistic about large claimed improvements, but don't ignore harmful changes.

Case 1:  $A > 0$

$r > 1 + \epsilon$     Cap reward for increasing probability of a good action too much

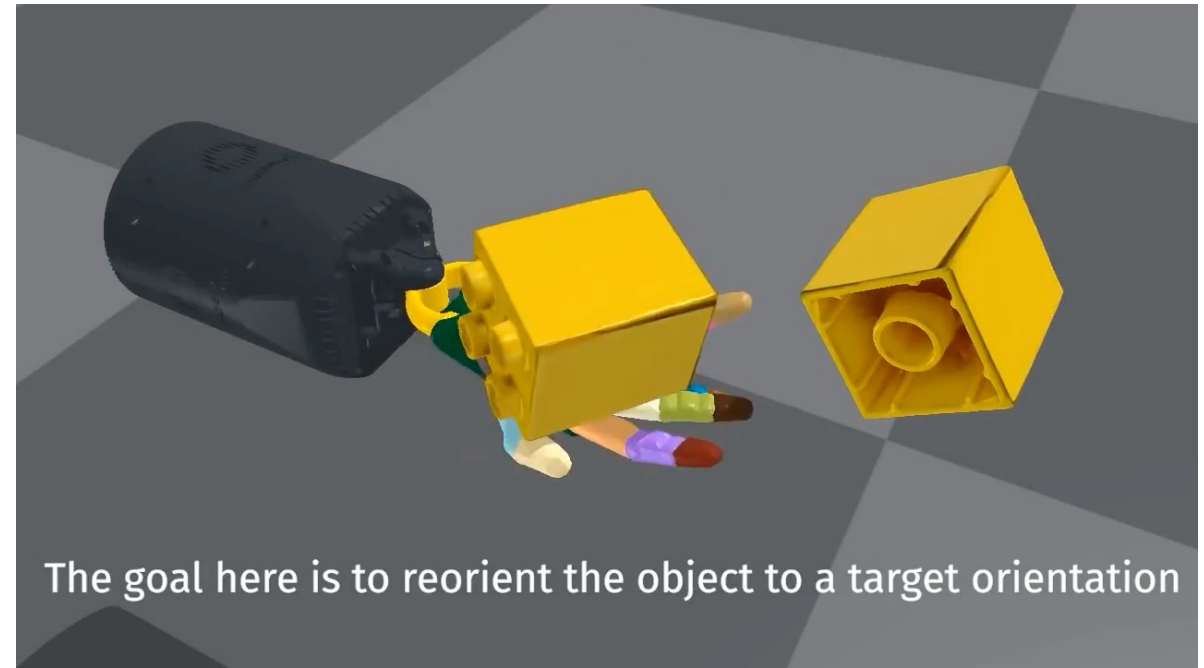
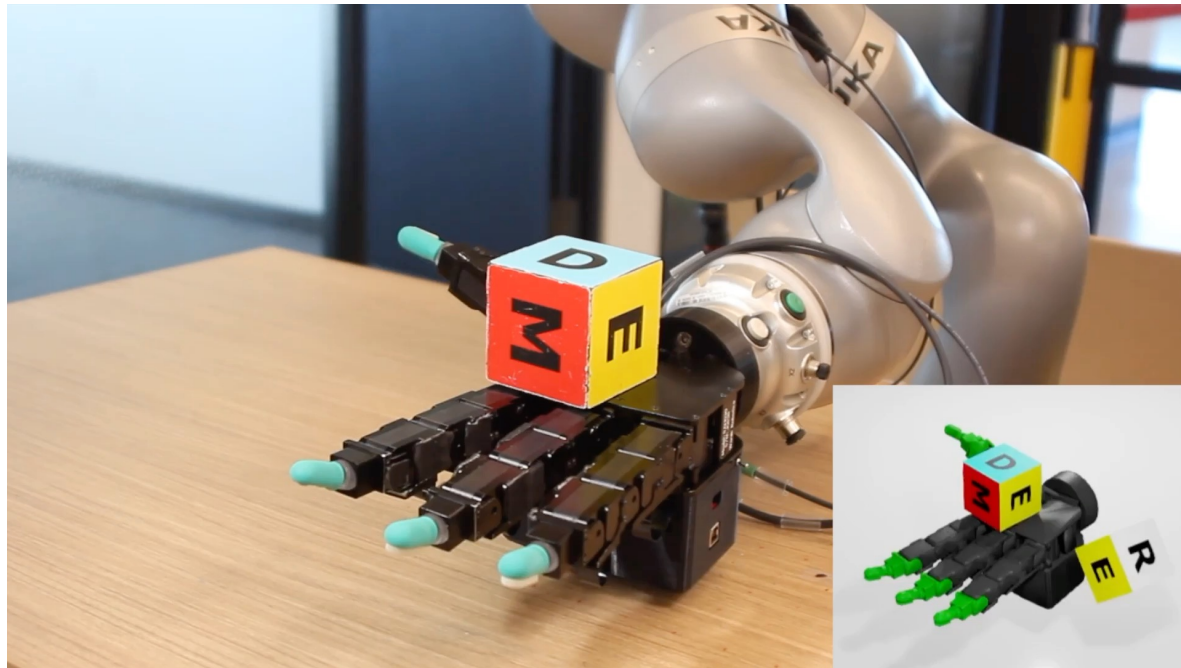
$r < 1 - \epsilon$     Keep penalizing decreasing probability of a good action

Case 2:  $A < 0$

$r > 1 + \epsilon$     Keep penalizing increasing probability of a bad action

$r < 1 - \epsilon$     Cap reward for decreasing probability of a bad action too much

# PPO in Action



# PPO in Action



Scene 1: Attacking Mid

**ACTIONS** **OBSERVATIONS**

Action: Ability Nethertoxin

|                          |                          |                          |                          |                          |                                     |                          |                          |
|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|-------------------------------------|--------------------------|--------------------------|
|                          |                          |                          |                          |                          |                                     |                          |                          |
| <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |

Target Necrophos

|                          |                          |                          |                          |                          |                                     |                          |                          |
|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|-------------------------------------|--------------------------|--------------------------|
|                          |                          |                          |                          |                          |                                     |                          |                          |
| <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |

Offset X

|                          |                          |                          |                          |                          |                                     |                          |                          |                          |
|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|-------------------------------------|--------------------------|--------------------------|--------------------------|
| -400                     | -300                     | -200                     | -100                     | 0                        | 100                                 | 200                      | 300                      | 400                      |
| <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |

Offset Y

|                          |                          |                          |                          |                                     |                          |                          |                          |                          |
|--------------------------|--------------------------|--------------------------|--------------------------|-------------------------------------|--------------------------|--------------------------|--------------------------|--------------------------|
| -400                     | -300                     | -200                     | -100                     | 0                                   | 100                      | 200                      | 300                      | 400                      |
| <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |

Action Delay

|                                     |                          |                          |                          |
|-------------------------------------|--------------------------|--------------------------|--------------------------|
|                                     |                          |                          |                          |
| <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |



# So should we just use PPO for everything?

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

Logan Engstrom<sup>1\*</sup>, Andrew Ilyas<sup>1\*</sup>, Shibani Santurkar<sup>1</sup>, Dimitris Tsipras<sup>1</sup>,  
Firdaus Janoos<sup>2</sup>, Larry Rudolph<sup>1,2</sup>, and Aleksander Mądry<sup>1</sup>

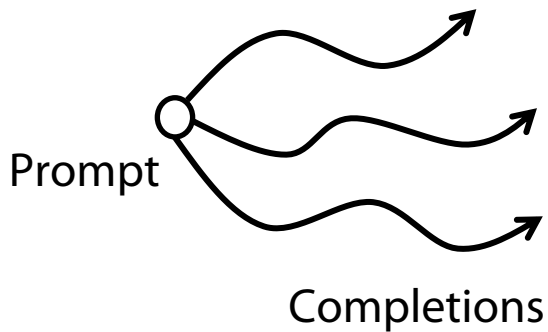
Open question!

| 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?



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

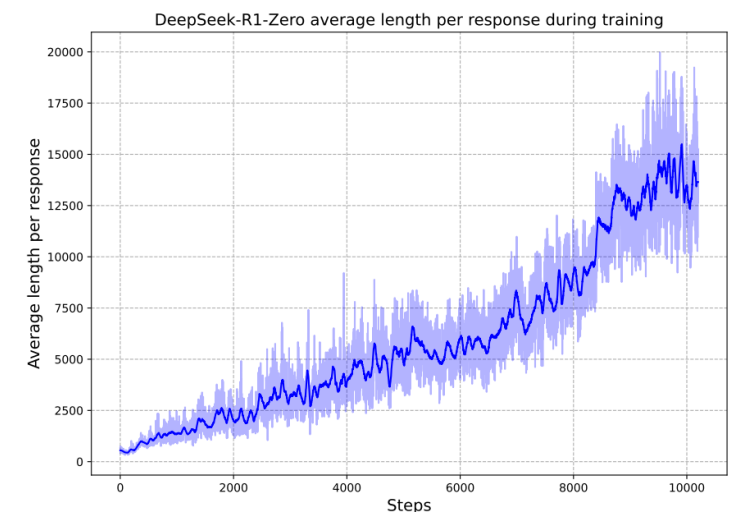
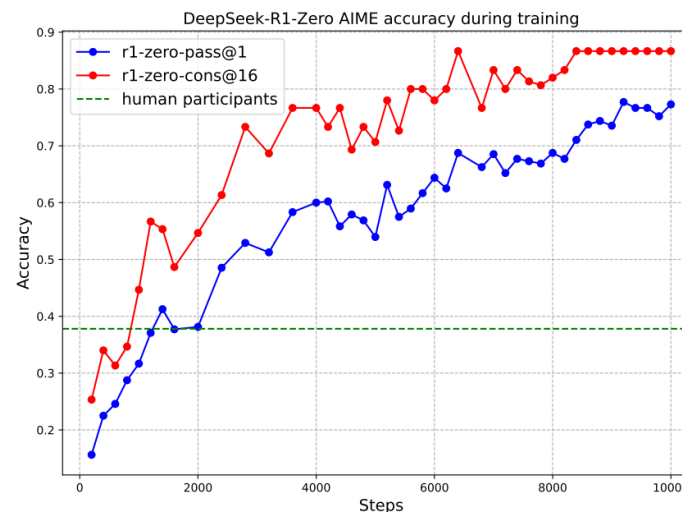


$$A(s, a) = \frac{r(s, a) - \text{mean}(r_1, r_2, \dots, r_G)}{\text{std}(r_1, r_2, \dots, r_G)}$$

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 Agarwal<sup>1</sup>, Mikael Henaff<sup>2</sup>, Sham Kakade<sup>3,1</sup>, and Wen Sun<sup>\*4</sup>

# Lecture outline

---

**Recap: Policy Gradient and Variance Reduction**



**Natural Policy Gradients and Covariant Parameterization**



**Trust Region Policy Optimization**

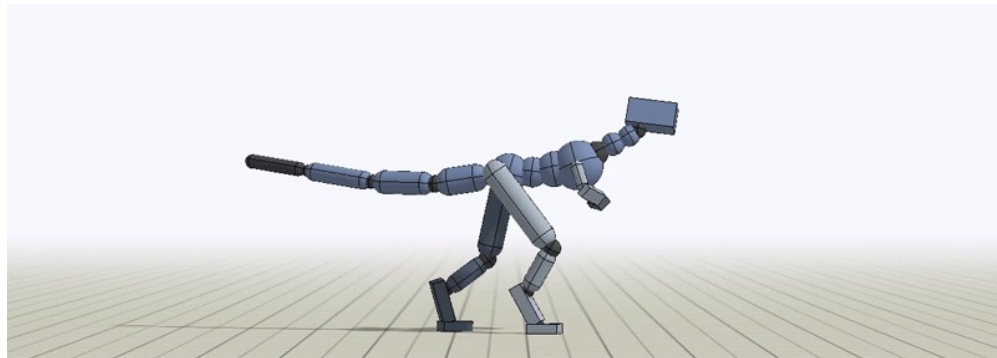
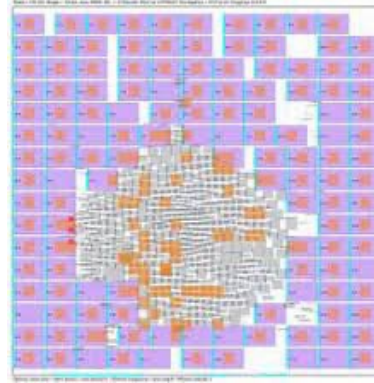


