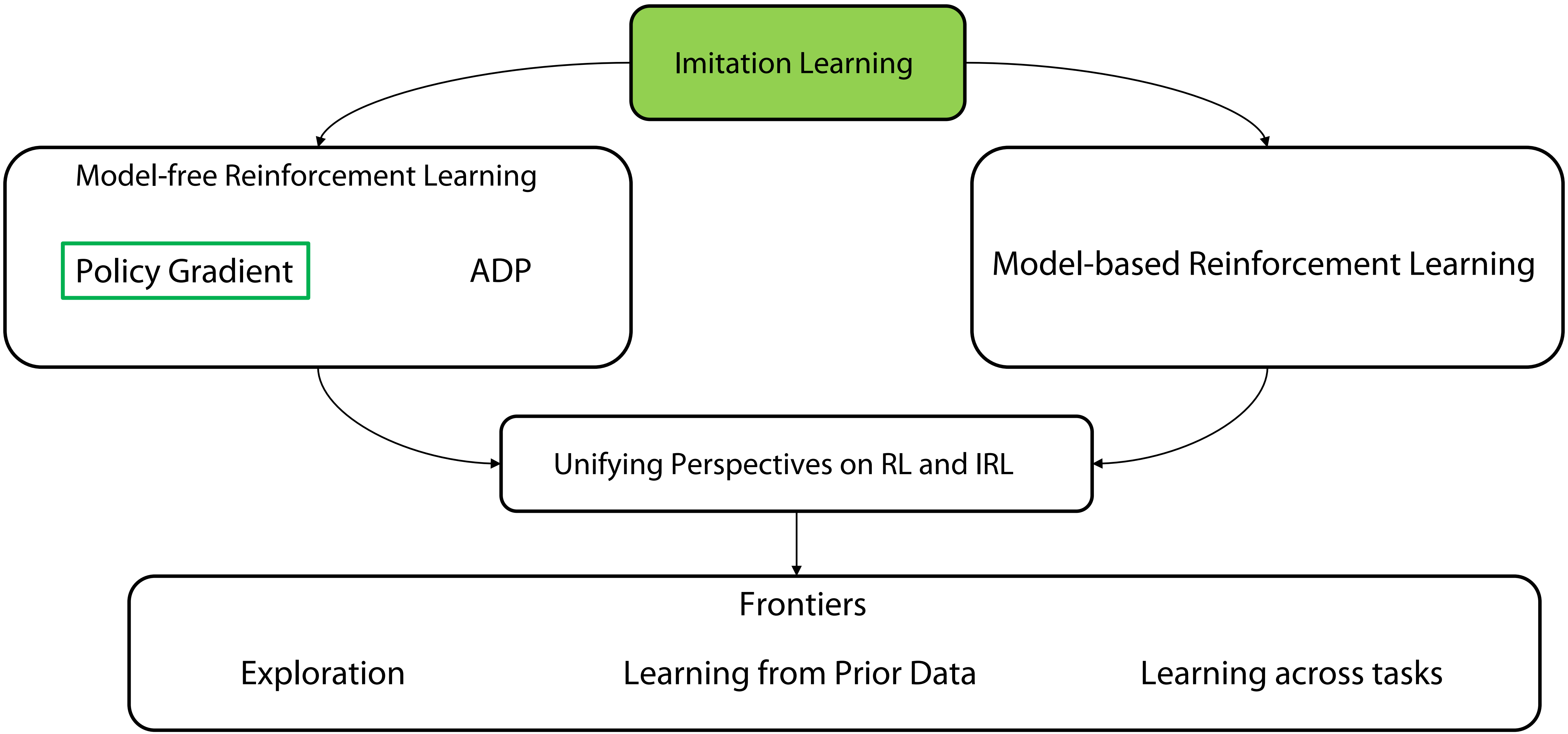
Reinforcement Learning
Spring 2024

Abhishek Gupta

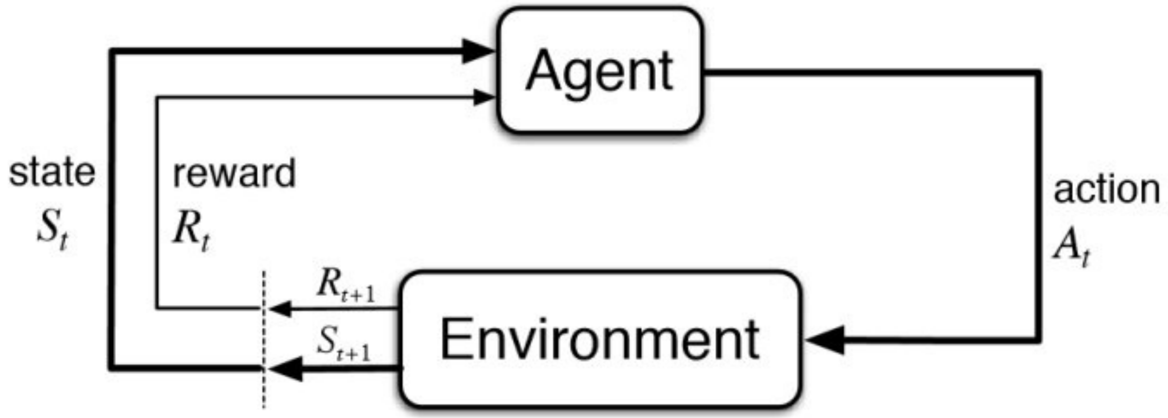
TAs: Patrick Yin, Qiuyu Chen

Slides inspired by Sergey Levine, Liyiming Ke

Class Structure



How should we optimize this 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 it's own +/-

Taking the gradient of return

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

$$\nabla_\theta J(\theta) = \mathbb{E}_{\substack{s_0 \sim p(s_0) \\ s_{t+1} \sim p(s_{t+1}|s_t, a_t) \\ a_t \sim \pi(a_t|s_t)}} \left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \sum_{t'=0}^{T} r(s_t, a_t) \right]$$

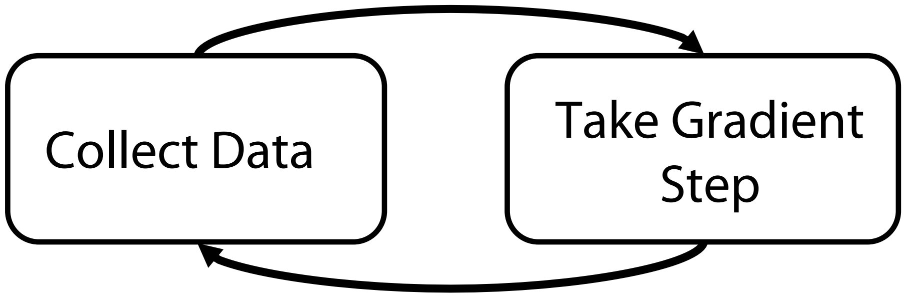
$$\approx \frac{1}{N} \sum_{i=0}^{N} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t^i|s_t^i) \sum_{t'=0}^{T} r(s_{t'}^i, a_{t'}^i)$$ (approximating using samples)

(Monte-Carlo approximation)

Resulting Algorithm (REINFORCE)

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

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



REINFORCE algorithm:

On-policy →

1. sample $\{\tau^i\}$ from $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ (run it on the robot)
2. $\nabla_\theta J(\theta) \approx \sum_i \left(\sum_t \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i|\mathbf{s}_t^i)\right) \left(\sum_t r(\mathbf{s}_t^i, \mathbf{a}_t^i)\right)$
3. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

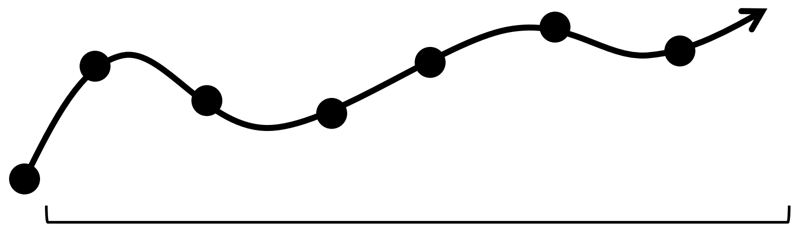
What makes policy gradient challenging?

Hard to tell what matters without many samples

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

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

What we do



Single sample estimate

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

Couples together all actions

Susceptible to scale variations

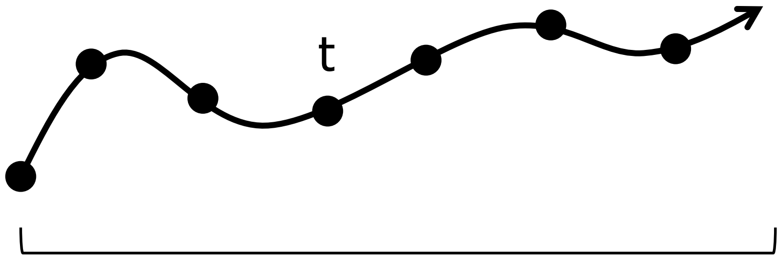
Susceptible to lucky samples

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 <u>**"return-to-go"**</u>

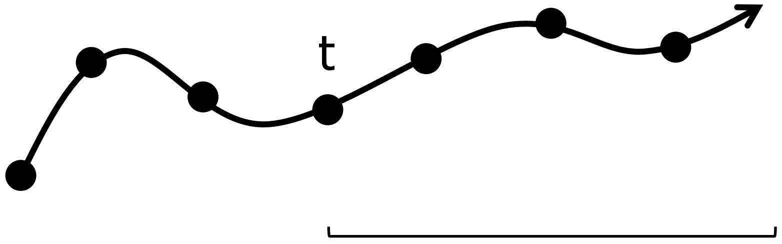
$$\approx \frac{1}{N}\sum_{i=0}^{N}\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t^i|s_t^i)\underbrace{\sum_{t'=0}^{T} r(s_{t'}^i, a_{t'}^i)}_{\text{Includes } t' < t}$$



Full trajectory return

Ignore past terms ⇩

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



Return to go

Lecture outline

Reducing the Variance of Policy Gradient with Baselines

↓

Covariant Parameterization - Natural Policy Gradient

↓

Trust Region Policy Optimization

↓

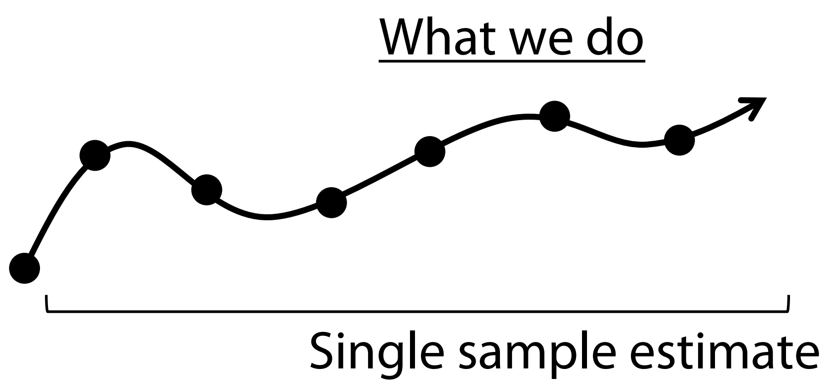
Proximal Policy Optimization

What makes policy gradient challenging?

