CS 287 Lecture 18 (Fall 2019)

RL I: Policy Gradients

Pieter Abbeel
UC Berkeley EECS

Many slides adapted from Thrun, Burgard and Fox, Probabilistic Robotics

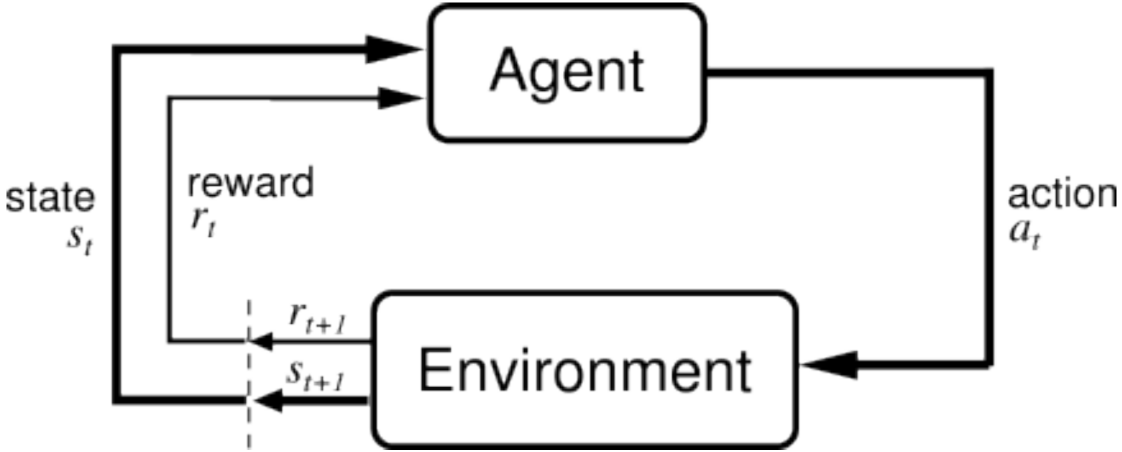
Outline for Today's Lecture

- Super-quick Refresher: Markov Decision Processes (MDPs)
- Reinforcement Learning
- Policy Optimization
- Model-free Policy Optimization: Finite Differences
- Model-free Policy Optimization: Cross-Entropy Method
- Model-free Policy Optimization: Policy Gradients
  - Policy Gradient basic derivation
  - Temporal decomposition
  - Baseline subtraction
  - Value function estimation
  - Advantage Estimation (A2C/A3C/GAE)
  - Trust Region Policy Optimization (TRPO)
  - Proximal Policy Optimization (PPO)

Outline for Today's Lecture

- ***Super-quick Refresher: Markov Decision Processes (MDPs)***
- Reinforcement Learning
- Policy Optimization
- Model-free Policy Optimization: Finite Differences
- Model-free Policy Optimization: Cross-Entropy Method
- Model-free Policy Optimization: Policy Gradients
  - Policy Gradient basic derivation
  - Temporal decomposition
  - Baseline subtraction
  - Value function estimation
  - Advantage Estimation (A2C/A3C/GAE)
  - Trust Region Policy Optimization (TRPO)
  - Proximal Policy Optimization (PPO)

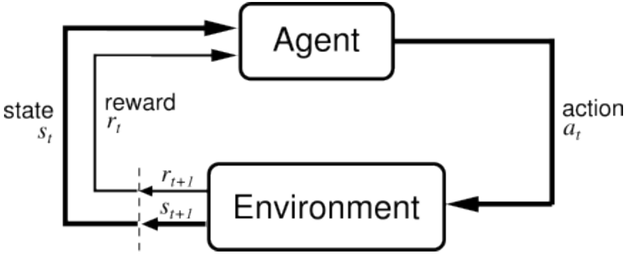
Markov Decision Process



Assumption: agent gets to observe the state

[Drawing from Sutton and Barto, Reinforcement Learning: An Introduction, 1998]

Markov Decision Process (S, A, T, R, γ, H)



Given:

- S: set of states
- A: set of actions
- T: S x A x S x {0,1,…,H} → [0,1] $T_t(s,a,s') = P(s_{t+1} = s' \mid s_t = s, a_t = a)$
- R: S x A x S x {0, 1, …, H} → $\mathbb{R}$ $R_t(s,a,s')$ = reward for $(s_{t+1} = s', s_t = s, a_t = a)$
- γ in (0,1]: discount factor H: horizon over which the agent will act

Goal:

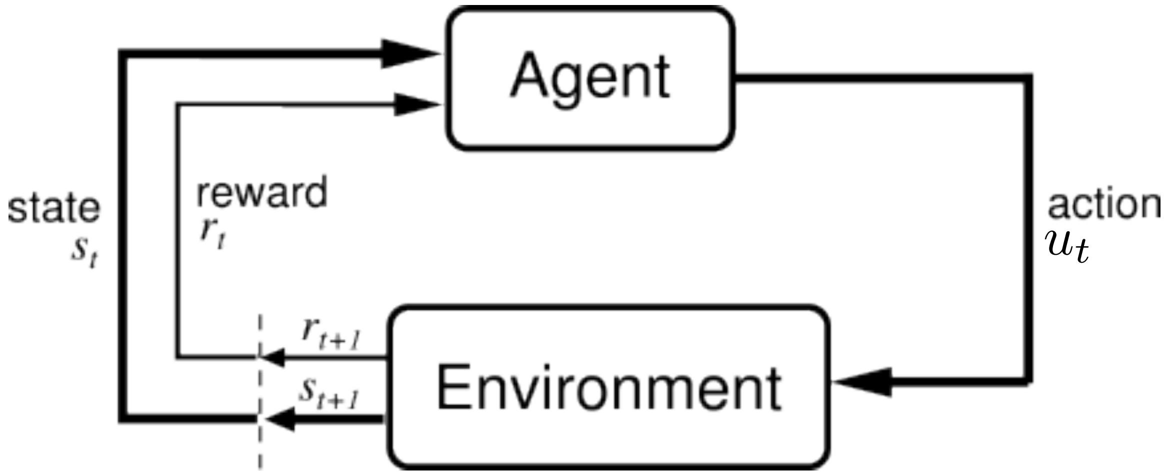
- Find π*: S x {0, 1, …, H} → A that maximizes expected sum of rewards, i.e.,

$$\pi^* = \arg\max_{\pi} \mathrm{E}[\sum_{t=0}^{H} \gamma^t R_t(S_t, A_t, S_{t+1}) | \pi]$$

Outline for Today’s Lecture

- Super-quick Refresher: Markov Decision Processes (MDPs)
- ***Reinforcement Learning***
- Policy Optimization
- Model-free Policy Optimization: Finite Differences
- Model-free Policy Optimization: Cross-Entropy Method
- Model-free Policy Optimization: Policy Gradients
  - Policy Gradient basic derivation
  - Temporal decomposition
  - Baseline subtraction
  - Value function estimation
  - Advantage Estimation (A2C/A3C/GAE)
  - Trust Region Policy Optimization (TRPO)
  - Proximal Policy Optimization (PPO)

Reinforcement Learning

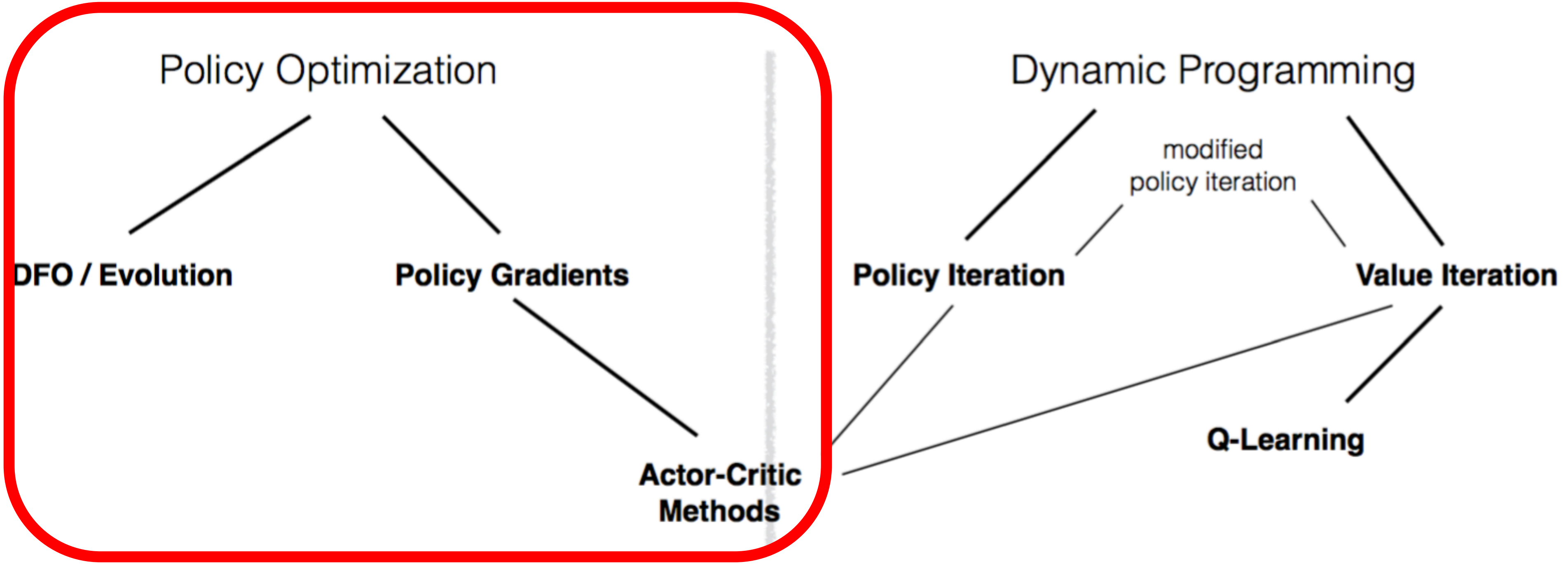


Still an MDP

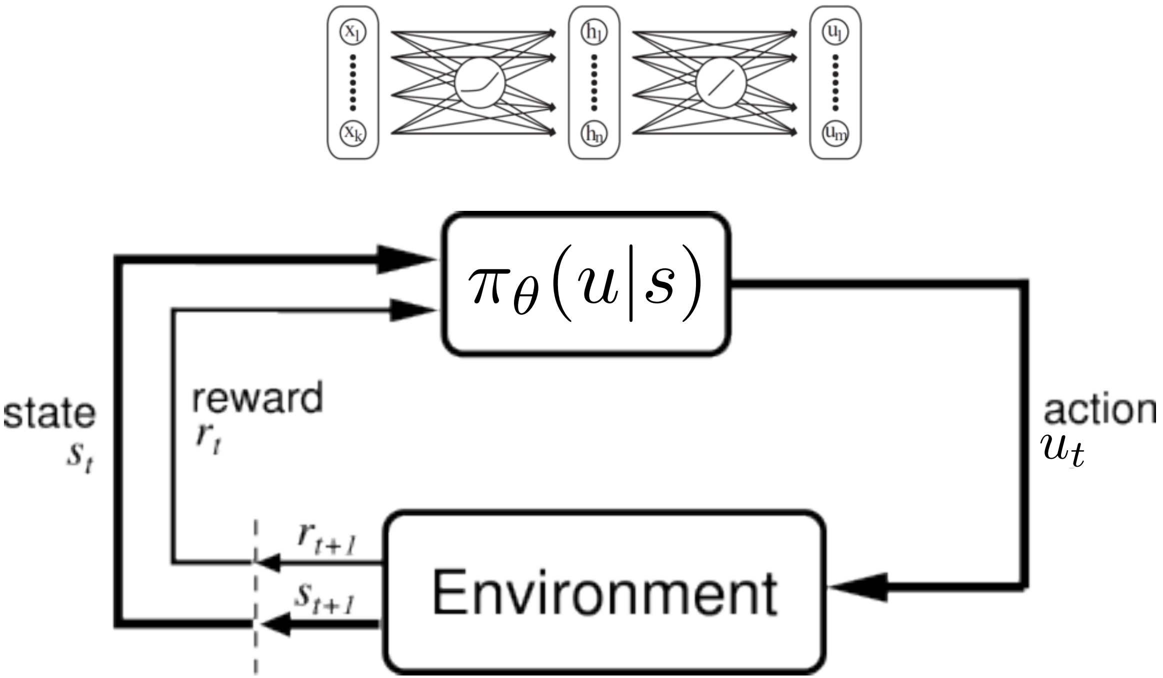
BUT: MDP not given to us, agent needs to learn to optimize reward through trial and error

[Figure source: Sutton & Barto, 1998]

Policy Optimization in the RL Landscape

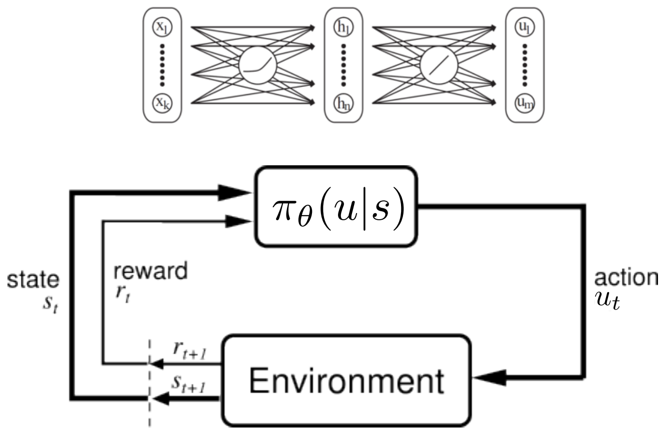


Policy Optimization



[Figure source: Sutton & Barto, 1998]

Policy Optimization



[Figure source: Sutton & Barto, 1998]

- Consider control policy parameterized by parameter vector θ

$$\max_\theta \quad \mathrm{E}[\sum_{t=0}^{H} R(s_t)|\pi_\theta]$$

- Stochastic policy class (smooths out the problem):

  $\pi_\theta(u|s)$: probability of action u in state s

Why Policy Optimization

- Often π can be simpler than Q or V
  - E.g., robotic grasp
- V: doesn't prescribe actions
  - Would need dynamics model (+ compute 1 Bellman back-up)
- Q: need to be able to efficiently solve $\arg\max_u Q_\theta(s,u)$
  - Challenge for continuous / high-dimensional action spaces*

*some recent work (partially) addressing this:

NAF: Gu, Lillicrap, Sutskever, Levine ICML 2016
Input Convex NNs: Amos, Xu, Kolter arXiv 2016
Deep Energy Q: Haarnoja, Tang, Abbeel, Levine, ICML 2017

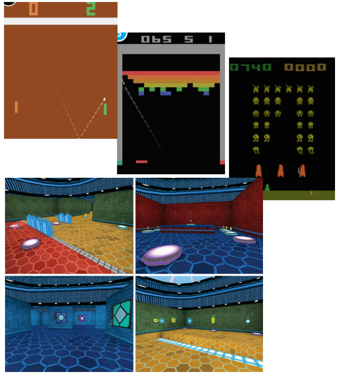
Pioneering Policy Optimization Success Stories

Kohl and Stone, 2004

Ng et al, 2004

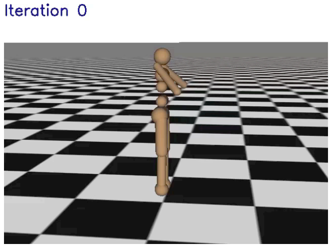
Tedrake et al, 2005

Kober and Peters, 2009

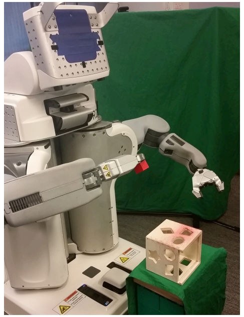


Mnih et al, 2015
(A3C)

Silver et al, 2014
(DPG)
Lillicrap et al, 2015
(DDPG)



Schulman et al, 2016 (TRPO + GAE)



Levine*, Finn*, et al, 2016
(GPS)

Silver*, Huang*, et al, 2016
(AlphaGo**)

	Policy Optimization	Dynamic Programming
Conceptually:	Optimize what you care about	Indirect, exploit the problem structure, self-consistency
Empirically:	More compatible with rich architectures (including recurrence) More versatile More compatible with auxiliary objectives	More compatible with exploration and off-policy learning More sample-efficient when they work

Note: We have done policy optimization before!

