Reinforcement Learning – Policy Optimization

Pieter Abbeel

UC Berkeley EECS

Policy Optimization

- 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 taking action u in state s

Learning to Trot/Run

Before learning (hand-tuned)

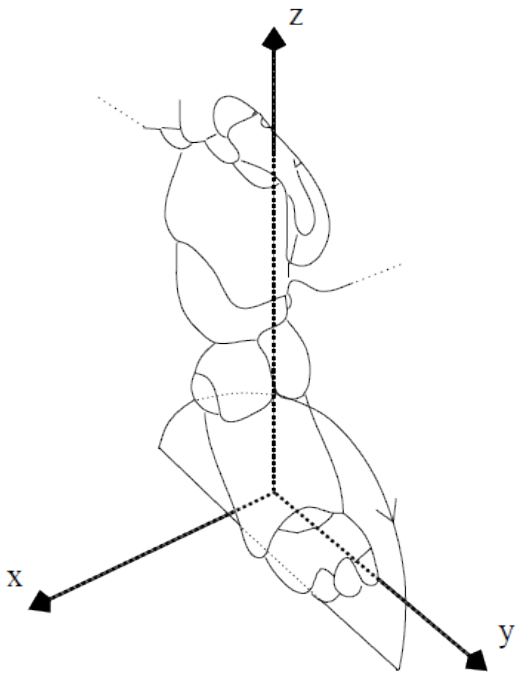
After learning

[Policy search was done through trials on the actual robot.]

Kohl and Stone, ICRA2004

Learning to Trot/Run

- 12 parameters define the Aibo's gait:
  - The front locus (3 parameters: height, x-pos., y-pos.)
  - The rear locus (3 parameters)
  - Locus length
  - Locus skew multiplier in the x-y plane (for turning)
  - The height of the front of the body
  - The height of the rear of the body
  - The time each foot takes to move through its locus
  - The fraction of time each foot spends on the ground



Kohl and Stone, ICRA2004

[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).

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

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

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

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

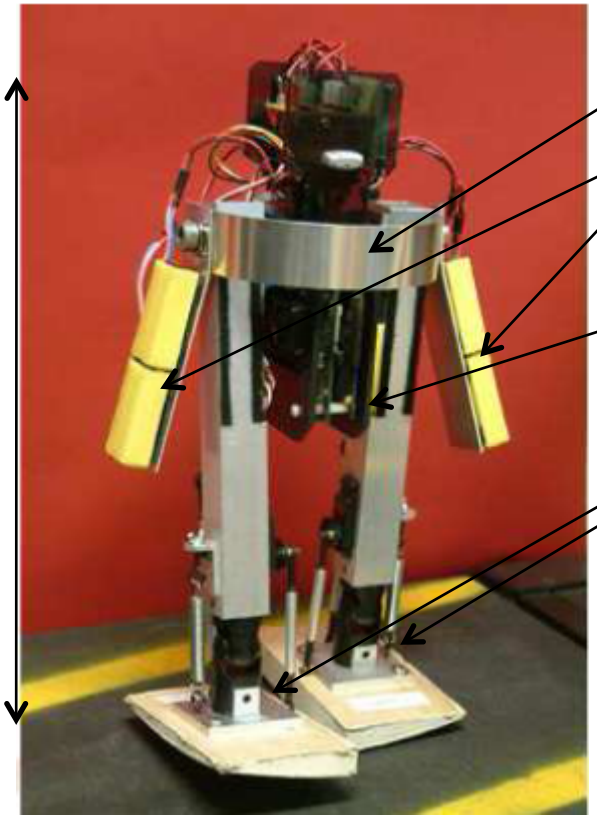
Ball-In-A-Cup

[Kober and Peters, 2009]

Learning to Walk in 20 Minutes

[Tedrake, Zhang, Seung 2005]

Learning to Walk in 20 Minutes



passive hip joint [1DOF]

Arms: coupled to opposite leg to reduce yaw moment

freely swinging load [1DOF]

2 x 2 (roll, pitch) position controlled servo motors [4 DOF]

9DOFs:
* 6 internal DOFs
* 3 DOFs for the robot's orientation (always assumed in contact with ground at a single point, absolute (x,y) ignored)

[Tedrake, Zhang, Seung 2005]

Natural gait down 0.03 radians ramp: 0.8Hz, 6.5cm steps

Learning to Walk in 20 Minutes

[Tedrake, Zhang, Seung 2005]

