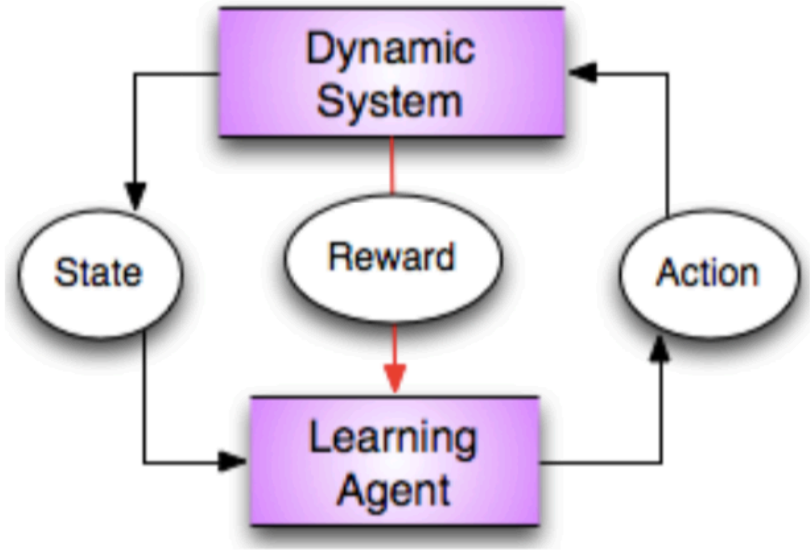
Deep Reinforcement Learning through Policy Optimization

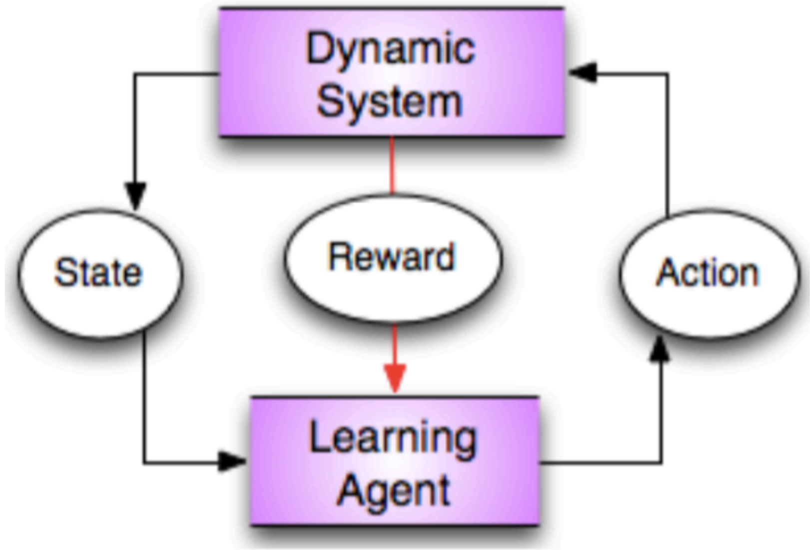
Pieter Abbeel
Open AI / Berkeley AI Research Lab

Slides made in collaboration with John Schulman

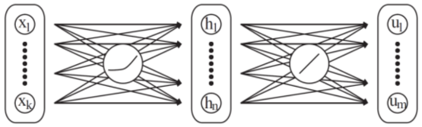
Reinforcement Learning



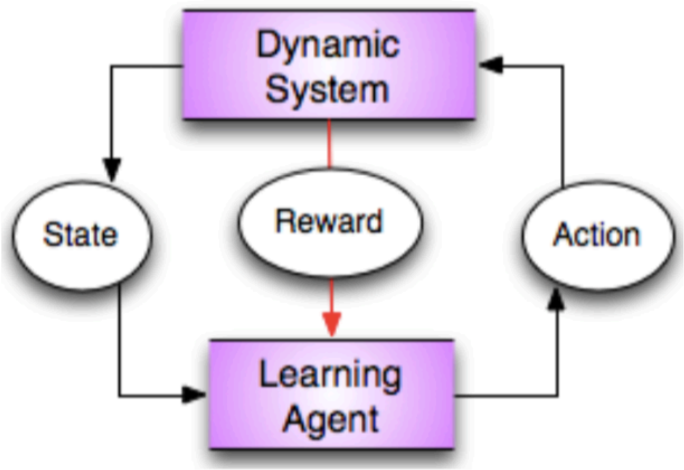
Policy Optimization



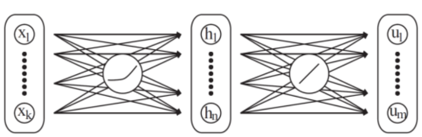
$$\pi_\theta(u|s)$$



Policy Optimization



$$\pi_\theta(u|s)$$



- Consider control policy parameterized by parameter vector θ

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

- Often 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 $\arg\max_a Q_\theta(s,a)$
  - Challenge for continuous / high-dimensional action spaces

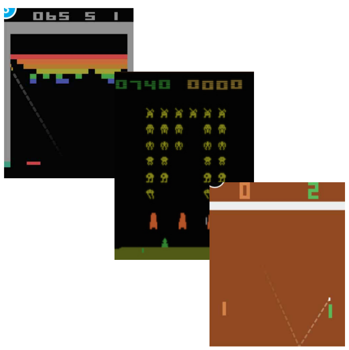
Example Policy Optimization Success Stories

Kohl and Stone, 2004

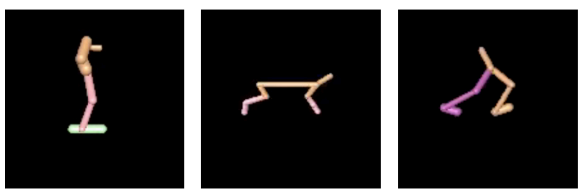
Ng et al, 2004

Tedrake et al, 2005

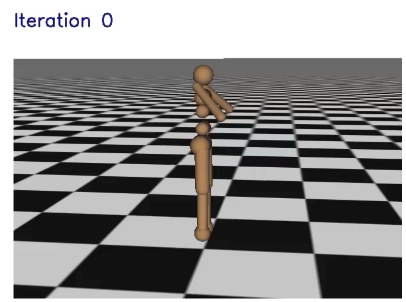
Kober and Peters, 2009



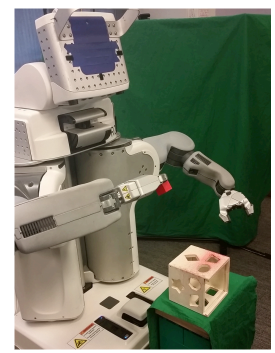
Schulman et al, 2015 (TRPO)
Mnih et al, 2015 (A3C)



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

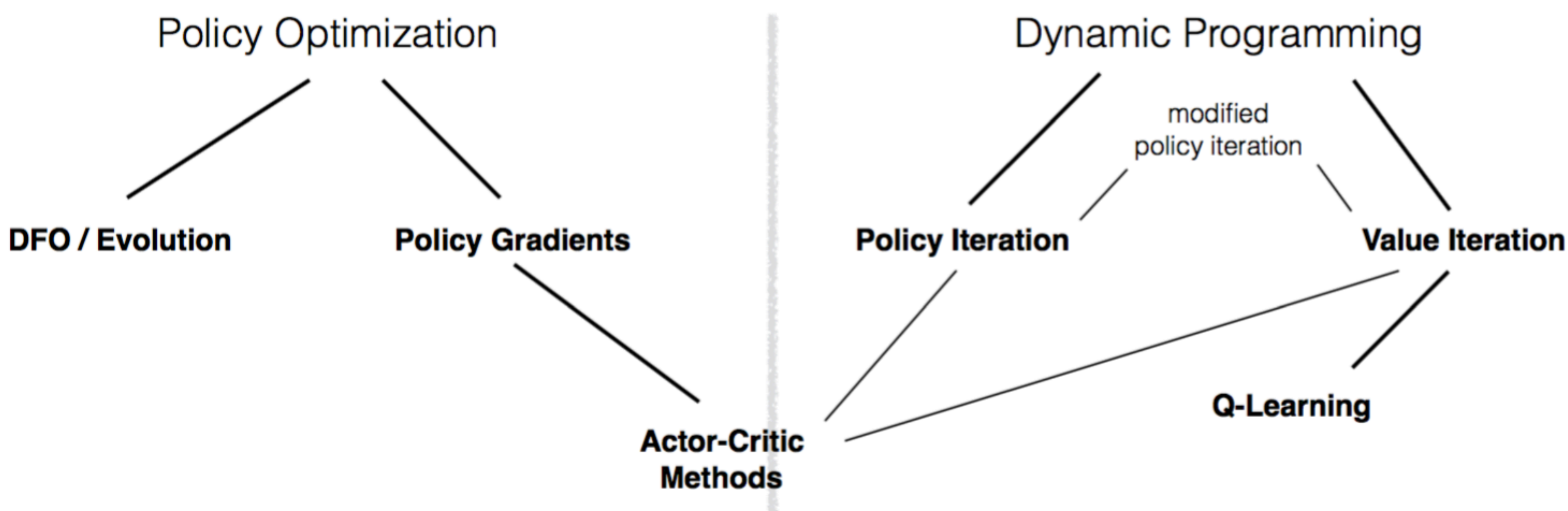


Schulman et al, 2016 (TRPO + GAE)



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

Policy Optimization in the RL Landscape



Outline

- Derivative free methods
  - Cross Entropy Method (CEM) / Finite Differences / Fixing Random Seed
- Likelihood Ratio (LR) Policy Gradient
  - Derivation / Connection w/Importance Sampling
- Natural Gradient / Trust Regions (-> TRPO)
- Actor-Critic (-> GAE, A3C)
- Path Derivatives (PD) (-> DPG, DDPG, SVG)
- Stochastic Computation Graphs (generalizes LR / PD)
- Guided Policy Search (GPS)
- Inverse Reinforcement Learning

Outline

- ***Derivative free methods***
  - ***Cross Entropy Method (CEM) / Finite Differences / Fixing Random Seed***
- Likelihood Ratio (LR) Policy Gradient
  - Derivation / Connection w/Importance Sampling
- Natural Gradient / Trust Regions (-> TRPO)
- Actor-Critic (-> GAE, A3C)
- Path Derivatives (PD) (-> DPG, DDPG, SVG)
- Stochastic Computation Graphs (generalizes LR / PD)
- Guided Policy Search (GPS)
- Inverse Reinforcement Learning

Cross-Entropy Method

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

- Views *U* as a black box
- Ignores all other information other than *U* collecting during episode

= evolutionary algorithm

population: $P_{\mu^{(i)}}(\theta)$

CEM:

for iter i = 1, 2, …

for population member e = 1, 2, …

sample $\theta^{(e)} \sim P_{\mu^{(i)}}(\theta)$

execute roll-outs under $\pi_{\theta^{(e)}}$

store $(\theta^{(e)}, u^{(e)})$

endfor

$\mu^{(i+1)} = \arg\max_\mu \sum_{\bar{e}} \log P_\mu(\theta^{(\bar{e})})$

where $\bar{e}$ indexes over top p %

endfor

Cross-Entropy Method

- Works embarrassingly well

Method	Mean Score	Reference
Nonreinforcement learning		
Hand-coded	631,167	Dellacherie (Fahey, 2003)
Genetic algorithm	586,103	(Böhm et al., 2004)
Reinforcement learning		
Relational reinforcement learning+kernel-based regression	≈50	Ramon and Driessens (2004)
Policy iteration	3183	Bertsekas and Tsitsiklis (1996)
Least squares policy iteration	<3000	Lagoudakis, Parr, and Littman (2002)
Linear programming + Bootstrap	4274	Farias and van Roy (2006)
Natural policy gradient	≈6800	Kakade (2001)
CE+RL	21,252	
CE+RL, constant noise	72,705	
CE+RL, decreasing noise	348,895	

István Szita and András Lörincz. "Learning Tetris using the noisy cross-entropy method". In: *Neural computation* 18.12 (2006), pp. 2936–2941

Approximate Dynamic Programming Finally Performs Well in the Game of Tetris

[NIPS 2013]

Victor Gabillon
INRIA Lille - Nord Europe,
Team SequeL, FRANCE
victor.gabillon@inria.fr

Mohammad Ghavamzadeh*
INRIA Lille - Team SequeL
& Adobe Research
mohammad.ghavamzadeh@inria.fr

Bruno Scherrer
INRIA Nancy - Grand Est,
Team Maia, FRANCE
bruno.scherrer@inria.fr

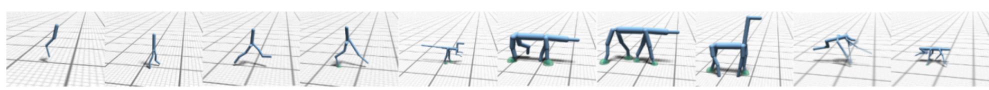
Cross-Entropy Method

- Covariance Matrix Adaptation (CMA) has become standard in graphics [Hansen, Ostermeier, 1996]

Optimal Gait and Form for Animal Locomotion

Kevin Wampler* Zoran Popović

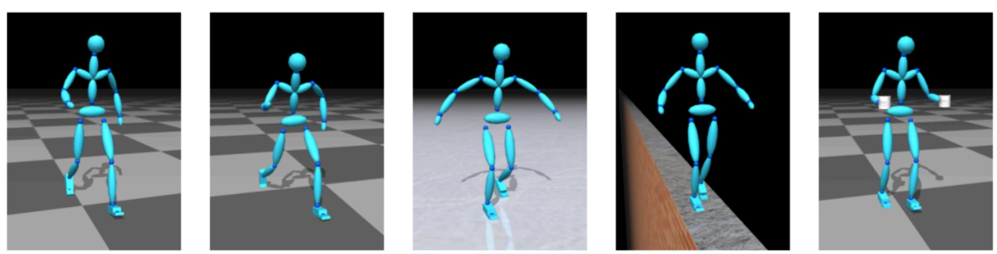
University of Washington



Optimizing Walking Controllers for Uncertain Inputs and Environments

Jack M. Wang David J. Fleet Aaron Hertzmann

University of Toronto



Cross-Entropy Method

- Caveat: best when number of parameters is relatively small
  - i.e., number of population members comparable to or larger than number of parameters

→ in practice OK if low-dimensional θ and willing to do do many runs

→ Easy to implement baseline to compare with!

Black Box Gradient Computation

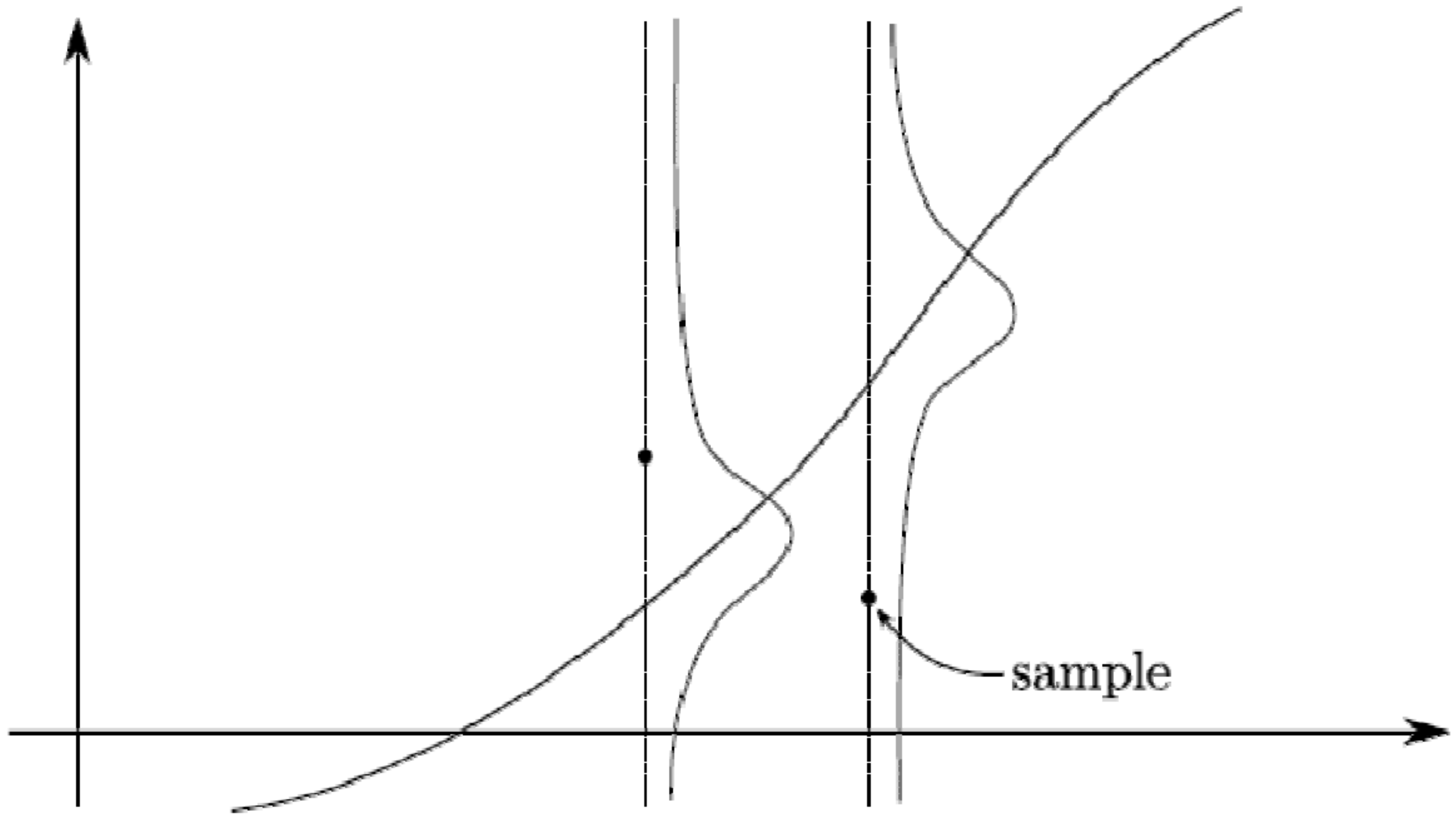
We can compute the gradient g using standard finite difference methods, as follows:

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

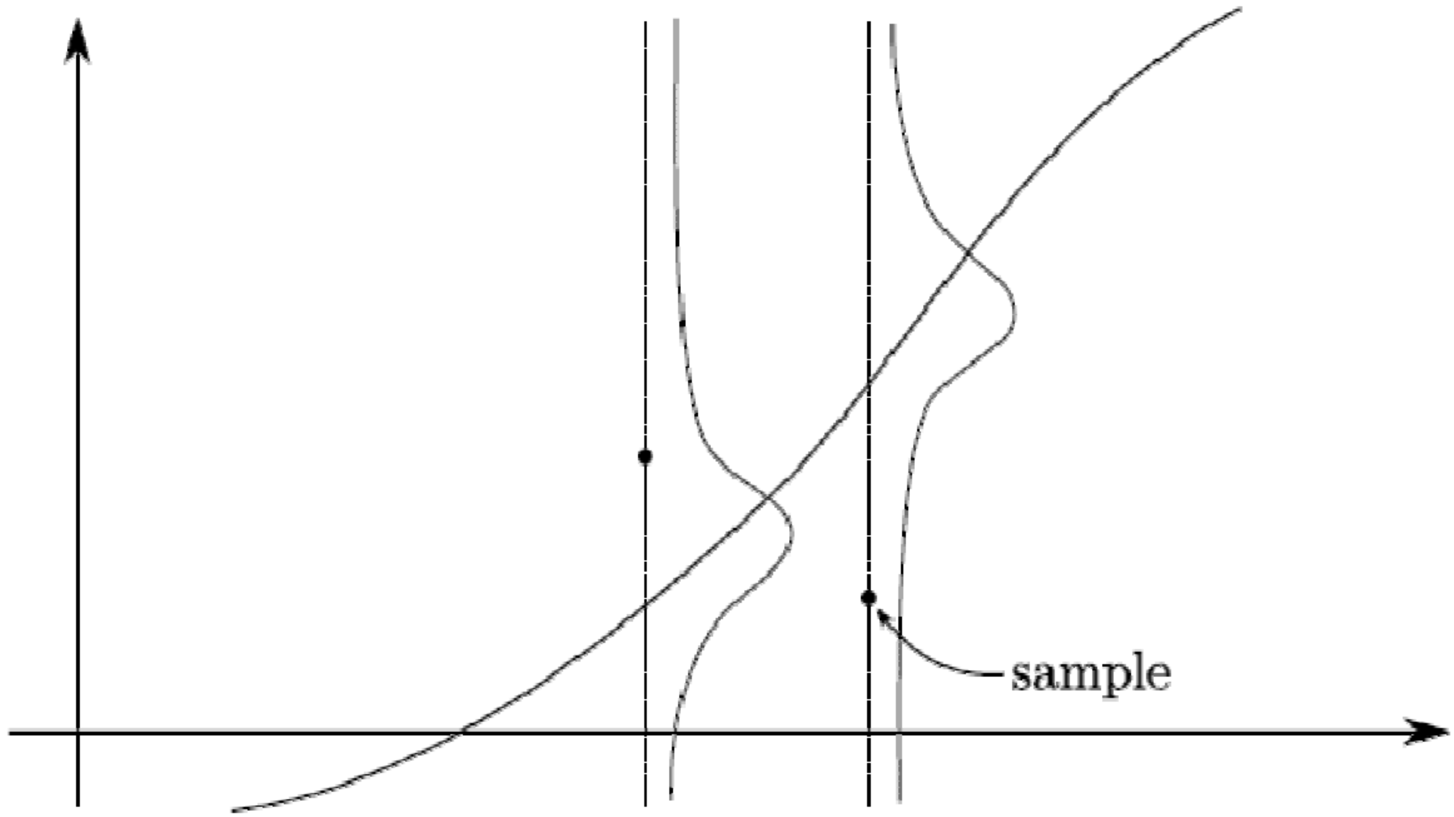
Where:

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

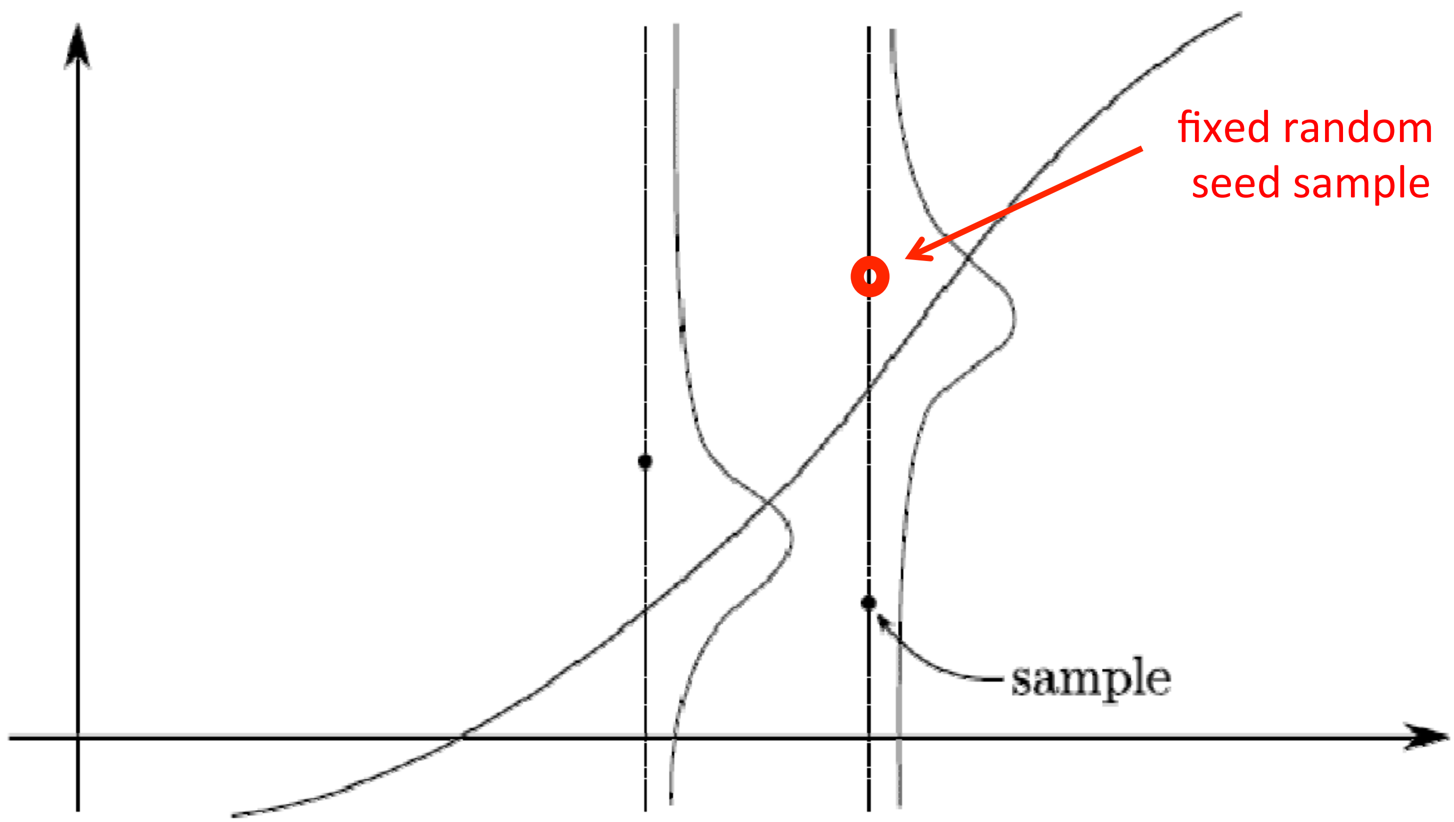
Challenge: Noise Can Dominate



Solution 1: Average over many samples



Solution 2: Fix random seed

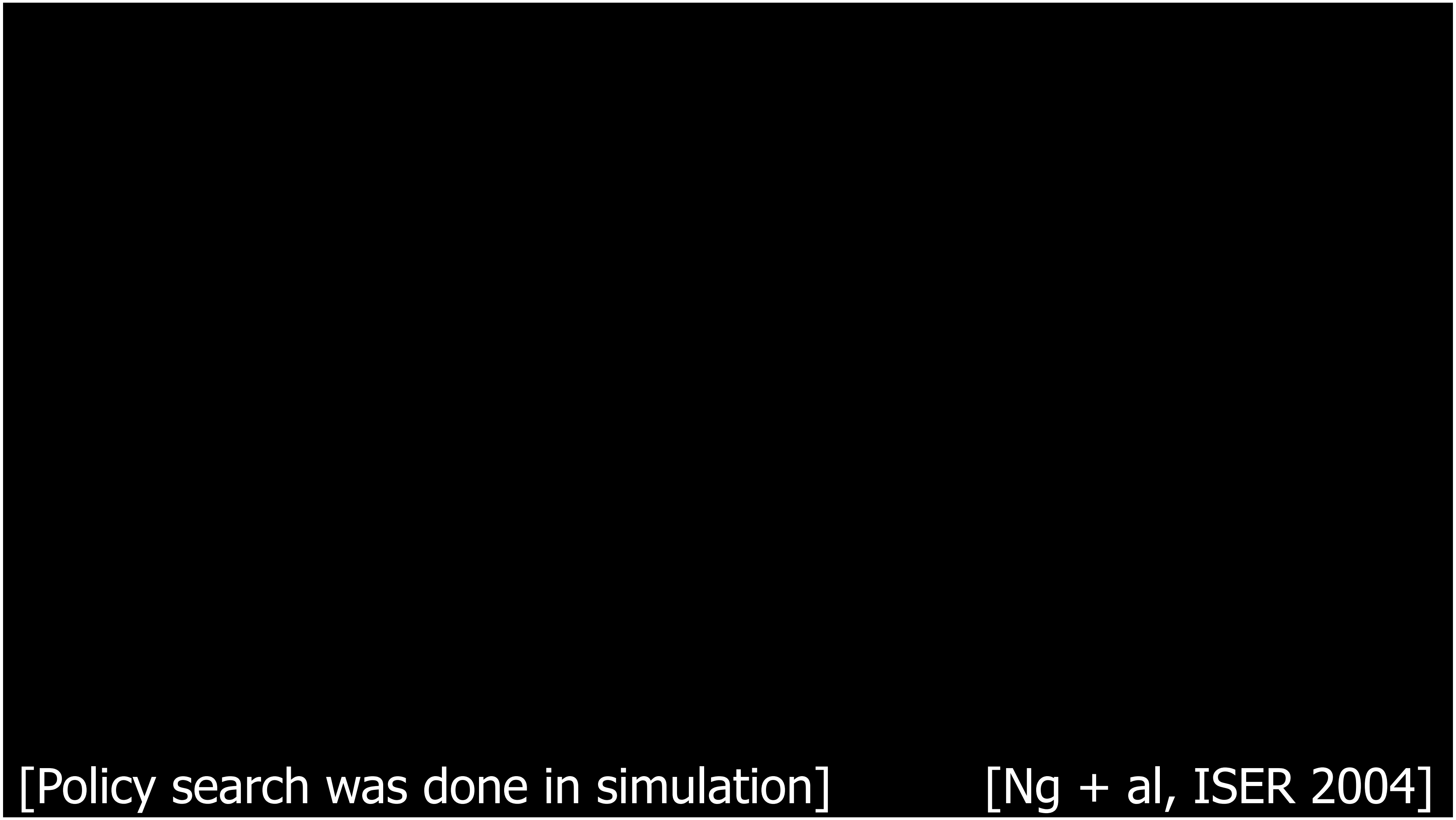


Solution 2: Fix random seed

- 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

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


[Policy search was done in simulation]
[Ng + al, ISER 2004]

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).

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

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

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

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

Outline

- Derivative free methods
  - Cross Entropy Method (CEM) / Finite Differences / Fixing Random Seed
- ***Likelihood Ratio (LR) Policy Gradient***
  - ***Derivation / Connection w/Importance Sampling***
- Natural Gradient / Trust Regions (-> TRPO)
- Actor-Critic (-> GAE, A3C)
- Path Derivatives (PD) (-> DPG, DDPG, SVG)
- Stochastic Computation Graphs (generalizes LR / PD)
- Guided Policy Search (GPS)
- Inverse Reinforcement Learning

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

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

Likelihood Ratio Policy Gradient

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

Taking the gradient w.r.t. θ gives

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

Likelihood Ratio Policy Gradient

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

Taking the gradient w.r.t. θ gives

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

Likelihood Ratio Policy Gradient

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

Taking the gradient w.r.t. θ gives

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

Likelihood Ratio Policy Gradient

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

Taking the gradient w.r.t. θ gives

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

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:

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

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, 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, 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, 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, 2011]

Derivation from Importance Sampling

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

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

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

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

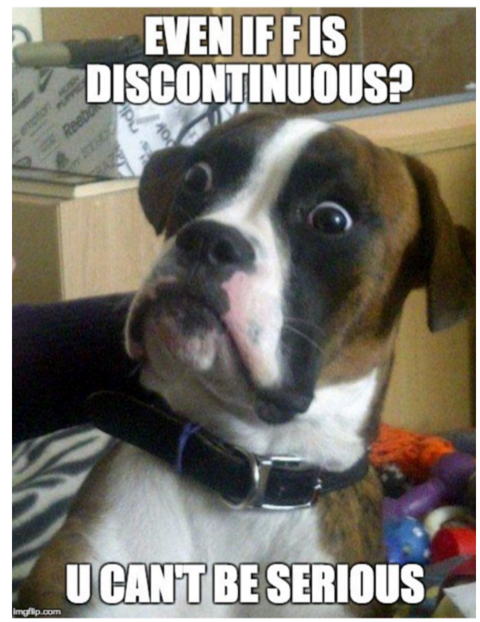
Note: Suggests we can also look at more than just gradient!

[Tang&Abbeel, 2011]

Likelihood Ratio Gradient: Validity

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

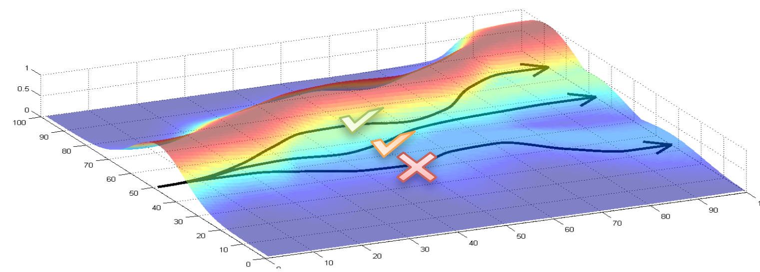
- Valid even if R is discontinuous, and unknown, or sample space (of paths) is a discrete set



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 (see Path Derivative later)

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

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

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

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

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

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:

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

Likelihood Ratio Gradient Estimate

- As formulated thus far: unbiased but very noisy
- Fixes that lead to real-world practicality
  - Baseline
  - Temporal structure

  - Also: KL-divergence trust region / natural gradient (= general trick, equally applicable to perturbation analysis and finite differences)

Likelihood Ratio Gradient: Baseline

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

- To build intuition, let's assume R > 0
  - Then tries to increase probabilities of all paths

→ 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

Good choice for b?