- iLQR
- Optimization-based Control: Collocation, Shooting, MPC, Contact Invariant Optimization

→ But these assumed access to the dynamics model, which we don't have available now

Note: in 3rd lecture on RL we'll cover model-based RL, which learns the dynamics model, and can use above methods

Outline for Today's Lecture

- Super-quick Refresher: Markov Decision Processes (MDPs)
- Reinforcement Learning
- Policy Optimization
- ***Model-free Policy Optimization: Finite Differences***
- Model-free Policy Optimization: Cross-Entropy Method
- Model-free Policy Optimization: Policy Gradients
  - Policy Gradient standard derivation
  - Temporal decomposition
  - Policy Gradient importance sampling derivation
  - Baseline subtraction
  - Value function estimation
  - Advantage Estimation (A2C/A3C/GAE)
  - Trust Region Policy Optimization (TRPO)
  - Proximal Policy Optimization (PPO)

Black Box Gradient Computation

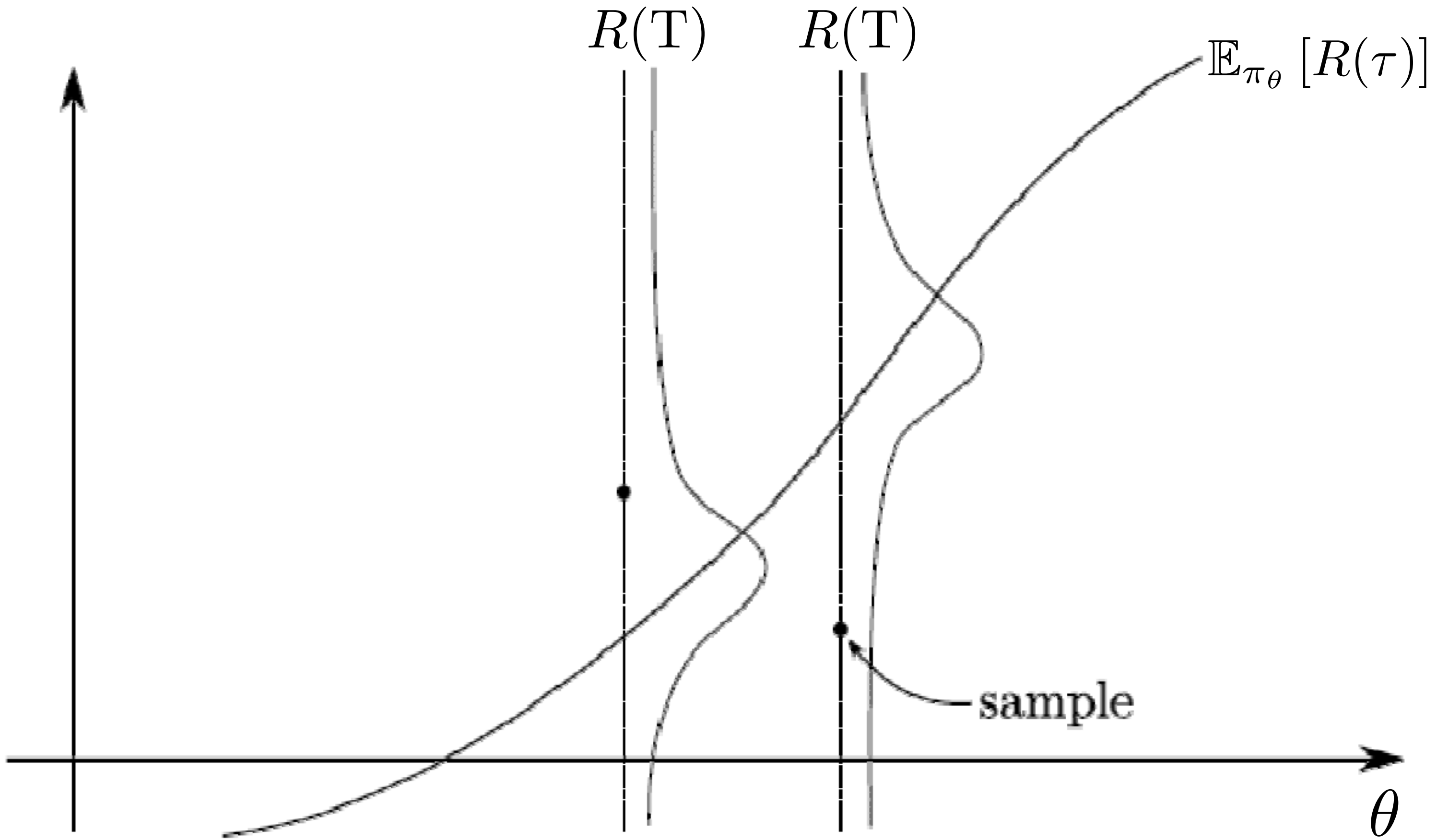
We can compute the gradient g using standard finite difference methods, as follows:

$$\frac{\partial U}{\partial \theta_j}(\theta) = \frac{U(\theta + \epsilon e_j) - U(\theta - \epsilon e_j)}{2\epsilon}$$

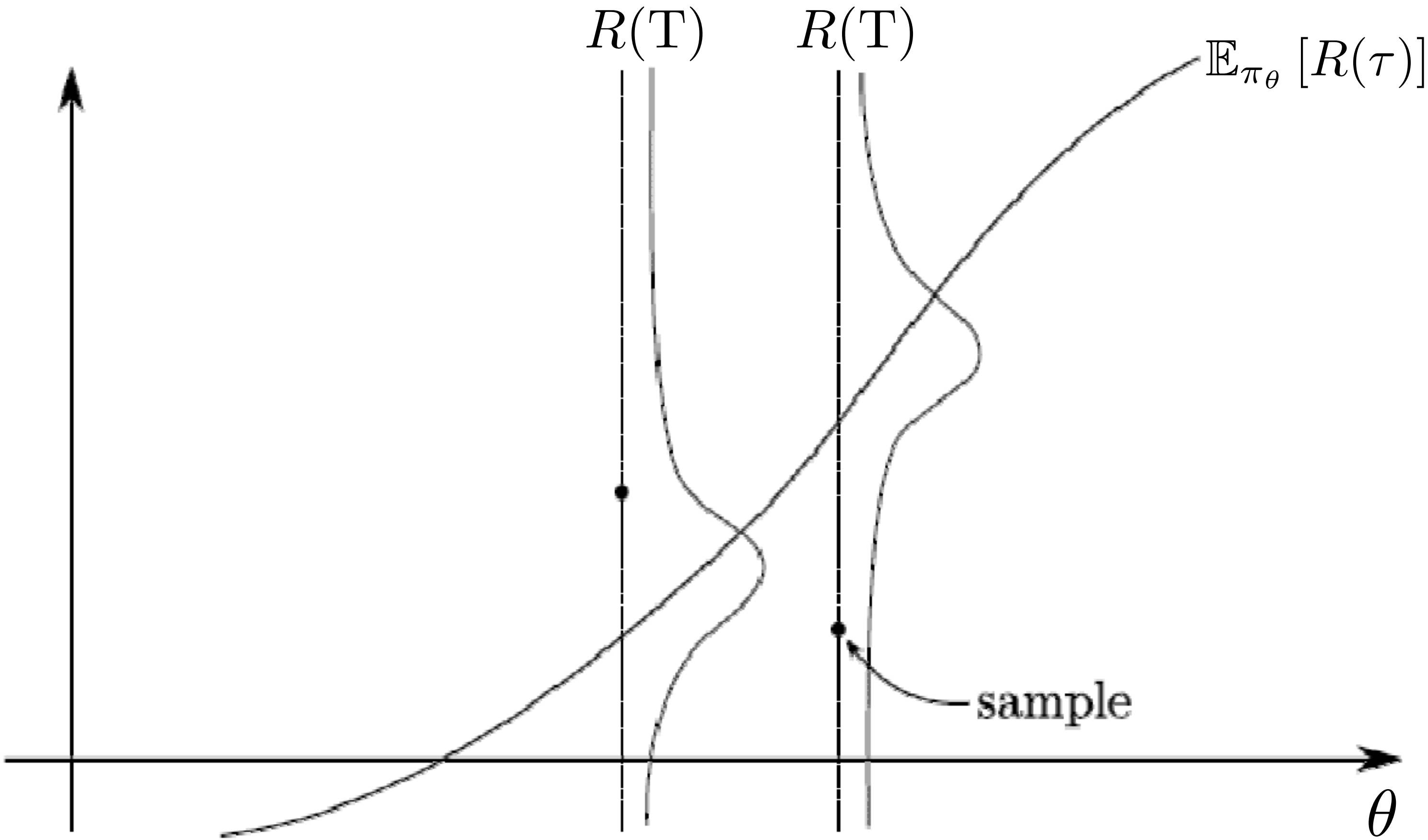
Where:

$$e_j = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \begin{array}{l} \\ \\ \\ \\ \leftarrow \; j\text{'th entry} \\ \\ \\ \\ \end{array}$$

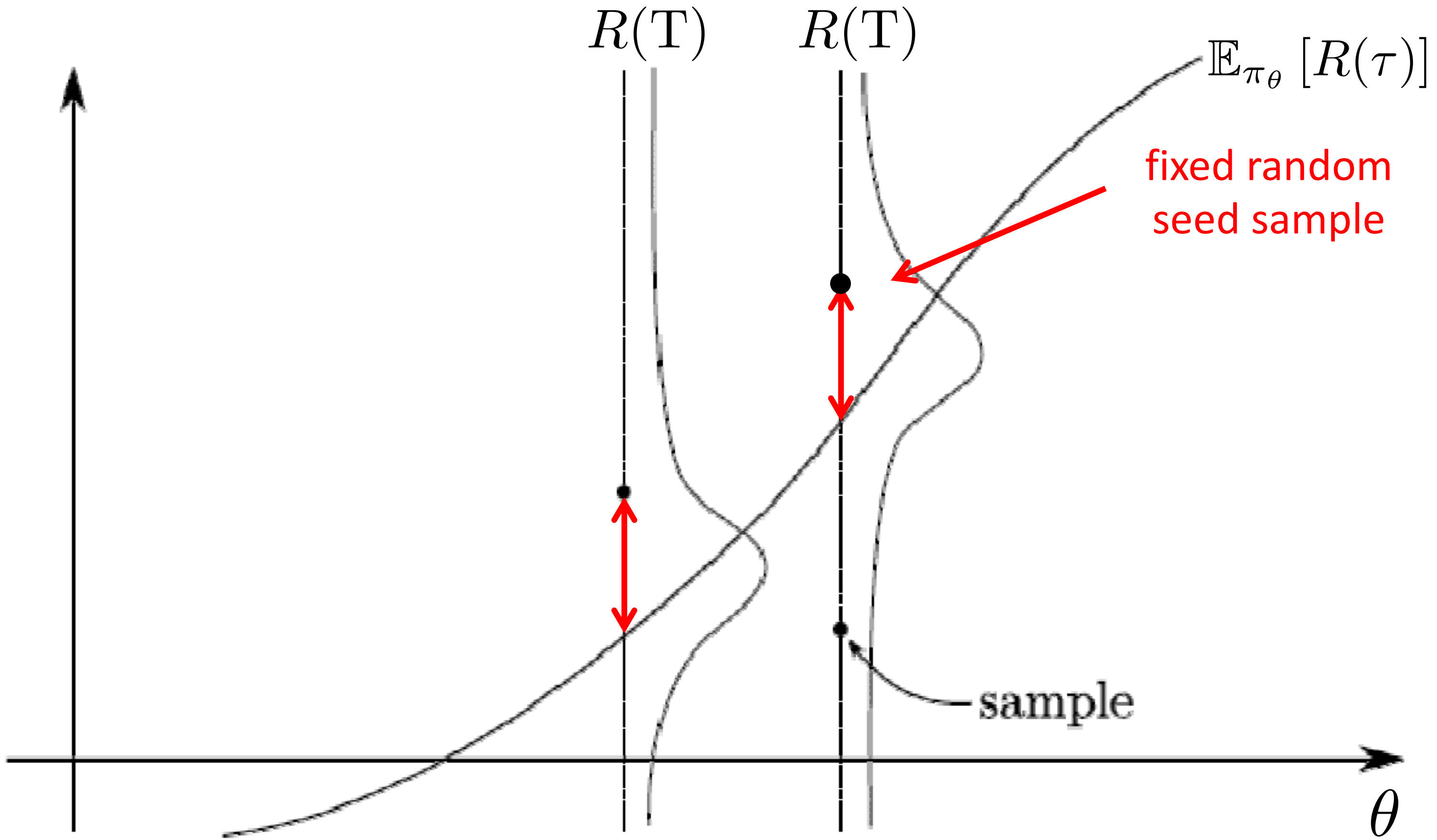
Challenge: Noise Can Dominate



Solution 1: Average over many samples



Solution 2: Fix random seed



Solution 2: Fix random seed

- Randomness in policy and dynamics
  - But can often only control randomness in policy..
- Example: wind influence on a helicopter is stochastic, but if we assume the same wind pattern across trials, this will make the different choices of θ more readily comparable
- *Note: equally applicable to evolutionary methods*