**Proximal Policy Optimization**

# Pros/Cons of Policy Gradient Methods

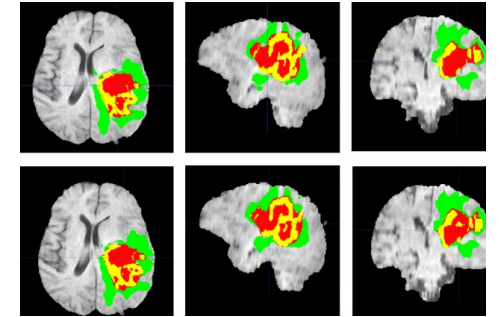
## Pros

- 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

# 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  $\eta$ ,*

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

and for

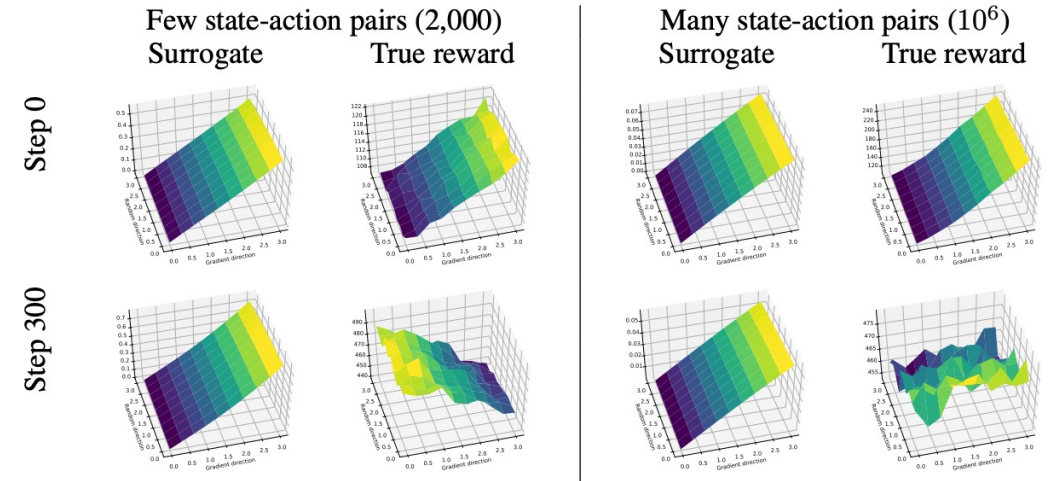
$$N \geq \frac{\|\Sigma_{K^*}\|}{\mu} \log \frac{C(K_0) - C(K^*)}{\varepsilon} \\ \times \text{poly} \left( \frac{C(K_0)}{\mu \sigma_{\min}(Q)}, \|A\|, \|B\|, \|R\|, \frac{1}{\sigma_{\min}(R)} \right),$$

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

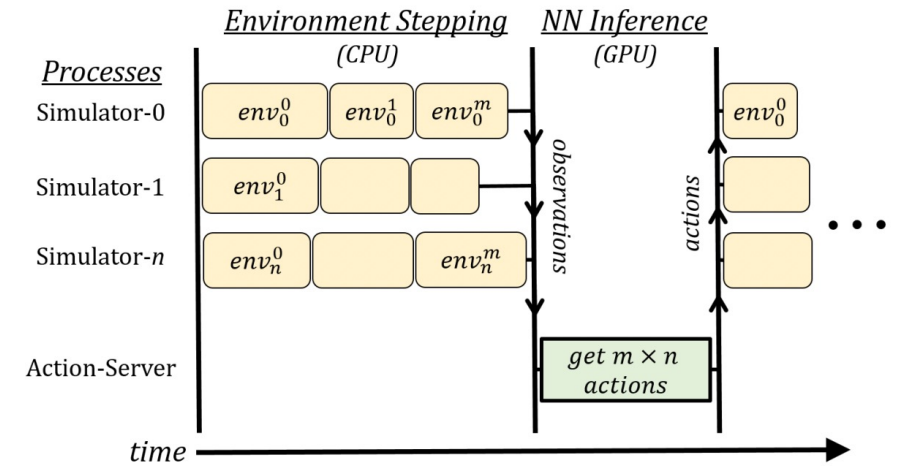
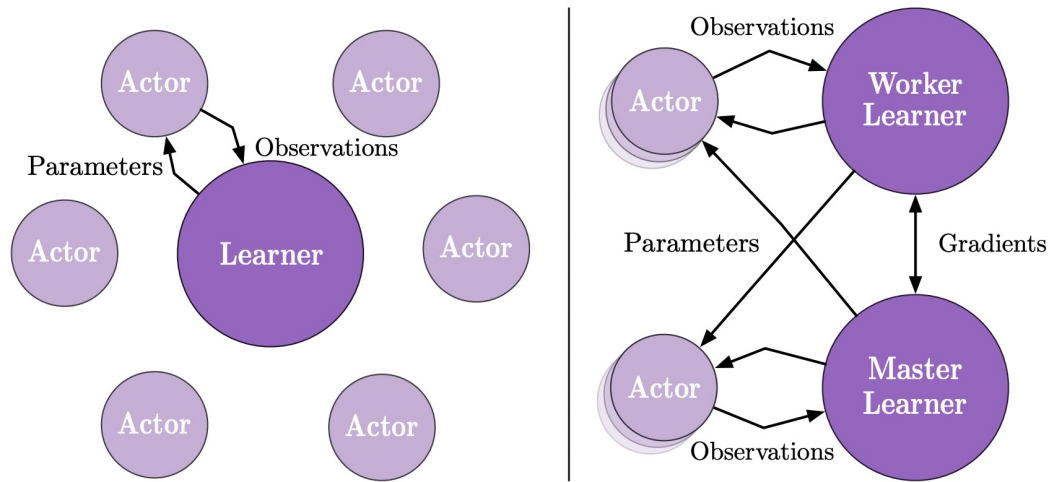
### Practical Algorithms Deviate from Theory