$U(\theta)$ [in practice estimate]

Likelihood Ratio and Temporal Structure

- Current estimate:

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

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

- Future actions do not depend on past rewards, hence can lower variance by instead using:

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

Outline

- Derivative free methods
  - Cross Entropy Method (CEM) / Finite Differences / Fixing Random Seed
- Likelihood Ratio (LR) Policy Gradient
  - Derivation / Connection w/Importance Sampling
- ***Natural Gradient / Trust Regions (-> TRPO)***
- Actor-Critic (-> GAE, A3C)
- Path Derivatives (PD) (-> DPG, DDPG, SVG)
- Stochastic Computation Graphs (generalizes LR / PD)
- Guided Policy Search (GPS)
- Inverse Reinforcement Learning

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:

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

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

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:

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

- 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:

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

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! ☺

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

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:

$$\begin{aligned} KL(P(\tau;\theta)||P(\tau;\theta+\delta\theta)) &= \sum_\tau P(\tau;\theta)\log\frac{P(\tau;\theta)}{P(\tau;\theta+\delta\theta)} \\ &= \sum_\tau P(\tau;\theta)\log\frac{P(s_0)\prod_{t=0}^{H-1}\pi_\theta(u_t|s_t)P(s_{t+1}|s_t,u_t)}{P(s_0)\prod_{t=0}^{H-1}\pi_{\theta+\delta\theta}(u_t|s_t)P(s_{t+1}|s_t,u_t)} \\ &= \sum_\tau P(\tau;\theta)\log\frac{\prod_{t=0}^{H-1}\pi_\theta(u_t|s_t)}{\prod_{t=0}^{H-1}\pi_{\theta+\delta\theta}(u_t|s_t)} \\ &\approx \frac{1}{M}\sum_{(s,u)\text{ in roll-outs under }\theta}\pi_\theta(u|s)\log\frac{\pi_\theta(u|s)}{\pi_{\theta+\delta\theta}(u|s)} \\ &= \frac{1}{M}\sum_{(s,u)\sim\theta} KL(\pi_\theta(u|s)||\pi_{\theta+\delta\theta}(u|s)) \end{aligned}$$

dynamics cancels out! ☺

Evaluating the KL

- Our problem:

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

- Has become:

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

Evaluating the KL

- Our problem:

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

- Has become:

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

- 2nd order approximation to KL:

Evaluating the KL

- Our problem:

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

- Has become:

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

- 2[nd] order approximation to KL:

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

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:

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

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

- If constraint moved to objective → natural policy gradient
  - [Kakade 2002, Bagnell & Schneider 2003, Peters & Schaal 2003]

- But keeping as constraint tends to be beneficial [Schulman et al 2015]
  - Can be done through dual gradient descent on Lagrangian

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]

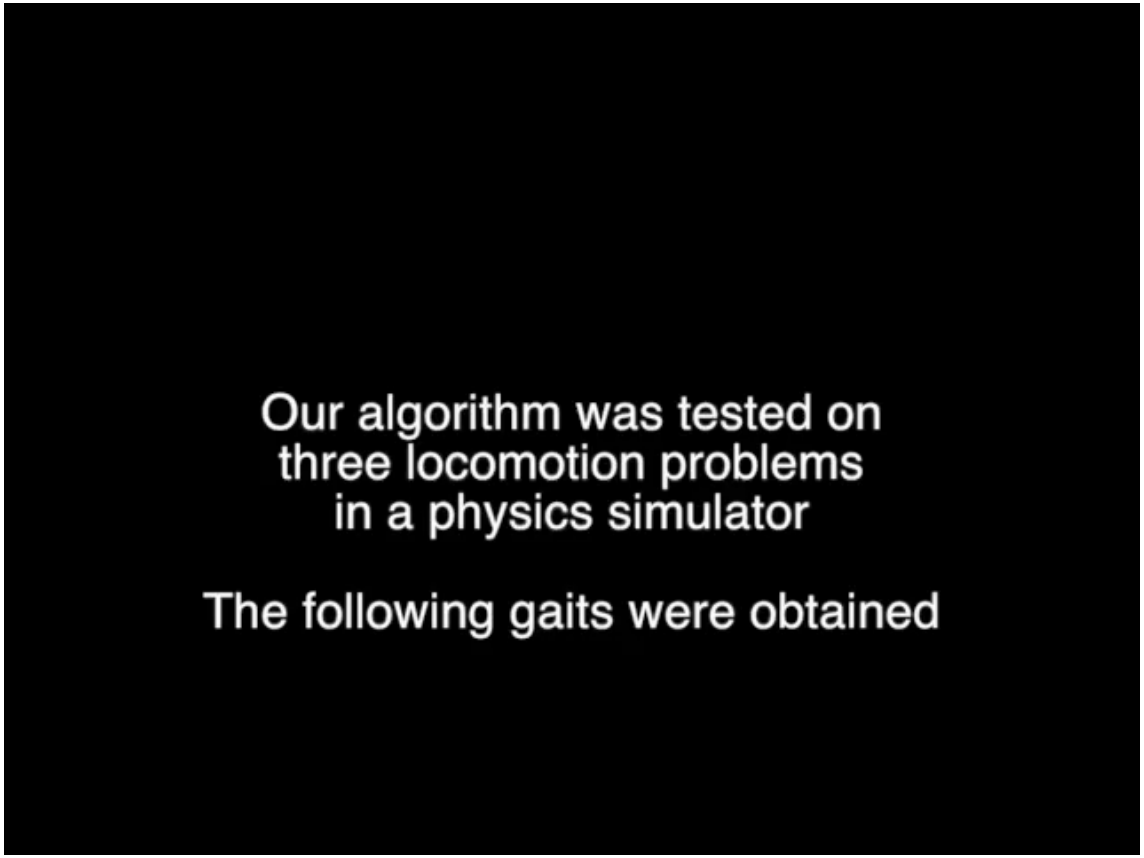
Evaluating the KL

- Our problem:

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

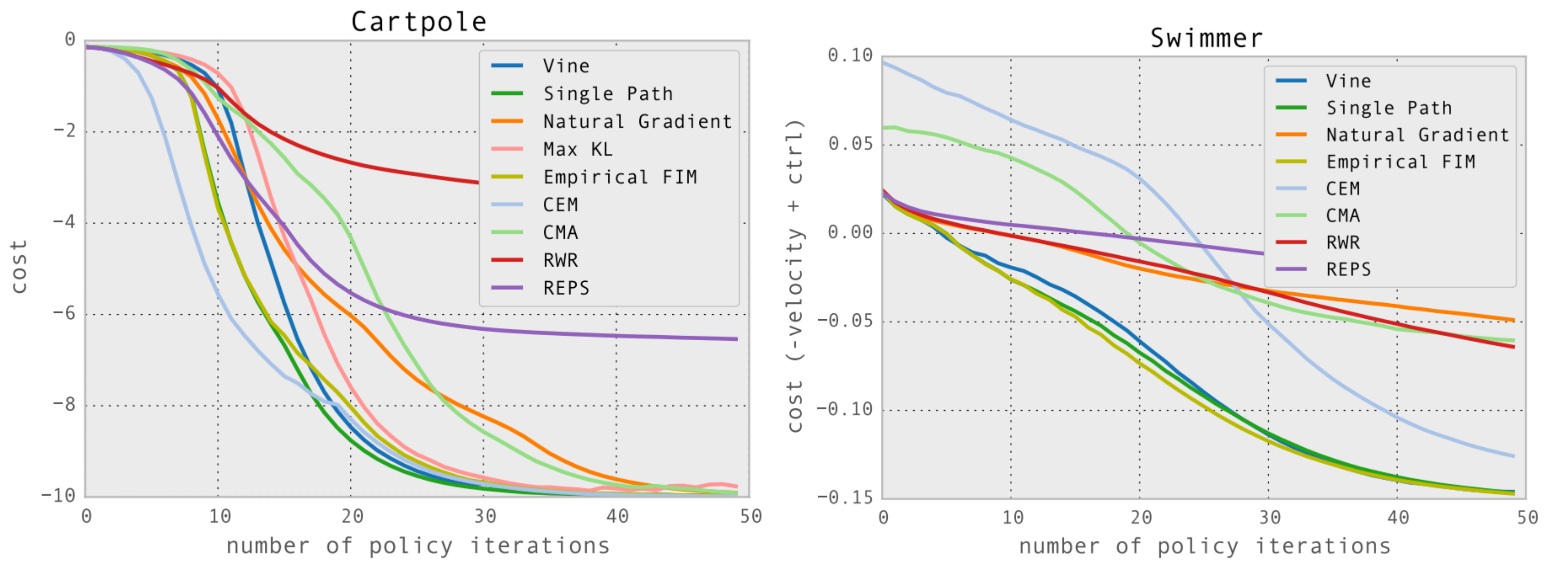
- Done?
  - Deep RL → θ high-dimensional, and building / inverting F_θ impractical
    - Efficient scheme through conjugate gradient [Schulman et al, 2015, TRPO]
  - Can we do even better?
    - Replace objective by surrogate loss that's higher order approximation yet equally efficient to evaluate [Schulman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

Experiments in Locomotion

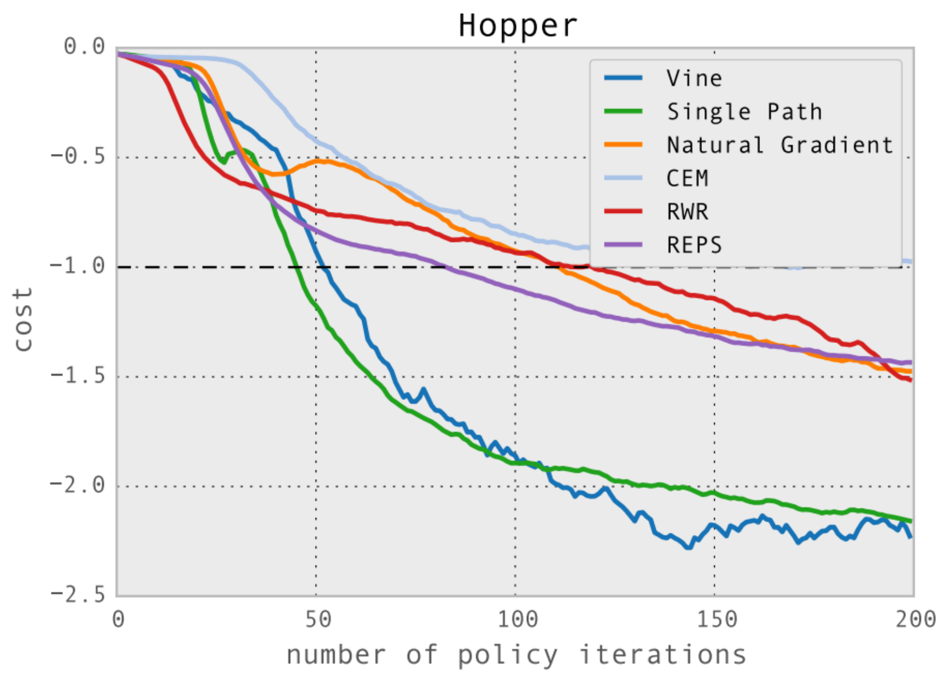


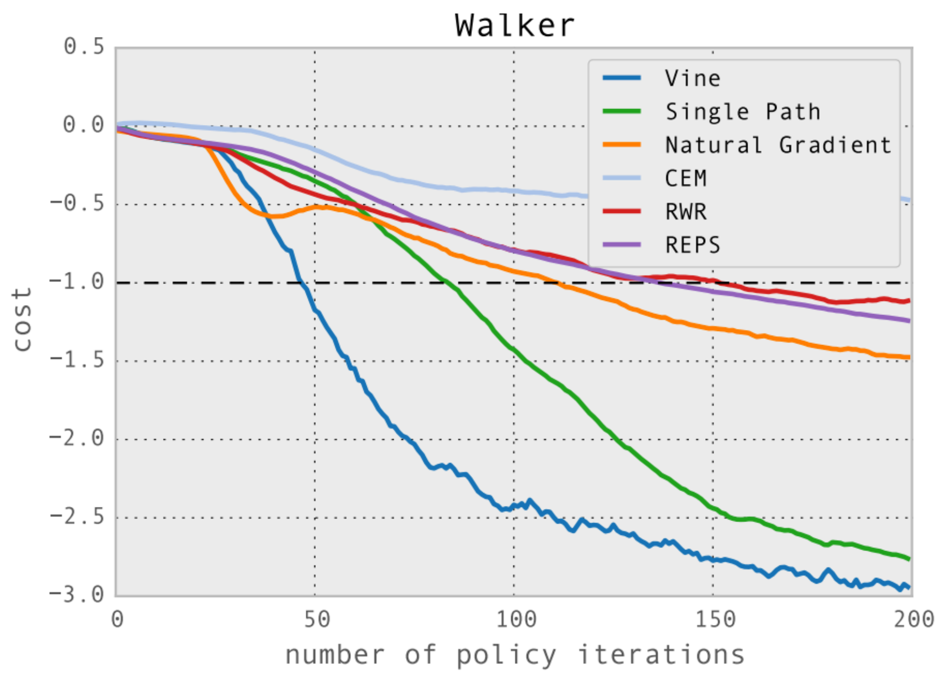
[Schulman, Levine, Moritz, Jordan, Abbeel, 2014]

Learning Curves -- Comparison

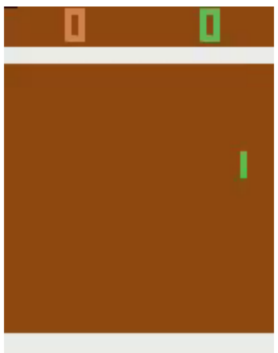


Learning Curves -- Comparison

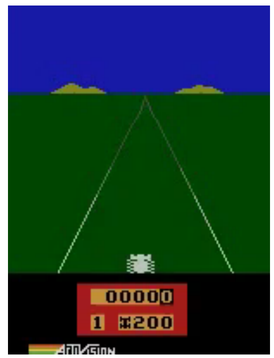




Atari Games

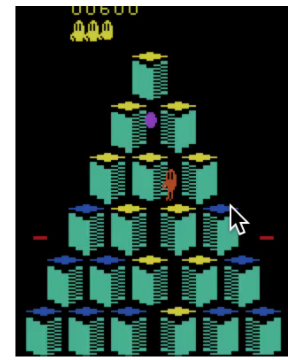


Pong



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]
- …

Outline

- Derivative free methods
  - Cross Entropy Method (CEM) / Finite Differences / Fixing Random Seed
- Likelihood Ratio (LR) Policy Gradient
  - Derivation / Connection w/Importance Sampling
- Natural Gradient / Trust Regions (-> TRPO)
- ***Actor-Critic (-> GAE, A3C)***
- Path Derivatives (PD) (-> DPG, DDPG, SVG)
- Stochastic Computation Graphs (generalizes LR / PD)
- Guided Policy Search (GPS)
- Inverse Reinforcement Learning

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

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

Recall Our Likelihood Ratio PG Estimator

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

- 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

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

- 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

Recall Our Likelihood Ratio PG Estimator

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

- 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

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

- 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

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

Reducing Variance by Function Approximation

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

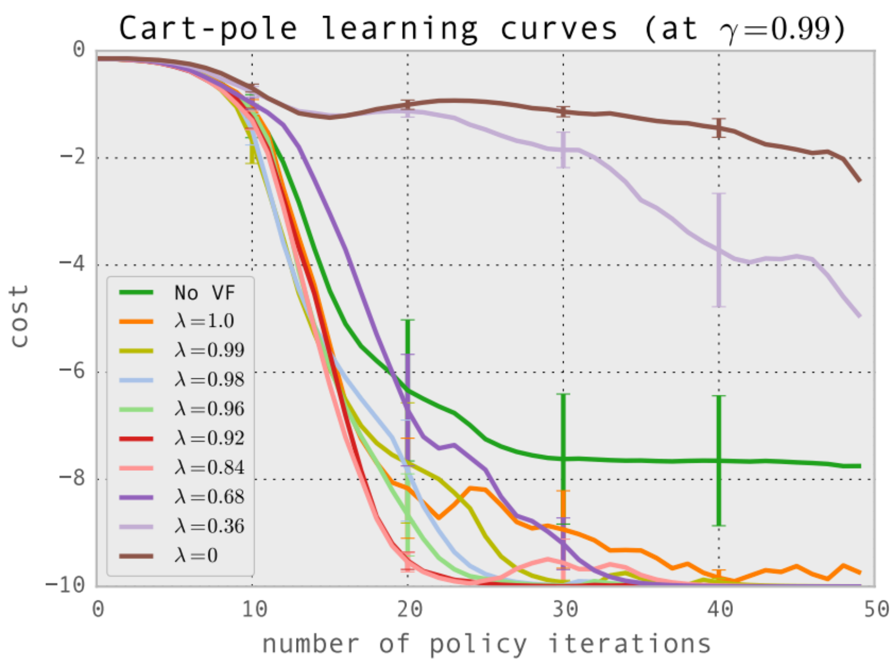
- ***Generalized Advantage Estimation*** uses an exponentially weighted average of sample estimates of these

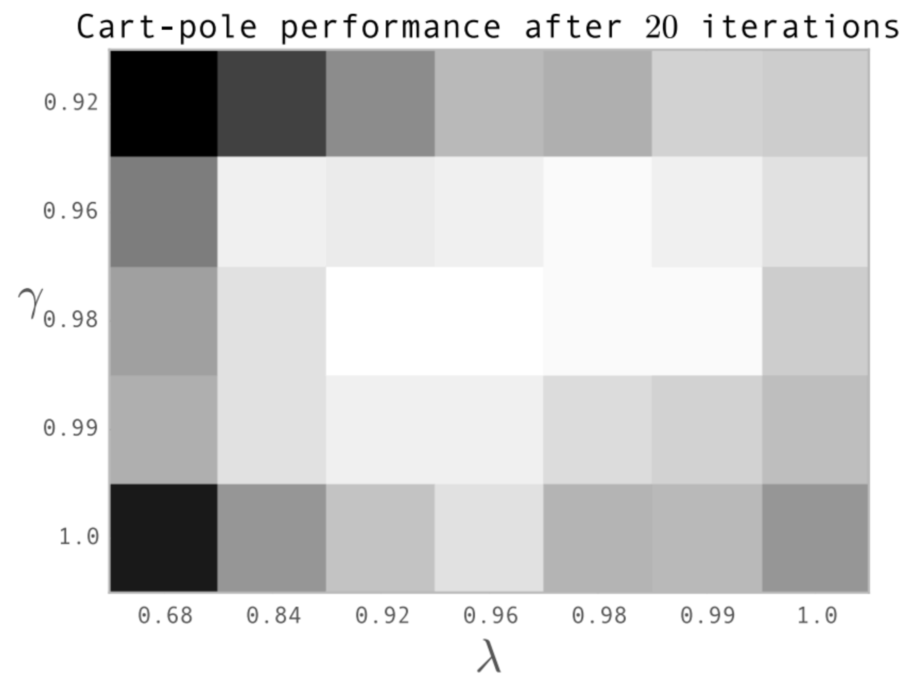
Reducing Variance by Function Approximation