[Ng & Jordan, 2000] provide theoretical analysis of gains from fixing randomness ("pegasus")

Learning to Hover

x, y, z: x points forward along the helicopter, y sideways to the right, z downward.

n_x, n_y, n_z: rotation vector that brings helicopter back to "level" position (expressed in the helicopter frame).

$$u_{collective} = \theta_1 \cdot f_1(z^* - z) + \theta_2 \cdot \dot{z}$$

$$u_{elevator} = \theta_3 \cdot f_2(x^* - x) + \theta_4 f_4(\dot{x}) + \theta_5 \cdot q + \theta_6 \cdot n_y$$

$$u_{aileron} = \theta_7 \cdot f_3(y^* - y) + \theta_8 f_5(\dot{y}) + \theta_9 \cdot p + \theta_{10} \cdot n_x$$

$$u_{rudder} = \theta_{11} \cdot r + \theta_{12} \cdot n_z$$

Example: Sidewinding

[Andrew Ng]

[Video: SNAKE – climbStep+side

Example: Learning to Walk

Initial

A Learning Trial

After Learning [1K Trials]

[Kohl and Stone, ICRA 2004]

Example: Learning to Walk

Initial

[Kohl and Stone, ICRA 2004]

[Video: AIBO WALK – in

Example: Learning to Walk

Training

[Kohl and Stone, ICRA 2004]

[Video: AIBO WALK – tr

Example: Learning to Walk

Finished

[Kohl and Stone, ICRA 2004]

[Video: AIBO WALK – fi

Finite Differences

- Can work well!
- Most success in low-dimensional spaces…

Outline for Today's Lecture

- Super-quick Refresher: Markov Decision Processes (MDPs)
- Reinforcement Learning
- Policy Optimization
- Model-free Policy Optimization: Finite Differences
- ***Model-free Policy Optimization: Cross-Entropy Method***
- Model-free Policy Optimization: Policy Gradients
  - Policy Gradient standard derivation
  - Temporal decomposition
  - Policy Gradient importance sampling derivation
  - Baseline subtraction
  - Value function estimation
  - Advantage Estimation (A2C/A3C/GAE)
  - Trust Region Policy Optimization (TRPO)
  - Proximal Policy Optimization (PPO)

Evolutionary Methods

$$\max_{\theta} U(\theta) = \max_{\theta} \mathrm{E}[\sum_{t=0}^{H} R(s_t)|\pi_\theta]$$

- General Algorithm:
  - Make some random change to the parameters
  - If the result improves, keep the change
  - Repeat

Cross-Entropy Method

CEM:

Initialize $\mu \in \mathbb{R}^d, \sigma \in \mathbb{R}^d_{>0}$

for iteration = 1, 2, …

Sample n parameters $\theta_i \sim N(\mu, \mathrm{diag}(\sigma^2))$

For each θ_i, perform one rollout to get return $R(\tau_i)$

Select the top k% of θ, and fit a new diagonal Gaussian to those samples. Update μ, σ

endfor

Cross-Entropy Method

- Very simple and can work surprisingly well
- Very scalable
- Does not take advantage of any temporal structure

Outline for Today's Lecture

- Super-quick Refresher: Markov Decision Processes (MDPs)
- Reinforcement Learning
- Policy Optimization
- Model-free Policy Optimization: Finite Differences
- Model-free Policy Optimization: Cross-Entropy Method
- ***Model-free Policy Optimization: Policy Gradients***
  - Policy Gradient standard derivation
  - Temporal decomposition
  - Policy Gradient importance sampling derivation
  - Baseline subtraction
  - Value function estimation
  - Advantage Estimation (A2C/A3C/GAE)
  - Trust Region Policy Optimization (TRPO)
  - Proximal Policy Optimization (PPO)

Likelihood Ratio Policy Gradient

We let τ denote a state-action sequence $s_0, u_0, \ldots, s_H, u_H$. We overload notation: $R(\tau) = \sum_{t=0}^{H} R(s_t, u_t)$.

$$U(\theta) = \mathrm{E}[\sum_{t=0}^{H} R(s_t, u_t); \pi_\theta] = \sum_{\tau} P(\tau; \theta) R(\tau)$$

In our new notation, our goal is to find θ:

$$\max_{\theta} U(\theta) = \max_{\theta} \sum_{\tau} P(\tau; \theta) R(\tau)$$

Likelihood Ratio Policy Gradient

$$U(\theta) = \sum_{\tau} P(\tau; \theta) R(\tau)$$

Taking the gradient w.r.t. θ gives

$$\nabla_\theta U(\theta) = \nabla_\theta \sum_{\tau} P(\tau; \theta) R(\tau)$$

[Aleksandrov, Sysoyev, & Shemeneva, 1968]
[Rubinstein, 1969]
[Glynn, 1986]
[Reinforce, Williams 1992]
[GPOMDP, Baxter & Bartlett, 2001]

Likelihood Ratio Policy Gradient

$$U(\theta) = \sum_{\tau} P(\tau;\theta)R(\tau)$$

Taking the gradient w.r.t. θ gives

$$\begin{aligned}\nabla_\theta U(\theta) &= \nabla_\theta \sum_{\tau} P(\tau;\theta)R(\tau) \\ &= \sum_{\tau} \nabla_\theta P(\tau;\theta)R(\tau)\end{aligned}$$

[Aleksandrov, Sysoyev, & Shemeneva, 1968]
[Rubinstein, 1969]
[Glynn, 1986]
[Reinforce, Williams 1992]
[GPOMDP, Baxter & Bartlett, 2001]

Likelihood Ratio Policy Gradient

$$U(\theta) = \sum_{\tau} P(\tau;\theta) R(\tau)$$

Taking the gradient w.r.t. θ gives

$$\begin{aligned}
\nabla_\theta U(\theta) &= \nabla_\theta \sum_{\tau} P(\tau;\theta) R(\tau) \\
&= \sum_{\tau} \nabla_\theta P(\tau;\theta) R(\tau) \\
&= \sum_{\tau} \frac{P(\tau;\theta)}{P(\tau;\theta)} \nabla_\theta P(\tau;\theta) R(\tau)
\end{aligned}$$

[Aleksandrov, Sysoyev, & Shemeneva, 1968]
[Rubinstein, 1969]
[Glynn, 1986]
[Reinforce, Williams 1992]
[GPOMDP, Baxter & Bartlett, 2001]

Likelihood Ratio Policy Gradient

$$U(\theta) = \sum_{\tau} P(\tau; \theta) R(\tau)$$

Taking the gradient w.r.t. θ gives

$$\begin{aligned}
\nabla_\theta U(\theta) &= \nabla_\theta \sum_{\tau} P(\tau; \theta) R(\tau) \\
&= \sum_{\tau} \nabla_\theta P(\tau; \theta) R(\tau) \\
&= \sum_{\tau} \frac{P(\tau; \theta)}{P(\tau; \theta)} \nabla_\theta P(\tau; \theta) R(\tau) \\
&= \sum_{\tau} P(\tau; \theta) \frac{\nabla_\theta P(\tau; \theta)}{P(\tau; \theta)} R(\tau)
\end{aligned}$$

[Aleksandrov, Sysoyev, & Shemeneva, 1968]
[Rubinstein, 1969]
[Glynn, 1986]
[Reinforce, Williams 1992]
[GPOMDP, Baxter & Bartlett, 2001]

Likelihood Ratio Policy Gradient

$$U(\theta) = \sum_\tau P(\tau;\theta)R(\tau)$$

Taking the gradient w.r.t. θ gives

$$\begin{aligned}
\nabla_\theta U(\theta) &= \nabla_\theta \sum_\tau P(\tau;\theta)R(\tau) \\
&= \sum_\tau \nabla_\theta P(\tau;\theta)R(\tau) \\
&= \sum_\tau \frac{P(\tau;\theta)}{P(\tau;\theta)} \nabla_\theta P(\tau;\theta)R(\tau) \\
&= \sum_\tau P(\tau;\theta) \frac{\nabla_\theta P(\tau;\theta)}{P(\tau;\theta)} R(\tau) \\
&= \sum_\tau P(\tau;\theta) \nabla_\theta \log P(\tau;\theta) R(\tau)
\end{aligned}$$

[Aleksandrov, Sysoyev, & Shemeneva, 1968]
[Rubinstein, 1969]
[Glynn, 1986]
[Reinforce, Williams 1992]
[GPOMDP, Baxter & Bartlett, 2001]

Likelihood Ratio Policy Gradient

$$U(\theta) = \sum_{\tau} P(\tau;\theta)R(\tau)$$

Taking the gradient w.r.t. θ gives

$$\begin{aligned}
\nabla_\theta U(\theta) &= \nabla_\theta \sum_{\tau} P(\tau;\theta)R(\tau) \\
&= \sum_{\tau} \nabla_\theta P(\tau;\theta)R(\tau) \\
&= \sum_{\tau} \frac{P(\tau;\theta)}{P(\tau;\theta)} \nabla_\theta P(\tau;\theta)R(\tau) \\
&= \sum_{\tau} P(\tau;\theta) \frac{\nabla_\theta P(\tau;\theta)}{P(\tau;\theta)} R(\tau) \\
&= \sum_{\tau} P(\tau;\theta) \nabla_\theta \log P(\tau;\theta) R(\tau)
\end{aligned}$$

Approximate with the empirical estimate for m sample paths under policy π_θ:

$$\nabla_\theta U(\theta) \approx \hat{g} = \frac{1}{m} \sum_{i=1}^{m} \nabla_\theta \log P(\tau^{(i)};\theta) R(\tau^{(i)})$$

[Aleksandrov, Sysoyev, & Shemeneva, 1968]
[Rubinstein, 1969]
[Glynn, 1986]
[Reinforce, Williams 1992]
[GPOMDP, Baxter & Bartlett, 2001]

Likelihood Ratio Gradient: Validity

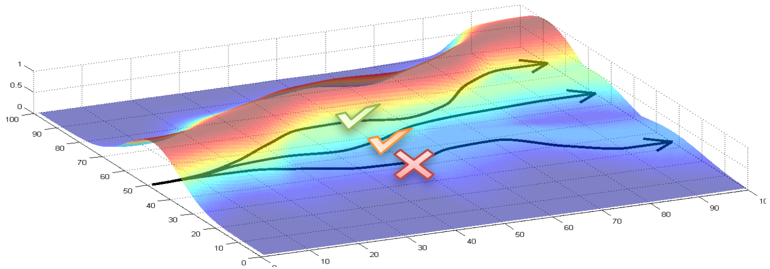
$$\nabla U(\theta) \approx \hat{g} = \frac{1}{m} \sum_{i=1}^{m} \nabla_\theta \log P(\tau^{(i)}; \theta) R(\tau^{(i)})$$

- Valid even when
  - R is discontinuous and/or unknown
  - Sample space (of paths) is a discrete set

Likelihood Ratio Gradient: Intuition

$$\nabla U(\theta) \approx \hat{g} = \frac{1}{m}\sum_{i=1}^{m} \nabla_\theta \log P(\tau^{(i)};\theta) R(\tau^{(i)})$$

- Gradient tries to:
  - Increase probability of paths with positive R
  - Decrease probability of paths with negative R



! Likelihood ratio changes probabilities of experienced paths, does not try to change the paths (<-> Path Derivative)

Let's Decompose Path into States and Actions

$$\nabla_\theta \log P(\tau^{(i)};\theta) = \nabla_\theta \log \left[\prod_{t=0}^{H} \underbrace{P(s_{t+1}^{(i)}|s_t^{(i)}, u_t^{(i)})}_{\text{dynamics model}} \cdot \underbrace{\pi_\theta(u_t^{(i)}|s_t^{(i)})}_{\text{policy}} \right]$$

Let's Decompose Path into States and Actions

$$\nabla_\theta \log P(\tau^{(i)};\theta) = \nabla_\theta \log \left[\prod_{t=0}^{H} \underbrace{P(s_{t+1}^{(i)}|s_t^{(i)}, u_t^{(i)})}_{\text{dynamics model}} \cdot \underbrace{\pi_\theta(u_t^{(i)}|s_t^{(i)})}_{\text{policy}} \right]$$

$$= \nabla_\theta \left[\sum_{t=0}^{H} \log P(s_{t+1}^{(i)}|s_t^{(i)}, u_t^{(i)}) + \sum_{t=0}^{H} \log \pi_\theta(u_t^{(i)}|s_t^{(i)}) \right]$$

Let’s Decompose Path into States and Actions

$$\nabla_\theta \log P(\tau^{(i)};\theta) = \nabla_\theta \log \left[\prod_{t=0}^{H} \underbrace{P(s_{t+1}^{(i)}|s_t^{(i)}, u_t^{(i)})}_{\text{dynamics model}} \cdot \underbrace{\pi_\theta(u_t^{(i)}|s_t^{(i)})}_{\text{policy}} \right]$$

$$= \nabla_\theta \left[\sum_{t=0}^{H} \log P(s_{t+1}^{(i)}|s_t^{(i)}, u_t^{(i)}) + \sum_{t=0}^{H} \log \pi_\theta(u_t^{(i)}|s_t^{(i)}) \right]$$

$$= \nabla_\theta \sum_{t=0}^{H} \log \pi_\theta(u_t^{(i)}|s_t^{(i)})$$

Let's Decompose Path into States and Actions

$$
\begin{aligned}
\nabla_\theta \log P(\tau^{(i)};\theta) &= \nabla_\theta \log \left[\prod_{t=0}^{H} \underbrace{P(s_{t+1}^{(i)} | s_t^{(i)}, u_t^{(i)})}_{\text{dynamics model}} \cdot \underbrace{\pi_\theta(u_t^{(i)} | s_t^{(i)})}_{\text{policy}} \right] \\
&= \nabla_\theta \left[\sum_{t=0}^{H} \log P(s_{t+1}^{(i)} | s_t^{(i)}, u_t^{(i)}) + \sum_{t=0}^{H} \log \pi_\theta(u_t^{(i)} | s_t^{(i)}) \right] \\
&= \nabla_\theta \sum_{t=0}^{H} \log \pi_\theta(u_t^{(i)} | s_t^{(i)}) \\
&= \sum_{t=0}^{H} \underbrace{\nabla_\theta \log \pi_\theta(u_t^{(i)} | s_t^{(i)})}_{\text{no dynamics model required!!}}
\end{aligned}
$$