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

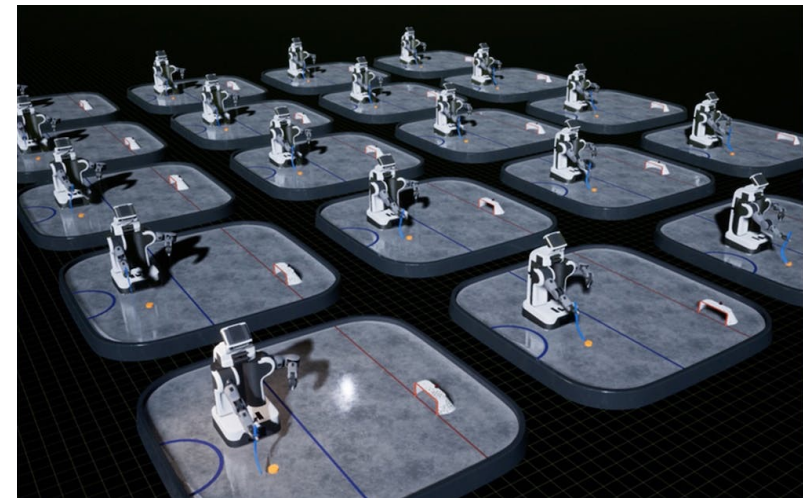
# Frontiers of Policy Gradient Research

## 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

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

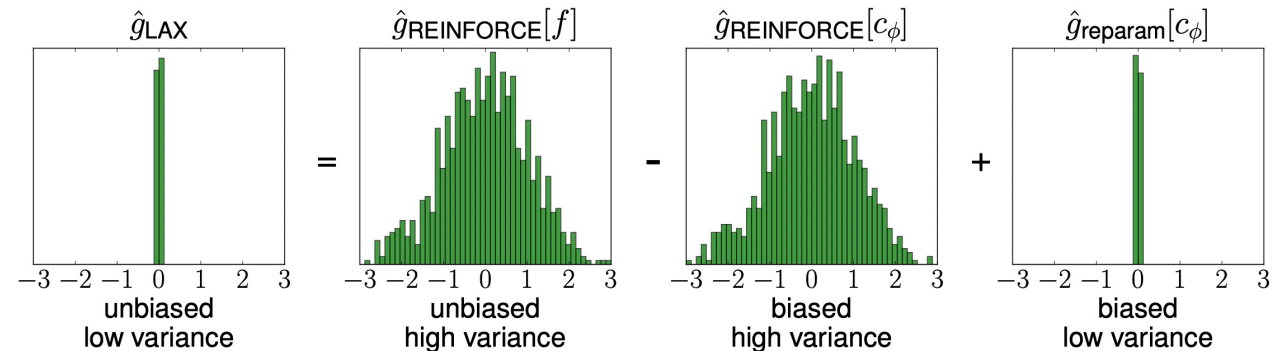
$$\nabla_{\theta} \eta(\pi_{\theta}) = \mathbb{E}_{\rho_{\pi}, \pi} \left[ \sum_{i=1}^m \nabla_{\theta} \log \pi_{\theta}(a_t^i|s_t) \left( \hat{Q}(s_t, a_t) - b_i(s_t, a_t^{-i}) \right) \right]$$