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

- ***Generalized Advantage Estimation*** uses an exponentially weighted average of sample estimates of these
- ~ TD(lambda) / eligibility traces (Sutton and Barto, 1990)

Illustrate effect of gamma and kappa

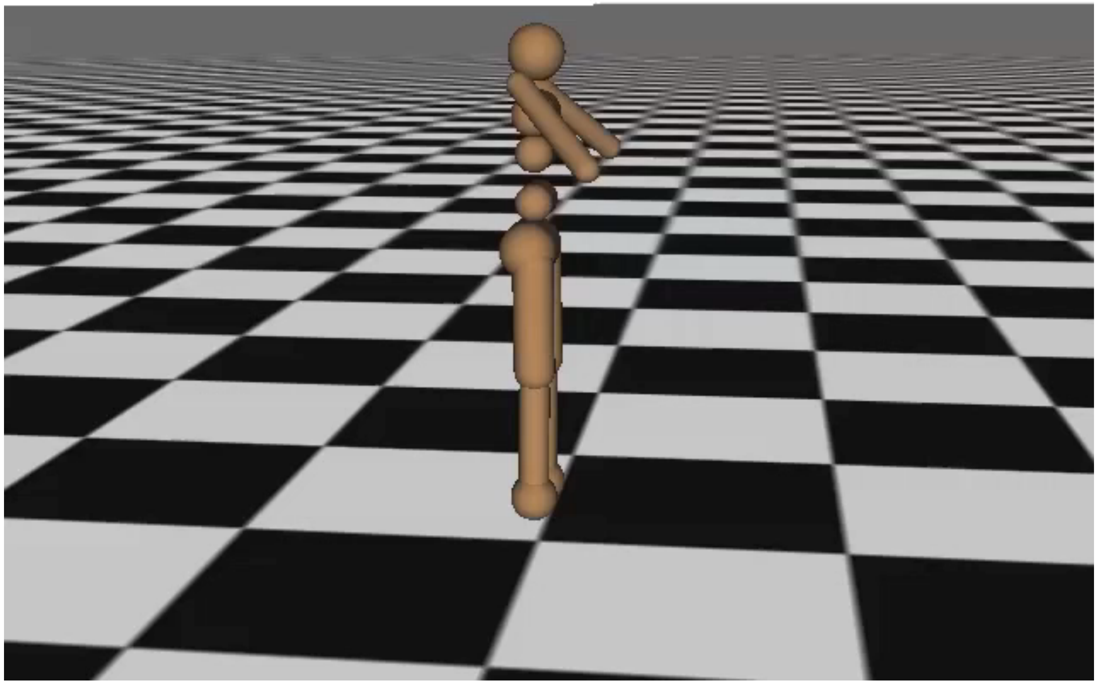




[Schulman et al, 2016 -- GAE]

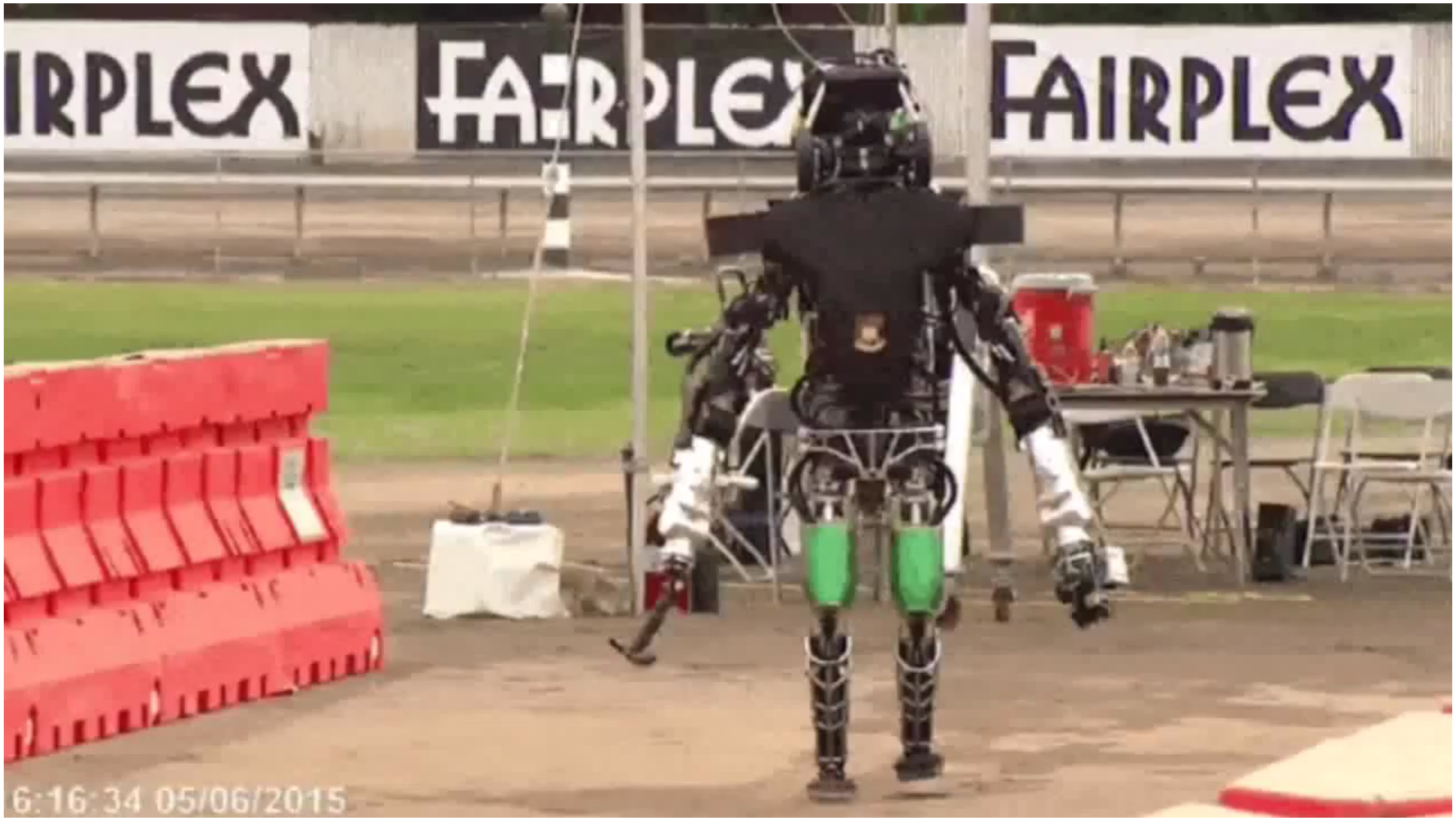
Learning Locomotion (TRPO + GAE)

Iteration 0



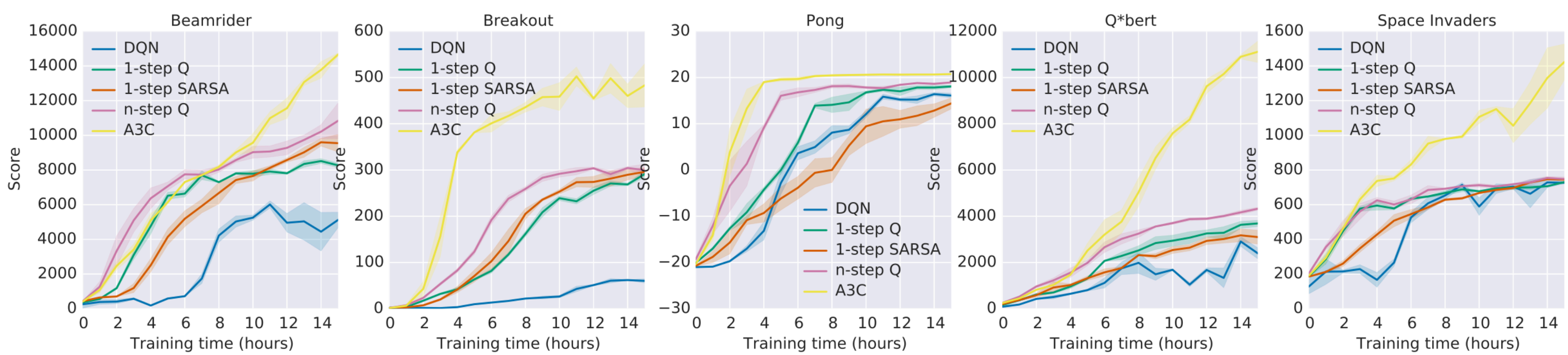
[Schulman, Moritz, Levine, Jordan, Abbeel, 2016]

In Contrast: Darpa Robotics Challenge



Async Advantage Actor Critic (A3C)

- [Mnih et al, 2015]
  - Likelihood Ratio Policy Gradient
  - Generalized Advantage Estimation

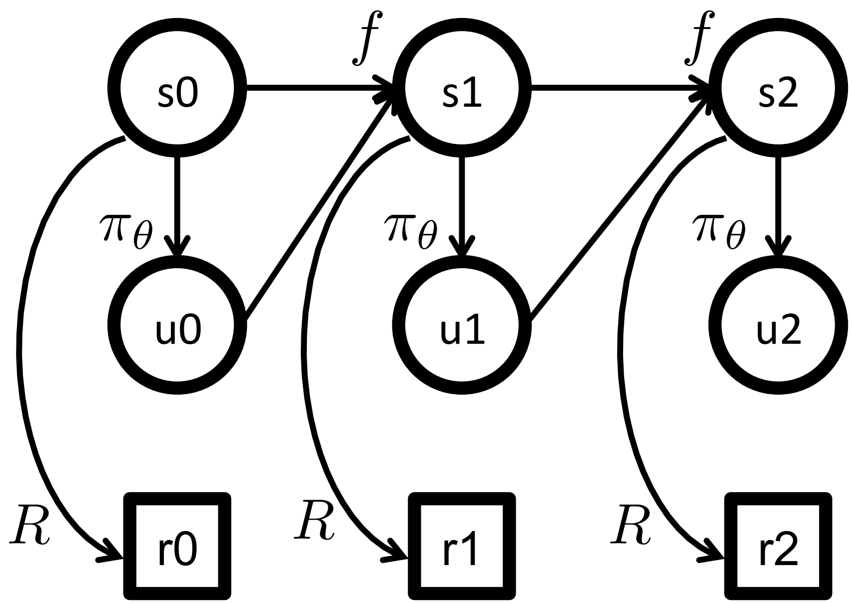


Outline

- Derivative free methods
  - Cross Entropy Method (CEM) / Finite Differences / Fixing Random Seed
- Likelihood Ratio (LR) Policy Gradient
  - Derivation / Connection w/Importance Sampling
- Natural Gradient / Trust Regions (-> TRPO)
- Actor-Critic (-> GAE, A3C)
- ***Path Derivatives (PD) (-> DPG, DDPG, SVG)***
- Stochastic Computation Graphs (generalizes LR / PD)
- Guided Policy Search (GPS)
- Inverse Reinforcement Learning

Gradient-Based Policy Optimization

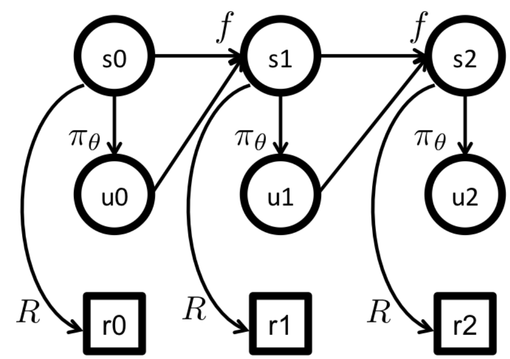
$$\max_{\theta} \; U(\theta) = \max_{\theta} \; \mathrm{E}[\sum_{t=0}^{H} R(s_t) | \pi_\theta]$$



Path Derivative

- Reminder of optimization objective:

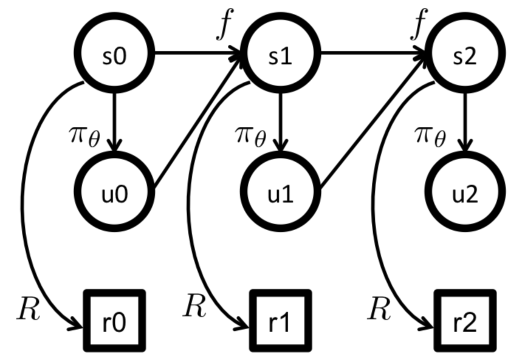
$$\max_{\theta} \; U(\theta) = \max_{\theta} \; \mathrm{E}[\sum_{t=0}^{H} R(s_t) | \pi_\theta]$$



Path Derivative

- Reminder of optimization objective:

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

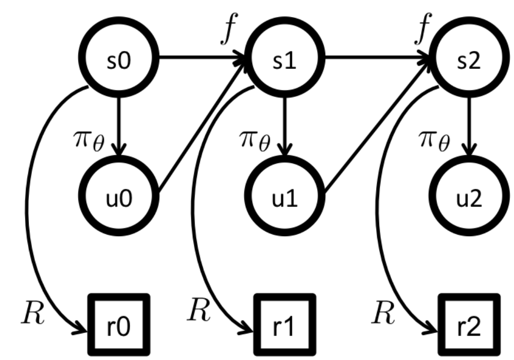


- Can compute gradient estimate along current roll-out:

Path Derivative

- Reminder of optimization objective:

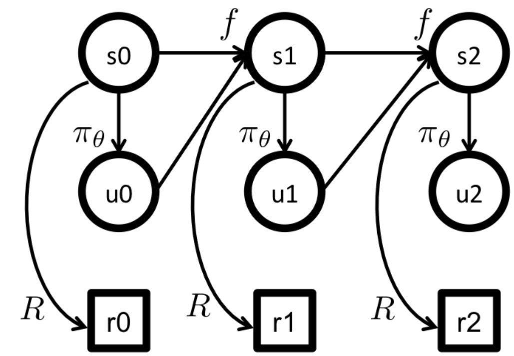
$$\max_{\theta} \; U(\theta) = \max_{\theta} \; \mathrm{E}[\sum_{t=0}^{H} R(s_t) | \pi_\theta]$$



- Can compute gradient estimate along current roll-out:

$$\frac{\partial U}{\partial \theta_i} = \sum_{t=0}^{H} \frac{\partial R}{\partial s}(s_t) \frac{\partial s_t}{\partial \theta_i}$$

Path Derivative



- Reminder of optimization objective:

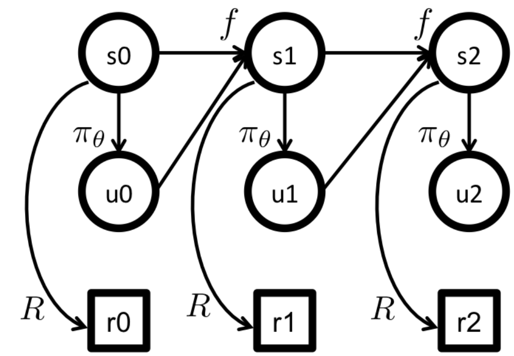
$$\max_\theta \; U(\theta) = \max_\theta \; \mathrm{E}[\sum_{t=0}^{H} R(s_t) | \pi_\theta]$$

- Can compute gradient estimate along current roll-out:

$$\frac{\partial U}{\partial \theta_i} = \sum_{t=0}^{H} \frac{\partial R}{\partial s}(s_t) \frac{\partial s_t}{\partial \theta_i}$$

$$\frac{\partial s_t}{\partial \theta_i} = \frac{\partial f}{\partial s}(s_{t-1}, u_{t-1}) \frac{\partial s_{t-1}}{\partial \theta_i} + \frac{\partial f}{\partial s}(s_{t-1}, u_{t-1}) \frac{\partial u_{t-1}}{\partial \theta_i}$$

Path Derivative



- Reminder of optimization objective:

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

- Can compute gradient estimate along current roll-out:

$$\frac{\partial U}{\partial \theta_i} = \sum_{t=0}^{H} \frac{\partial R}{\partial s}(s_t) \frac{\partial s_t}{\partial \theta_i}$$

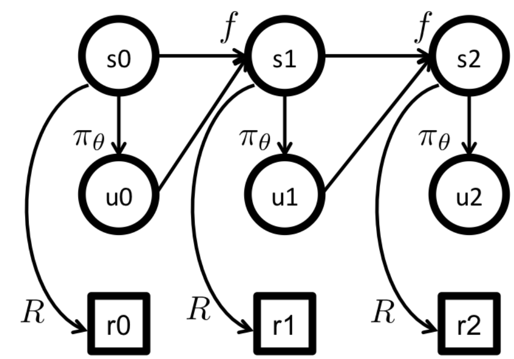
$$\frac{\partial s_t}{\partial \theta_i} = \frac{\partial f}{\partial s}(s_{t-1}, u_{t-1}) \frac{\partial s_{t-1}}{\partial \theta_i} + \frac{\partial f}{\partial s}(s_{t-1}, u_{t-1}) \frac{\partial u_{t-1}}{\partial \theta_i}$$

$$\frac{\partial u_t}{\partial \theta_i} = \frac{\partial \pi_\theta}{\partial \theta_i}(s_t, \theta) + \frac{\partial \pi_\theta}{\partial s}(s_t, \theta) \frac{\partial s_t}{\partial \theta_i}$$

Path Derivative

- Reminder of optimization objective:

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



- Can compute gradient estimate along current roll-out:

$$\frac{\partial U}{\partial \theta_i} = \sum_{t=0}^{H} \frac{\partial R}{\partial s}(s_t) \frac{\partial s_t}{\partial \theta_i}$$

$$\frac{\partial s_t}{\partial \theta_i} = \frac{\partial f}{\partial s}(s_{t-1}, u_{t-1}) \frac{\partial s_{t-1}}{\partial \theta_i} + \frac{\partial f}{\partial s}(s_{t-1}, u_{t-1}) \frac{\partial u_{t-1}}{\partial \theta_i}$$

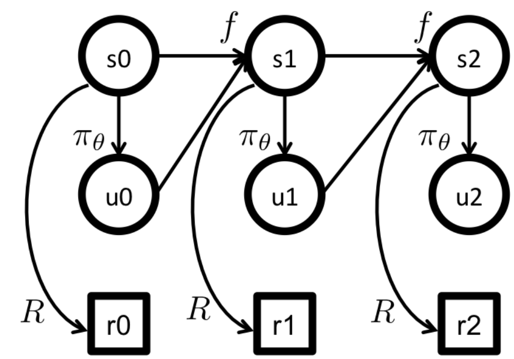
$$\frac{\partial u_t}{\partial \theta_i} = \frac{\partial \pi_\theta}{\partial \theta_i}(s_t, \theta) + \frac{\partial \pi_\theta}{\partial s}(s_t, \theta) \frac{\partial s_t}{\partial \theta_i}$$

Assumes:

- **f known**
- **f deterministic**

Path Derivative for Stochastic f – Additive Noise

$$s_{t+1} = f(s_t, u_t) + w_t$$



→ for any sample trajectory, can backsolve for the value of w_t and then apply same idea as for deterministic f for each sample trajectory

Path Derivative for Stochastic f – Reparameterization Trick

Path Derivative for Stochastic f – Reparameterization Trick

- Original: $$s_{t+1} = f_{\mathrm{STOCH}}(s_t, u_t, \theta)$$
- Reparameterized: $$s_{t+1} = f_{\mathrm{DET}}(s_t, u_t, \theta, \xi_{\mathrm{STOCH}})$$

Path Derivative for Stochastic f – Reparameterization Trick

- Original: $$s_{t+1} = f_{\text{STOCH}}(s_t, u_t, \theta)$$
- Reparameterized: $$s_{t+1} = f_{\text{DET}}(s_t, u_t, \theta, \xi_{\text{STOCH}})$$
- E.g. $$s_{t+1} \sim \mathcal{N}(g(s_t, u_t, \theta), \sigma^2)$$

Path Derivative for Stochastic f – Reparameterization Trick

- Original: $$s_{t+1} = f_{\text{STOCH}}(s_t, u_t, \theta)$$

- Reparameterized: $$s_{t+1} = f_{\text{DET}}(s_t, u_t, \theta, \xi_{\text{STOCH}})$$

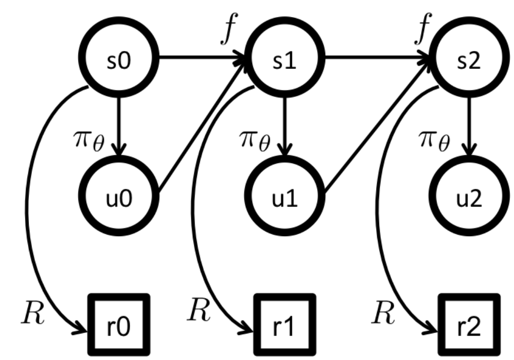
- E.g. $$s_{t+1} \sim \mathcal{N}(g(s_t, u_t, \theta), \sigma^2)$$

  → $$s_{t+1} = g(s_t, u_t, \theta) + \sigma\xi$$

SVG, (D)DPG

- SVG: $$\max_\theta \; U(\theta) = \max_\theta \; \mathrm{E}[\sum_{t=0}^{H} R(s_t)|\pi_\theta]$$
  - Computes gradient estimate along current roll-out:

$$\frac{\partial U}{\partial \theta_i} = \sum_{t=0}^{H} \frac{\partial R}{\partial s}(s_t)\frac{\partial s_t}{\partial \theta_i}$$
$$\frac{\partial s_t}{\partial \theta_i} = \frac{\partial f}{\partial s}(s_{t-1}, u_{t-1})\frac{\partial s_{t-1}}{\partial \theta_i} + \frac{\partial f}{\partial s}(s_{t-1}, u_{t-1})\frac{\partial u_{t-1}}{\partial \theta_i}$$
$$\frac{\partial u_t}{\partial \theta_i} = \frac{\partial \pi_\theta}{\partial \theta_i}(s_t, \theta) + \frac{\partial \pi_\theta}{\partial s}(s_t, \theta)\frac{\partial s_t}{\partial \theta_i}$$



[SVG: Heess et al, 2015; DPG: Silver, 2014, DDPG Lillicrap et al, 2015]

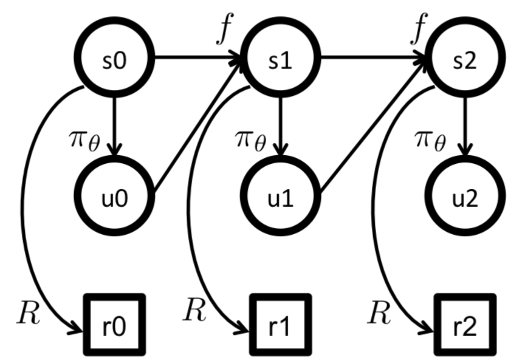
SVG, (D)DPG

- SVG: $\max_\theta U(\theta) = \max_\theta \mathrm{E}[\sum_{t=0}^{H} R(s_t)|\pi_\theta]$
  - Computes gradient estimate along current roll-out:

$$\frac{\partial U}{\partial \theta_i} = \sum_{t=0}^{H} \frac{\partial R}{\partial s}(s_t)\frac{\partial s_t}{\partial \theta_i}$$

$$\frac{\partial s_t}{\partial \theta_i} = \frac{\partial f}{\partial s}(s_{t-1}, u_{t-1})\frac{\partial s_{t-1}}{\partial \theta_i} + \frac{\partial f}{\partial s}(s_{t-1}, u_{t-1})\frac{\partial u_{t-1}}{\partial \theta_i}$$

$$\frac{\partial u_t}{\partial \theta_i} = \frac{\partial \pi_\theta}{\partial \theta_i}(s_t, \theta) + \frac{\partial \pi_\theta}{\partial s}(s_t, \theta)\frac{\partial s_t}{\partial \theta_i}$$



- DPG, DDPG replace sum of future rewards by fitted Q

  $\frac{d}{du}Q^\pi(s,u)$ says how to improve action

[SVG: Heess et al, 2015; DPG: Silver, 2014, DDPG Lillicrap et al, 2015]

Outline

- Derivative free methods
  - Cross Entropy Method (CEM) / Finite Differences / Fixing Random Seed
- Likelihood Ratio (LR) Policy Gradient
  - Derivation / Connection w/Importance Sampling
- Natural Gradient / Trust Regions (-> TRPO)
- Actor-Critic (-> GAE, A3C)
- Path Derivatives (PD) (-> DPG, DDPG, SVG)
- ***Stochastic Computation Graphs (generalizes LR / PD)***
- Guided Policy Search (GPS)
- Inverse Reinforcement Learning

Stochastic Computation Graphs

- For any stochastic neural net
  - Can mix and match likelihood ratio and path derivative
  - If black-box node: might need to place stochastic node in front of it and use likelihood ratio
  - This includes recurrent neural net policies etc.

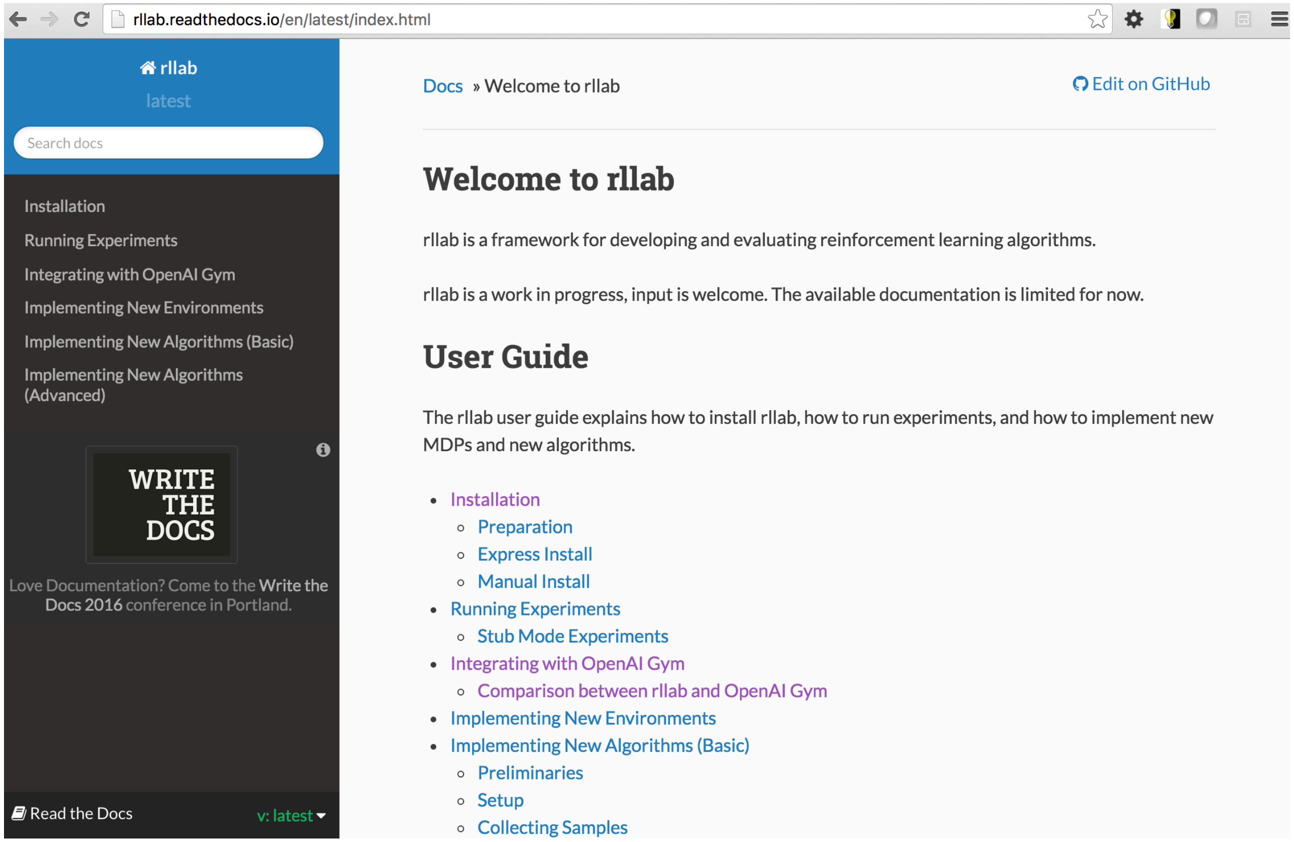
- Details: Schulman, Heess, Weber, Abbeel, NIPS 2015

[redacted]

Benchmarking [Duan et al, ICML 2016]

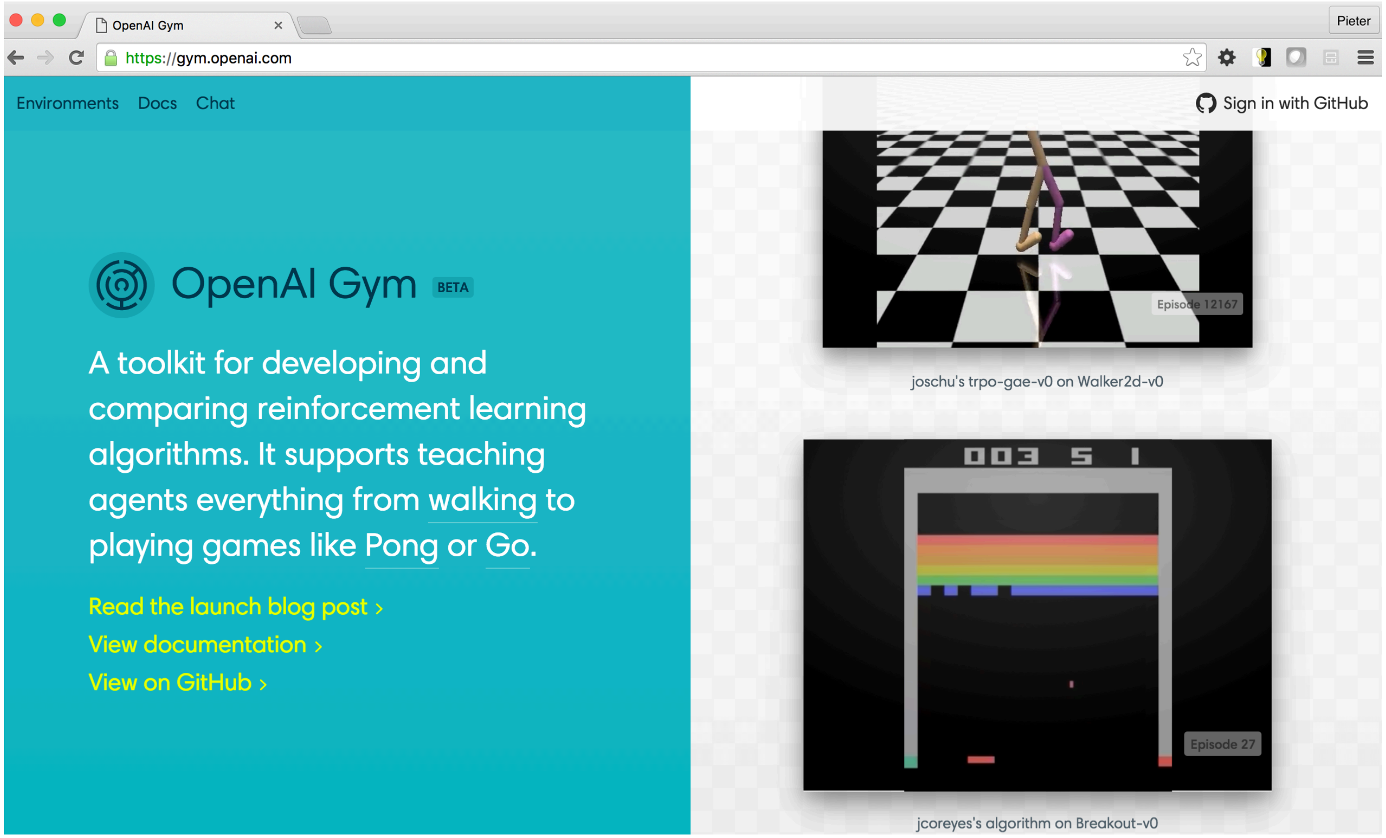
Task	Random	REINFORCE	TNPG	RWR	REPS	TRPO	CEM	CMA-ES	DDPG
Cart-Pole Balancing	77.1 ± 0.0	4693.7 ± 14.0	**3986.4 ± 748.9**	**4861.5 ± 12.3**	565.6 ± 137.6	**4869.8 ± 37.6**	4815.4 ± 4.8	2440.4 ± 568.3	4634.4 ± 87.8
Inverted Pendulum*	−153.4 ± 0.2	13.4 ± 18.0	**209.7 ± 55.5**	84.7 ± 13.8	−113.3 ± 4.6	**247.2 ± 76.1**	38.2 ± 25.7	−40.1 ± 5.7	40.0 ± 244.6
Mountain Car	−415.4 ± 0.0	−67.1 ± 1.0	**-66.5 ± 4.5**	−79.4 ± 1.1	−275.6 ± 166.3	**-61.7 ± 0.9**	−66.0 ± 2.4	−85.0 ± 7.7	−288.4 ± 170.3
Acrobot	−1904.5 ± 1.0	−508.1 ± 91.0	−395.8 ± 121.2	−352.7 ± 35.9	−1001.5 ± 10.8	−326.0 ± 24.4	−436.8 ± 14.7	−785.6 ± 13.1	**-223.6 ± 5.8**
Double Inverted Pendulum*	149.7 ± 0.1	4116.5 ± 65.2	**4455.4 ± 37.6**	3614.8 ± 368.1	446.7 ± 114.8	**4412.4 ± 50.4**	2566.2 ± 178.9	1576.1 ± 51.3	2863.4 ± 154.0
Swimmer*	−1.7 ± 0.1	92.3 ± 0.1	**96.0 ± 0.2**	60.7 ± 5.5	3.8 ± 3.3	**96.0 ± 0.2**	68.8 ± 2.4	64.9 ± 1.4	85.8 ± 1.8
Hopper	8.4 ± 0.0	714.0 ± 29.3	**1155.1 ± 57.9**	553.2 ± 71.0	86.7 ± 17.6	**1183.3 ± 150.0**	63.1 ± 7.8	20.3 ± 14.3	267.1 ± 43.5
2D Walker	−1.7 ± 0.0	506.5 ± 78.8	**1382.6 ± 108.2**	136.0 ± 15.9	−37.0 ± 38.1	**1353.8 ± 85.0**	84.5 ± 19.2	77.1 ± 24.3	318.4 ± 181.6
Half-Cheetah	−90.8 ± 0.3	1183.1 ± 69.2	**1729.5 ± 184.6**	376.1 ± 28.2	34.5 ± 38.0	**1914.0 ± 120.1**	330.4 ± 274.8	441.3 ± 107.6	**2148.6 ± 702.7**
Ant*	13.4 ± 0.7	548.3 ± 55.5	**706.0 ± 127.7**	37.6 ± 3.1	39.0 ± 9.8	**730.2 ± 61.3**	49.2 ± 5.9	17.8 ± 15.5	326.2 ± 20.8
Simple Humanoid	41.5 ± 0.2	128.1 ± 34.0	**255.0 ± 24.5**	93.3 ± 17.4	28.3 ± 4.7	**269.7 ± 40.3**	60.6 ± 12.9	28.7 ± 3.9	99.4 ± 28.1
Full Humanoid	13.2 ± 0.1	262.2 ± 10.5	**288.4 ± 25.2**	46.7 ± 5.6	41.7 ± 6.1	**287.0 ± 23.4**	36.9 ± 2.9	N/A ± N/A	119.0 ± 31.2
Cart-Pole Balancing (LS)*	77.1 ± 0.0	420.9 ± 265.5	**945.1 ± 27.8**	68.9 ± 1.5	898.1 ± 22.1	**960.2 ± 46.0**	227.0 ± 223.0	68.0 ± 1.6	
Inverted Pendulum (LS)	−122.1 ± 0.1	−13.4 ± 3.2	**0.7 ± 6.1**	−107.4 ± 0.2	−87.2 ± 8.0	**4.5 ± 4.1**	−81.2 ± 33.2	−62.4 ± 3.4	
Mountain Car (LS)	−83.0 ± 0.0	−81.2 ± 0.6	**-65.7** ± 9.0	−81.7 ± 0.1	−82.6 ± 0.4	**-64.2 ± 9.5**	**-68.9 ± 1.3**	**-73.2 ± 0.6**	
Acrobot (LS)*	−393.2 ± 0.0	−128.9 ± 11.6	**-84.6 ± 2.9**	−235.9 ± 5.3	−379.5 ± 1.4	**-83.3 ± 9.9**	−149.5 ± 15.3	−159.9 ± 7.5	
Cart-Pole Balancing (NO)*	101.4 ± 0.1	616.0 ± 210.8	**916.3 ± 23.0**	93.8 ± 1.2	99.6 ± 7.2	606.2 ± 122.2	181.4 ± 32.1	104.4 ± 16.0	
Inverted Pendulum (NO)	−122.2 ± 0.1	6.5 ± 1.1	**11.5 ± 0.5**	−110.0 ± 1.4	−119.3 ± 4.2	**10.4 ± 2.2**	−55.6 ± 16.7	−80.3 ± 2.8	
Mountain Car (NO)	−83.0 ± 0.0	−74.7 ± 7.8	**-64.5 ± 8.6**	−81.7 ± 0.1	−82.9 ± 0.1	**-60.2 ± 2.0**	−67.4 ± 1.4	−73.5 ± 0.5	
Acrobot (NO)*	−393.5 ± 0.0	**-186.7 ± 31.3**	**-164.5 ± 13.4**	−233.1 ± 0.4	−258.5 ± 14.0	**-149.6 ± 8.6**	−213.4 ± 6.3	−236.6 ± 6.2	
Cart-Pole Balancing (SI)*	76.3 ± 0.1	431.7 ± 274.1	**980.5 ± 7.3**	69.0 ± 2.8	702.4 ± 196.4	**980.3 ± 5.1**	746.6 ± 93.2	71.6 ± 2.9	
Inverted Pendulum (SI)	−121.8 ± 0.2	−5.3 ± 5.6	**14.8 ± 1.7**	−108.7 ± 4.7	−92.8 ± 23.9	**14.1 ± 0.9**	−51.8 ± 10.6	−63.1 ± 4.8	
Mountain Car (SI)	−82.7 ± 0.0	−63.9 ± 0.2	**-61.8 ± 0.4**	−81.4 ± 0.1	−80.7 ± 2.3	**-61.6 ± 0.4**	−63.9 ± 1.0	−66.9 ± 0.6	
Acrobot (SI)*	−387.8 ± 1.0	**-169.1 ± 32.3**	**-156.6 ± 38.9**	−233.2 ± 2.6	−216.1 ± 7.7	**-170.9 ± 40.3**	−250.2 ± 13.7	−245.0 ± 5.5	
Swimmer + Gathering	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
Ant + Gathering	−5.8 ± 5.0	−0.1 ± 0.1	−0.4 ± 0.1	−5.5 ± 0.5	−6.7 ± 0.7	−0.4 ± 0.0	−4.7 ± 0.7	N/A ± N/A	−0.3 ± 0.3
Swimmer + Maze	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
Ant + Maze	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	N/A ± N/A	0.0 ± 0.0