Likelihood Ratio Gradient Estimate

The following expression provides us with an unbiased estimate of the gradient, and we can compute it without access to a dynamics model:

$$\hat{g} = \frac{1}{m} \sum_{i=1}^{m} \nabla_\theta \log P(\tau^{(i)};\theta) R(\tau^{(i)})$$

Here:

$$\nabla_\theta \log P(\tau^{(i)};\theta) = \sum_{t=0}^{H} \underbrace{\nabla_\theta \log \pi_\theta(u_t^{(i)}|s_t^{(i)})}_{\text{no dynamics model required!!}}$$

Unbiased means:

$$\mathrm{E}[\hat{g}] = \nabla_\theta U(\theta)$$

Outline for Today's Lecture

- Super-quick Refresher: Markov Decision Processes (MDPs)
- Reinforcement Learning
- Policy Optimization
- Model-free Policy Optimization: Finite Differences
- Model-free Policy Optimization: Cross-Entropy Method
- Model-free Policy Optimization: Policy Gradients
  - Policy Gradient standard derivation
  - Temporal decomposition
  - ***Policy Gradient importance sampling derivation***
  - Baseline subtraction
  - Value function estimation
  - Advantage Estimation (A2C/A3C/GAE)
  - Trust Region Policy Optimization (TRPO)
  - Proximal Policy Optimization (PPO)

Derivation from Importance Sampling

$$U(\theta) = \mathbb{E}_{\tau\sim\theta_{\text{old}}}\left[\frac{P(\tau|\theta)}{P(\tau|\theta_{\text{old}})}R(\tau)\right]$$

[Tang&Abbeel, NeurIPS 2011]

Derivation from Importance Sampling

$$U(\theta) = \mathbb{E}_{\tau \sim \theta_{\text{old}}} \left[\frac{P(\tau|\theta)}{P(\tau|\theta_{\text{old}})} R(\tau) \right]$$

$$\nabla_\theta U(\theta) = \mathbb{E}_{\tau \sim \theta_{\text{old}}} \left[\frac{\nabla_\theta P(\tau|\theta)}{P(\tau|\theta_{\text{old}})} R(\tau) \right]$$

[Tang&Abbeel, NeurIPS 2011]

Derivation from Importance Sampling

$$U(\theta) = \mathbb{E}_{\tau \sim \theta_{\text{old}}} \left[\frac{P(\tau|\theta)}{P(\tau|\theta_{\text{old}})} R(\tau) \right]$$

$$\nabla_\theta U(\theta) = \mathbb{E}_{\tau \sim \theta_{\text{old}}} \left[\frac{\nabla_\theta P(\tau|\theta)}{P(\tau|\theta_{\text{old}})} R(\tau) \right]$$

$$\nabla_\theta \left. U(\theta) \right|_{\theta=\theta_{\text{old}}} = \mathbb{E}_{\tau \sim \theta_{\text{old}}} \left[\frac{\nabla_\theta \left. P(\tau|\theta) \right|_{\theta_{\text{old}}}}{P(\tau|\theta_{\text{old}})} R(\tau) \right]$$

[Tang&Abbeel, NeurIPS 2011]

Derivation from Importance Sampling

$$U(\theta) = \mathbb{E}_{\tau \sim \theta_{\text{old}}} \left[\frac{P(\tau|\theta)}{P(\tau|\theta_{\text{old}})} R(\tau) \right]$$

$$\nabla_\theta U(\theta) = \mathbb{E}_{\tau \sim \theta_{\text{old}}} \left[\frac{\nabla_\theta P(\tau|\theta)}{P(\tau|\theta_{\text{old}})} R(\tau) \right]$$

$$\nabla_\theta \left. U(\theta) \right|_{\theta=\theta_{\text{old}}} = \mathbb{E}_{\tau \sim \theta_{\text{old}}} \left[\frac{\nabla_\theta \left. P(\tau|\theta) \right|_{\theta_{\text{old}}}}{P(\tau|\theta_{\text{old}})} R(\tau) \right]$$

$$= \mathbb{E}_{\tau \sim \theta_{\text{old}}} \left[\nabla_\theta \log \left. P(\tau|\theta) \right|_{\theta_{\text{old}}} R(\tau) \right]$$

[Tang&Abbeel, NeurIPS 2011]

Derivation from Importance Sampling

$$U(\theta) = \mathbb{E}_{\tau \sim \theta_{\text{old}}} \left[\frac{P(\tau|\theta)}{P(\tau|\theta_{\text{old}})} R(\tau) \right]$$

$$\nabla_\theta U(\theta) = \mathbb{E}_{\tau \sim \theta_{\text{old}}} \left[\frac{\nabla_\theta P(\tau|\theta)}{P(\tau|\theta_{\text{old}})} R(\tau) \right]$$

$$\nabla_\theta \left. U(\theta) \right|_{\theta=\theta_{\text{old}}} = \mathbb{E}_{\tau \sim \theta_{\text{old}}} \left[\frac{\nabla_\theta \left. P(\tau|\theta) \right|_{\theta_{\text{old}}}}{P(\tau|\theta_{\text{old}})} R(\tau) \right]$$

$$= \mathbb{E}_{\tau \sim \theta_{\text{old}}} \left[\nabla_\theta \log \left. P(\tau|\theta) \right|_{\theta_{\text{old}}} R(\tau) \right]$$

Suggests we can also look at more than just gradient!

E.g., can use importance sampled objective as "surrogate loss" (locally) [[→ later: PPO]]

[Tang&Abbeel, NeurIPS 2011]

Outline for Today's Lecture

- Super-quick Refresher: Markov Decision Processes (MDPs)
- Reinforcement Learning
- Policy Optimization
- Model-free Policy Optimization: Finite Differences
- Model-free Policy Optimization: Cross-Entropy Method
- Model-free Policy Optimization: Policy Gradients
  - Policy Gradient standard derivation
  - Temporal decomposition
  - Policy Gradient importance sampling derivation
  - ***Baseline subtraction and temporal structure***
  - Value function estimation
  - Advantage Estimation (A2C/A3C/GAE)
  - Trust Region Policy Optimization (TRPO)
  - Proximal Policy Optimization (PPO)

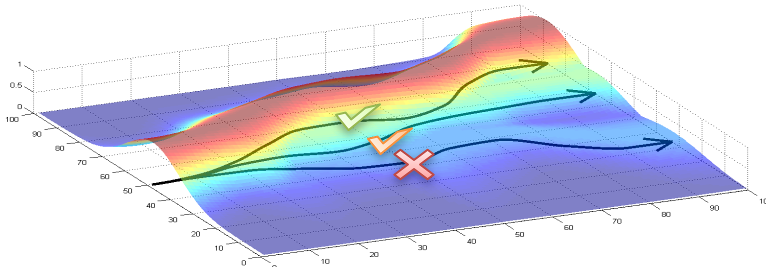
Likelihood Ratio Gradient Estimate

- As formulated thus far: unbiased but very noisy
- Fixes that lead to real-world practicality
  - Baseline
  - Temporal structure
  - [later] Trust region / natural gradient

Likelihood Ratio Gradient: Intuition

$$\nabla U(\theta) \approx \hat{g} = \frac{1}{m} \sum_{i=1}^{m} \nabla_\theta \log P(\tau^{(i)}; \theta) R(\tau^{(i)})$$

- Gradient tries to:
  - Increase probability of paths with positive R
  - Decrease probability of paths with negative R



! Likelihood ratio changes probabilities of experienced paths, does not try to change the paths (<-> Path Derivative)

Likelihood Ratio Gradient: Baseline

$$\nabla U(\theta) \approx \hat{g} = \frac{1}{m}\sum_{i=1}^{m} \nabla_\theta \log P(\tau^{(i)};\theta) R(\tau^{(i)})$$

→ Consider baseline b: $\nabla U(\theta) \approx \hat{g} = \frac{1}{m}\sum_{i=1}^{m} \nabla_\theta \log P(\tau^{(i)};\theta)(R(\tau^{(i)}) - b)$

still unbiased!

[Williams 1992]

$$\begin{aligned}
&\mathbb{E}\left[\nabla_\theta \log P(\tau;\theta) b\right] \\
&= \sum_\tau P(\tau;\theta) \nabla_\theta \log P(\tau;\theta) b \\
&= \sum_\tau P(\tau;\theta) \frac{\nabla_\theta P(\tau;\theta)}{P(\tau;\theta)} b \\
&= \sum_\tau \nabla_\theta P(\tau;\theta) b \\
&= \nabla_\theta \left(\sum_\tau P(\tau) b\right) \\
&= \nabla_\theta (b) \\
&= 0
\end{aligned}$$

$$= b\nabla_\theta (\sum_\tau P(\tau)) = b \times 0$$

OK as long as baseline doesn't depend on action in logprob(action)

Likelihood Ratio and Temporal Structure

- Current estimate:

$$\hat{g} = \frac{1}{m}\sum_{i=1}^{m} \nabla_\theta \log P(\tau^{(i)};\theta)(R(\tau^{(i)}) - b)$$

$$= \frac{1}{m}\sum_{i=1}^{m} \left(\sum_{t=0}^{H-1} \nabla_\theta \log \pi_\theta(u_t^{(i)}|s_t^{(i)})\right)\left(\sum_{t=0}^{H-1} R(s_t^{(i)}, u_t^{(i)}) - b\right)$$

$$= \frac{1}{m}\sum_{i=1}^{m} \left(\sum_{t=0}^{H-1} \nabla_\theta \log \pi_\theta(u_t^{(i)}|s_t^{(i)}) \left[\left(\sum_{k=0}^{t-1} R(s_k^{(i)}, u_k^{(i)})\right) + \left(\sum_{k=t}^{H-1} R(s_k^{(i)}, u_k^{(i)})\right) - b \right]\right)$$

Doesn't depend on $u_t^{(i)}$

Ok to depend on $s_t^{(i)}$

- Removing terms that don't depend on current action can lower variance:

$$\frac{1}{m}\sum_{i=1}^{m}\sum_{t=0}^{H-1} \nabla_\theta \log \pi_\theta(u_t^{(i)}|s_t^{(i)}) \left(\sum_{k=t}^{H-1} R(s_k^{(i)}, u_k^{(i)}) - b(s_t^{(i)})\right)$$

[Policy Gradient Theorem: Sutton et al, NIPS 1999; GPOMDP: Bartlett & Baxter, JAIR 2001; Survey: Peters & Schaal, IROS 2006]

Baseline Choices

- Good choice for b?
  - Constant baseline: $b = \mathbb{E}\left[R(\tau)\right] \approx \frac{1}{m}\sum_{i=1}^{m} R(\tau^{(i)})$
  - Optimal Constant baseline: $b = \frac{\sum_i \left(\nabla_\theta \log P(\tau^{(i)};\theta)\right)^2 R(\tau^{(i)})}{\sum_i \left(\nabla_\theta \log P(\tau^{(i)};\theta)\right)^2}$
  - Time-dependent baseline: $b_t = \frac{1}{m}\sum_{i=1}^{m}\sum_{k=t}^{H-1} R(s_k^{(i)}, u_k^{(i)})$
  - State-dependent expected return:

$$b(s_t) = \mathbb{E}\left[r_t + r_{t+1} + r_{t+2} + \ldots + r_{H-1}\right] = V^\pi(s_t)$$

→ Increase logprob of action proportionally to how much its returns are better than the expected return under the current policy

[See: Greensmith, Bartlett, Baxter, JMLR 2004 for variance reduction techniques.]

Outline for Today's Lecture

- Super-quick Refresher: Markov Decision Processes (MDPs)
- Reinforcement Learning
- Policy Optimization
- Model-free Policy Optimization: Finite Differences
- Model-free Policy Optimization: Cross-Entropy Method
- Model-free Policy Optimization: Policy Gradients
  - Policy Gradient standard derivation
  - Temporal decomposition
  - Policy Gradient importance sampling derivation
  - Baseline subtraction & temporal structure
  - ***Value function estimation***
  - Advantage Estimation (A2C/A3C/GAE)
  - Trust Region Policy Optimization (TRPO)
  - Proximal Policy Optimization (PPO)

Monte Carlo Estimation of V^π

$$\frac{1}{m}\sum_{i=1}^{m}\sum_{t=0}^{H-1}\nabla_\theta \log \pi_\theta(u_t^{(i)}|s_t^{(i)})\left(\sum_{k=t}^{H-1}R(s_k^{(i)},u_k^{(i)}) - \underbrace{V^\pi(s_k^{(i)})}\right)$$

How to estimate?