Hard to tell what matters without many samples

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

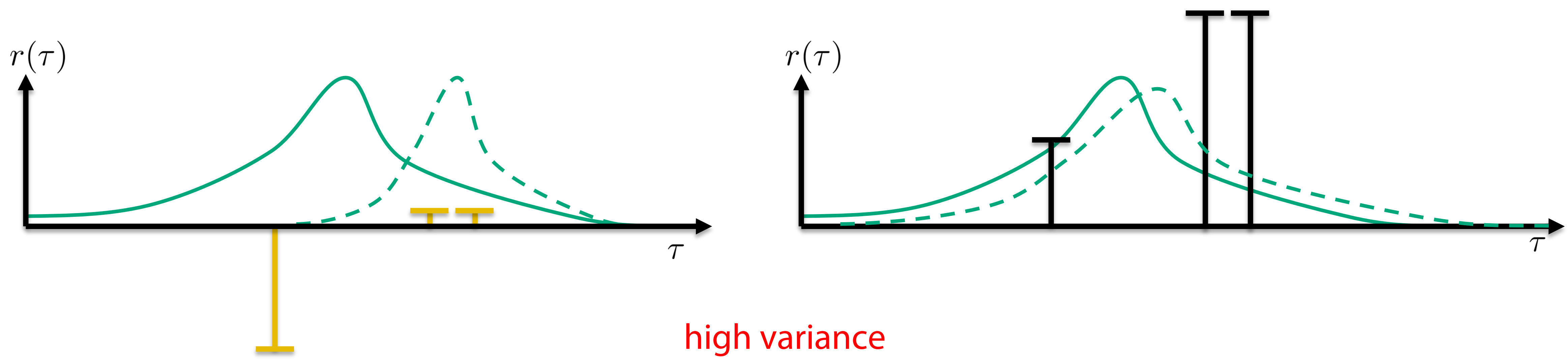
$$\approx \frac{1}{N} \sum_{i=0}^{N} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t^i | s_t^i) \sum_{t'=0}^{T} r(s_{t'}^i, a_{t'}^i)$$



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

Susceptible to scale variations

Policy gradient is susceptible to scale variations



high variance

Arbitrarily uncentered, scaled returns can lead to huge variance:

→ Imagine all rewards were positive, every action would be pushed up, some more than others

→ What if instead, we pushed down some actions and pushed up some others (even if rewards are positive)

Credit: Sergey Levine

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

But does this increase bias??

Variance Reduction with a Baseline

$$\int_{\mathcal{S}} \int_{\mathcal{A}} p(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t|s_t) \left[\sum_{t'=t}^{T} r(s_{t'}, a_{t'}) - b(s_t) \right] ds_t \, da_t$$

$$\int_{\mathcal{S}} \int_{\mathcal{A}} p(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t|s_t) \left[\sum_{t'=t}^{T} r(s_{t'}, a_{t'}) \right] ds_t \, da_t - \int_{\mathcal{S}} \int_{\mathcal{A}} p(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t|s_t) b(s_t) \, ds_t \, da_t$$

Let us show this is 0!

Variance Reduction with a Baseline

$$\int\int p(s_t, a_t)\nabla_\theta \log \pi_\theta(a_t|s_t)\left[b(s_t)\right] ds_t da_t = \int\int p(s_t)\pi_\theta(a_t|s_t)\nabla_\theta \log \pi_\theta(a_t|s_t)\left[b(s_t)\right] ds_t da_t$$

$$= \int p(s_t)b(s_t)\int \pi_\theta(a_t|s_t)\nabla_\theta \log \pi_\theta(a_t|s_t) da_t ds_t$$

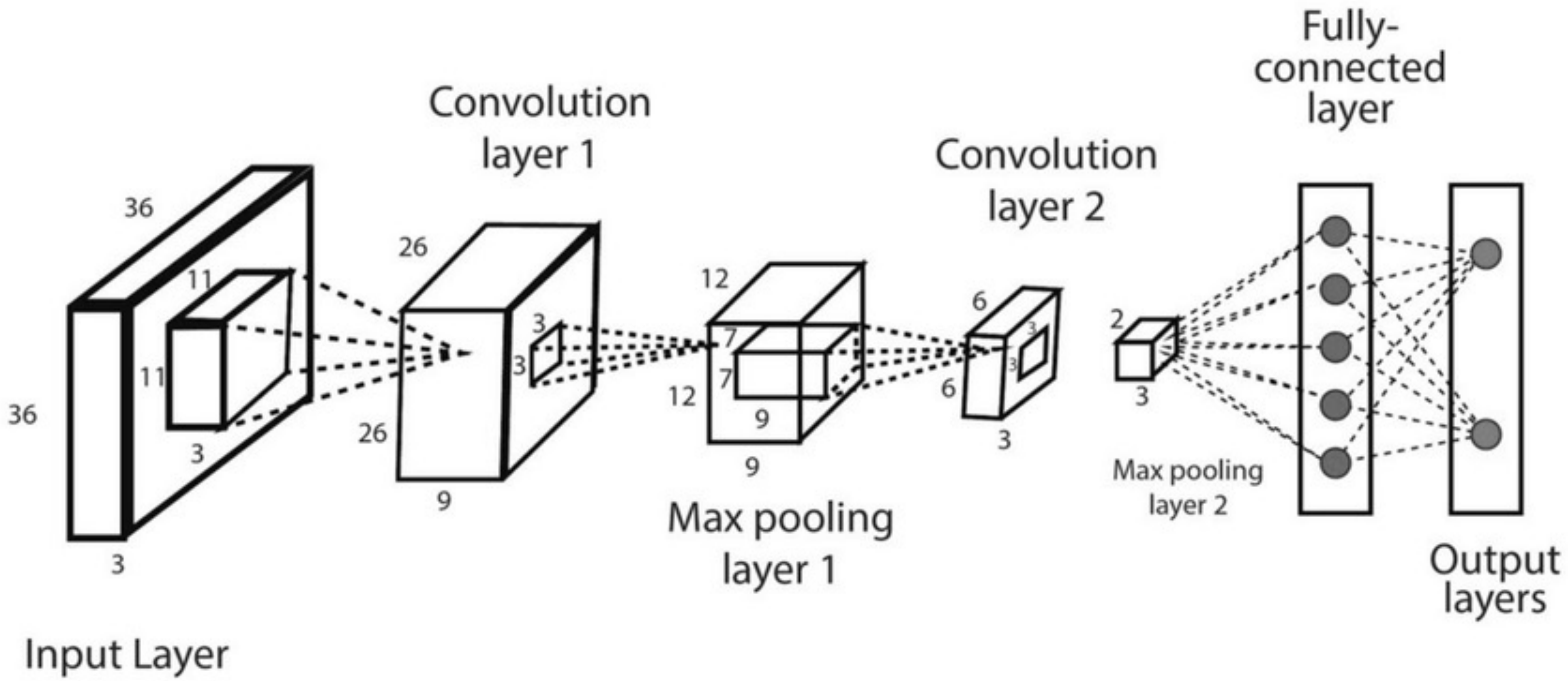
$$= \int p(s_t)b(s_t)\int \nabla_\theta \pi_\theta(a_t|s_t) da_t ds_t$$

$$= \int p(s_t)b(s_t)\nabla_\theta \int \pi_\theta(a_t|s_t) da_t ds_t = \int p(s_t)b(s_t)\nabla_\theta(1) ds_t = 0$$

Unbiased!

Learning Baselines

Baselines are typically learned as deep neural nets from $R^s \rightarrow R^1$



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

Minimize with Monte-Carlo regression at every iteration, club with policy gradient

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

Reducing the Variance of Policy Gradient with Baselines

↓

Covariant Parameterization - Natural Policy Gradient

↓

Trust Region Policy Optimization

↓

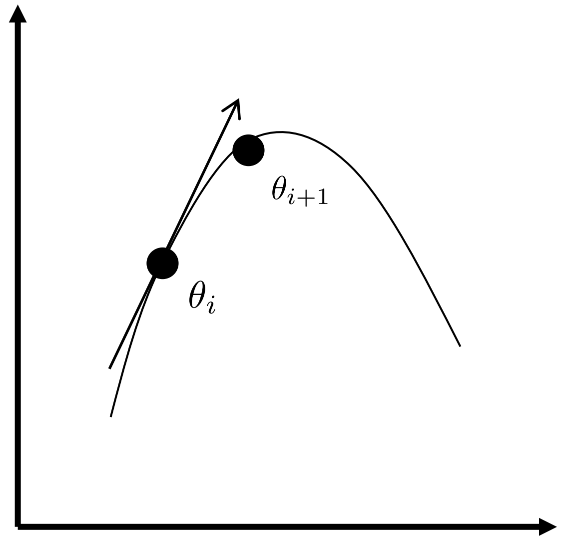
Proximal Policy Optimization

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!

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



Take a deeper look at REINFORCE

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

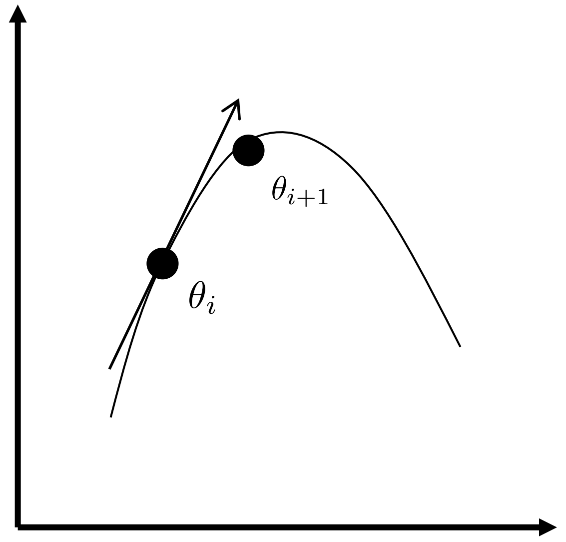
Gradient ascent is steepest ascent on linear approximation under the Euclidean metric!