rllab



[Duan et al]

Open AI Gym



OpenAI Gym
Pieter
https://gym.openai.com
Environments
Docs
Chat
Sign in with GitHub
OpenAI Gym BETA
A toolkit for developing and comparing reinforcement learning algorithms. It supports teaching agents everything from walking to playing games like Pong or Go.
Read the launch blog post ›
View documentation ›
View on GitHub ›
Episode 12167
joschu's trpo-gae-v0 on Walker2d-v0
Episode 27
jcoreyes's algorithm on Breakout-v0

Outline

- Derivative free methods
  - Cross Entropy Method (CEM) / Finite Differences / Fixing Random Seed
- Likelihood Ratio (LR) Policy Gradient
  - Derivation / Connection w/Importance Sampling
- Natural Gradient / Trust Regions (-> TRPO)
- Actor-Critic (-> GAE, A3C)
- Path Derivatives (PD) (-> DPG, DDPG, SVG)
- Stochastic Computation Graphs (generalizes LR / PD)
- ***Guided Policy Search (GPS)***
- Inverse Reinforcement Learning

Goal

- Find parameterized policy $\pi_\theta(\mathbf{u}_t|\mathbf{x}_t)$ that optimizes:

$$J(\theta) = \sum_{t=1}^{T} \mathrm{E}_{\pi_\theta(\mathbf{x}_t,\mathbf{u}_t)}[l(\mathbf{x}_t, \mathbf{u}_t)]$$

- Notation: $$\pi_\theta(\tau) = p(\mathbf{x}_1) \prod_{t=1}^{T} p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)\pi_\theta(\mathbf{u}_t|\mathbf{x}_t)$$

$$\tau = \{\mathbf{x}_1, \mathbf{u}_1, \ldots, \mathbf{x}_T, \mathbf{u}_T\}$$

- RL takes lots of data… Can we reduce to supervised learning?

Naïve Solution

- Step 1:
  - Consider sampled problem instances $i = 1, 2, \ldots, I$
  - Find a trajectory-centric controller $\pi_i(\mathbf{u}_t|\mathbf{x}_t)$ for each problem instance
- Step 2:
  - Supervised training of neural net to match all $\pi_i(\mathbf{u}_t|\mathbf{x}_t)$

$$\pi_\theta \leftarrow \arg\min_\theta \sum_i D_{\mathrm{KL}}(p_i(\tau)||\pi_\theta(\tau))$$

- ISSUES:
  - Compounding error (Ross, Gordon, Bagnell JMLR 2011 "Dagger")
  - Mismatch train vs. test E.g., Blind peg, Vision,...

(Generic) Guided Policy Search

- Optimization formulation:

$$\min_{\theta, p_1, \ldots, p_N} \sum_{i=1}^{N} \sum_{t=1}^{T} E_{p_i(\mathbf{x}_t, \mathbf{u}_t)}[\ell(\mathbf{x}_t, \mathbf{u}_t)] \text{ such that } p_i(\mathbf{u}_t|\mathbf{x}_t) = \pi_\theta(\mathbf{u}_t|\mathbf{x}_t) \;\; \forall \mathbf{x}_t, \mathbf{u}_t, t, i. \qquad (1)$$

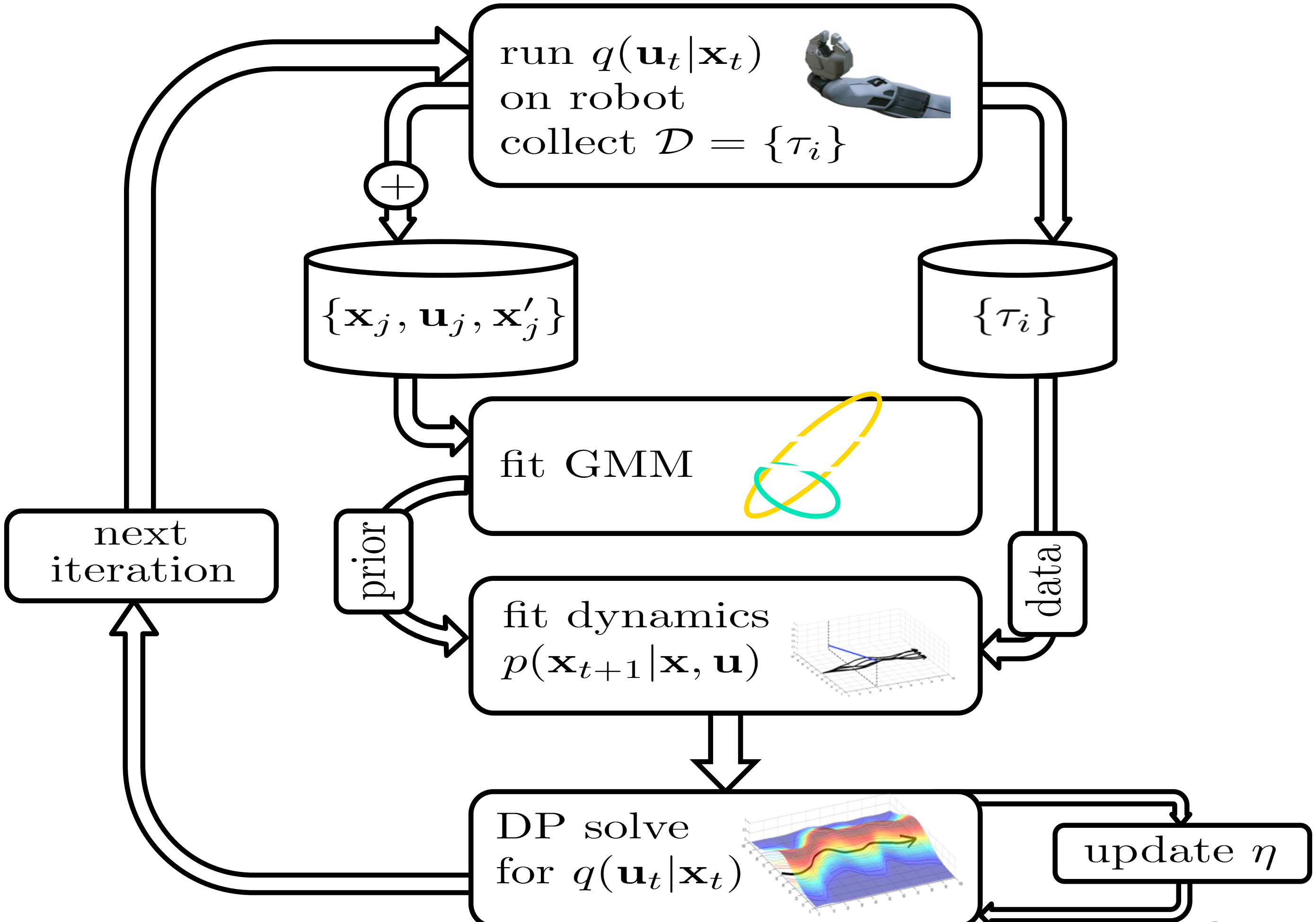
Particular form of the constraint varies depending on the specific method:

Dual gradient descent: Levine and Abbeel, NIPS 2014

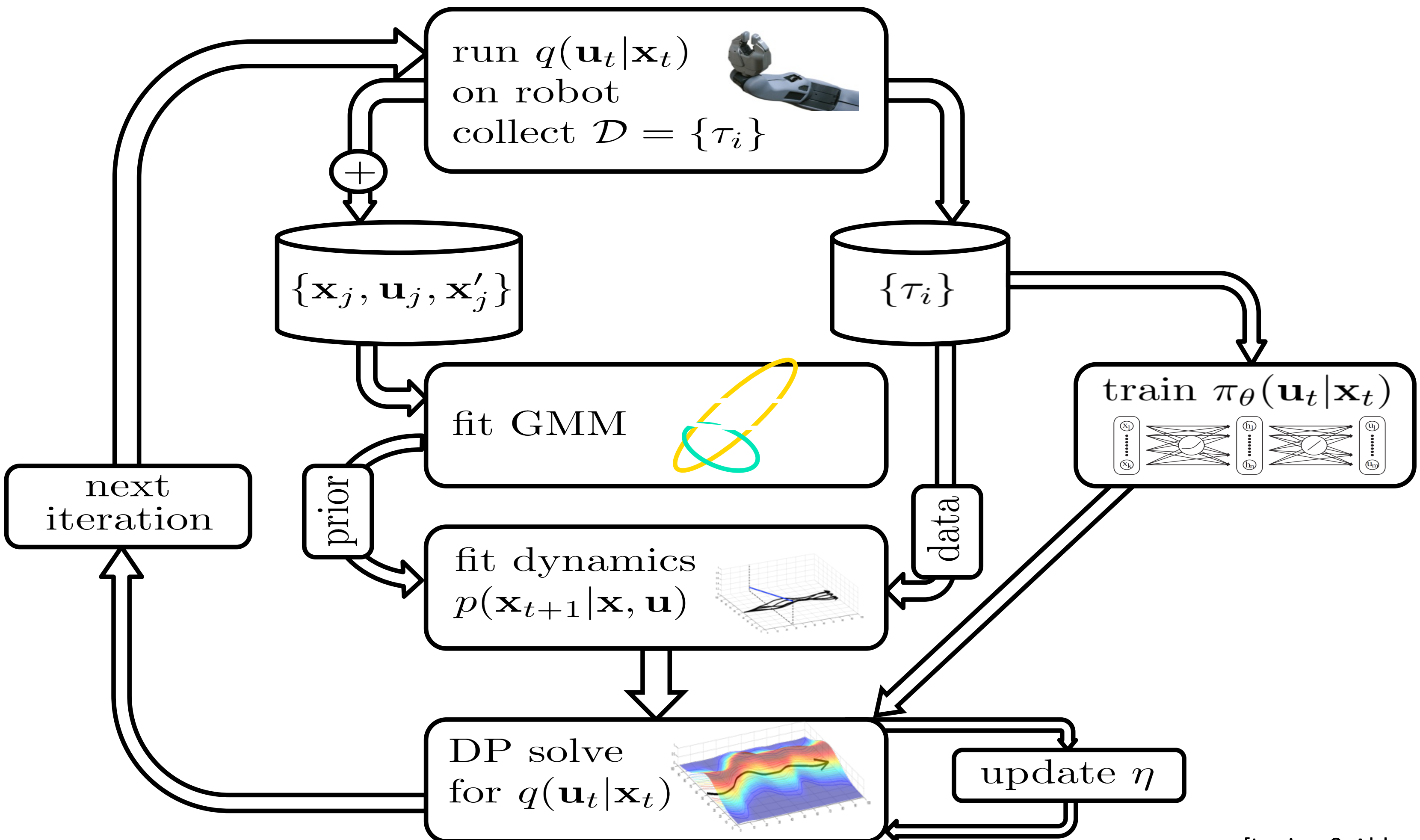
Penalty methods: Mordatch, Lowrey, Andrew, Popovic, Todorov, NIPS 2016

ADMM: Mordatch and Todorov, RSS 2014

Bregman ADMM:Levine, Finn, Darrell, Abbeel, JMLR 2016

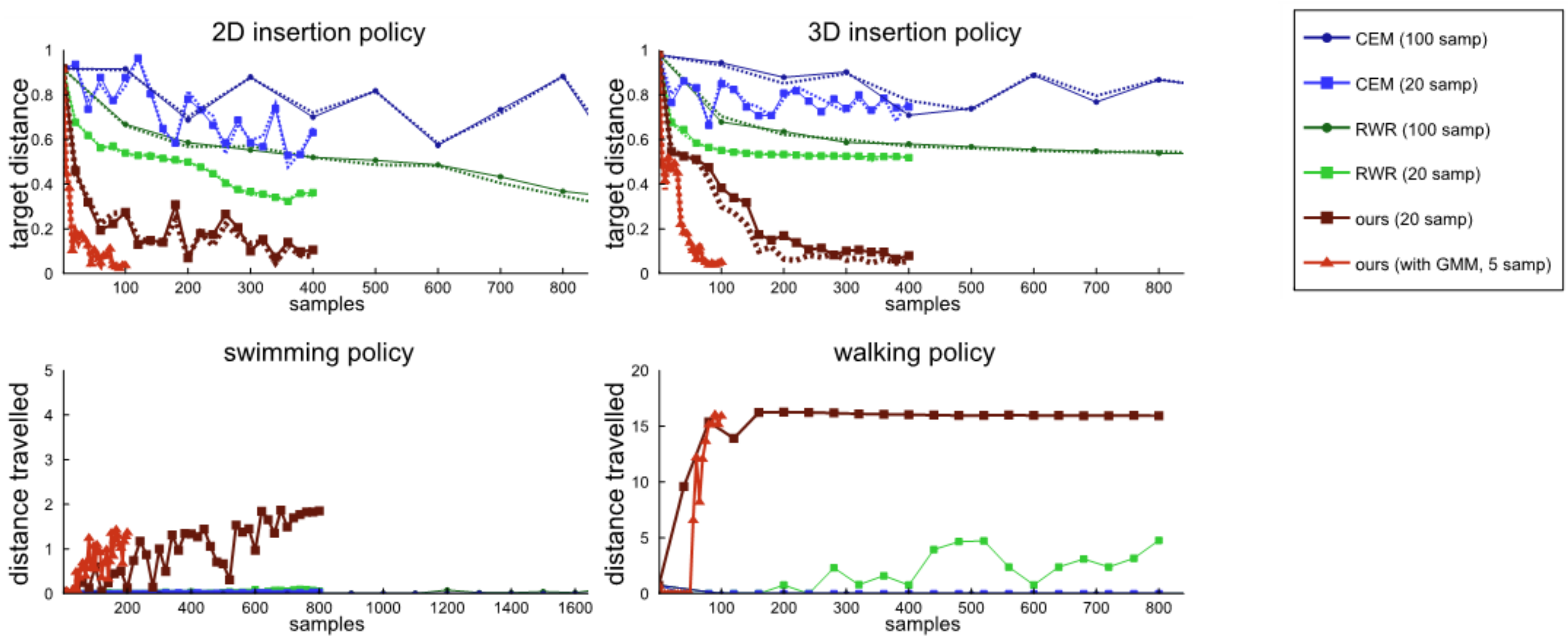


[Levine & Abbeel, NIPS 2014]