Gradient-Free Methods

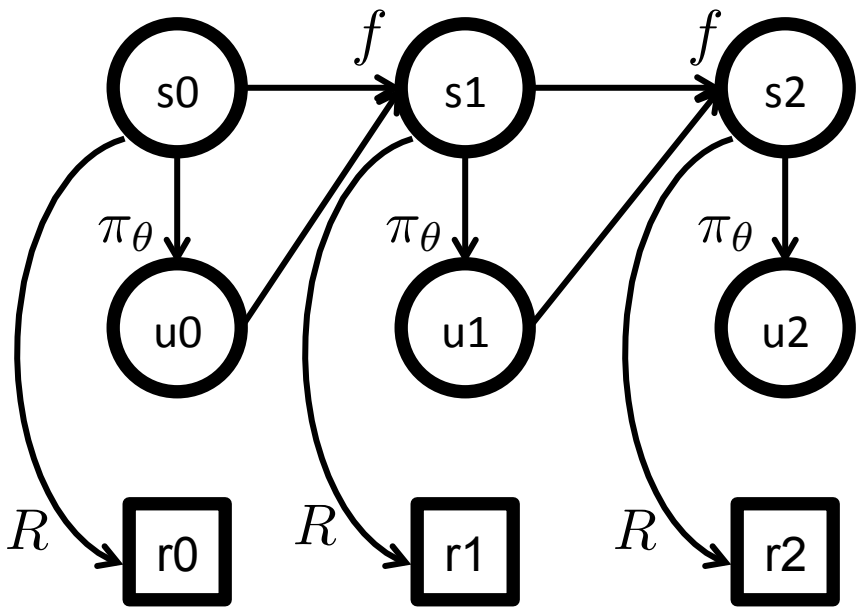
$$\max_{\theta} \; U(\theta) = \max_{\theta} \; \mathrm{E}[\sum_{t=0}^{H} R(s_t)|\pi_\theta]$$

- Cross-Entropy Method (CEM)
- Covariance Matrix Adaptation (CMA)
  - Dynamics model: stochastic: OK; unknown: OK
  - Policy class: stochastic: OK
  - Downside: gradient-free methods slower than gradient-based methods

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

Gradient-Based Policy Optimization

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



Overview of Methods / Settings

	Dynamics					**Policy**		
	D+K	**D+U**	**S+K+R**	**S+K**	**S+U**	**D**	**S+R**	**S**
PD	+		+			+	+	
LR	+	+	+	+	+		+	+

D: deterministic; S: stochastic; K: known; U: unknown; R: reparameterizable;
PD: path derivative (=perturbation analysis)
LR: likelihood ratio (=score function)

Questions

- When more than one is applicable, which one is best?
- When dynamics is only available as black-box, but derivatives aren't available – finite differences based derivatives?
  - Vs. directly finite differences / gradient-free on the policy
  - Note: finite difference tricky (impractical?) when can't control random seed
- What if model is unknown, but estimate available