- Init $V^\pi_{\phi_0}$
  - Collect trajectories $\tau_1,\ldots,\tau_m$
  - Regress against empirical return:

$$\phi_{i+1} \leftarrow \arg\min_\phi \frac{1}{m}\sum_{i=1}^{m}\sum_{t=0}^{H-1}\left(V^\pi_\phi(s_t^{(i)}) - (\sum_{k=t}^{H-1}R(s_k^{(i)},u_k^{(i)}))\right)^2$$

Bootstrap Estimation of V^π

- Bellman Equation for V^π

$$V^\pi(s) = \sum_u \pi(u|s) \sum_{s'} P(s'|s,u)[R(s,u,s') + \gamma V^\pi(s')]$$

- Init $V^\pi_{\phi_0}$
  - Collect data {s, u, s', r}
  - Fitted V iteration:

$$\phi_{i+1} \leftarrow \min_\phi \sum_{(s,u,s',r)} \|r + V^\pi_{\phi_i}(s') - V_\phi(s)\|_2^2 + \lambda\|\phi - \phi_i\|_2^2$$

Vanilla Policy Gradient

Algorithm 1 "Vanilla" policy gradient algorithm

Initialize policy parameter θ, baseline b
for iteration$=1, 2, \ldots$ **do**
    Collect a set of trajectories by executing the current policy
    At each timestep in each trajectory, compute
    the *return* $R_t = \sum_{t'=t}^{T-1} \gamma^{t'-t} r_{t'}$, and
    the *advantage estimate* $\hat{A}_t = R_t - b(s_t)$.
    Re-fit the baseline, by minimizing $\|b(s_t) - R_t\|^2$,
    summed over all trajectories and timesteps.
    Update the policy, using a policy gradient estimate $\hat{g}$,
    which is a sum of terms $\nabla_\theta \log \pi(a_t \mid s_t, \theta)\hat{A}_t$
end for

~ [Williams, 1992]

Outline for Today's Lecture

- Super-quick Refresher: Markov Decision Processes (MDPs)
- Reinforcement Learning
- Policy Optimization
- Model-free Policy Optimization: Finite Differences
- Model-free Policy Optimization: Cross-Entropy Method
- Model-free Policy Optimization: Policy Gradients
  - Policy Gradient standard derivation
  - Temporal decomposition
  - Policy Gradient importance sampling derivation
  - Baseline subtraction & temporal structure
  - Value function estimation
  - ***Advantage Estimation (A2C/A3C/GAE)***
  - Trust Region Policy Optimization (TRPO)
  - Proximal Policy Optimization (PPO)

Recall Our Likelihood Ratio PG Estimator

$$\frac{1}{m}\sum_{i=1}^{m}\sum_{t=0}^{H-1}\nabla_\theta \log \pi_\theta(u_t^{(i)}|s_t^{(i)})\left(\sum_{k=t}^{H-1} R(s_k^{(i)}, u_k^{(i)}) - V^\pi(s_k^{(i)})\right)$$

Recall Our Likelihood Ratio PG Estimator

$$\frac{1}{m}\sum_{i=1}^{m}\sum_{t=0}^{H-1}\nabla_\theta \log \pi_\theta(u_t^{(i)}|s_t^{(i)})\left(\underbrace{\sum_{k=t}^{H-1} R(s_k^{(i)},u_k^{(i)})} - V^\pi(s_k^{(i)})\right)$$

Recall Our Likelihood Ratio PG Estimator

$$\frac{1}{m}\sum_{i=1}^{m}\sum_{t=0}^{H-1}\nabla_\theta \log \pi_\theta(u_t^{(i)}|s_t^{(i)})\left(\underbrace{\sum_{k=t}^{H-1} R(s_k^{(i)},u_k^{(i)})} - V^\pi(s_k^{(i)})\right)$$

- Estimation of Q from *single* roll-out

$$Q^\pi(s,u) = \mathbb{E}[r_0 + r_1 + r_2 + \cdots | s_0 = s, a_0 = a]$$

Recall Our Likelihood Ratio PG Estimator

$$\frac{1}{m}\sum_{i=1}^{m}\sum_{t=0}^{H-1}\nabla_\theta \log \pi_\theta(u_t^{(i)}|s_t^{(i)})\left(\underbrace{\sum_{k=t}^{H-1}R(s_k^{(i)},u_k^{(i)})} - V^\pi(s_k^{(i)})\right)$$

- Estimation of Q from *single* roll-out

$$Q^\pi(s,u) = \mathbb{E}[r_0 + r_1 + r_2 + \cdots | s_0 = s, a_0 = a]$$

- = high variance per sample based / no generalization

Further Refinements

$$\frac{1}{m}\sum_{i=1}^{m}\sum_{t=0}^{H-1}\nabla_\theta \log \pi_\theta(u_t^{(i)}|s_t^{(i)}) \left(\underbrace{\sum_{k=t}^{H-1} R(s_k^{(i)}, u_k^{(i)})} - V^\pi(s_k^{(i)}) \right)$$

- Estimation of Q from *single* roll-out

$$Q^\pi(s,u) = \mathbb{E}[r_0 + r_1 + r_2 + \cdots | s_0 = s, a_0 = a]$$

- = high variance per sample based / no generalization
  - Reduce variance by discounting

Recall Our Likelihood Ratio PG Estimator

$$\frac{1}{m}\sum_{i=1}^{m}\sum_{t=0}^{H-1}\nabla_\theta \log \pi_\theta(u_t^{(i)}|s_t^{(i)})\left(\underbrace{\sum_{k=t}^{H-1} R(s_k^{(i)},u_k^{(i)})} - V^\pi(s_k^{(i)})\right)$$

- Estimation of Q from *single* roll-out

$$Q^\pi(s,u) = \mathbb{E}[r_0 + r_1 + r_2 + \cdots | s_0 = s, a_0 = a]$$

- = high variance per sample based / no generalization
  - Reduce variance by discounting
  - Reduce variance by function approximation (=critic)

Variance Reduction by Discounting

$$Q^{\pi}(s, u) = \mathbb{E}[r_0 + r_1 + r_2 + \cdots | s_0 = s, a_0 = a]$$

→ introduce discount factor as a hyperparameter to improve estimate of Q:

$$Q^{\pi,\gamma}(s, u) = \mathbb{E}[r_0 + \gamma r_1 + \gamma^2 r_2 + \cdots | s_0 = s, a_0 = a]$$

Reducing Variance by Function Approximation

$$Q^{\pi,\gamma}(s,u) = \mathbb{E}[r_0 + \gamma r_1 + \gamma^2 r_2 + \cdots \mid s_0 = s, u_0 = u]$$

Reducing Variance by Function Approximation

$$
\begin{aligned}
Q^{\pi,\gamma}(s,u) &= \mathbb{E}[r_0 + \gamma r_1 + \gamma^2 r_2 + \cdots \mid s_0 = s, u_0 = u] \\
&= \mathbb{E}[r_0 + \gamma V^{\pi}(s_1) \mid s_0 = s, u_0 = u]
\end{aligned}
$$

Reducing Variance by Function Approximation

$$\begin{aligned} Q^{\pi,\gamma}(s,u) &= \mathbb{E}[r_0 + \gamma r_1 + \gamma^2 r_2 + \cdots \mid s_0 = s, u_0 = u] \\ &= \mathbb{E}[r_0 + \gamma V^{\pi}(s_1) \mid s_0 = s, u_0 = u] \\ &= \mathbb{E}[r_0 + \gamma r_1 + \gamma^2 V^{\pi}(s_2) \mid s_0 = s, u_0 = u] \end{aligned}$$

Reducing Variance by Function Approximation

$$\begin{aligned} Q^{\pi,\gamma}(s,u) &= \mathbb{E}[r_0 + \gamma r_1 + \gamma^2 r_2 + \cdots \mid s_0 = s, u_0 = u] \\ &= \mathbb{E}[r_0 + \gamma V^\pi(s_1) \mid s_0 = s, u_0 = u] \\ &= \mathbb{E}[r_0 + \gamma r_1 + \gamma^2 V^\pi(s_2) \mid s_0 = s, u_0 = u] \\ &= \mathbb{E}[r_0 + \gamma r_1 + + \gamma^2 r_2 + \gamma^3 V^\pi(s_3) \mid s_0 = s, u_0 = u] \\ &= \cdots \end{aligned}$$

- ***Async Advantage Actor Critic (A3C)*** *[Mnih et al, 2016]*
  - $\hat{Q}$ one of the above choices (e.g. k=5 step lookahead)

Reducing Variance by Function Approximation

$$\begin{aligned}
Q^{\pi,\gamma}(s,u) &= \mathbb{E}[r_0 + \gamma r_1 + \gamma^2 r_2 + \cdots \mid s_0 = s, u_0 = u] && \color{red}{(1-\lambda)} \\
&= \mathbb{E}[r_0 + \gamma V^\pi(s_1) \mid s_0 = s, u_0 = u] && \color{red}{(1-\lambda)\lambda} \\
&= \mathbb{E}[r_0 + \gamma r_1 + \gamma^2 V^\pi(s_2) \mid s_0 = s, u_0 = u] && \color{red}{(1-\lambda)\lambda^2} \\
&= \mathbb{E}[r_0 + \gamma r_1 + + \gamma^2 r_2 + \gamma^3 V^\pi(s_3) \mid s_0 = s, u_0 = u] \\
&&& \color{red}{(1-\lambda)\lambda^3} \\
&= \cdots
\end{aligned}$$

- ***Generalized Advantage Estimation (GAE)*** *[Schulman et al, ICLR 2016]*
  - $\hat{Q}$ = lambda exponentially weighted average of all the above

Reducing Variance by Function Approximation

$$\begin{aligned}
Q^{\pi,\gamma}(s,u) &= \mathbb{E}[r_0 + \gamma r_1 + \gamma^2 r_2 + \cdots \mid s_0 = s, u_0 = u] && (1-\lambda) \\
&= \mathbb{E}[r_0 + \gamma V^\pi(s_1) \mid s_0 = s, u_0 = u] && (1-\lambda)\lambda \\
&= \mathbb{E}[r_0 + \gamma r_1 + \gamma^2 V^\pi(s_2) \mid s_0 = s, u_0 = u] && (1-\lambda)\lambda^2 \\
&= \mathbb{E}[r_0 + \gamma r_1 + +\gamma^2 r_2 + \gamma^3 V^\pi(s_3) \mid s_0 = s, u_0 = u] \\
&= \cdots && (1-\lambda)\lambda^3
\end{aligned}$$

- ***Generalized Advantage Estimation (GAE)*** *[Schulman et al, ICLR 2016]*
  - $\hat{Q}$ = lambda exponentially weighted average of all the above
- ~ TD(lambda) / eligibility traces [Sutton and Barto, 1990]

Actor-Critic with A3C or GAE

- Policy Gradient + Generalized Advantage Estimation:
  - Init π_{θ_0} $V^{\pi}_{\phi_0}$
  - Collect roll-outs {s, u, s', r} and $\hat{Q}_i(s,u)$
  - Update: $$\phi_{i+1} \leftarrow \min_{\phi} \sum_{(s,u,s',r)} \|\hat{Q}_i(s,u) - V^{\pi}_{\phi}(s)\|_2^2 + \kappa \|\phi - \phi_i\|_2^2$$
    $$\theta_{i+1} \leftarrow \theta_i + \alpha \frac{1}{m} \sum_{k=1}^{m} \sum_{t=0}^{H-1} \nabla_\theta \log \pi_{\theta_i}(u_t^{(k)} | s_t^{(k)}) \left(\hat{Q}_i(s_t^{(k)}, u_t^{(k)}) - V^{\pi}_{\phi_i}(s_t^{(k)}) \right)$$

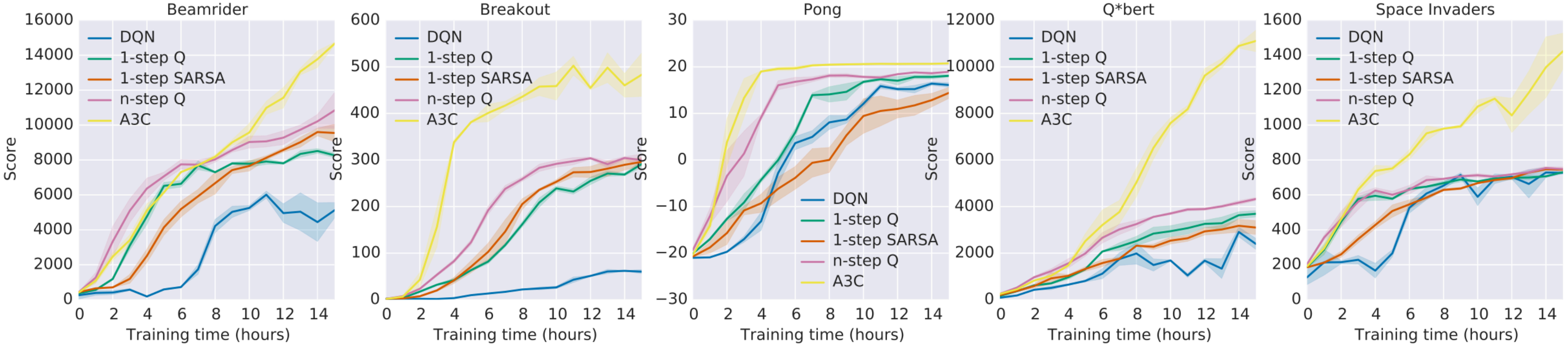
<u>Note</u>: many variations, e.g. could instead use 1-step for V, full roll-out for pi:

$$\phi_{i+1} \leftarrow \min_{\phi} \sum_{(s,u,s',r)} \|r + V^{\pi}_{\phi_i}(s') - V_{\phi}(s)\|_2^2 + \lambda \|\phi - \phi_i\|_2^2$$

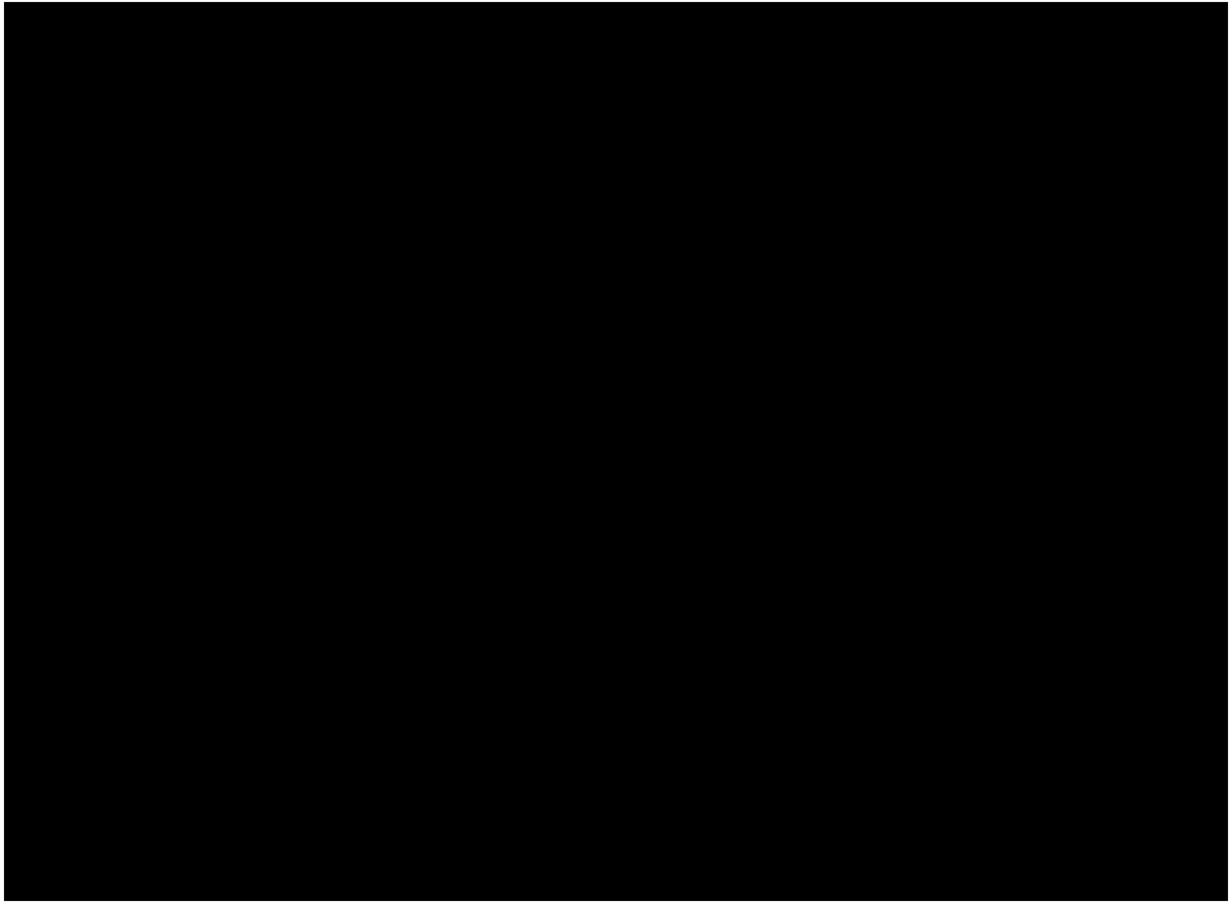
$$\theta_{i+1} \leftarrow \theta_i + \alpha \ \frac{1}{m} \sum_{k=1}^{m} \sum_{t=0}^{H-1} \nabla_\theta \log \pi_{\theta_i}(u_t^{(k)} | s_t^{(k)}) \left(\sum_{t'=t}^{H-1} r_{t'}^{(k)} - V^{\pi}_{\phi_i}(s_{t'}^{(k)}) \right)$$