[Levine & Abbeel, NIPS 2014]

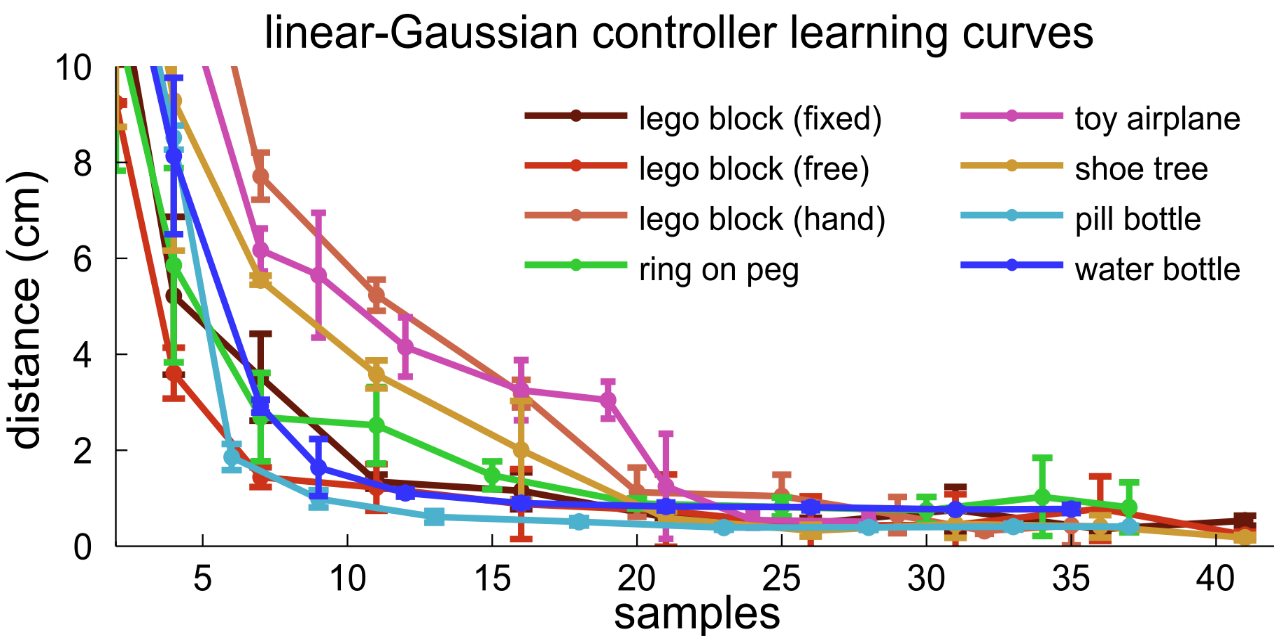
Comparison



2D insertion policy
target distance
samples
3D insertion policy
target distance
samples
swimming policy
distance travelled
samples
walking policy
distance travelled
samples
CEM (100 samp)
CEM (20 samp)
RWR (100 samp)
RWR (20 samp)
ours (20 samp)
ours (with GMM, 5 samp)

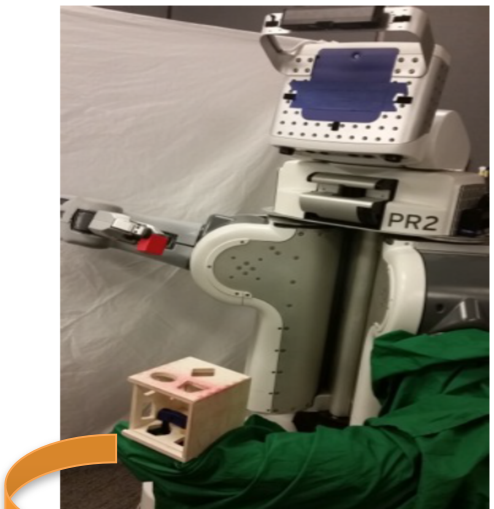
Block Stacking – Learning the Controller for a Single Instance

Linear-Gaussian Controller Learning Curves

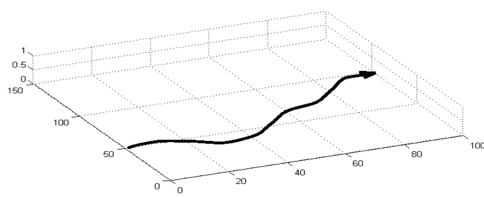


Instrumented Training

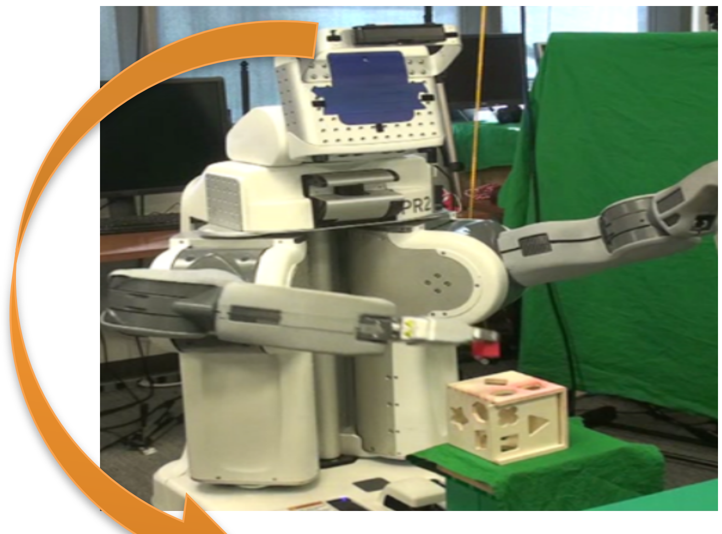
training time



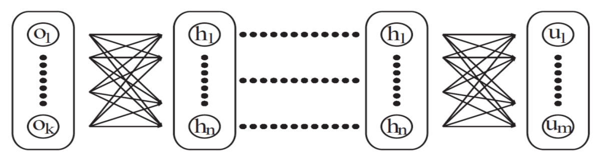
$$\mathbf{x}_t \longrightarrow \mathbf{u}_t$$



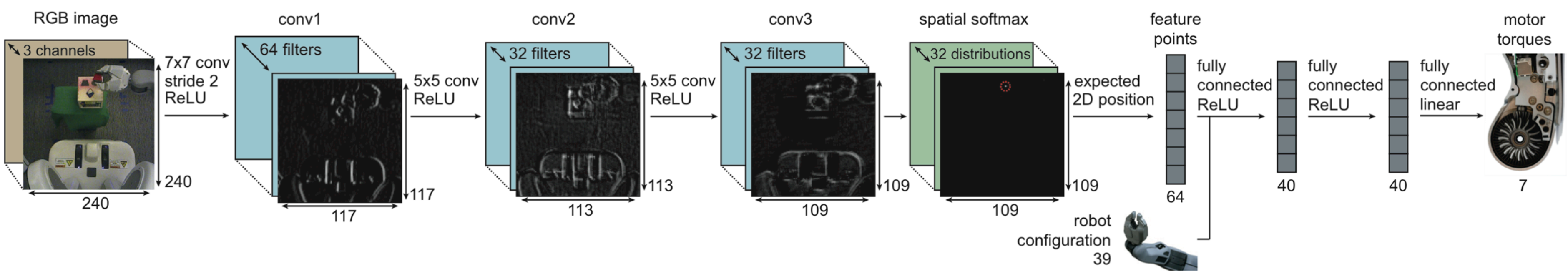
test time



$$\mathbf{o}_t \longrightarrow \mathbf{u}_t$$

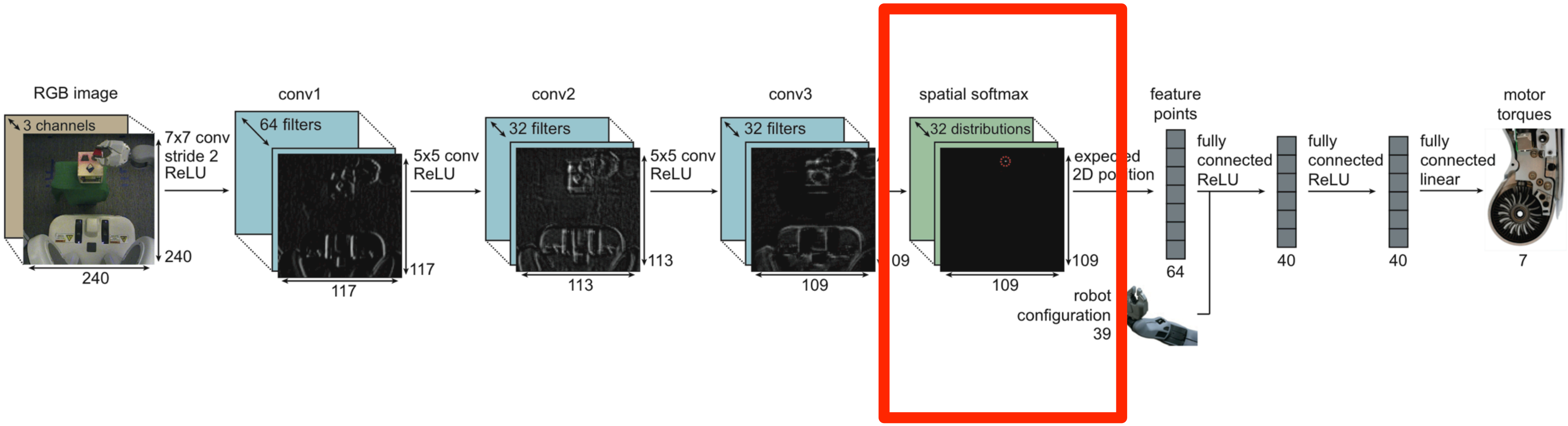


Architecture (92,000 parameters)



[Levine*, Finn*, Darrell, Abbeel, JMLR 2016

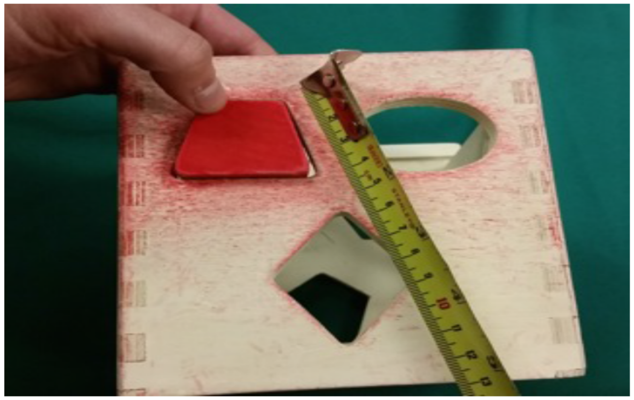
π_θ Deep Spatial Neural Net Architecture

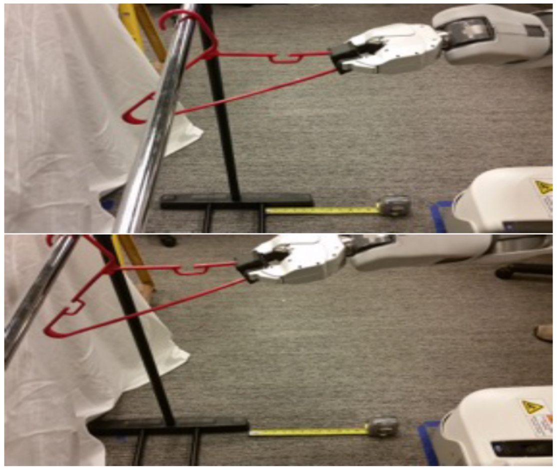


(92,000 parameters)

[Levine*, Finn*, Darrell, Abbeel, JMLR 2016

Experimental Tasks





[Levine*, Finn*, Darrell, Abbeel, JMLR 2016

Learning

[Levine*, Finn*, Darrell, Abbeel, JMLR 2016

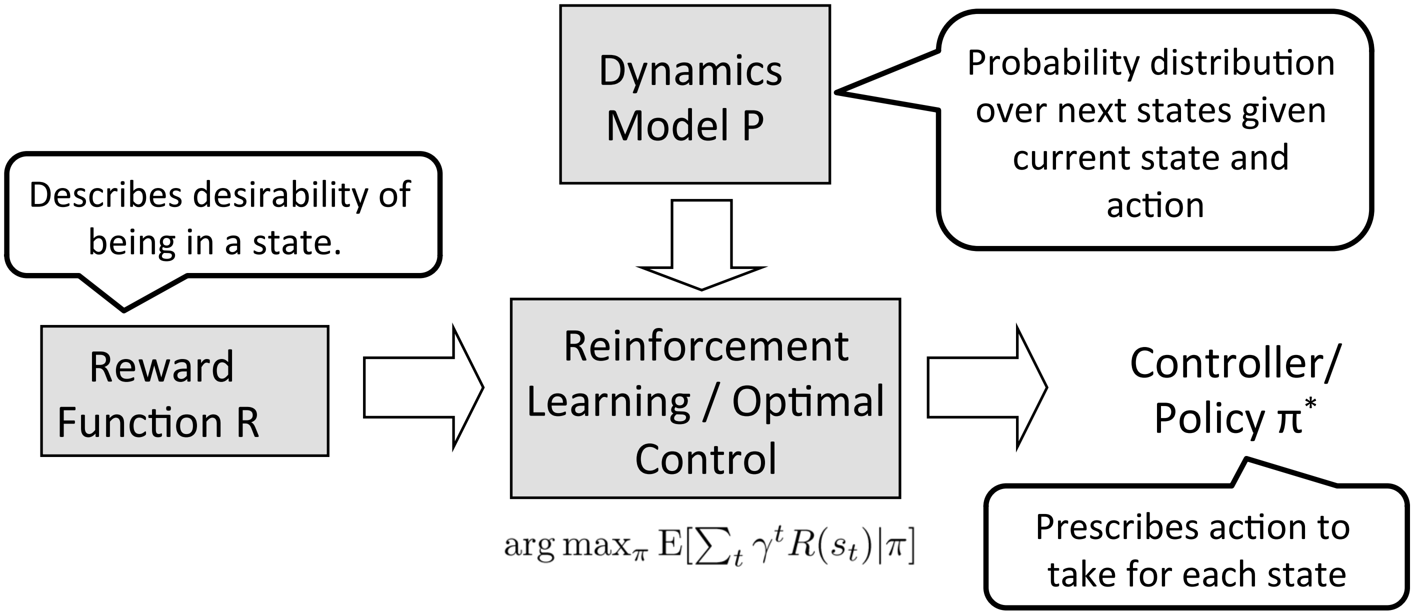
Learned Skills

[Levine*, Finn*, Darrell, Abbeel, JMLR 2016

Outline

- Derivative free methods
  - Cross Entropy Method (CEM) / Finite Differences / Fixing Random Seed
- Likelihood Ratio (LR) Policy Gradient
  - Derivation / Connection w/Importance Sampling
- Natural Gradient / Trust Regions (-> TRPO)
- Actor-Critic (-> GAE, A3C)
- Path Derivatives (PD) (-> DPG, DDPG, SVG)
- Stochastic Computation Graphs (generalizes LR / PD)
- Guided Policy Search (GPS)
- ***Inverse Reinforcement Learning***

High-level picture



$\arg\max_{\pi} \mathrm{E}[\sum_t \gamma^t R(s_t)|\pi]$

Inverse RL:

Given π^* and P, can we recover R?

More generally, given execution traces, can we recover R?