Gradient Computation – Unknown Model – Finite Differences

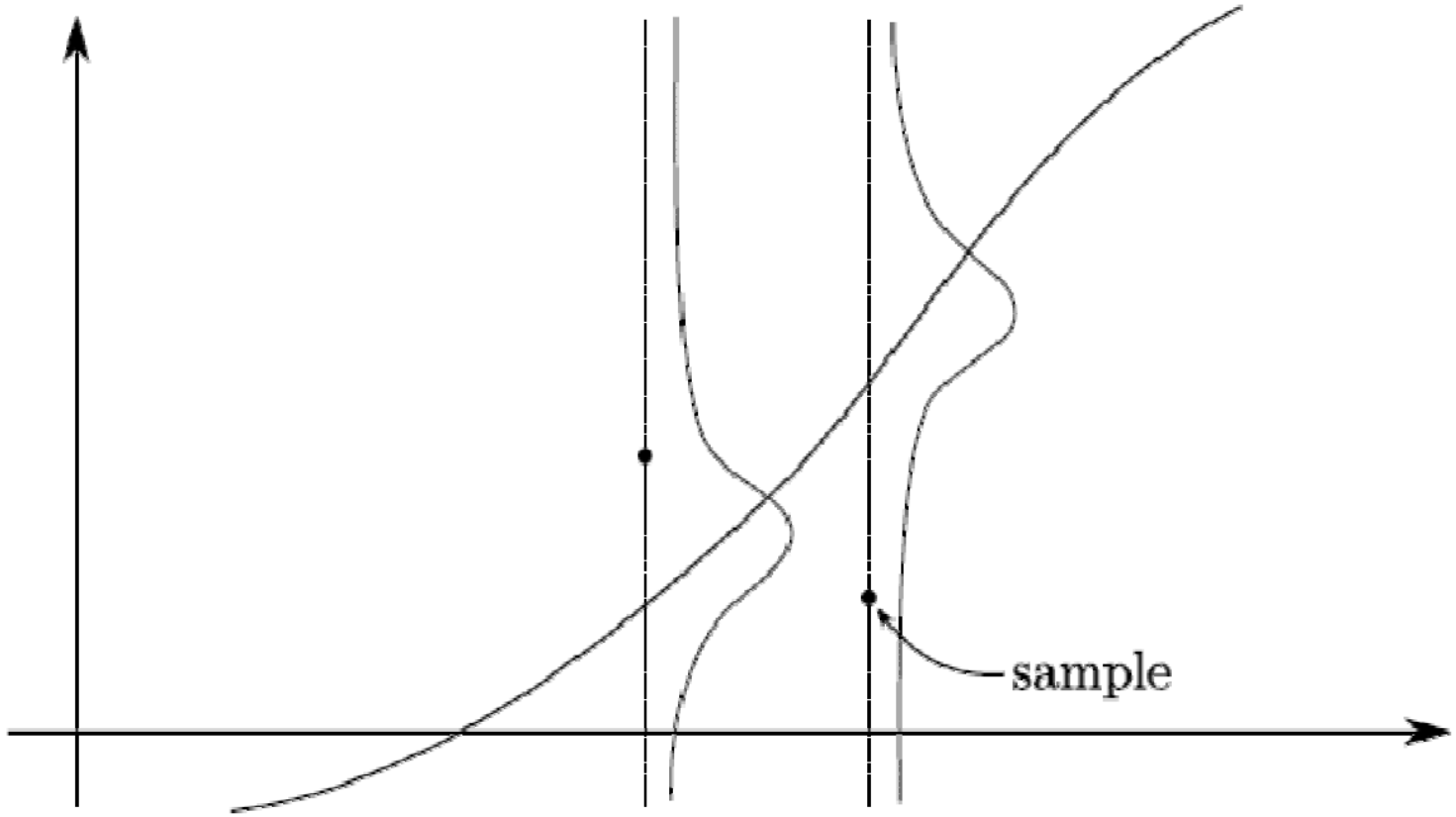
We can compute the gradient g using standard finite difference methods, as follows:

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

Where:

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

Noise Can Dominate



Finite Differences and Noise

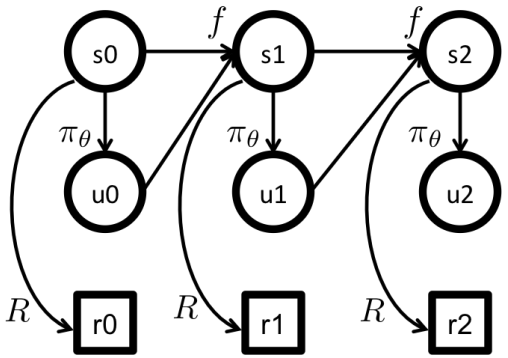
- Solution 1: Average over many samples

- Solution 2: Fix the randomness (if possible)
  - Intuition by 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
  - General instantiation: Fix the random seed; and the result is deterministic system
  - Ng & Jordan, 2000 provide theoretical analysis of gains from fixing randomness

Path Derivative for Dynamics: D+K; Policy: D

- 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:

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

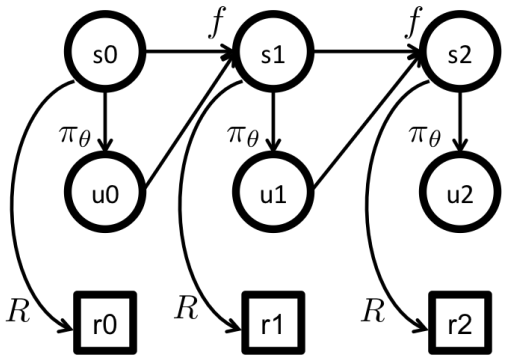
$$\frac{\partial s_t}{\partial \theta_i} = \frac{\partial f}{\partial s}(s_{t-1}, u_{t-1})\frac{\partial s_{t-1}}{\partial \theta_i} + \frac{\partial f}{\partial s}(s_{t-1}, u_{t-1})\frac{\partial u_{t-1}}{\partial \theta_i}$$

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

Path Derivative for Dynamics: S+K+R; Policy: S+R

- Reminder of optimization objective:

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



- (draw reparameterized graph on board)
- + average over multiple samples

Overview of Methods / Settings

	Dynamics					**Policy**		
	D+K	**D+U**	**S+K+R**	**S+K**	**S+U**	**D**	**S+R**	**S**
PD	+		+			+	+	
LR	+	+	+	+	+		+	+

D: deterministic; S: stochastic; K: known; U: unknown; R: reparameterizable;
PD: path derivative (=perturbation analysis)
LR: likelihood ratio (=score function)

Gradient Computation – Unknown Model – Likelihood Ratio

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 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:

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

[Note: Can also be derived/generalized through an importance sampling derivation – Tang and Abbeel, 2011]

Importance Sampling

- On board..