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



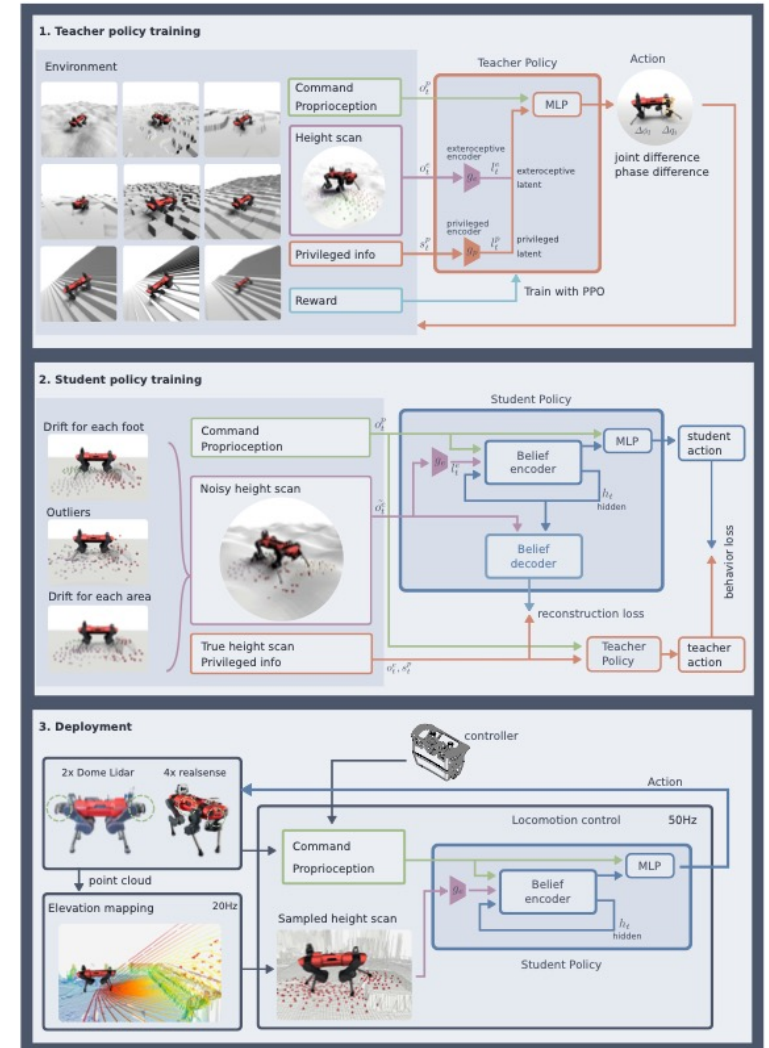
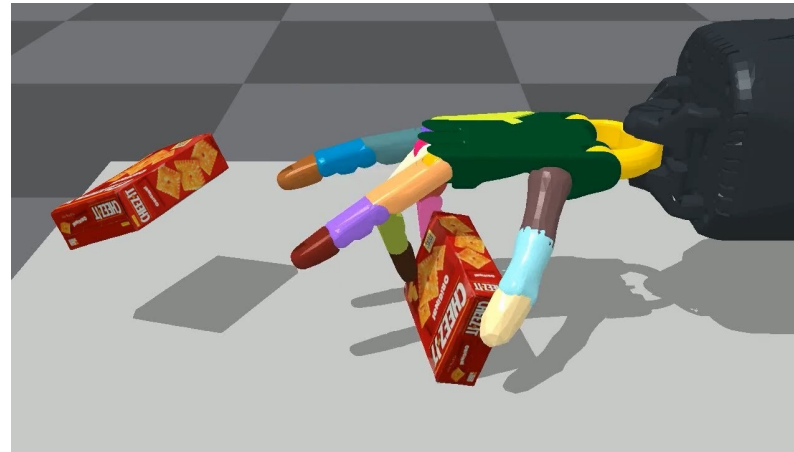
Q-Prop: Sample-Efficient Policy Gradient with An Off-Policy Critic, Gu et al '16

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

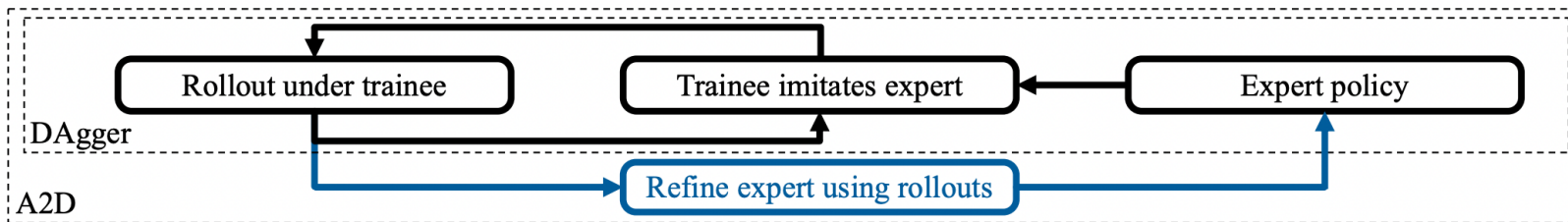
## Learning from High Dimensional Observations