Motivation for inverse RL

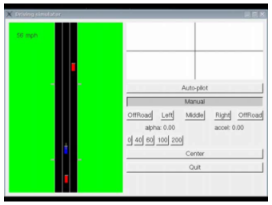
- Scientific inquiry
  - Model animal and human behavior
    - E.g., bee foraging, songbird vocalization. [See intro of Ng and Russell, 2000 for a brief overview.]
- Apprenticeship learning/Imitation learning through inverse RL
  - Presupposition: reward function provides the most succinct and transferable definition of the task
  - Has enabled advancing the state of the art in various robotic domains
- Modeling of other agents, both adversarial and cooperative

Outline

- Example applications
- Inverse RL vs. behavioral cloning
- Historical sketch of inverse RL
- Mathematical formulations for inverse RL
- Case studies

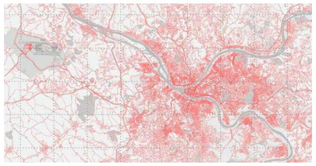
Examples

- Simulated highway driving
  - Abbeel and Ng, ICML 2004,
  - Syed and Schapire, NIPS 2007



- Aerial imagery based navigation
  - Ratliff, Bagnell and Zinkevich, ICML 2006

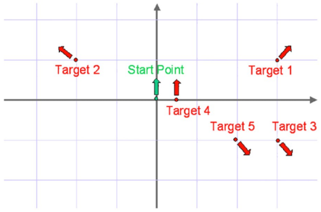
- Parking lot navigation
  - Abbeel, Dolgov, Ng and Thrun, IROS 2008



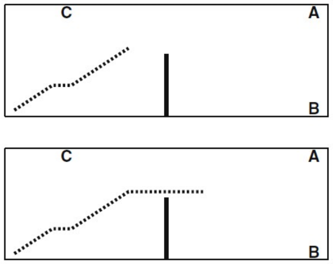
- Urban navigation
  - Ziebart, Maas, Bagnell and Dey, AAAI 2008

Examples (ctd)

- Human path planning
  - Mombaur, Truong and Laumond, AURO 2009



- Human goal inference
  - Baker, Saxe and Tenenbaum, Cognition 2009



- Quadruped locomotion
  - Ratliff, Bradley, Bagnell and Chestnutt, NIPS 2007
  - Kolter, Abbeel and Ng, NIPS 2008

Lecture outline

- Example applications
- *Inverse RL vs. behavioral cloning*
- Historical sketch of inverse RL
- Mathematical formulations for inverse RL
- Case studies

Problem setup

- Input:
  - State space, action space
  - *No* reward function
  - Transition model $P_{sa}(s_{t+1} \mid s_t, a_t)$
  - Teacher's demonstration: $s_0, a_0, s_1, a_1, s_2, a_2, \ldots$
    (= trace of the teacher's policy π^*)
- *Inverse RL:*
  - Can we recover R ?
- *Apprenticeship learning via inverse RL*
  - Can we then use this R to find a good policy ?
- *Behavioral cloning*
  - Can we directly learn the teacher's policy using supervised learning?

Behavioral cloning

- Formulate as standard machine learning problem
  - Fix a policy class
    - E.g., support vector machine, neural network, decision tree, deep belief net, …
  - Estimate a policy (=mapping from states to actions) from the training examples $(s_0, a_0), (s_1, a_1), (s_2, a_2), \ldots$

- Some of the most notable success stories:
  - Pomerleau, NIPS 1989: ALVINN
  - Sammut et al., ICML 1992: Learning to fly (flight sim)
  - Ross, Gordon, Bagnell 2011: DAgger

Inverse RL vs. Behavioral cloning

- **Which has the most succinct description: π^* vs. R^*?**

- Especially in planning oriented tasks, the reward function is often much more succinct than the optimal policy.

Lecture outline

- Example applications
- Inverse RL vs. behavioral cloning
- *Historical sketch of inverse RL*
- Mathematical formulations for inverse RL
- Case studies

Inverse RL history

- 1964, Kalman posed the inverse optimal control problem and solved it in the 1D input case
- 1994, Boyd+al.: a linear matrix inequality (LMI) characterization for the general linear quadratic setting
- 2000, Ng and Russell: first MDP formulation, reward function ambiguity pointed out and a few solutions suggested
- 2004, Abbeel and Ng: inverse RL for apprenticeship learning---reward feature matching
- 2006, Ratliff+al: max margin formulation

Inverse RL history

- 2007, Ratliff+al: max margin with boosting---enables large vocabulary of reward features
- 2007, Ramachandran and Amir [R&A], and Neu and Szepesvari: reward function as characterization of policy class
- 2008, Kolter, Abbeel and Ng: hierarchical max-margin
- 2008, Syed and Schapire: feature matching + game theoretic formulation
- 2008, Ziebart+al: feature matching + max entropy
- 2008, Abbeel+al: feature matching -- application to learning parking lot navigation style
- 2009, Baker, Saxe, Tenenbaum: same formulation as [R&A], investigation of understanding of human inverse planning inference
- 2009, Mombaur, Truong, Laumond: human path planning
- …

Lecture outline

- Example applications
- Inverse RL vs. behavioral cloning
- Historical sketch of inverse RL
- *Mathematical formulations for inverse RL*
- Case studies

Basic principle

- Find a reward function R* which explains the expert behavior.
- Find R* such that

$$\mathrm{E}[\sum_{t=0}^{\infty} \gamma^t R^*(s_t) | \pi^*] \geq \mathrm{E}[\sum_{t=0}^{\infty} \gamma^t R^*(s_t) | \pi] \quad \forall \pi$$

- In fact a convex feasibility problem, but many challenges:
  - R=0 is a solution, more generally: reward function ambiguity
  - We typically only observe expert traces rather than the entire expert policy π* --- how to compute left-hand side?
  - Assumes the expert is indeed optimal --- otherwise infeasible
  - Computationally: assumes we can enumerate all policies

Maxent Inverse RL

- Assume following noise model for demonstrator:

$$P(\tau) = \frac{1}{Z} e^{R(\tau)}$$

[Ziebart et al, 2008]

→ can run maximum likelihood (with regularization) to estimate R

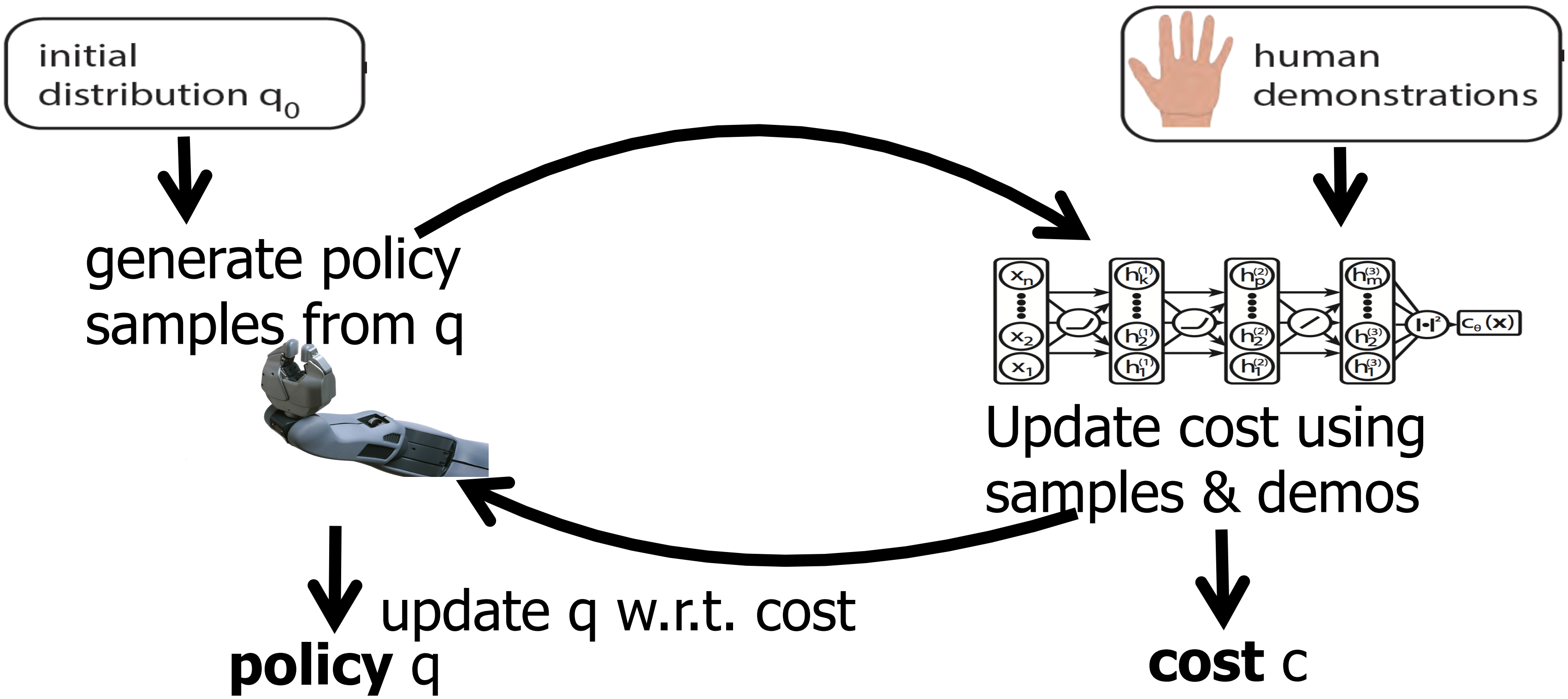
→ can have a deep net represent R

Estimating Z is tricky!

Ho et al, ICML 2016, Finn et all, ICML 2016

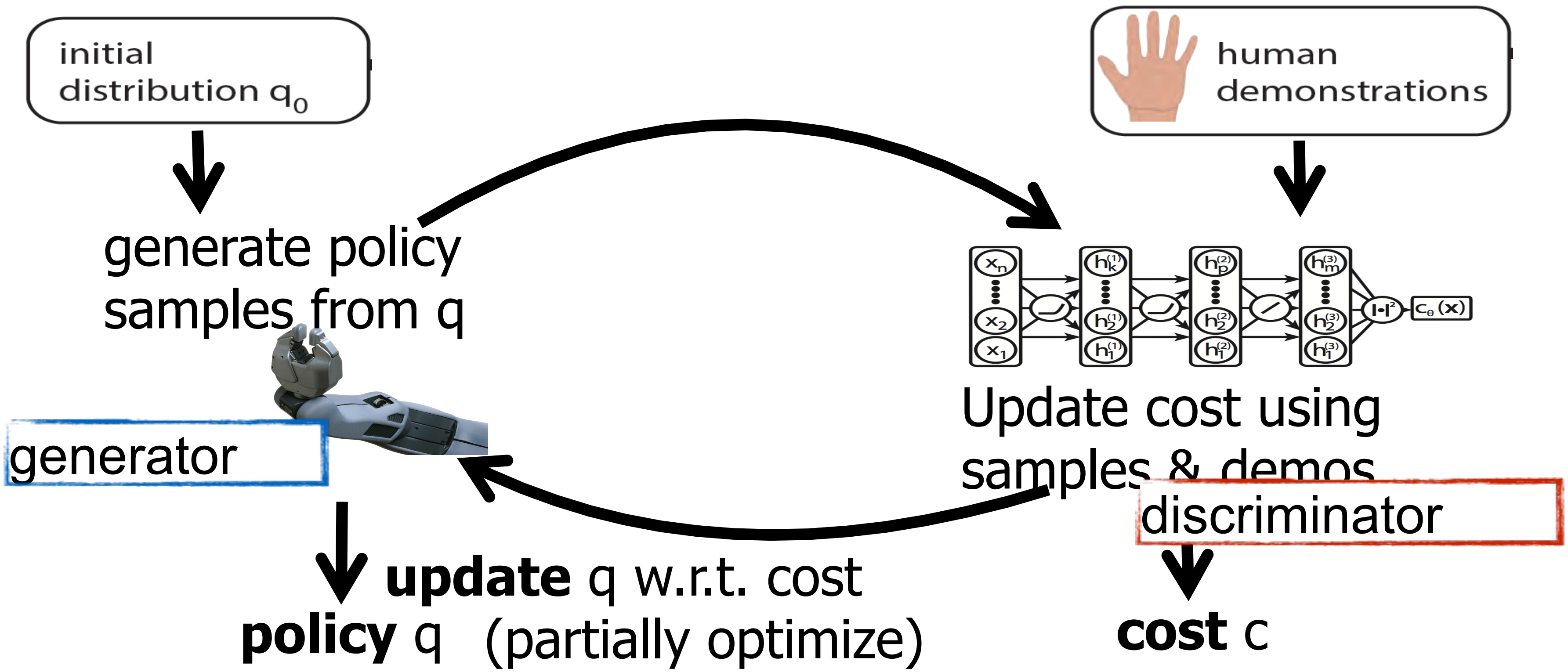
Guided Cost Learning

[Finn, Levine, Abbeel, ICML 2016]



Guided Cost Learning

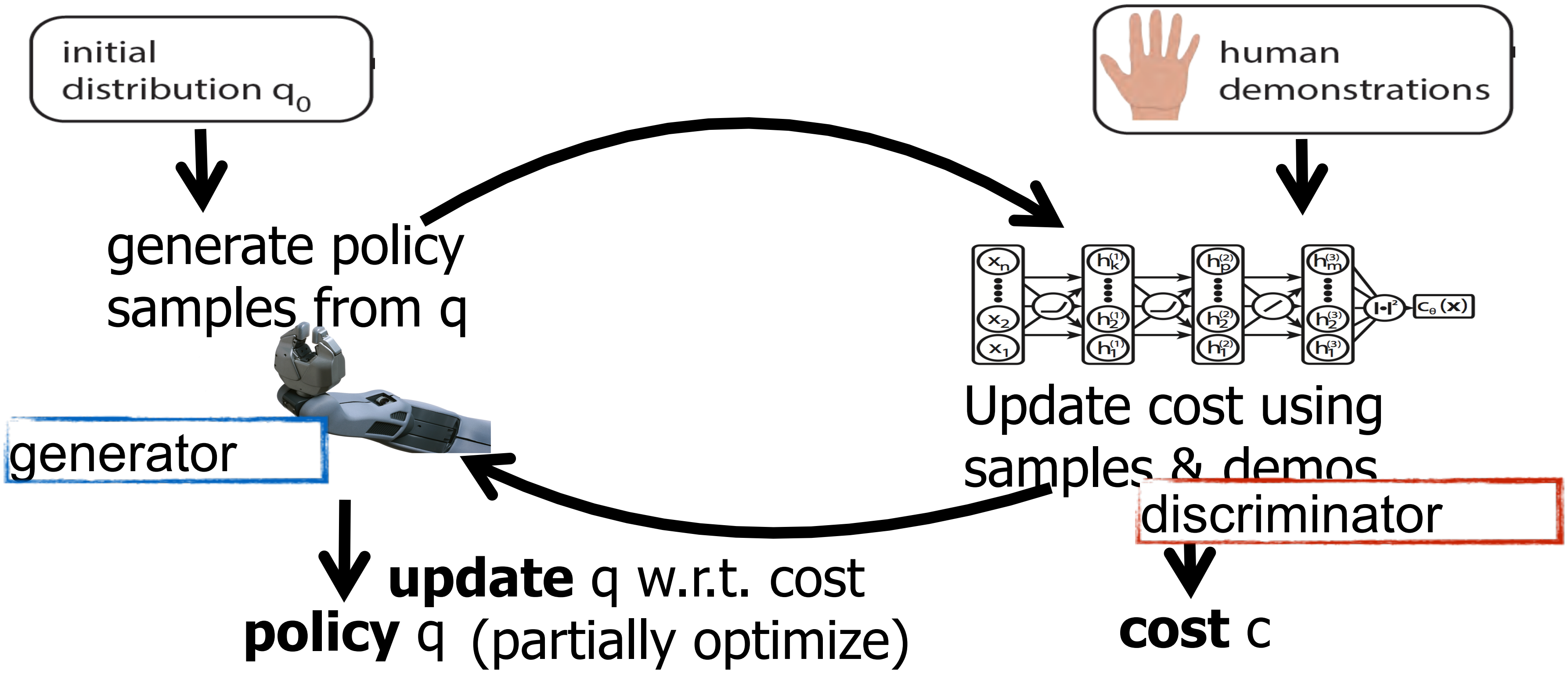
[Finn, Levine, Abbeel, ICML 2016]



update cost in inner loop of policy optimization

Guided Cost Learning

[Finn, Levine, Abbeel, ICML 2016]



Ho et al., ICML '16, arXiv '16

Kim & Bengio, arXiv '16

Inverse RL Summary

- Example applications
- Inverse RL vs. behavioral cloning
- Sketch of history of inverse RL
- Mathematical formulations for inverse RL

- Open directions: Active inverse RL, Inverse RL w.r.t. minmax control, partial observability, learning stage (rather than observing optimal policy), … ?

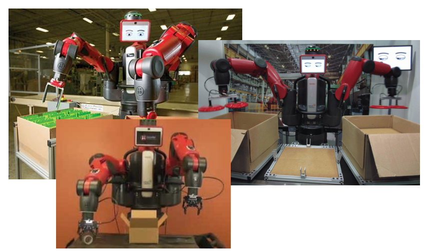
Outline

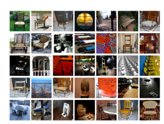
- Derivative free methods
  - Cross Entropy Method (CEM) / Finite Differences / Fixing Random Seed
- Likelihood Ratio (LR) Policy Gradient
  - Derivation / Connection w/Importance Sampling
- Natural Gradient / Trust Regions (-> TRPO)
- Actor-Critic (-> GAE, A3C)
- Path Derivatives (PD) (-> DPG, DDPG, SVG)
- Stochastic Computation Graphs (generalizes LR / PD)
- Guided Policy Search (GPS)
- Inverse Reinforcement Learning

Frontiers

- Shared and transfer learning

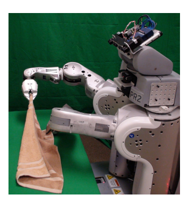






- Memory
  - Estimation
  - Temporal hierarchy / goal setting
- Exploration
- Model-based RL

- Applications



Thank you

Funding

- ONR
- Darpa
- NSF
- Berkeley AI Research Lab industrial affiliates program