Likelihood Ratio Gradient Estimate

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

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

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

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

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 with Baseline

- Gradient estimate with baseline:

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

- Crudely, increasing log-likelihood of paths with higher than baseline reward and decreasing log-likelihood of paths with lower than baseline reward
- Still unbiased? Yes!

$$\mathrm{E}\left[\frac{1}{m} \sum_{i=1}^{m} \nabla_\theta \log P(\tau^{(i)}; \theta) b\right] = 0$$

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:

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

Step-sizing and Trust Regions

- Naïve step-sizing: Line search
  - Step-sizing necessary as gradient is only first-order approximation
  - Line search in the 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:

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

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

KL Trust Region (a.k.a. natural gradient)

- Solve for best point within trust region:

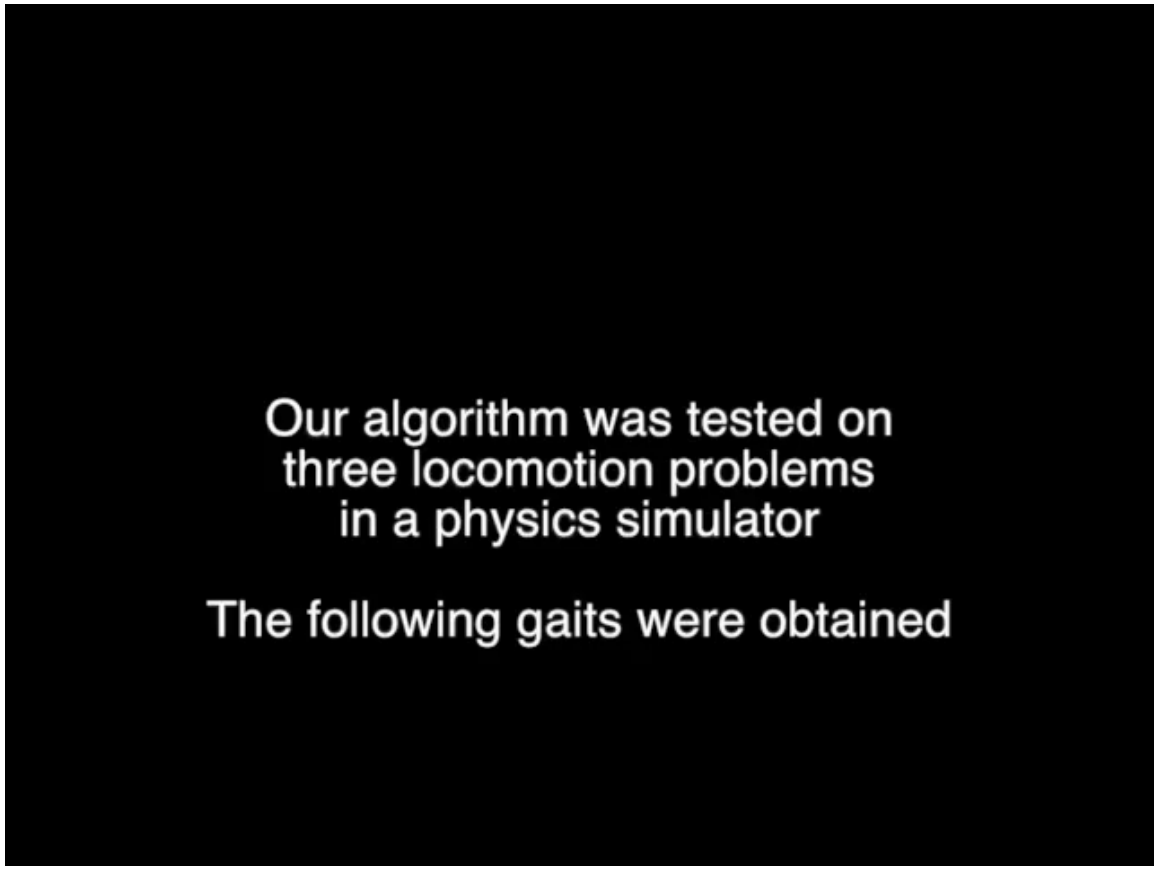
$$\max_{\delta\theta} \quad \hat{g}^\top \delta\theta$$
$$\text{s.t.} \quad KL(P(\tau;\theta)||P(\tau;\theta+\delta\theta)) \leq \varepsilon$$