Async Advantage Actor Critic (A3C)

- [Mnih et al, ICML 2016]
  - Likelihood Ratio Policy Gradient
  - n-step Advantage Estimation



A3C -- labyrinth

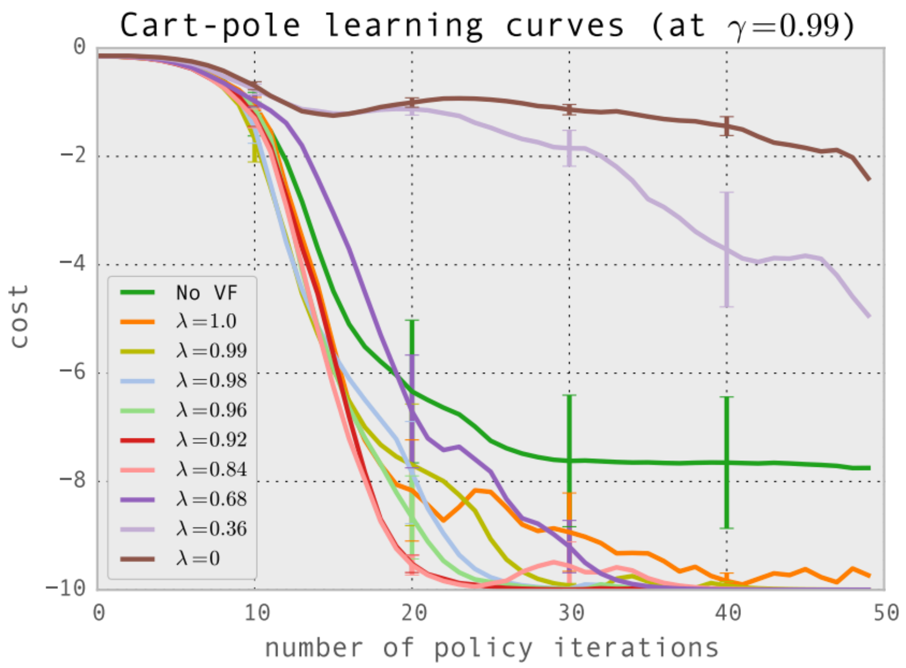


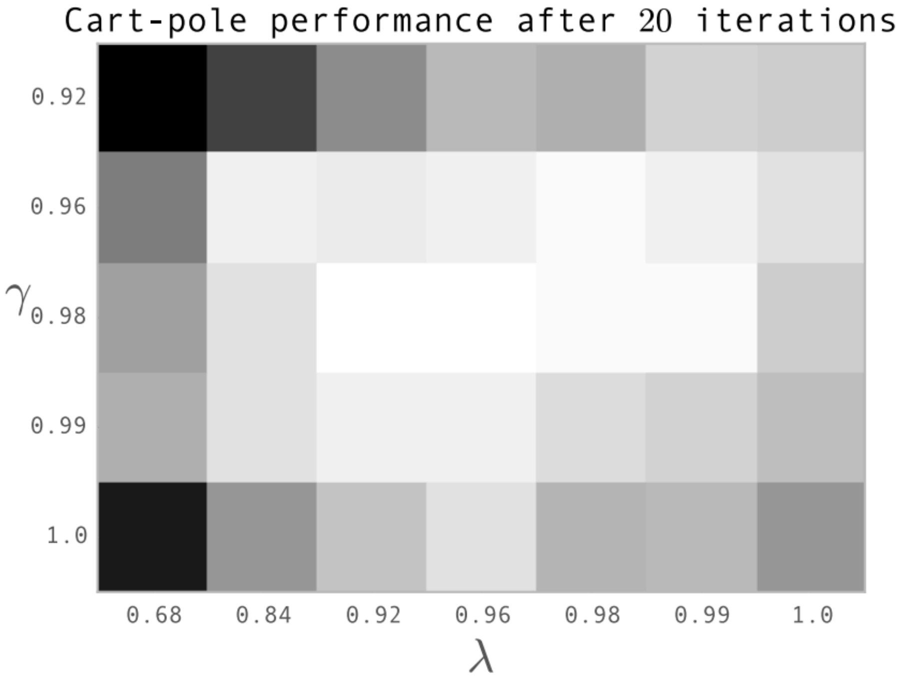
Example: Toddler Robot

[Tedrake, Zhang and Seung, 2005]

[Video: TODDLER – 40s]

GAE: Effect of gamma and lambda





[Schulman et al, 2016 -- GAE]

Outline for Today's Lecture

- Super-quick Refresher: Markov Decision Processes (MDPs)
- Reinforcement Learning
- Policy Optimization
- Model-free Policy Optimization: Finite Differences
- Model-free Policy Optimization: Cross-Entropy Method
- Model-free Policy Optimization: Policy Gradients
  - Policy Gradient standard derivation
  - Temporal decomposition
  - Policy Gradient importance sampling derivation
  - Baseline subtraction & temporal structure
  - Value function estimation
  - Advantage Estimation (A2C/A3C/GAE)
  - ***Trust Region Policy Optimization (TRPO)***
  - Proximal Policy Optimization (PPO)

Step-sizing and Trust Regions

- Step-sizing necessary as gradient is only first-order approximation

What’s in a step-size?

- Terrible step sizes, always an issue, but how about just not so great ones?

- Supervised learning
  - Step too far → next update will correct for it

- Reinforcement learning
  - Step too far → terrible policy
  - Next mini-batch: collected under this terrible policy!
  - Not clear how to recover short of going back and shrinking the step size

Step-sizing and Trust Regions

- Simple step-sizing: Line search in direction of gradient
  - Simple, but expensive (evaluations along the line)
  - Naïve: ignores where the first-order approximation is good/poor

Step-sizing and Trust Regions

- Advanced step-sizing: Trust regions
- First-order approximation from gradient is a good approximation within "trust region"

→ Solve for best point within trust region:

$$\max_{\delta\theta} \quad \hat{g}^\top \delta\theta$$

$$\text{s.t.} \quad KL(P(\tau;\theta)||P(\tau;\theta+\delta\theta)) \leq \varepsilon$$

Evaluating the KL

- **Our problem:**

$$\max_{\delta\theta} \ \hat{g}^\top \delta\theta$$
$$\text{s.t.} \ \ KL(P(\tau;\theta)||P(\tau;\theta+\delta\theta)) \leq \varepsilon$$

- **Recall:**

$$P(\tau;\theta) = P(s_0)\prod_{t=0}^{H-1}\pi_\theta(u_t|s_t)P(s_{t+1}|s_t,u_t)$$

Evaluating the KL

- Our problem:

$$\max_{\delta\theta} \ \hat{g}^\top \delta\theta$$
$$\text{s.t.} \ \ KL(P(\tau;\theta)||P(\tau;\theta+\delta\theta)) \leq \varepsilon$$

- Recall:

$$P(\tau;\theta) = P(s_0)\prod_{t=0}^{H-1}\pi_\theta(u_t|s_t)P(s_{t+1}|s_t,u_t)$$

- Hence:

$$KL(P(\tau;\theta)||P(\tau;\theta+\delta\theta)) = \sum_\tau P(\tau;\theta)\log\frac{P(\tau;\theta)}{P(\tau;\theta+\delta\theta)}$$

Evaluating the KL

- Our problem:

$$\max_{\delta\theta} \quad \hat{g}^\top \delta\theta$$
$$\text{s.t.} \quad KL(P(\tau;\theta)||P(\tau;\theta+\delta\theta)) \leq \varepsilon$$

- Recall:

$$P(\tau;\theta) = P(s_0)\prod_{t=0}^{H-1}\pi_\theta(u_t|s_t)P(s_{t+1}|s_t,u_t)$$

- Hence:

$$KL(P(\tau;\theta)||P(\tau;\theta+\delta\theta)) = \sum_\tau P(\tau;\theta)\log\frac{P(\tau;\theta)}{P(\tau;\theta+\delta\theta)}$$
$$= \sum_\tau P(\tau;\theta)\log\frac{P(s_0)\prod_{t=0}^{H-1}\pi_\theta(u_t|s_t)P(s_{t+1}|s_t,u_t)}{P(s_0)\prod_{t=0}^{H-1}\pi_{\theta+\delta\theta}(u_t|s_t)P(s_{t+1}|s_t,u_t)}$$

Evaluating the KL

- Our problem:

$$\max_{\delta\theta} \; \hat{g}^\top \delta\theta$$
$$\text{s.t.} \;\; KL(P(\tau;\theta)||P(\tau;\theta+\delta\theta)) \leq \varepsilon$$

- Recall:

$$P(\tau;\theta) = P(s_0)\prod_{t=0}^{H-1}\pi_\theta(u_t|s_t)P(s_{t+1}|s_t,u_t)$$

- Hence:

$$KL(P(\tau;\theta)||P(\tau;\theta+\delta\theta)) = \sum_\tau P(\tau;\theta)\log\frac{P(\tau;\theta)}{P(\tau;\theta+\delta\theta)}$$
$$= \sum_\tau P(\tau;\theta)\log\frac{P(s_0)\prod_{t=0}^{H-1}\pi_\theta(u_t|s_t)P(s_{t+1}|s_t,u_t)}{P(s_0)\prod_{t=0}^{H-1}\pi_{\theta+\delta\theta}(u_t|s_t)P(s_{t+1}|s_t,u_t)}$$

dynamics cancels out! ☺

$$= \sum_\tau P(\tau;\theta)\log\frac{\prod_{t=0}^{H-1}\pi_\theta(u_t|s_t)}{\prod_{t=0}^{H-1}\pi_{\theta+\delta\theta}(u_t|s_t)}$$

Evaluating the KL

- Our problem:

$$\max_{\delta\theta} \ \hat{g}^\top \delta\theta$$
$$\text{s.t.} \ \ KL(P(\tau;\theta)||P(\tau;\theta+\delta\theta)) \leq \varepsilon$$

- Recall:

$$P(\tau;\theta) = P(s_0)\prod_{t=0}^{H-1} \pi_\theta(u_t|s_t)P(s_{t+1}|s_t,u_t)$$

- Hence:

$$KL(P(\tau;\theta)||P(\tau;\theta+\delta\theta)) = \sum_\tau P(\tau;\theta)\log\frac{P(\tau;\theta)}{P(\tau;\theta+\delta\theta)}$$
$$= \sum_\tau P(\tau;\theta)\log\frac{P(s_0)\prod_{t=0}^{H-1}\pi_\theta(u_t|s_t)P(s_{t+1}|s_t,u_t)}{P(s_0)\prod_{t=0}^{H-1}\pi_{\theta+\delta\theta}(u_t|s_t)P(s_{t+1}|s_t,u_t)}$$

dynamics cancels out! ☺

$$= \sum_\tau P(\tau;\theta)\log\frac{\prod_{t=0}^{H-1}\pi_\theta(u_t|s_t)}{\prod_{t=0}^{H-1}\pi_{\theta+\delta\theta}(u_t|s_t)}$$
$$\approx \frac{1}{M}\sum_{\text{s,u in roll-outs under }\theta}\log\frac{\pi_\theta(u|s)}{\pi_{\theta+\delta\theta}(u|s)}$$

Evaluating the KL

- Our problem:

$$\max_{\delta\theta} \quad \hat{g}^\top \delta\theta$$
$$\text{s.t.} \quad KL(P(\tau;\theta)||P(\tau;\theta+\delta\theta)) \leq \varepsilon$$

- Has become:

$$\max_{\delta\theta} \quad \hat{g}^\top \delta\theta$$
$$\text{s.t.} \quad \frac{1}{M} \sum_{\text{(s,u)}\sim\theta} \log \frac{\pi_\theta(u|s)}{\pi_{\theta+\delta\theta}(u|s)} \leq \varepsilon$$

Evaluating the KL

- Our problem:

$$\max_{\delta\theta} \quad \hat{g}^\top \delta\theta$$
$$\text{s.t.} \quad KL(P(\tau;\theta)||P(\tau;\theta+\delta\theta)) \leq \varepsilon$$

- Has become:

$$\max_{\delta\theta} \quad \hat{g}^\top \delta\theta$$
$$\text{s.t.} \quad \frac{1}{M}\sum_{(\mathrm{s,u})\sim\theta} \log \frac{\pi_\theta(u|s)}{\pi_{\theta+\delta\theta}(u|s)} \leq \varepsilon$$

- How to enforce this constraint given complex policies like neural nets
  - 2nd approximation of KL Divergence
    - (1) First order approximation is constant
    - (2) Hessian is Fisher Information Matrix

$$[\mathcal{I}(\theta)]_{i,j} = \mathrm{E}\left[\left(\frac{\partial}{\partial\theta_i}\log f(X;\theta)\right)\left(\frac{\partial}{\partial\theta_j}\log f(X;\theta)\right)\middle|\theta\right].$$

Evaluating the KL

- Our problem:

$$\max_{\delta\theta} \quad \hat{g}^\top \delta\theta$$
$$\text{s.t.} \quad KL(P(\tau;\theta)||P(\tau;\theta+\delta\theta)) \le \varepsilon$$

- Has become:

$$\max_{\delta\theta} \quad \hat{g}^\top \delta\theta$$
$$\text{s.t.} \quad \frac{1}{M} \sum_{\text{(s,u)}\sim\theta} \log \frac{\pi_\theta(u|s)}{\pi_{\theta+\delta\theta}(u|s)} \le \varepsilon$$

- 2nd order approximation to KL:

$$KL(\pi_\theta(u|s)||\pi_{\theta+\delta\theta}(u|s) \approx \delta\theta^\top \left(\sum_{(s,u)\sim\theta} \nabla_\theta \log \pi_\theta(u|s) \nabla_\theta \log \pi_\theta(u|s)^\top \right) \delta\theta$$
$$= \delta\theta^\top F_\theta \delta\theta$$

Evaluating the KL

- Our problem:

$$\max_{\delta\theta} \quad \hat{g}^\top \delta\theta$$
$$\text{s.t.} \quad KL(P(\tau;\theta)||P(\tau;\theta+\delta\theta)) \leq \varepsilon$$

- Has become:

$$\max_{\delta\theta} \quad \hat{g}^\top \delta\theta$$
$$\text{s.t.} \quad \frac{1}{M}\sum_{(\mathrm{s,u})\sim\theta} \log\frac{\pi_\theta(u|s)}{\pi_{\theta+\delta\theta}(u|s)} \leq \varepsilon$$

- 2nd order approximation to KL:

$$KL(\pi_\theta(u|s)||\pi_{\theta+\delta\theta}(u|s) \approx \delta\theta^\top \left(\sum_{(s,u)\sim\theta} \nabla_\theta \log\pi_\theta(u|s)\nabla_\theta\log\pi_\theta(u|s)^\top\right)\delta\theta$$
$$= \delta\theta^\top F_\theta \delta\theta$$

→ Fisher matrix F_θ easily computed from gradient calculations

Evaluating the KL

- Our problem:
$$\max_{\delta\theta} \quad \hat{g}^\top \delta\theta$$
$$\text{s.t.} \quad \delta\theta^\top F_\theta \delta\theta \leq \varepsilon$$

Evaluating the KL

- Our problem:

$$\max_{\delta\theta} \quad \hat{g}^\top \delta\theta$$
$$\text{s.t.} \quad \delta\theta^\top F_\theta \delta\theta \leq \varepsilon$$

- Done?