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

$$(\theta - \theta_i)^T(\theta - \theta_i) \leq \epsilon$$ Quadratic Constraint

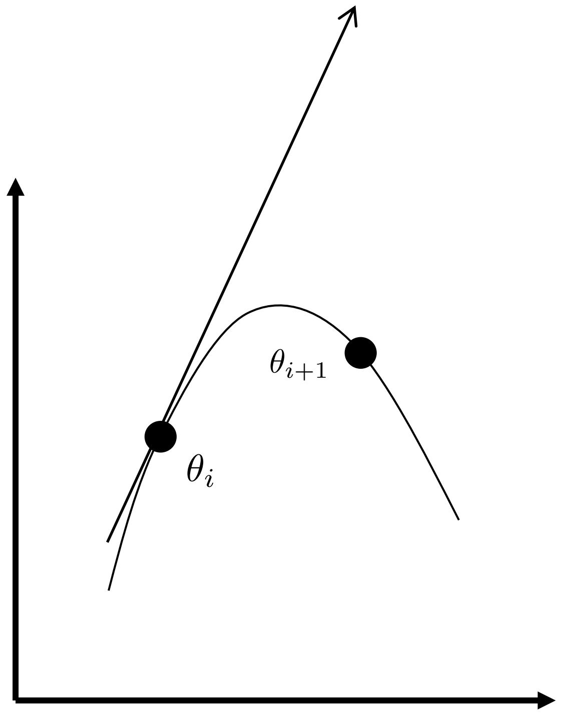
$$\downarrow$$

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



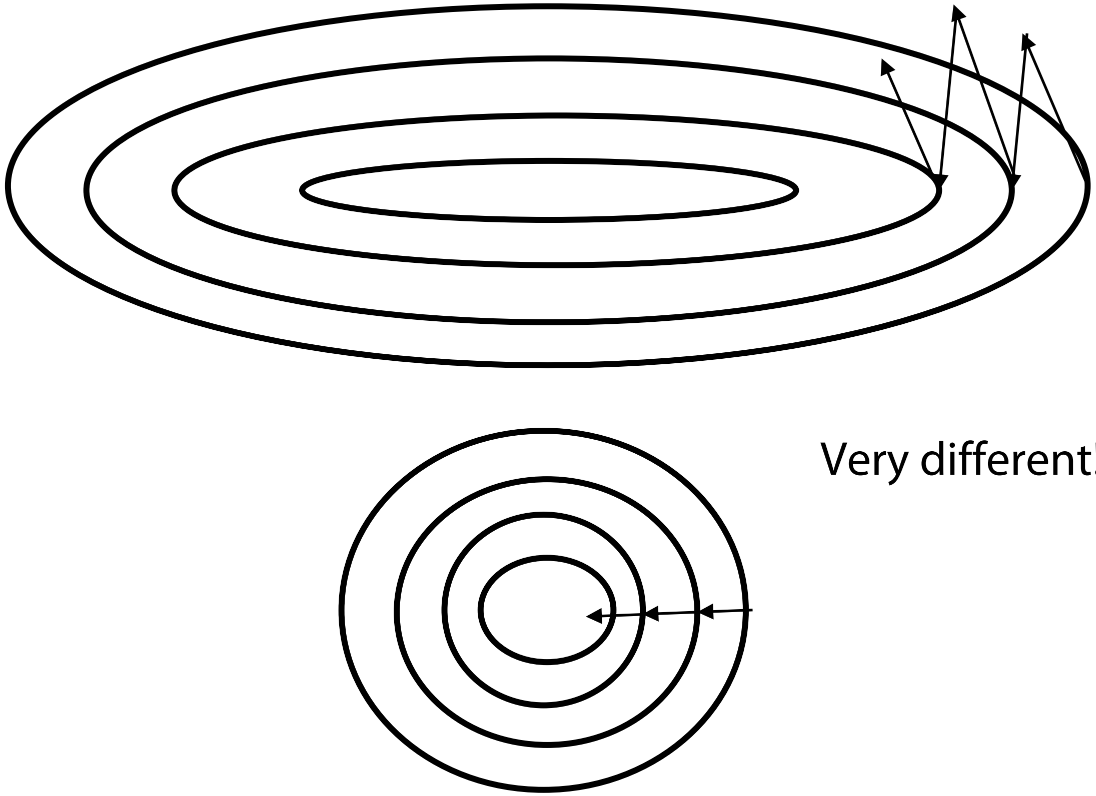
When might this fail?

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

<u>Sensitive to Policy Parameterization</u>

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

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

$$\phi_1 = \theta_1^2$$

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

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

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

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

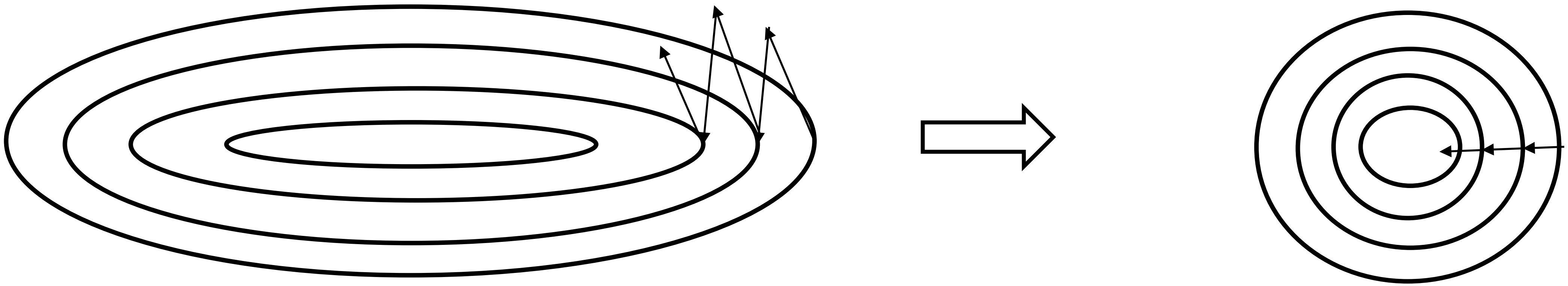
$$\nabla_{\phi_2} L = -\phi_2^{-2} = -\theta_2^2$$

↔ Not covariant!

Modified Constraint on Policy Gradient

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

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



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

Rescales according to G^{-1}

Adaptive choice of G can avoid sensitivity to policy parameterization!

Covariant Policy Gradient Updates

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

What should G be?

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

Let us use the constraint as KL divergence on the policy (2^{nd} order Taylor expansion)

Measures functional distance, not parameter distance

Resulting "Natural" Policy Gradient

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

2nd order approximation of KL → Fisher Information Metric

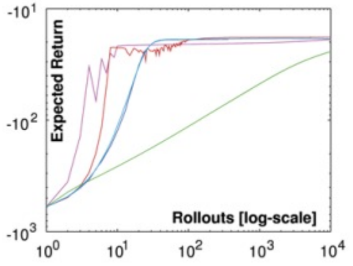
$$F = \mathbb{E}_{\pi_\theta}\left[(\nabla_\theta \log \pi_\theta)(\nabla_\theta \log \pi_\theta)^T\right]$$

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

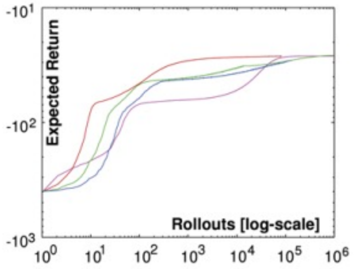
Resulting update $$\theta_{i+1} = \theta_i + \alpha F^{-1} \nabla_\theta J(\theta)|_{\theta=\theta_i}$$ Covariant to parameterization

Natural Policy Gradient in Action

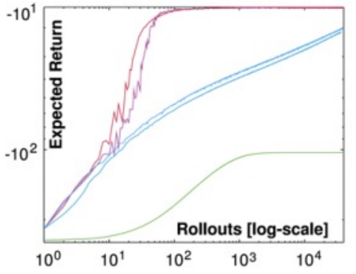
(a) Performance. (b) Imitation learning. (c) Initial reproduction. (d) After reinforcement learning.



(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

Peters, Schaal '08

Lecture outline

Reducing the Variance of Policy Gradient with Baselines

↓

Covariant Parameterization - Natural Policy Gradient

↓

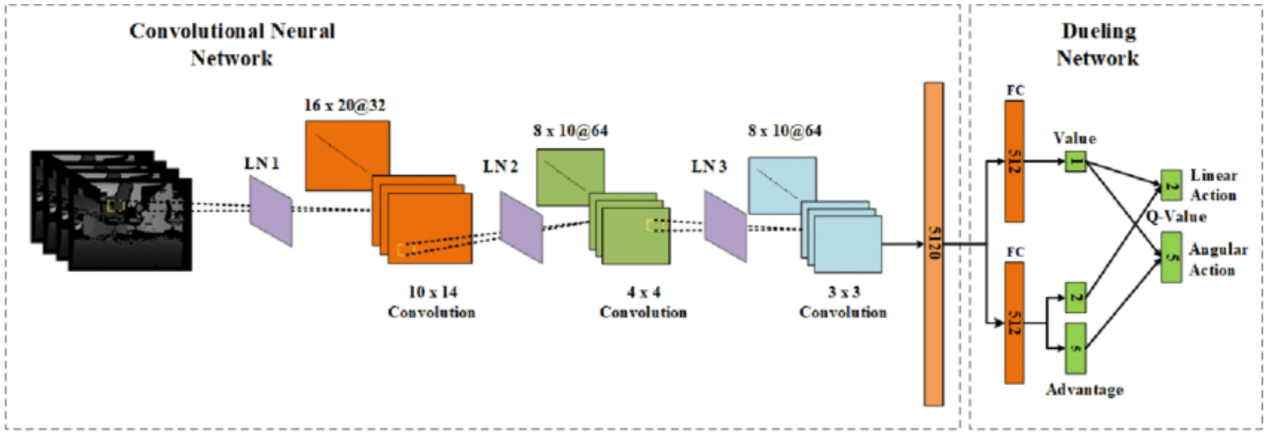
Trust Region Policy Optimization

↓

Proximal Policy Optimization

Natural Policy Gradient - is it enough?

Huge matrix inversion

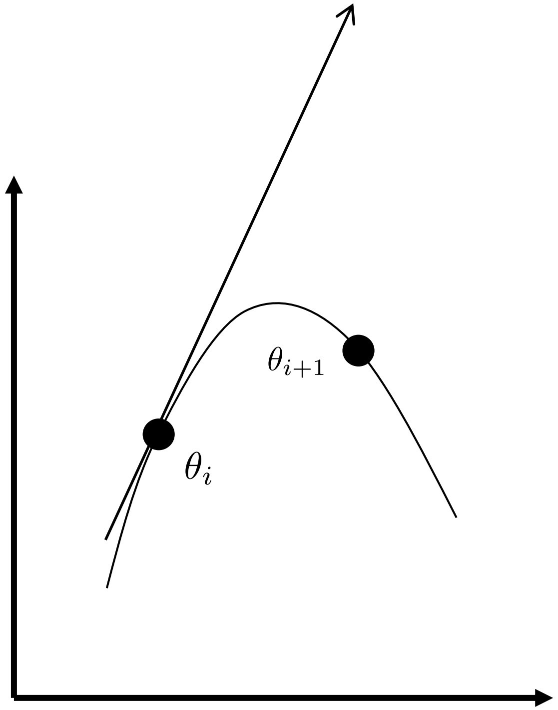


F – R^{dxd}

For a standard convnet – d is in the millions

Hessian is way out of memory / hard to invert!

Step-size?



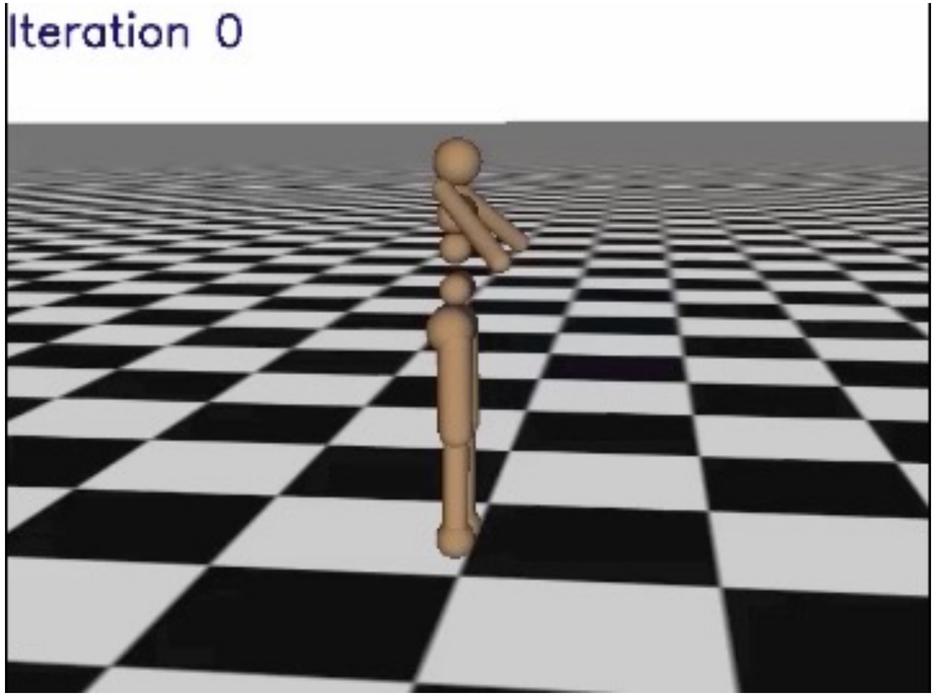
Can easily overstep and collapse performance

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

Trust Region Policy Optimization

3 key ideas:

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

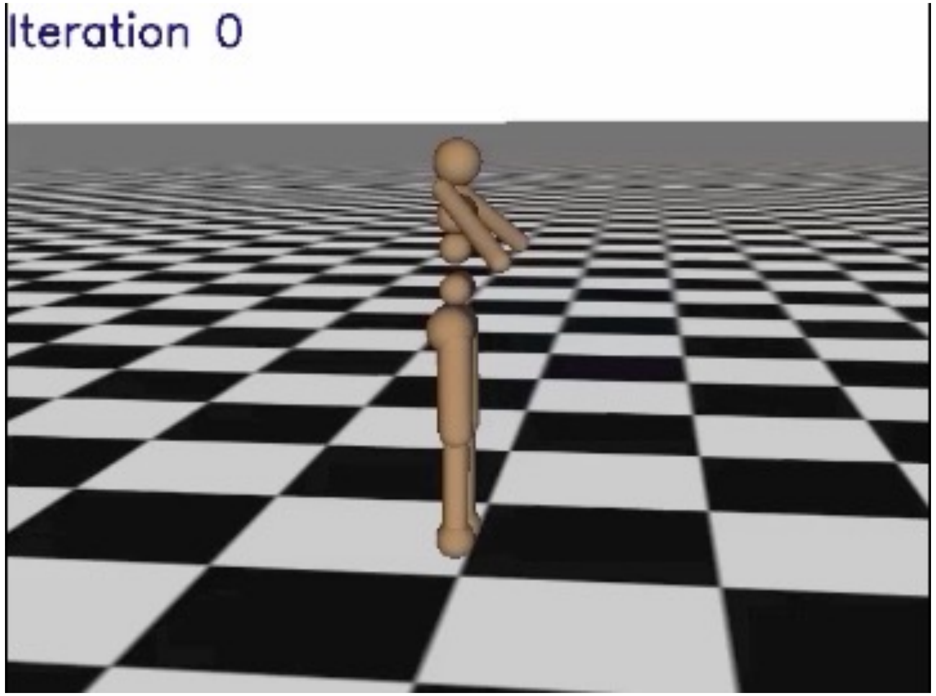


TRPO, Schulman '15

Trust Region Policy Optimization

3 key ideas:

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



TRPO, Schulman '15

Trust Region Policy Optimization – Importance Sampling

Cannot evaluate without resampling

Original Objective

$$J(\theta) = \mathbb{E}_{s\sim d_\pi(\theta), a\sim\theta}\left[A(s,a)\right]$$

$$= \mathbb{E}_{s\sim d_\pi(\theta), a\sim\theta_i}\left[\frac{\pi_\theta}{\pi_{\theta_i}} A(s,a)\right]$$

Importance Sampling (ish)

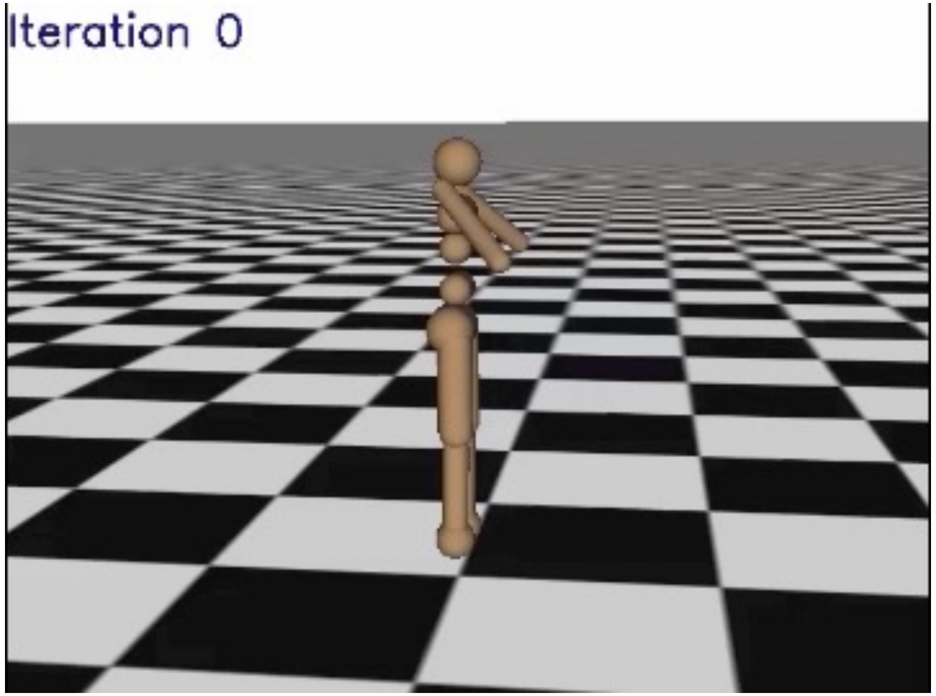
$$\approx \mathbb{E}_{s\sim d_\pi(\theta_i), a\sim\theta_i}\left[\frac{\pi_\theta}{\pi_{\theta_i}} A(s,a)\right]$$

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

Trust Region Policy Optimization

3 key ideas:

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



TRPO, Schulman '15

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

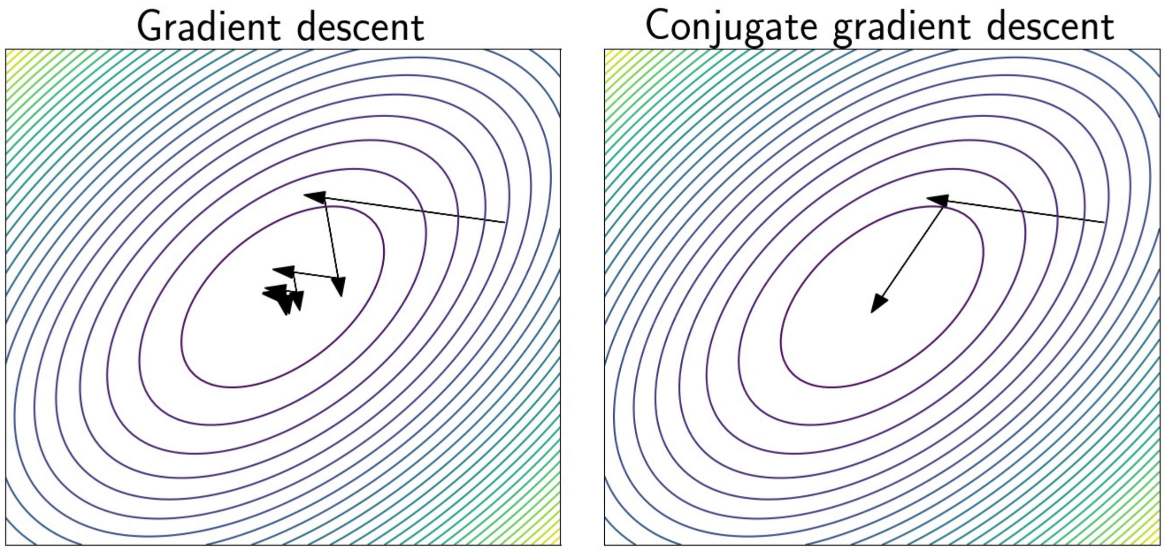
Challenging to compute F^{-1} and then get $F^{-1}g$

⇩

Convert into an iterative minimization problem!

⇩

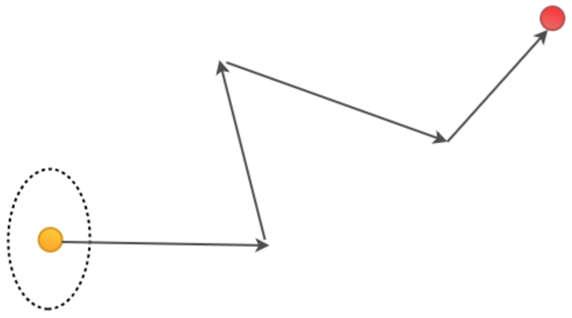
Solve with conjugate gradient



Do coordinate descent in geometry aligned orthogonal directions

https://www.cs.cmu.edu/~quake-papers/painless-conjugate-gradient.pdf

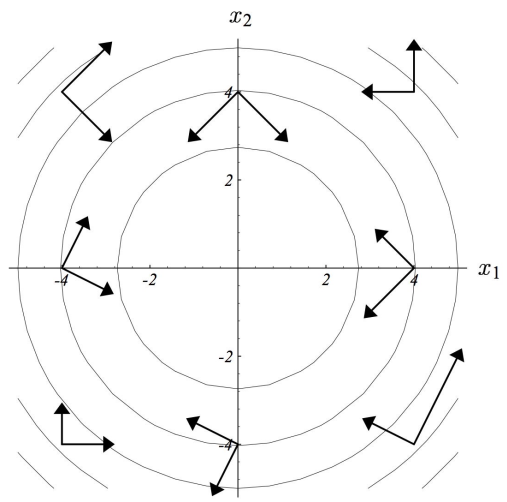
Trust Region Policy Optimization – Conjugate Gradient

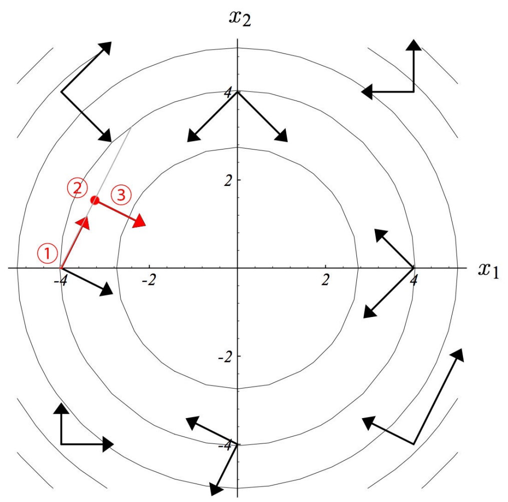


$$\min_x \frac{1}{2} x^T F x - x^T g + c$$

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

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





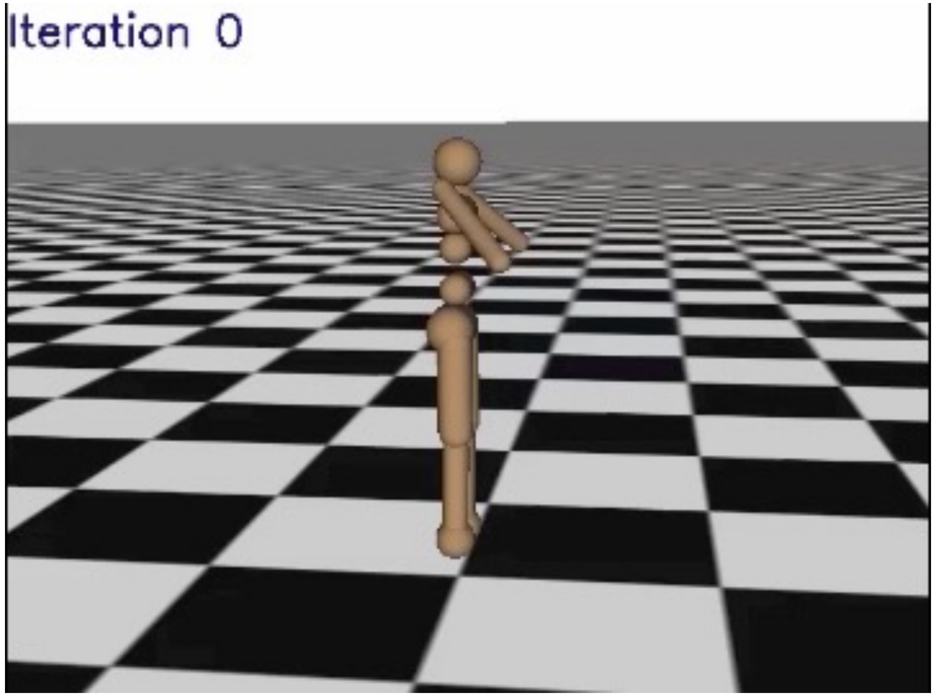
Converges in approx N steps!