- KL can be approximated efficiently with 2nd order expansion:

$$\begin{aligned} KL(P(\tau;\theta)||P(\tau;\theta+\delta\theta)) &\approx \sum_\tau P(\tau;\theta)\delta\theta^\top \nabla_\theta \log P(\tau;\theta) \nabla_\theta \log P(\tau;\theta)^\top \delta\theta \\ &= \delta\theta^\top \left(\sum_\tau P(\tau;\theta) \nabla_\theta \log P(\tau;\theta) \nabla_\theta \log P(\tau;\theta)^\top \right) \delta\theta \\ &= \delta\theta^\top G(\theta) \delta\theta \end{aligned}$$

G: Fisher Information Matrix

Experiments in Locomotion



[Schulman, Levine, Abbeel, 2014]

Actor-Critic Variant

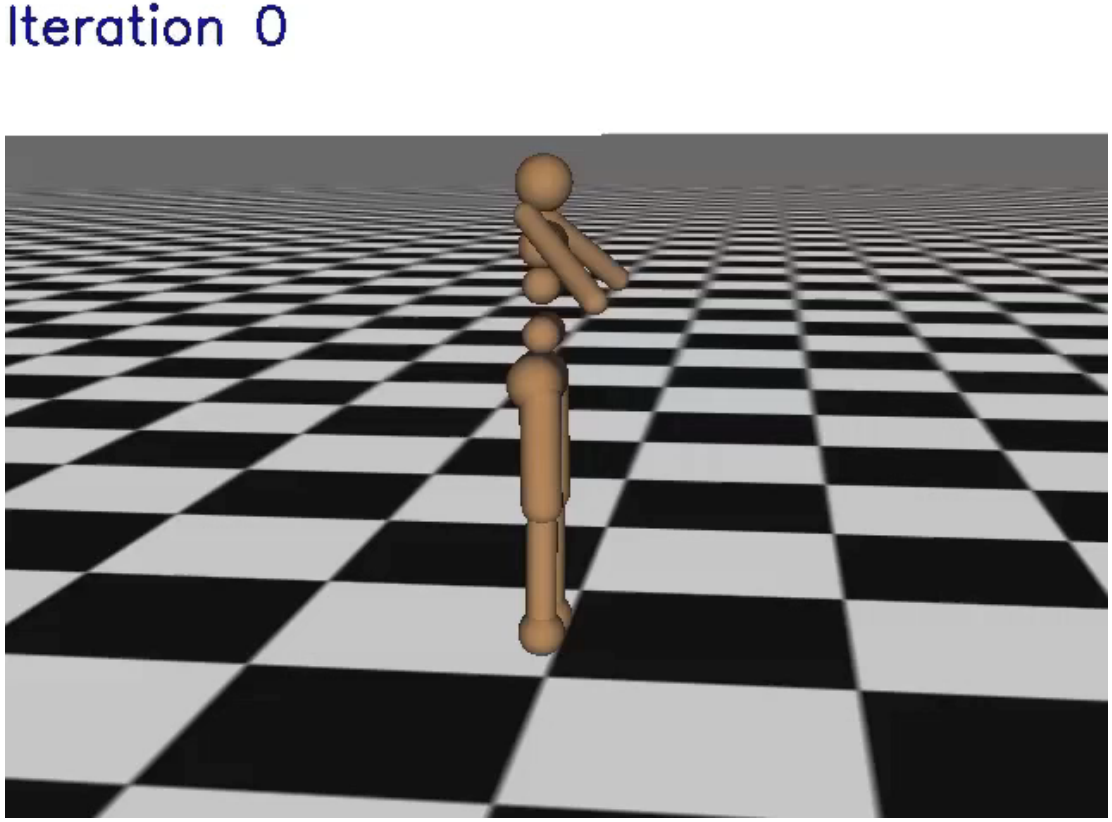
- Current estimate:

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

Sample based estimate of $Q(s_k^{(i)}, u_k^{(i)})$

- Actor-critic algorithms in parallel run an estimator for the Q-function, and substitute in the estimated Q value

Learning Locomotion



[Schulman, Moritz, Levine, Jordan, Abbeel, 2015]

In Contrast: Darpa Robotics Challenge

Thank you