Evaluating the KL

- Our problem:

$$\max_{\delta\theta} \quad \hat{g}^\top \delta\theta$$

$$\text{s.t.} \quad \delta\theta^\top F_\theta \delta\theta \leq \varepsilon$$

- Done?
  - Deep RL → θ high-dimensional, and building / inverting F_θ impractical

Evaluating the KL

- Our problem:

$$\max_{\delta\theta} \quad \hat{g}^\top \delta\theta$$
$$\text{s.t.} \quad \delta\theta^\top F_\theta \delta\theta \leq \varepsilon$$

- Done?
  - Deep RL → θ high-dimensional, and building / inverting F_θ impractical
    - Efficient scheme through conjugate gradient [Schulman et al, 2015, TRPO]

Evaluating the KL

- Our problem:

$$\max_{\delta\theta} \quad \hat{g}^\top \delta\theta$$
$$\text{s.t.} \quad \delta\theta^\top F_\theta \delta\theta \leq \varepsilon$$

- Done?
  - Deep RL → θ high-dimensional, and building / inverting F_θ impractical
    - Efficient scheme through conjugate gradient [Schulman et al, 2015, TRPO]
  - Can we do even better?

Evaluating the KL

- Our problem:

$$\max_{\delta\theta} \quad \hat{g}^\top \delta\theta$$
$$\text{s.t.} \quad \delta\theta^\top F_\theta \delta\theta \leq \varepsilon$$

- Done?
  - Deep RL → θ high-dimensional, and building / inverting F_θ impractical
    - Efficient scheme through conjugate gradient [Schulman et al, 2015, TRPO]
  - Can we do even better?
    - Replace objective by surrogate loss that's higher order approximation yet equally efficient to evaluate [Schulman et al, 2015, TRPO]

Evaluating the KL

- Our problem:

$$\max_{\delta\theta} \quad \hat{g}^\top \delta\theta$$
$$\text{s.t.} \quad \delta\theta^\top F_\theta \delta\theta \leq \varepsilon$$

- Done?
  - Deep RL → θ high-dimensional, and building / inverting F_θ impractical
    - Efficient scheme through conjugate gradient [Schulman et al, 2015, TRPO]
  - Can we do even better?
    - Replace objective by surrogate loss that's higher order approximation yet equally efficient to evaluate [Schulman et al, 2015, TRPO]
    - Note: the surrogate loss idea is generally applicable when likelihood ratio gradients are used

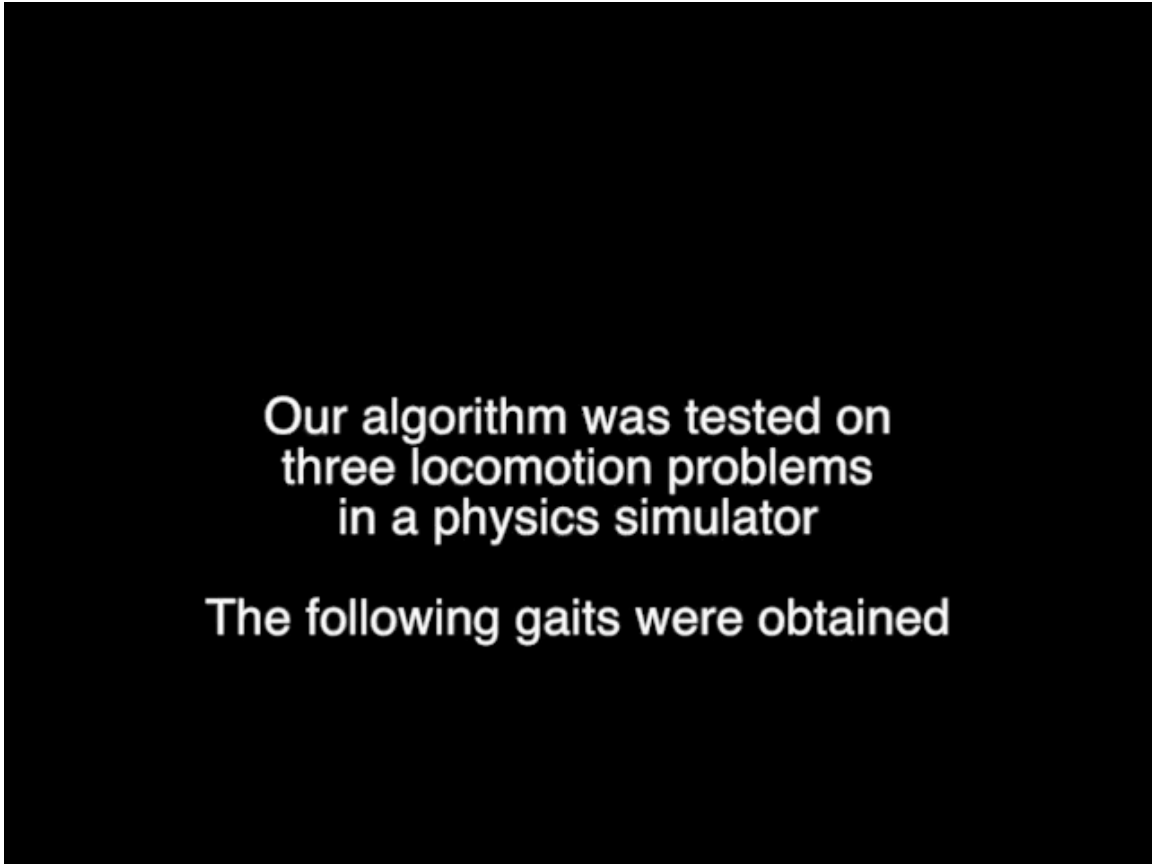
TRPO

Surrogate loss: $$\max_{\pi} L(\pi) = \mathbb{E}_{\pi_{\text{old}}}\left[\frac{\pi(a|s)}{\pi_{\text{old}}(a|s)} A^{\pi_{\text{old}}}(s,a)\right]$$

Constraint: $$\mathbb{E}_{\pi_{\text{old}}}\left[KL(\pi||\pi_{\text{old}})\right] \leq \epsilon$$

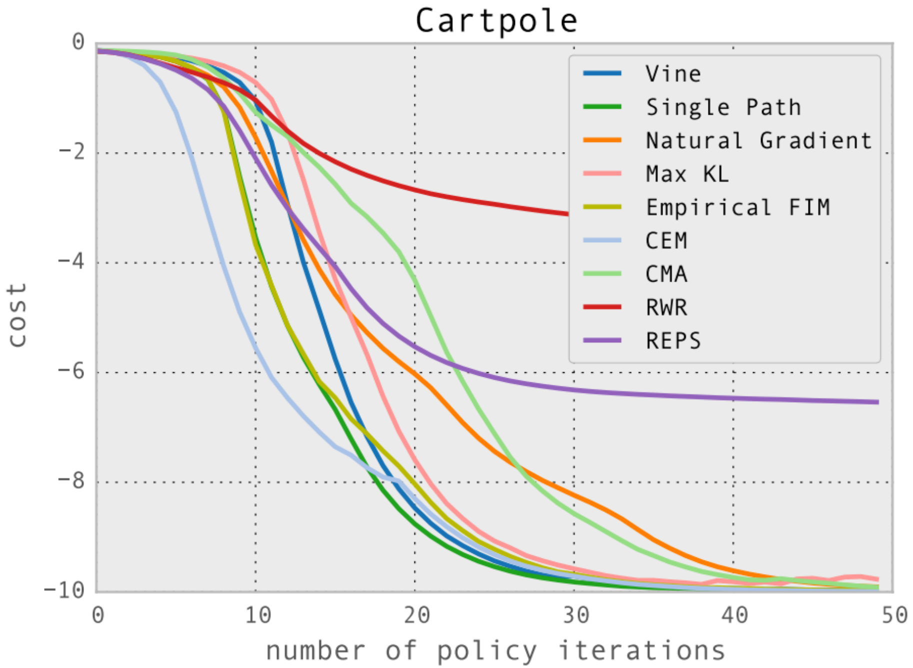
for iteration$=1, 2, \ldots$ **do**
    Run policy for T timesteps or N trajectories
    Estimate advantage function at all timesteps
    Compute policy gradient g
    Use CG (with Hessian-vector products) to compute $F^{-1}g$
    Do line search on surrogate loss and KL constraint
end for

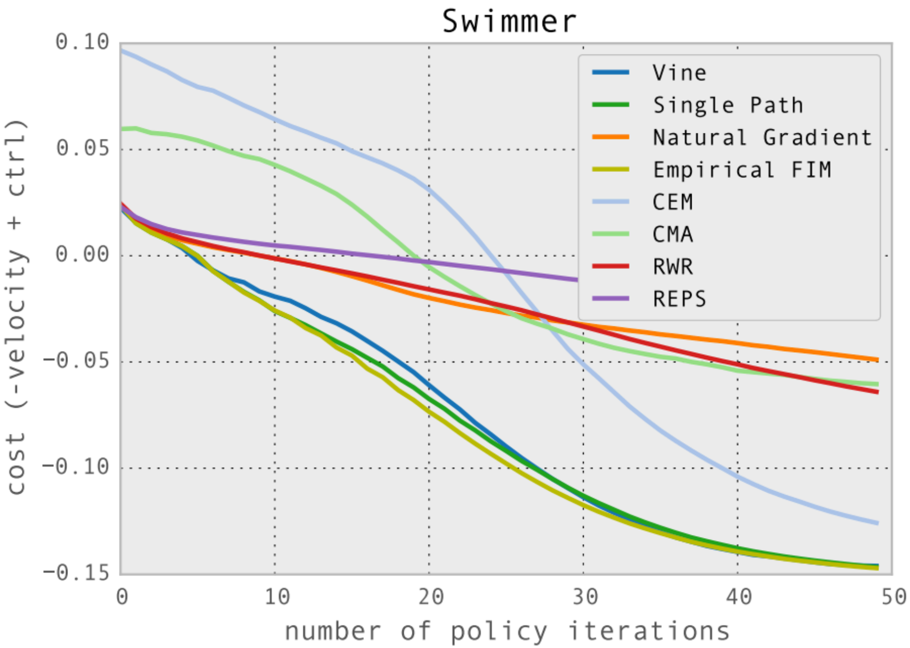
Experiments in Locomotion



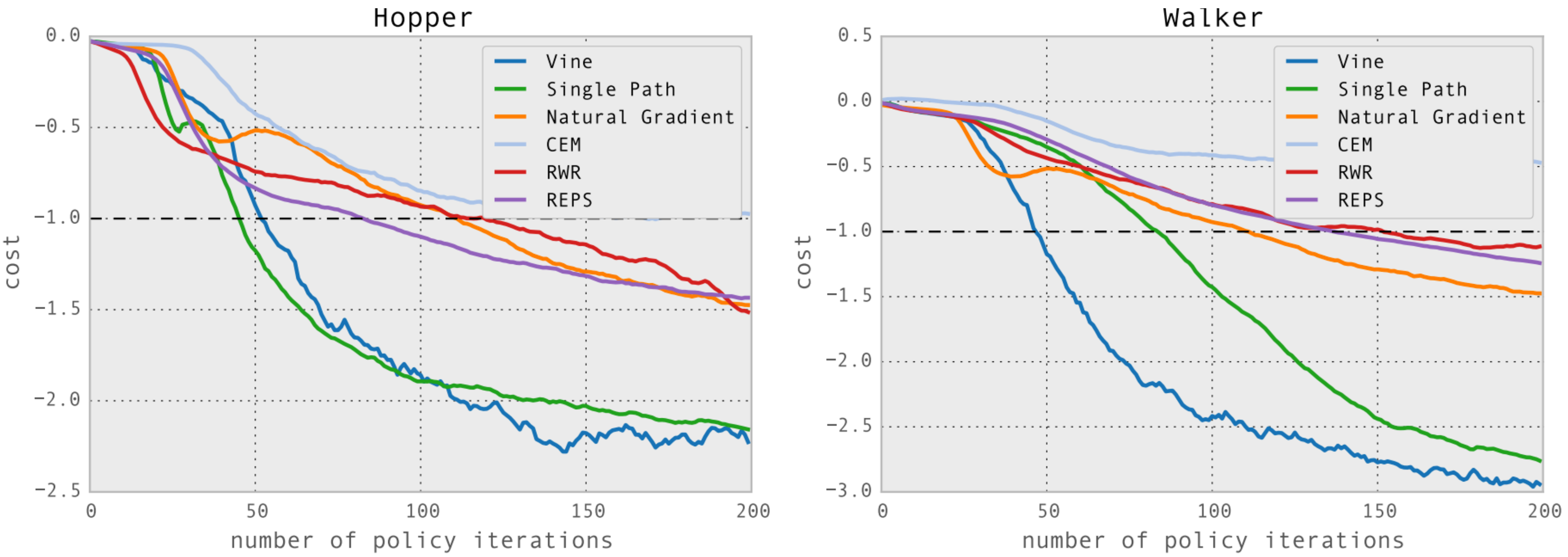
[Schulman, Levine, Moritz, Jordan, Abbeel, 2014]