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

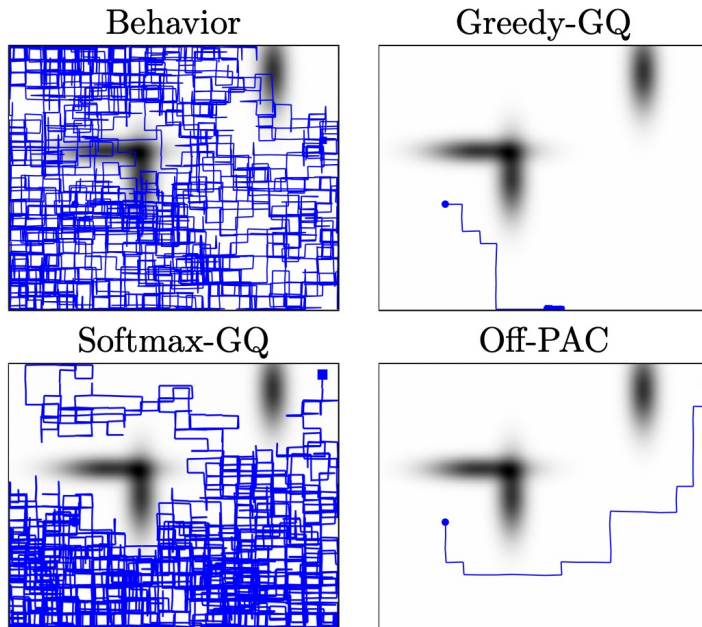
# Frontiers of Policy Gradient Research

## Bootstrapping from Prior/Off-Policy Data

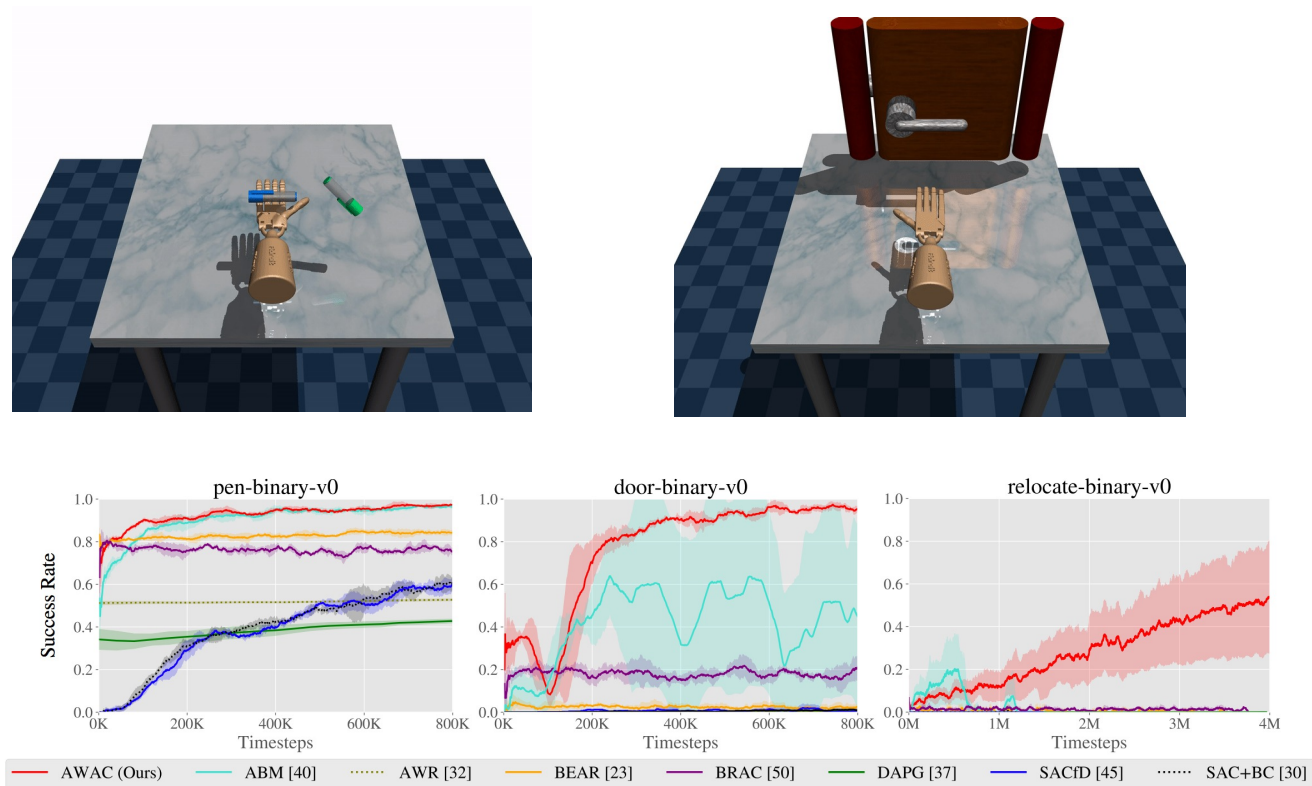
Off-policy policy gradient

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

Learning from Prior Data



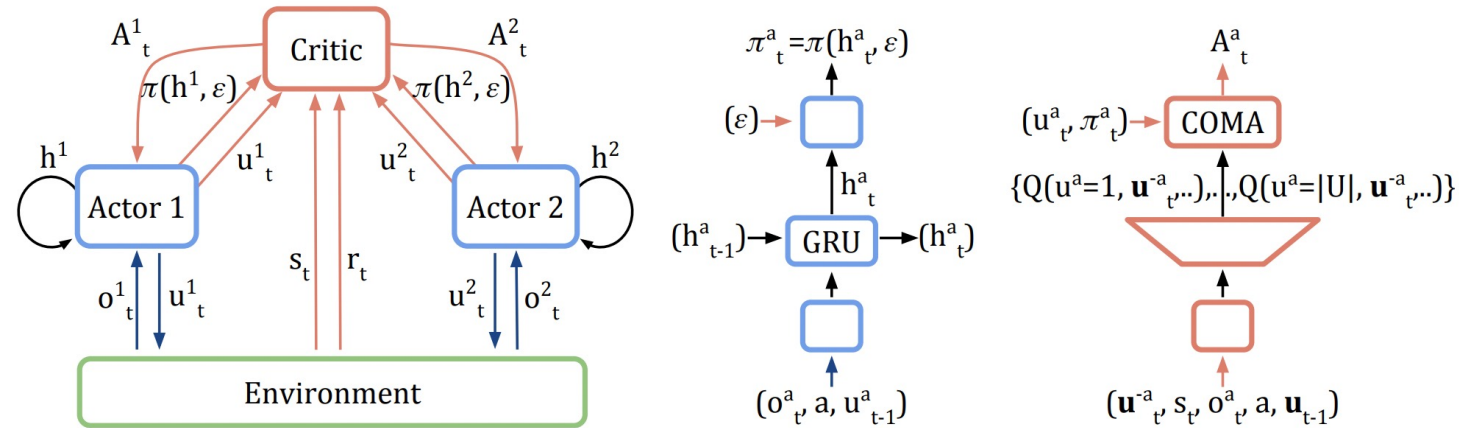
Off-Policy Actor-Critic, Degrís et al '13



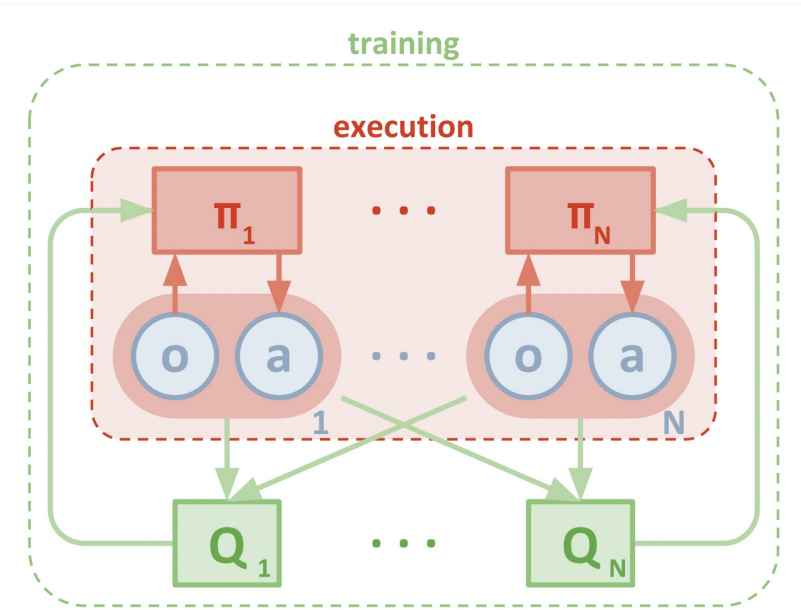
Advantage Weighted Actor Critic, Nair et al '20  
DDPGfD, Vecerik '17  
DAPG, Rajeswaran '17