Only requires matrix-vector product

Trust Region Policy Optimization

3 key ideas:

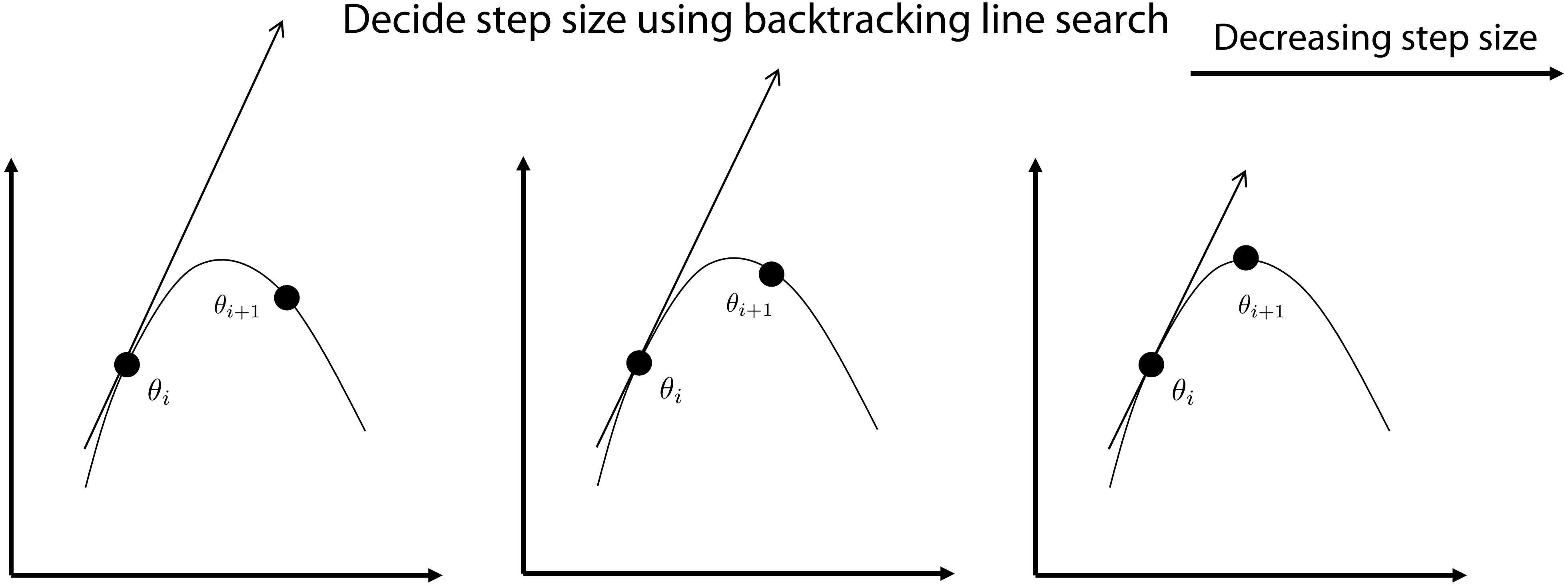
1. On-policy updates → importance sampled objective
2. Huge matrix inversion → conjugate gradient method
3. Step size may be too large → backtracking line search



TRPO, Schulman '15

Trust Region Policy Optimization – Backtracking line search

Decide step size using 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 - $\frac{1}{2}(ts)^T F(ts) = \epsilon \rightarrow t = \sqrt{\frac{2\epsilon}{s^T F s}}$
3. While $J(\theta_i + ts) < J(\theta_i)$, set $t = \beta t$

Backtracking

Trust Region Policy Optimization

3 key ideas:

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

Algorithm 3 Trust Region Policy Optimization

Input: initial policy parameters θ_0

for $k = 0, 1, 2, \ldots$ **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, Schulman '15

Can we say anything formal about updates?

$$\eta(\tilde{\pi}) \geq L_\pi(\tilde{\pi}) - C D_{\mathrm{KL}}^{\max}(\pi, \tilde{\pi}),$$
$$\text{where } C = \frac{4\epsilon\gamma}{(1-\gamma)^2}.$$

Ensures that policies are non-decreasing in performance

Performance difference lemma

$$\eta(\tilde{\pi}) = \eta(\pi) + \mathbb{E}_{\tau \sim \tilde{\pi}}\left[\sum_{t=0}^{\infty} \gamma^t A_\pi(s_t, a_t)\right] \Rightarrow$$

Express advantage in terms of TVD

Theorem 1. *Let* $\alpha = D_{\mathrm{TV}}^{\max}(\pi_{\mathrm{old}}, \pi_{\mathrm{new}})$. *Then the following bound holds:*

$$\eta(\pi_{\mathrm{new}}) \geq L_{\pi_{\mathrm{old}}}(\pi_{\mathrm{new}}) - \frac{4\epsilon\gamma}{(1-\gamma)^2}\alpha^2$$
$$\textit{where } \epsilon = \max_{s,a} |A_\pi(s,a)| \qquad (8)$$

Key idea: by bounding how different the policies are, we can bound how different returns are

TRPO, Schulman '15

TRPO in action

TRPO, Schulman '15

Why might TRPO not be enough?

Algorithm 3 Trust Region Policy Optimization

Input: initial policy parameters θ_0
for $k = 0, 1, 2, \ldots$ **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

TRPO, Schulman '15

Better Advantage Estimation - Generalized Advantage Estimation

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

High variance!

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

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

⋮

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

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

Variance ↑

Bias ↓

GAE, Schulman '15

Generalized Advantage Estimation

Sum up all the estimators in a geometric sum

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

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

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

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

Geometric sum

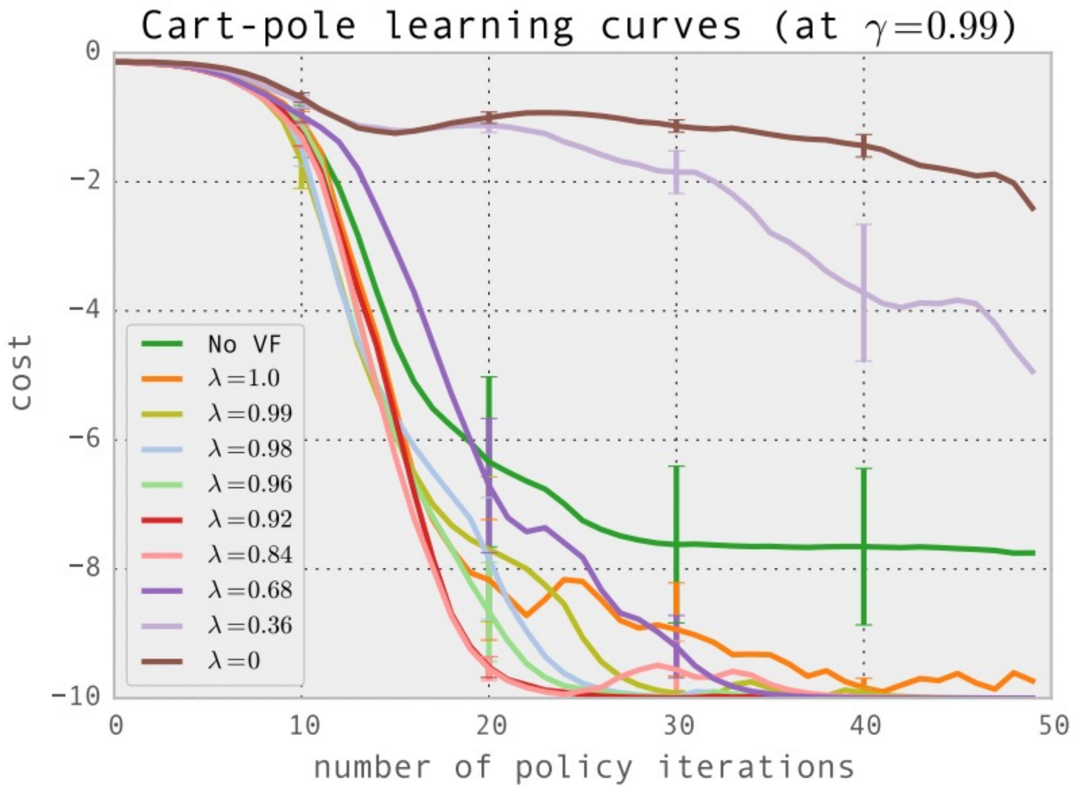
$$A_\lambda^\theta(s_1, a_1) = \sum_{j=1}^{N} \lambda^j A_j^\theta(s, a)$$

λ controls bias-variance tradeoff

Best of both worlds – very similar idea to eligibility traces

GAE, Schulman '15

Generalized Advantage Estimation in Action



GAE, Schulman '15

Lecture outline

Reducing the Variance of Policy Gradient with Baselines

↓

Covariant Parameterization - Natural Policy Gradient

↓

Trust Region Policy Optimization

↓

Proximal Policy Optimization

Avoiding Second Order Optimization

Algorithm 3 Trust Region Policy Optimization

Input: initial policy parameters θ_0
for $k = 0, 1, 2, \ldots$ **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

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

Restrict the amount the policy moves

⇩

Proximal Policy Optimization

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

PPO, Schulman '17

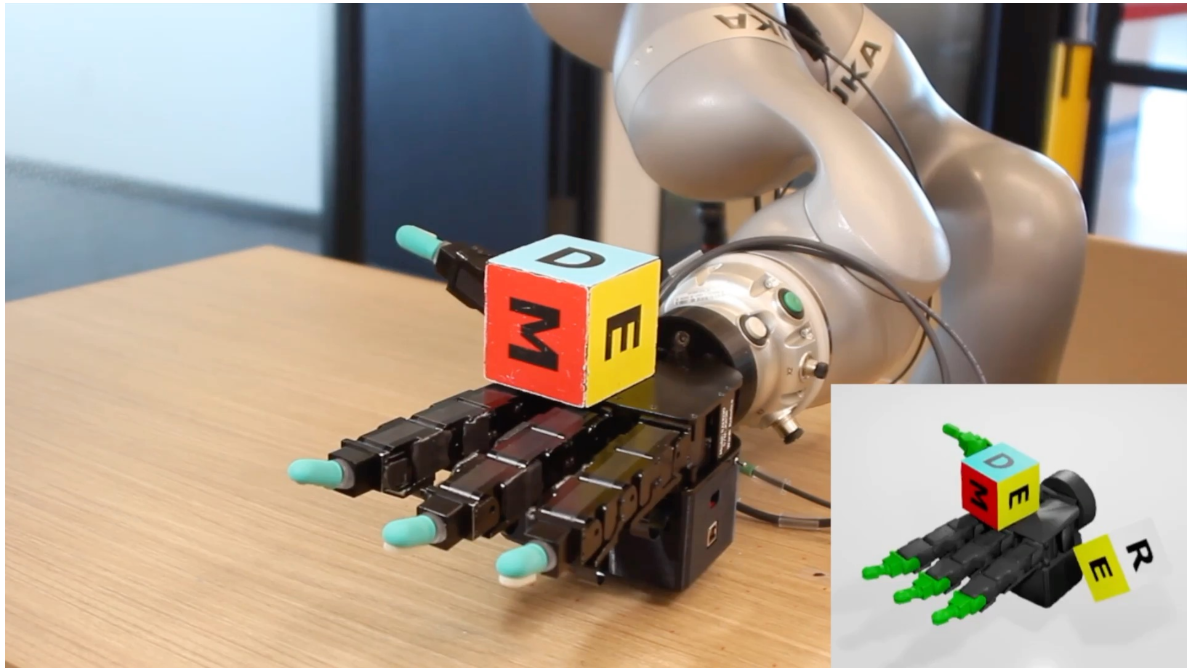
Proximal Policy Optimization Algorithm

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

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

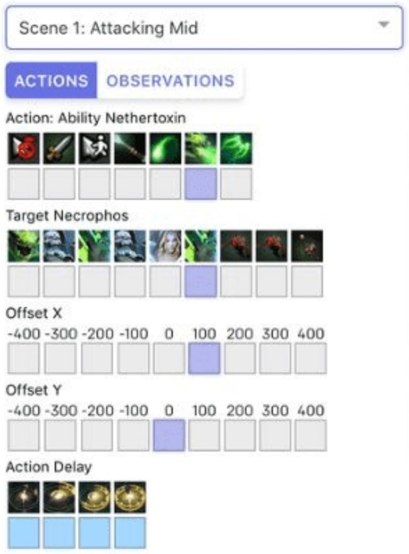
PPO, Schulman '17

PPO in Action



Dextreme, Handa '22
A system for general in-hand reorientation, Chen '21

PPO in Action



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 Santurkar[1], Dimitris Tsipras[1], Firdaus Janoos[2], Larry Rudolph[1,2], and Aleksander Mądry[1]

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!