Learning Curves -- Comparison

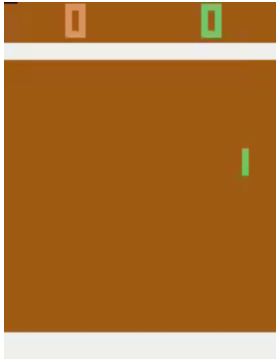




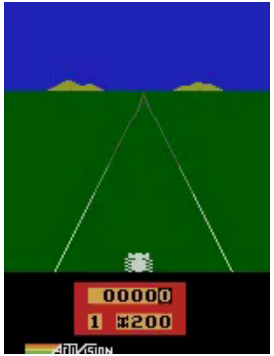
Learning Curves -- Comparison



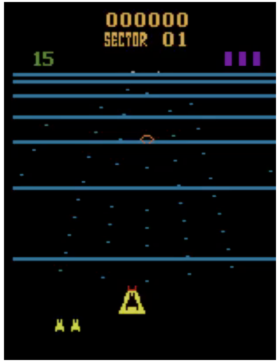
Atari Games



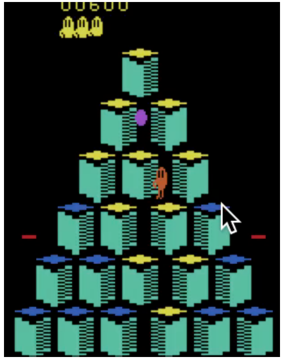
Pong



Enduro



Beamrider



Q*bert

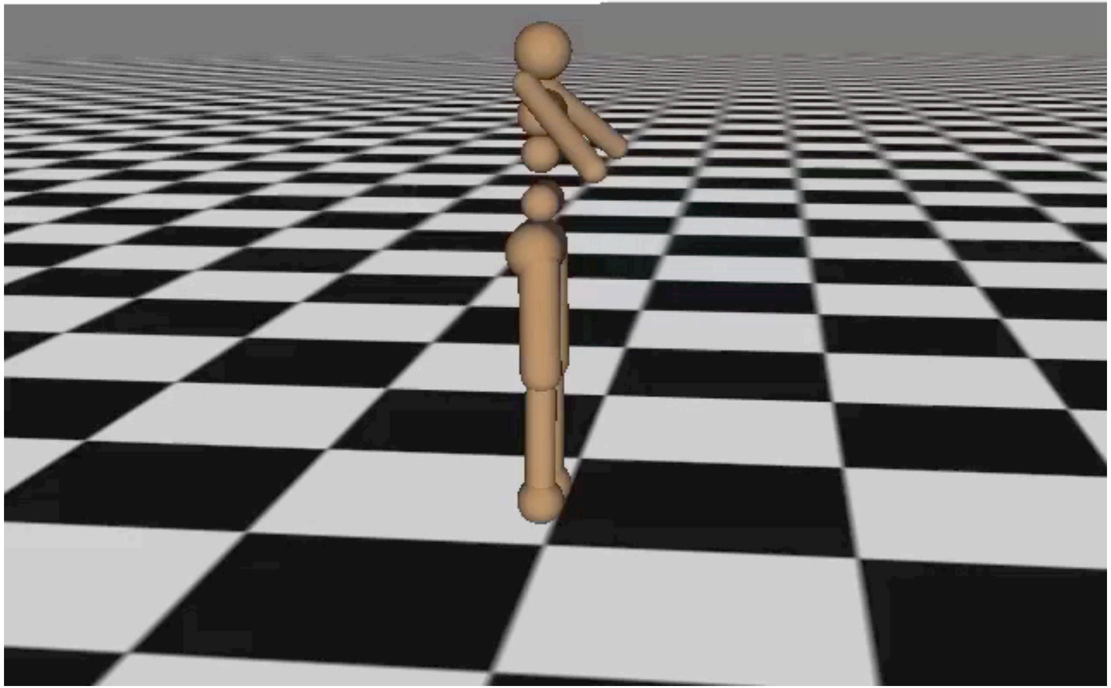
- Deep Q-Network (DQN) [Mnih et al, 2013/2015]
- Dagger with Monte Carlo Tree Search [Xiao-Xiao et al, 2014]
- Trust Region Policy Optimization [Schulman, Levine, Moritz, Jordan, Abbeel, 2015]
- ...

Natural Gradients Work

Task	Random	REINFORCE	TNPG
Cart-Pole Balancing	77.1 ± 0.0	4693.7 ± 14.0	**3986.4** $\pm$ **748.9**
Inverted Pendulum*	-153.4 ± 0.2	13.4 ± 18.0	**209.7** $\pm$ **55.5**
Mountain Car	-415.4 ± 0.0	-67.1 ± 1.0	**-66.5** $\pm$ **4.5**
Acrobot	-1904.5 ± 1.0	-508.1 ± 91.0	-395.8 ± 121.2
Double Inverted Pendulum*	149.7 ± 0.1	4116.5 ± 65.2	**4455.4** $\pm$ **37.6**
Swimmer*	-1.7 ± 0.1	92.3 ± 0.1	**96.0** $\pm$ **0.2**
Hopper	8.4 ± 0.0	714.0 ± 29.3	**1155.1** $\pm$ **57.9**
2D Walker	-1.7 ± 0.0	506.5 ± 78.8	**1382.6** $\pm$ **108.2**
Half-Cheetah	-90.8 ± 0.3	1183.1 ± 69.2	**1729.5** $\pm$ **184.6**
Ant*	13.4 ± 0.7	548.3 ± 55.5	**706.0** $\pm$ **127.7**
Simple Humanoid	41.5 ± 0.2	128.1 ± 34.0	**255.0** $\pm$ **24.5**
Full Humanoid	13.2 ± 0.1	262.2 ± 10.5	**288.4** $\pm$ **25.2**
Cart-Pole Balancing (LS)*	77.1 ± 0.0	420.9 ± 265.5	**945.1** $\pm$ **27.8**
Inverted Pendulum (LS)	-122.1 ± 0.1	-13.4 ± 3.2	**0.7** $\pm$ **6.1**
Mountain Car (LS)	-83.0 ± 0.0	-81.2 ± 0.6	**-65.7** $\pm$ 9.0
Acrobot (LS)*	-393.2 ± 0.0	-128.9 ± 11.6	**-84.6** $\pm$ **2.9**

Learning Locomotion (TRPO + GAE)

Iteration 0



[Schulman, Moritz, Levine, Jordan, Abbeel, 2016]

Outline for Today’s Lecture

- Super-quick Refresher: Markov Decision Processes (MDPs)
- Reinforcement Learning
- Policy Optimization
- Model-free Policy Optimization: Finite Differences
- Model-free Policy Optimization: Cross-Entropy Method
- Model-free Policy Optimization: Policy Gradients
  - Policy Gradient standard derivation
  - Temporal decomposition
  - Policy Gradient importance sampling derivation
  - Baseline subtraction & temporal structure
  - Value function estimation
  - Advantage Estimation (A2C/A3C/GAE)
  - Trust Region Policy Optimization (TRPO)
  - ***Proximal Policy Optimization (PPO)***

A better TRPO?

- Not easy to enforce trust region constraint for complex policy architectures
  - Networks that have stochasticity like dropout
  - Parameter sharing between policy and value function
- Conjugate Gradient implementation is complex
- Would be good to harness good first-order optimizers like Adam, RMSProp…

Proximal Policy Optimization V1 – “Dual Descent TRPO”

TRPO

$$\underset{\theta}{\text{maximize}} \quad \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\hat{A}_t\right]$$

$$\text{subject to} \quad \hat{\mathbb{E}}_t[\text{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)]] \leq \delta.$$

PPO v1

$$\max_\theta \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}\hat{A}_t\right] - \beta\left(\hat{\mathbb{E}}_t\left[\text{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)]\right] - \delta\right)$$

- Pseudocode:

  for iteration$=1, 2, \ldots$ **do**

  Run policy for T timesteps or N trajectories

  Estimate advantage function at all timesteps

  Do SGD on above objective for some number of epochs

  Do dual descent update for beta

Can we simplify further?

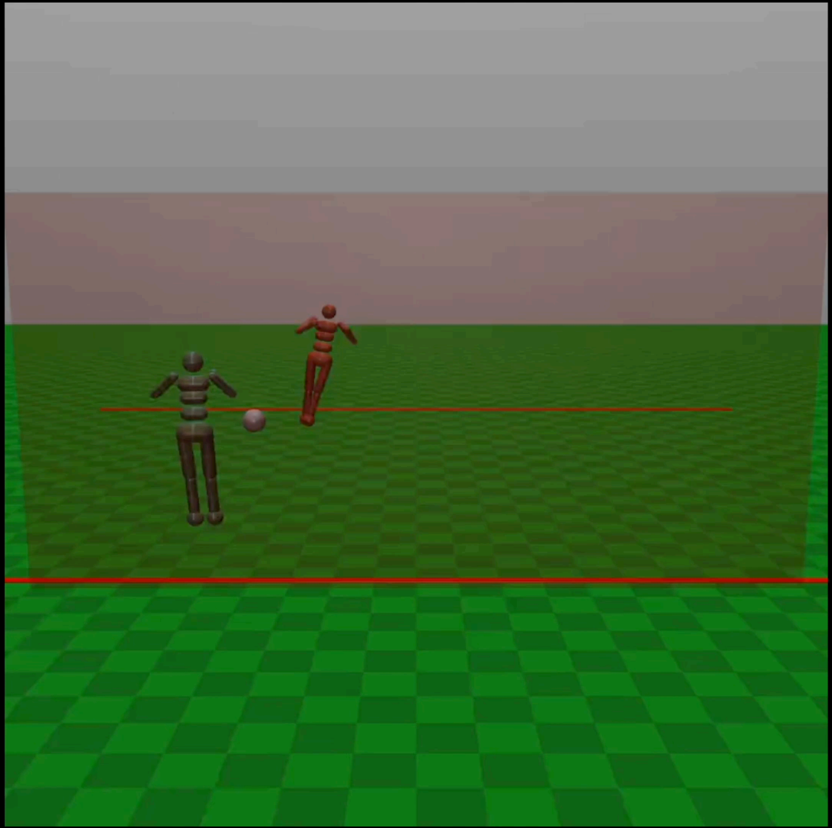
Proximal Policy Optimization V2 – "Clipped Surrogate Loss"

Let: $r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}$, so $r(\theta_{\text{old}}) = 1$

Optimize:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\Big[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)\hat{A}_t)\Big]$$

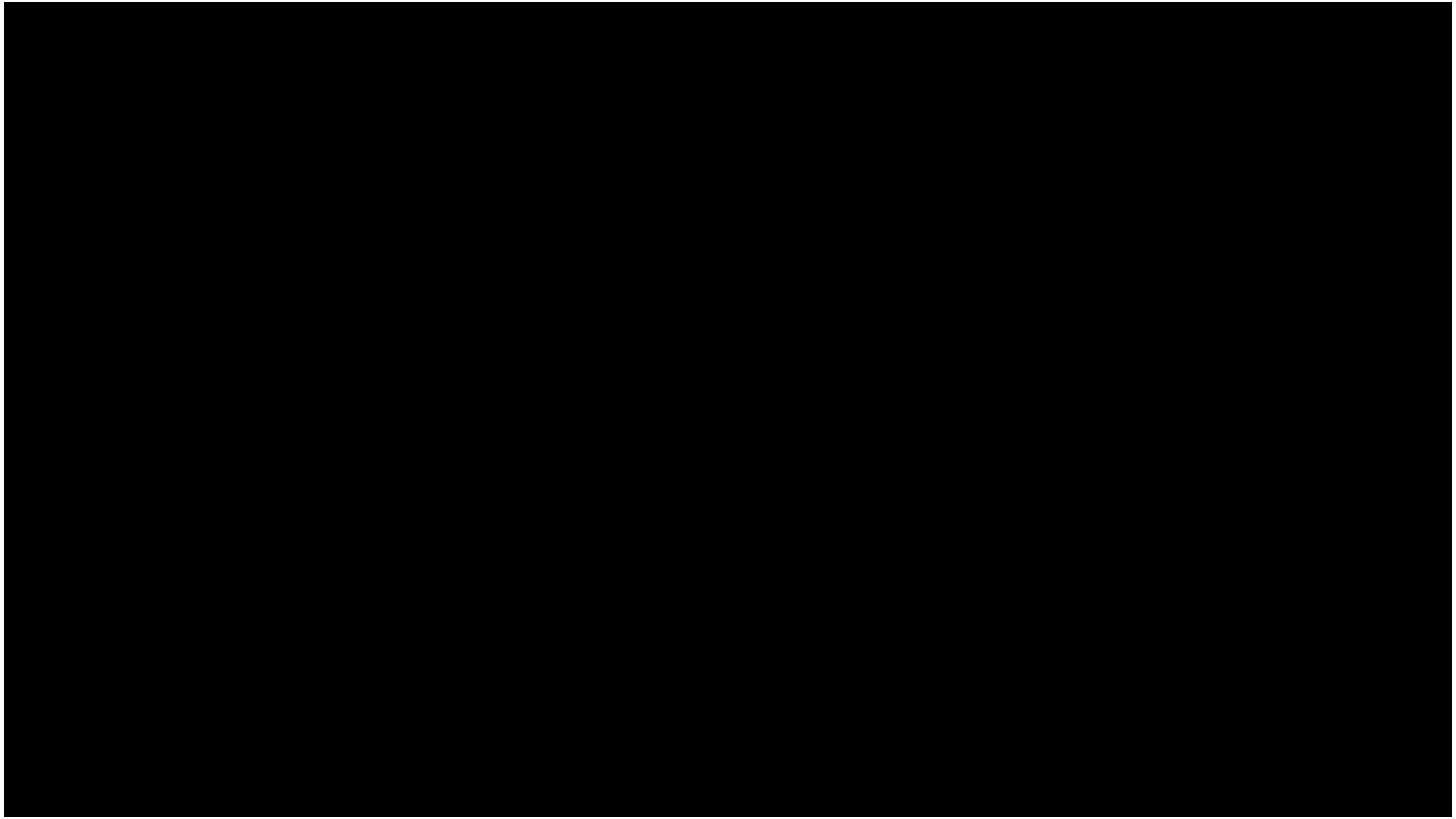
RL: Learning Soccer



[Bansal et al, 2017]

OpenAI-5 was trained with PPO

OpenAI In-Hand Re-Orientation



OpenAI Rubik's Cube