# Frontiers of Policy Gradient Research

## Multi-agent policy gradient



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



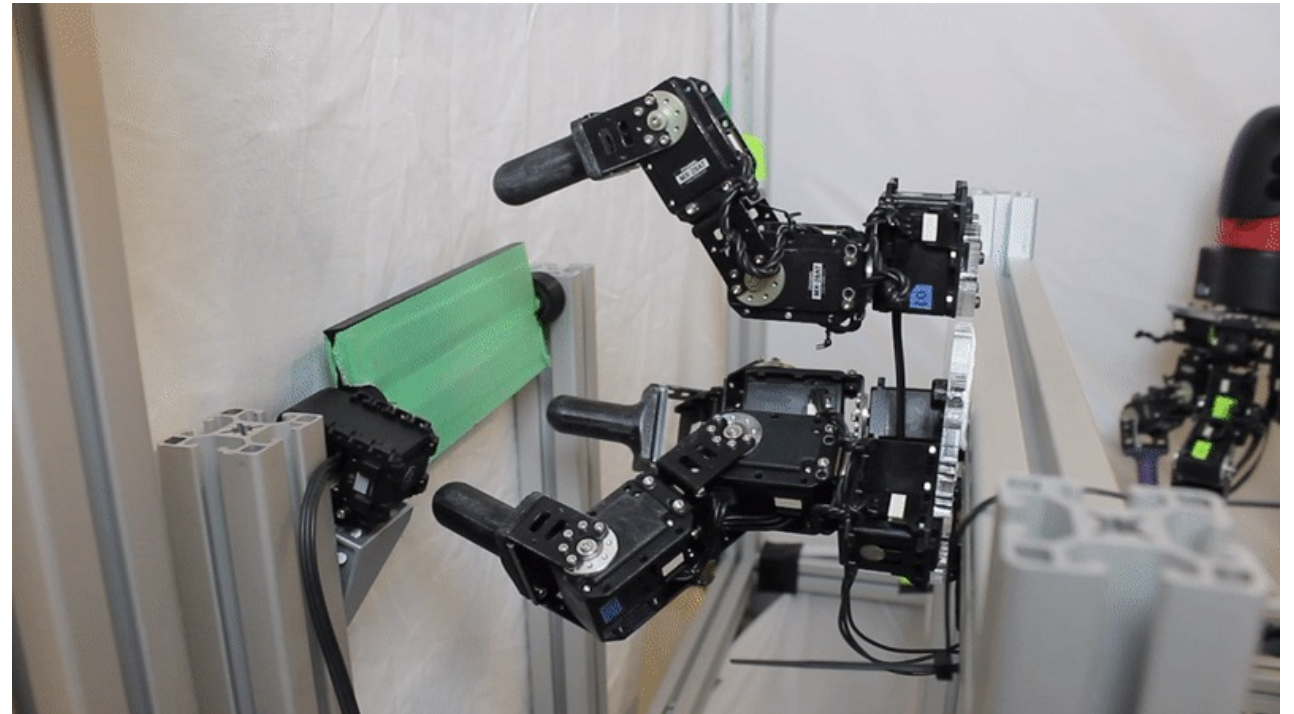
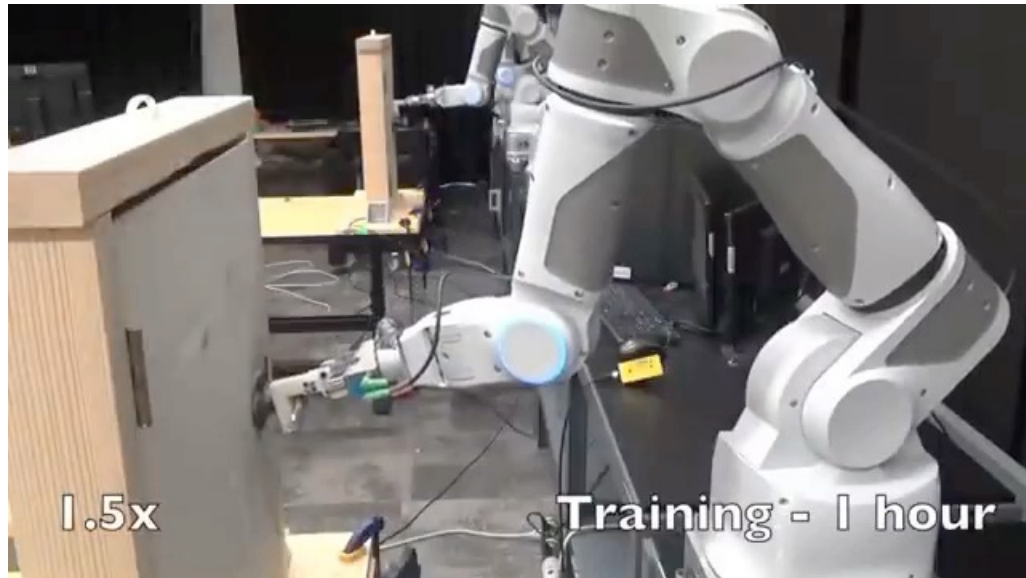
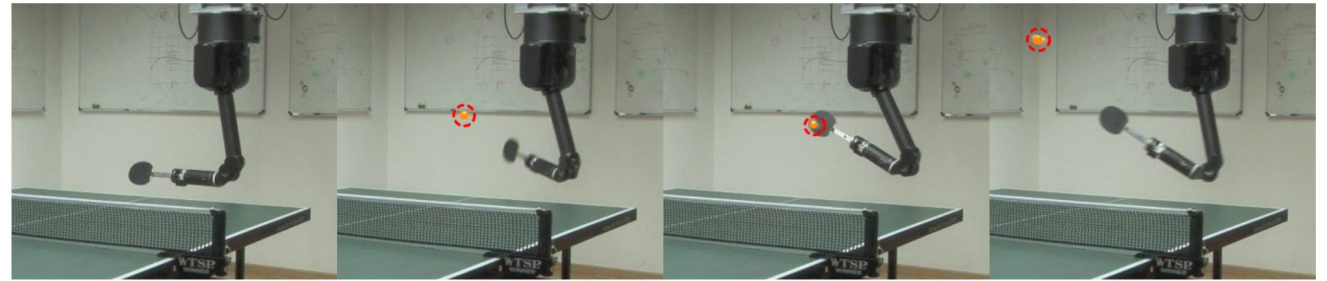
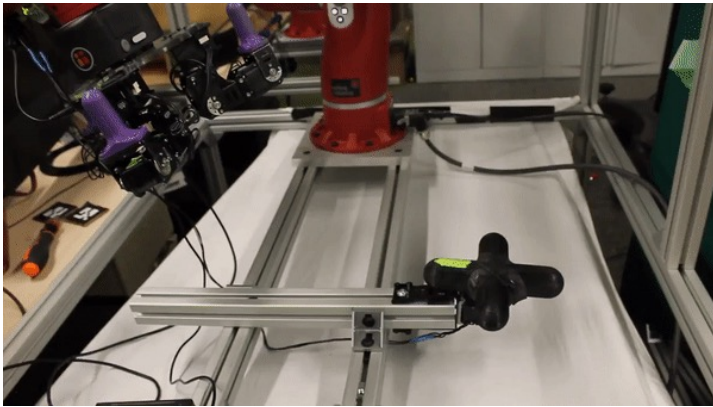
Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments, Lowe et al '17