Engstrom et al, '20

Lecture outline

Reducing the Variance of Policy Gradient with Baselines

↓

Covariant Parameterization - Natural Policy Gradient

↓

Trust Region Policy Optimization

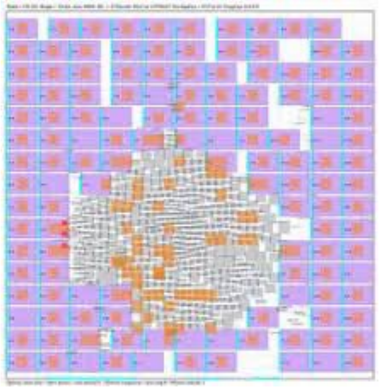
↓

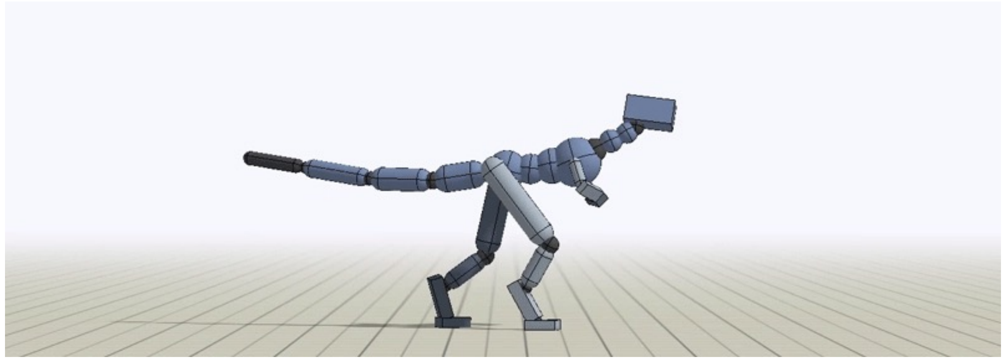
Proximal Policy Optimization

Pros/Cons of Policy Gradient Methods

<u>Pros</u>

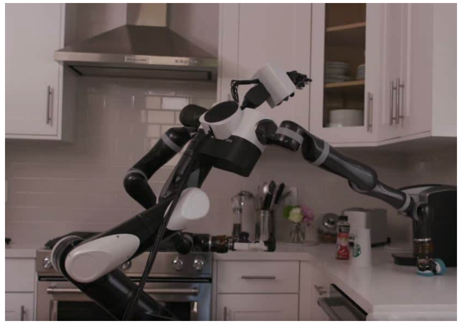
- Conceptually simple, easy to implement
- Stable, good asymptotic performance
- Compatible with deep models
- Require minimal modeling

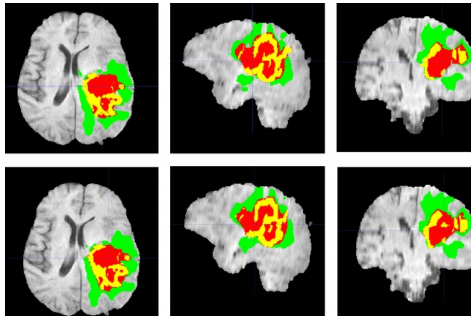




<u>Cons</u>

- 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 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

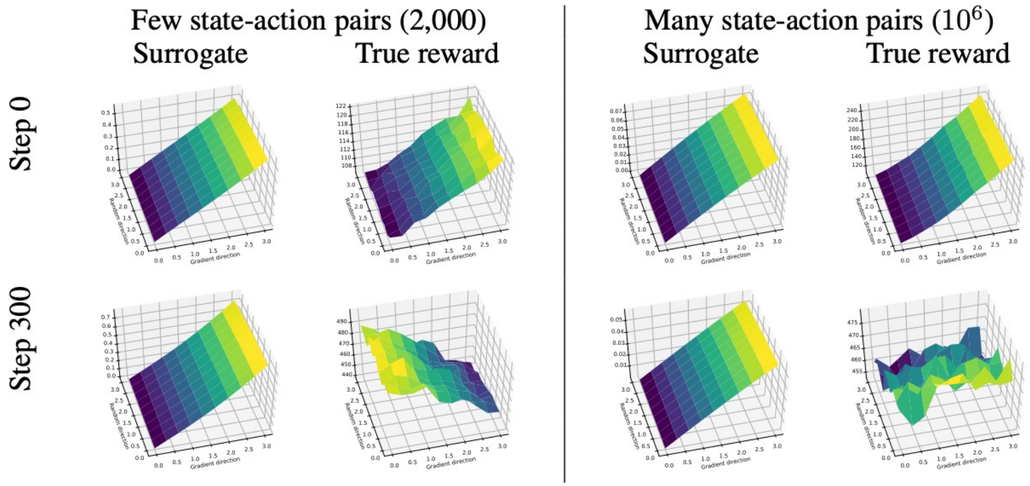
$$N \geq \frac{\|\Sigma_{K^*}\|}{\mu} \log \frac{C(K_0) - C(K^*)}{\varepsilon} \times \text{poly}\left(\frac{C(K_0)}{\mu\sigma_{min}(Q)}, \|A\|, \|B\|, \|R\|, \frac{1}{\sigma_{min}(R)}\right),$$

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

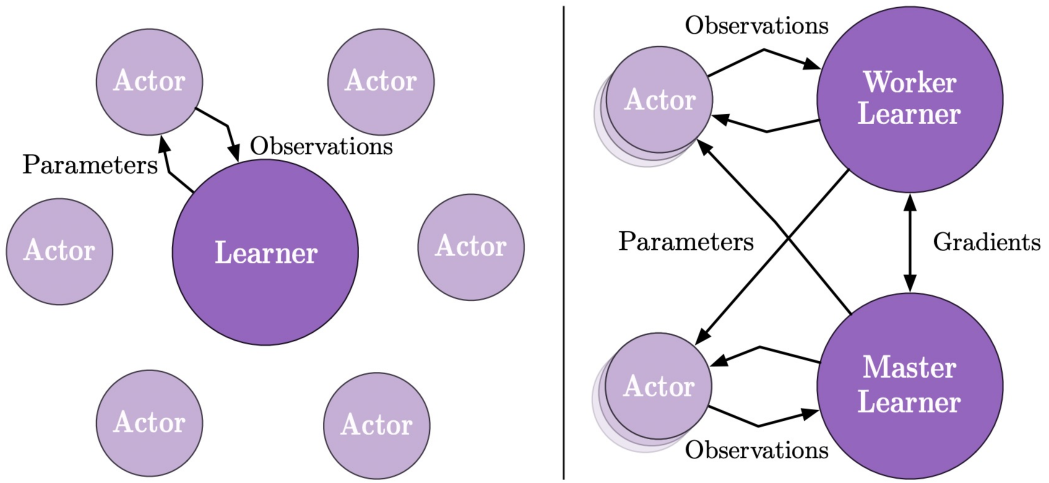
Practical Algorithms Deviate from Theory



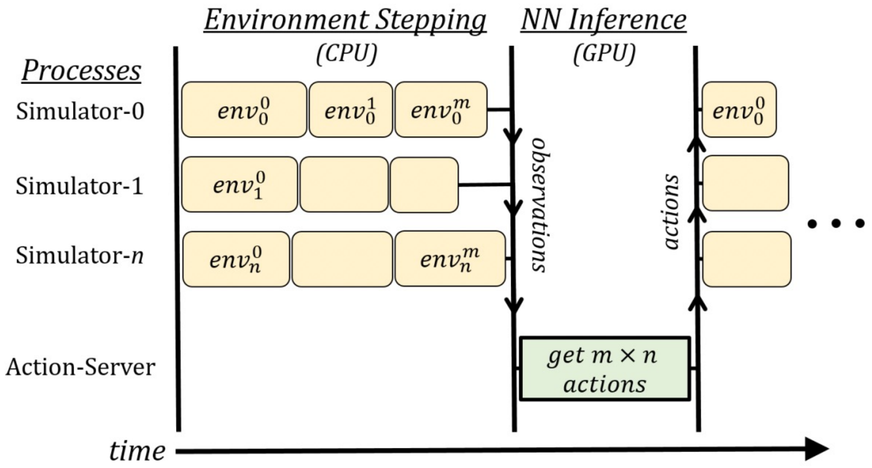
Is the Policy Gradient a Gradient?, Nota et al, '19
A Closer Look at Deep Policy Gradients, Ilyas et al '19
An Empirical Analysis of Proximal Policy Optimization with Kronecker-factored Natural Gradients, Song et al '18
What Matters In On-Policy Reinforcement Learning? A Large-Scale Empirical Study,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

Frontiers of Policy Gradient Research

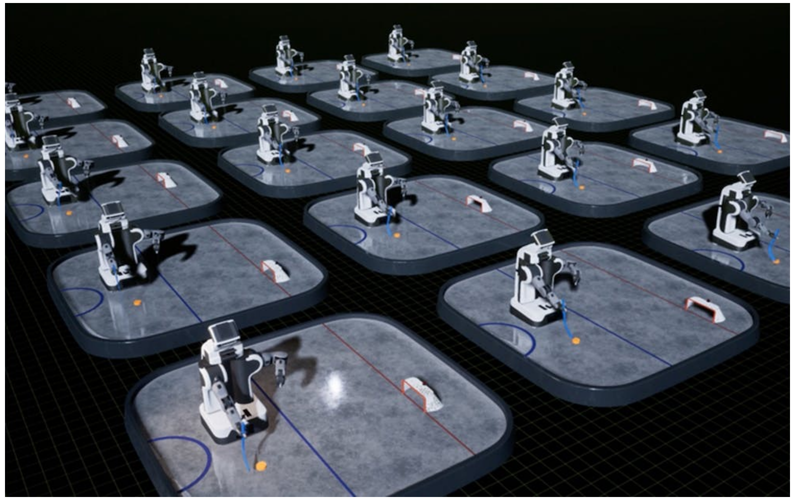
Asynchronous methods for large scale speedup