### Primary challenges:

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

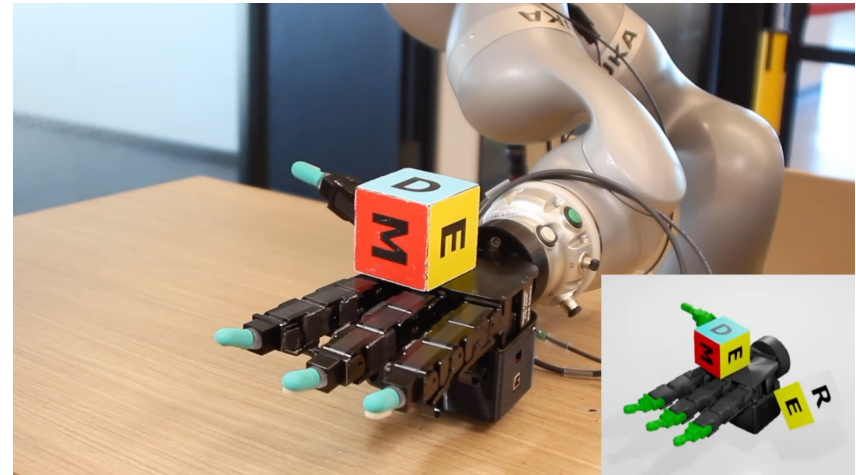
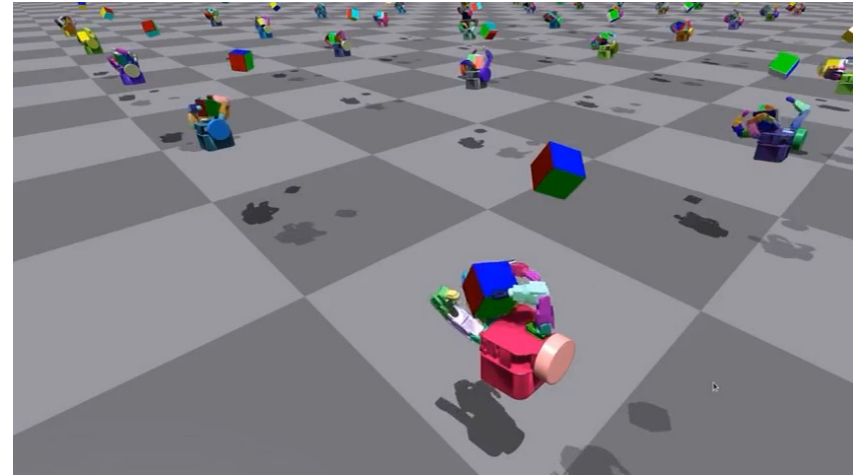
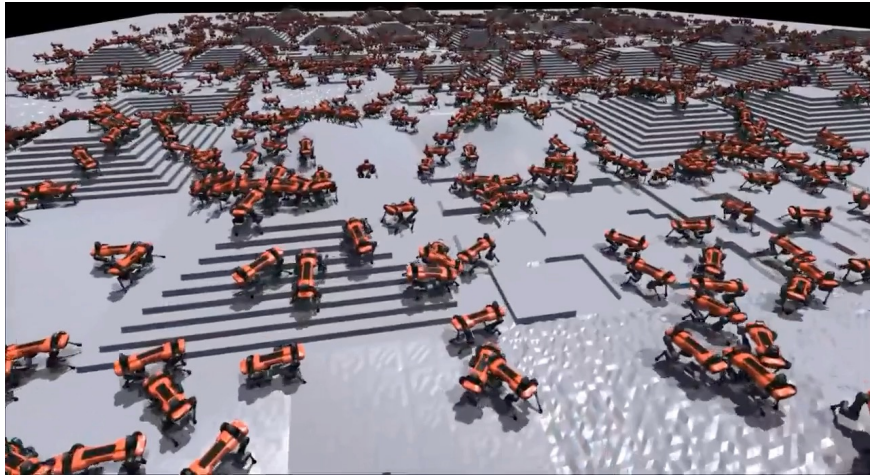
# How is this useful for robotics?

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