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

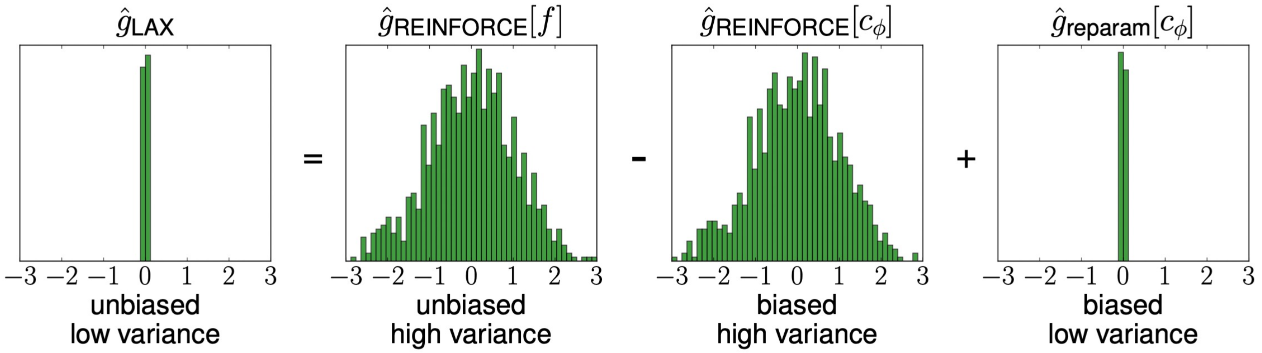
$$\pi_\theta(a_t|s_t) = \prod_{i=1}^{m} \pi_\theta(a_t^i|s_t)$$

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

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



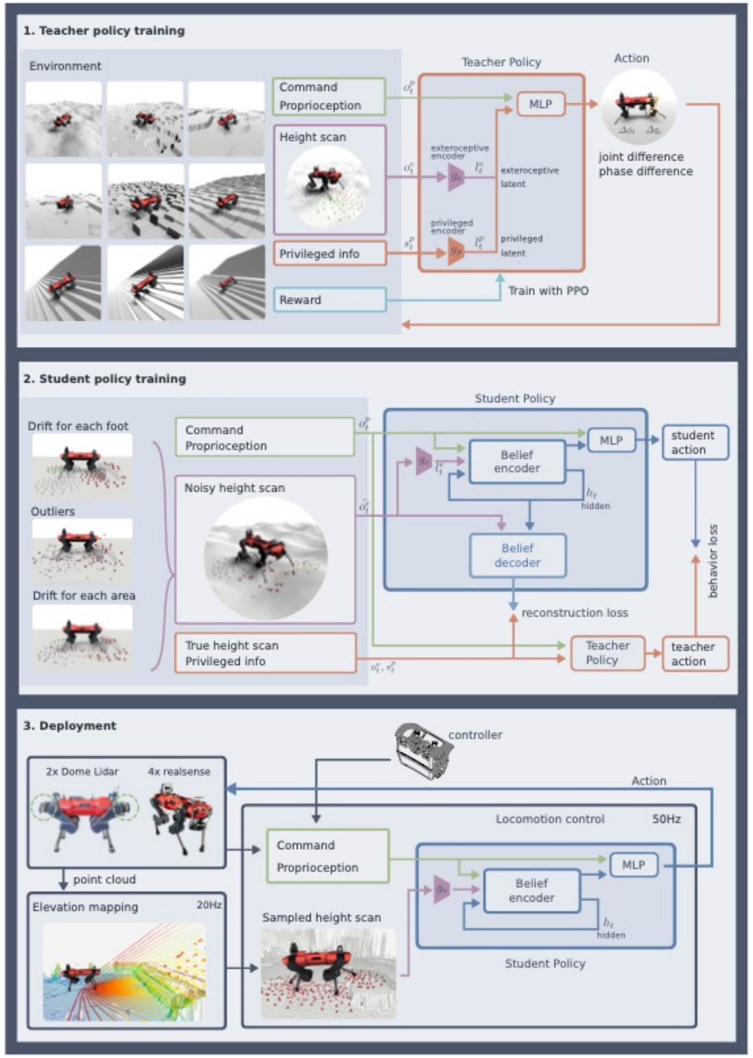
Q-Prop: Sample-Efficient Policy Gradient with An Off-Policy Critic, Gu et al '16
Backpropagation through the Void: Optimizing control variates for black-box gradient estimation, Grathwohl et al '17
Categorical Reparameterization with Gumbel-Softmax, Jang et al '16

Frontiers of Policy Gradient Research

Learning from High Dimensional Observations

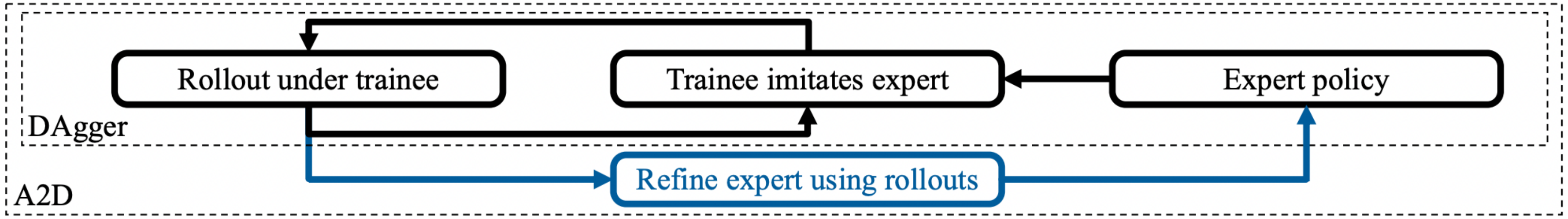
Learning Quadrupedal Locomotion over Challenging Terrain, Lee et al '20

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



Learning robust perceptive locomotion for quadrupedal robots in the wild, Miki et al '22

Challenging to provide guarantees in partially observed settings!



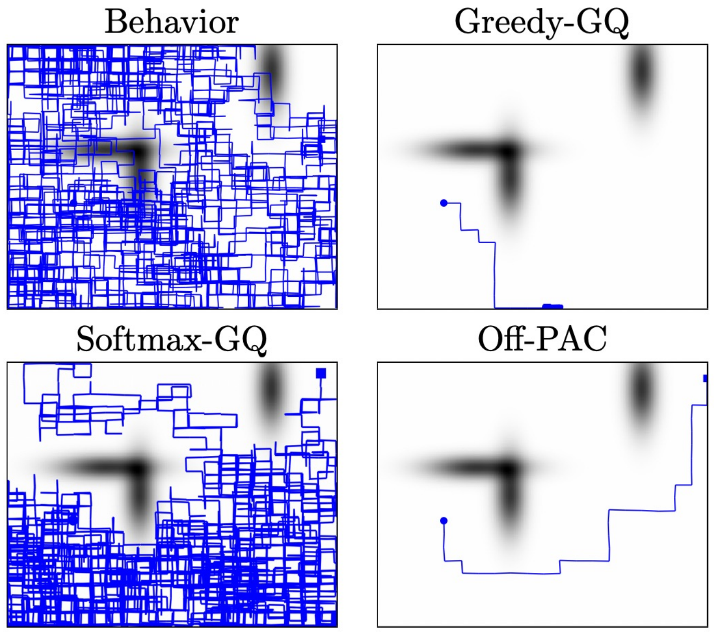
Robust Asymmetric Learning in POMDPs, Warrington et al '20

Frontiers of Policy Gradient Research

Bootstrapping from Prior/Off-Policy Data

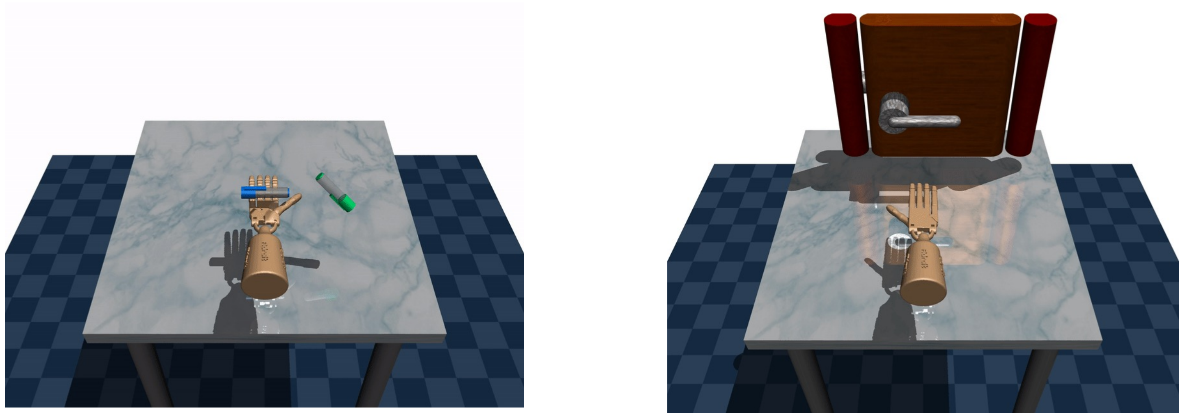
Off-policy policy gradient

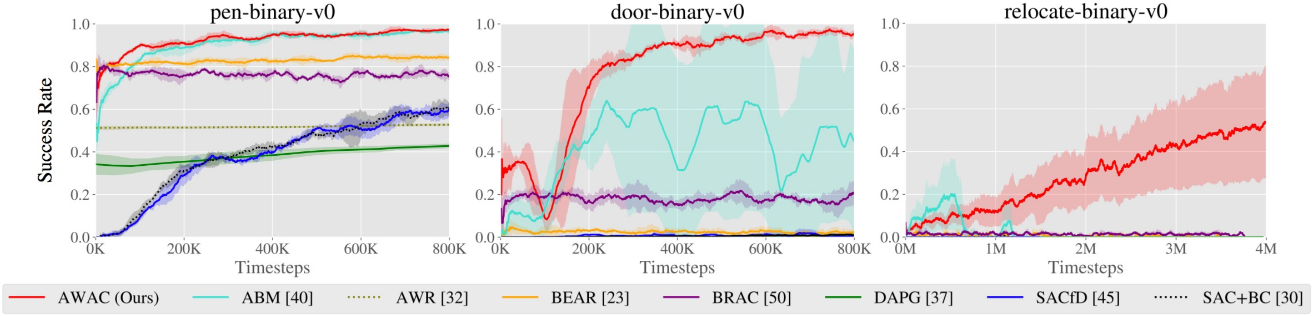
$$\mathbb{E}_\beta\left[\frac{\pi_\theta(a|s)}{\beta(a|s)} Q^\pi(s,a)\nabla_\theta \ln \pi_\theta(a|s)\right]$$



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

Learning from Prior Data

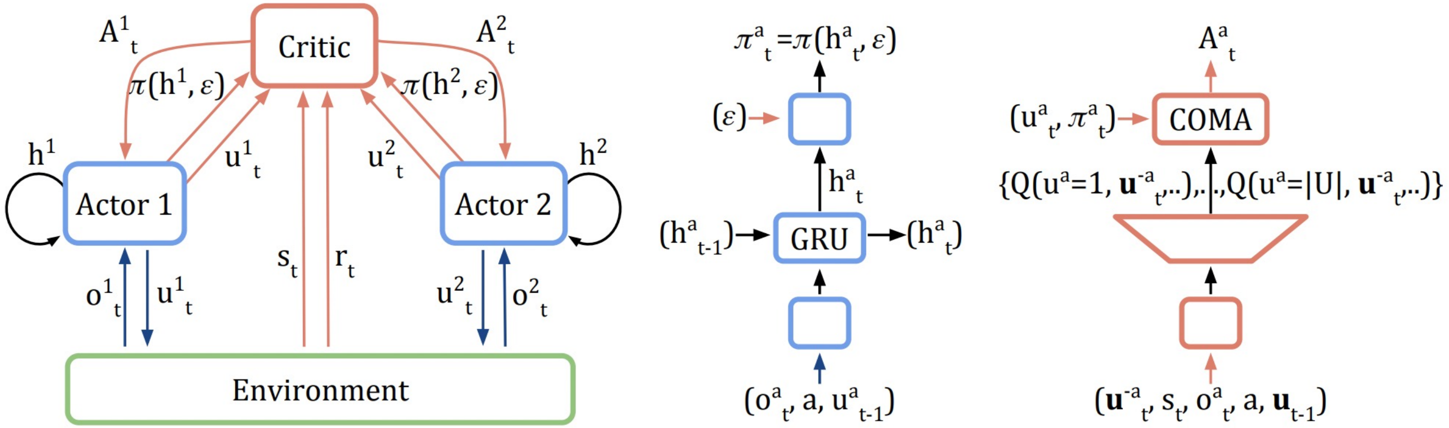




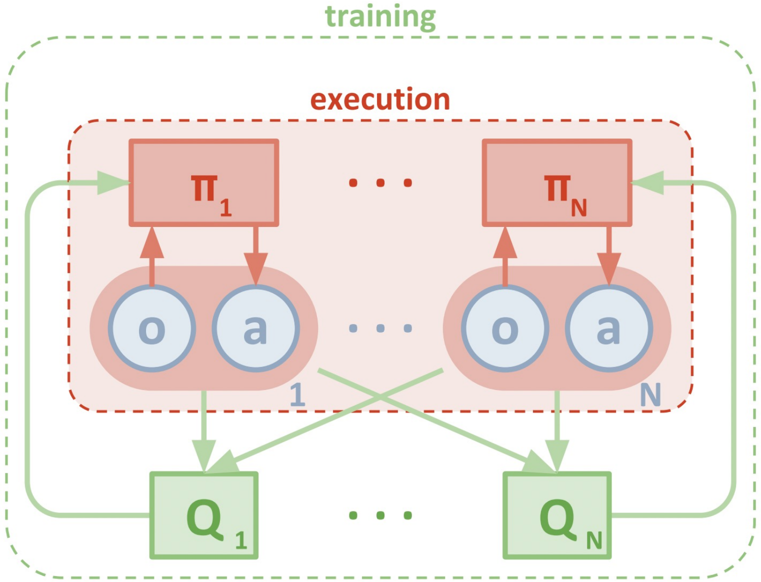
Advantage Weighted Actor Critic, Nair et al '20
DDPGfD, Vecerik '17
DAPG, Rajeswaran '17

Frontiers of Policy Gradient Research

Multi-agent policy gradient



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



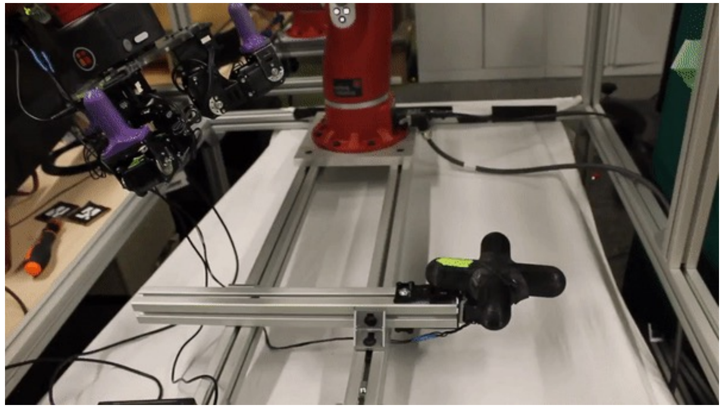
Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments, Lowe et al '17

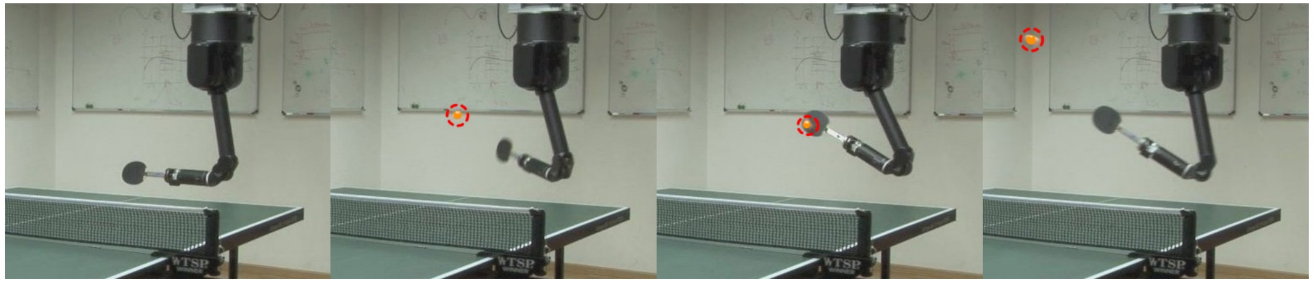
Primary challenges:

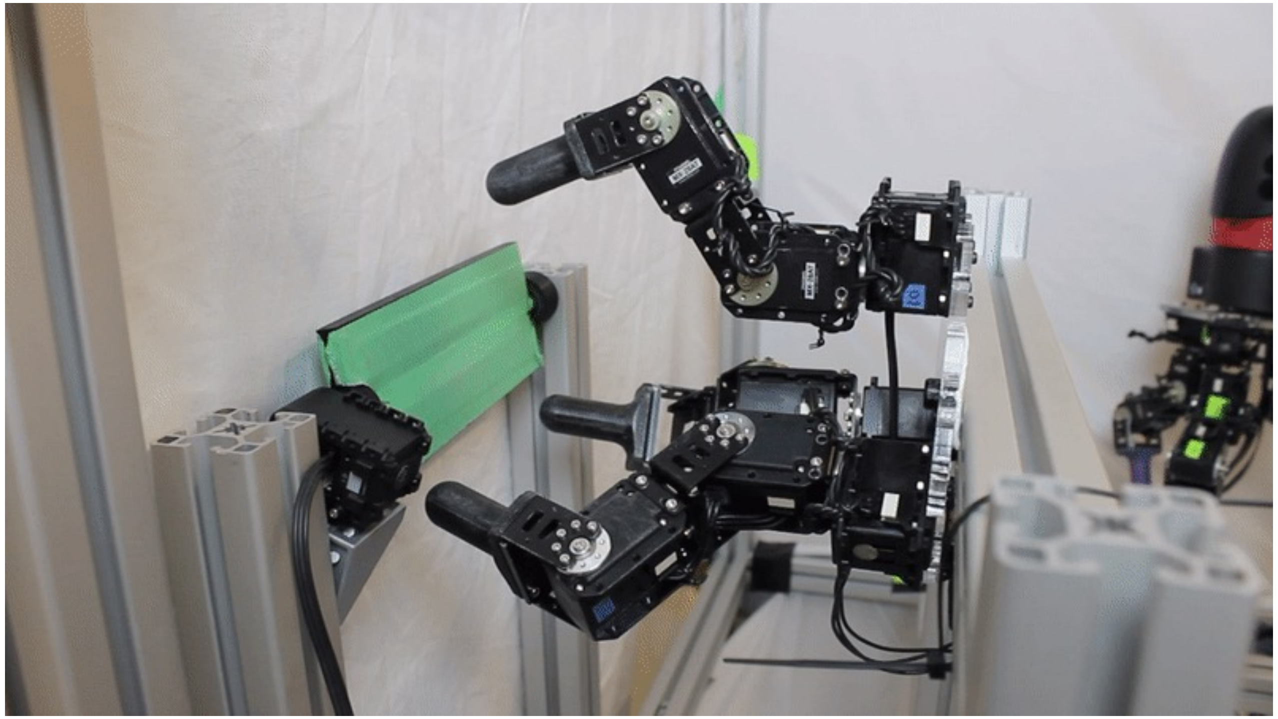
1. Non-stationarity
2. Data-efficiency
3. Communication

How is this useful for robotics?

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

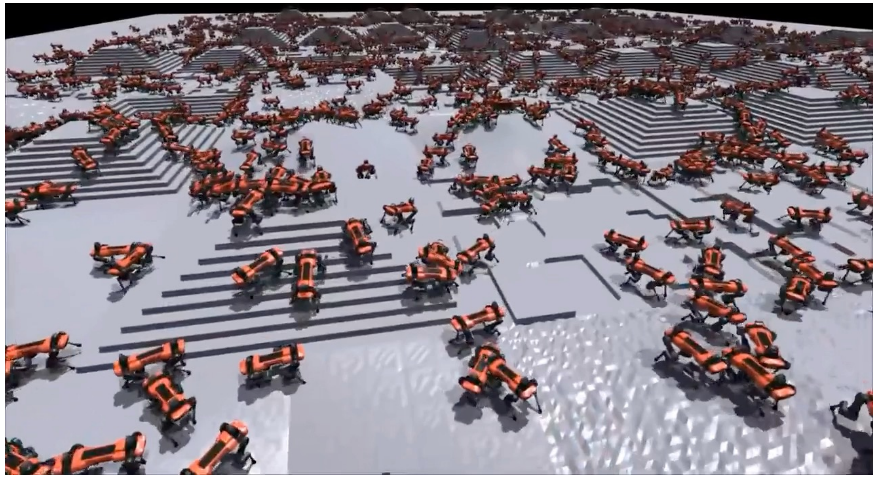


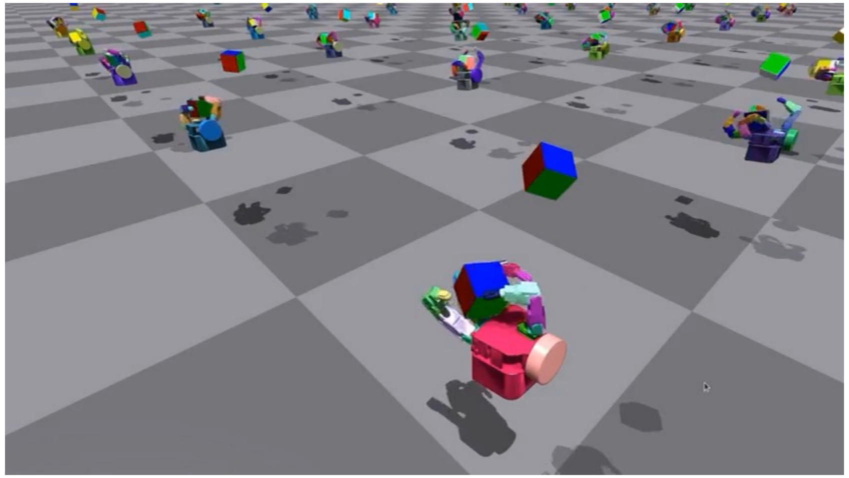


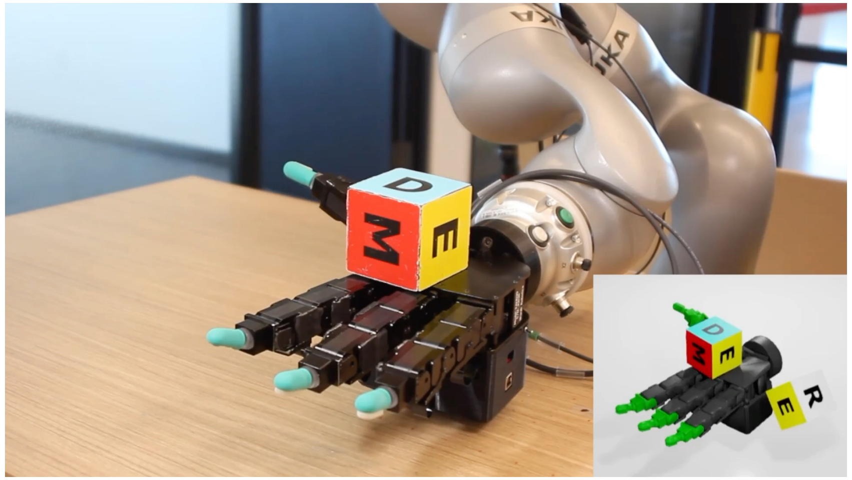


How is this useful for robotics?

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 it's 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

Fin.