Deep RL Foundations in 6 Lectures

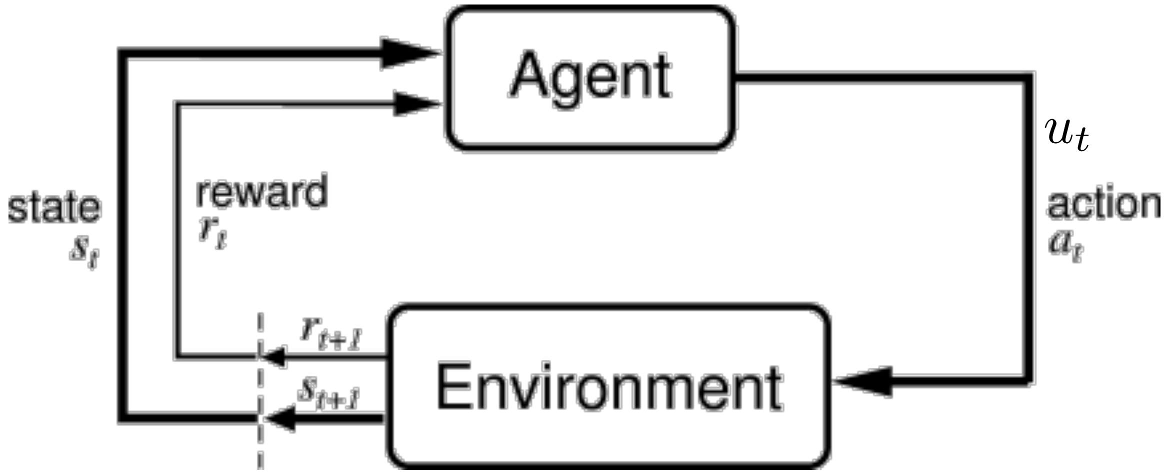
Lecture 4: TRPO, PPO

Pieter Abbeel

Lecture Series

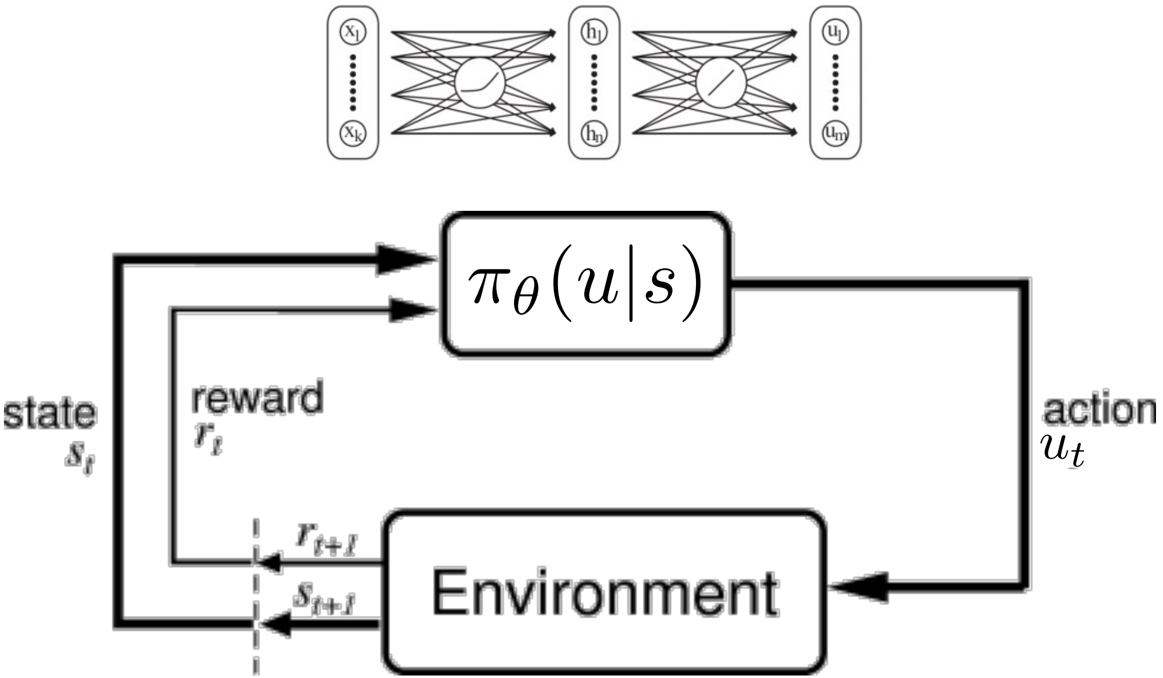
- Lecture 1: MDPs Foundations and Exact Solution Methods
- Lecture 2: Deep Q-Learning
- Lecture 3: Policy Gradients, Advantage Estimation
- ***<u>Lecture 4: TRPO, PPO</u>***
- Lecture 5: DDPG, SAC
- Lecture 6: Model-based RL

Reinforcement Learning



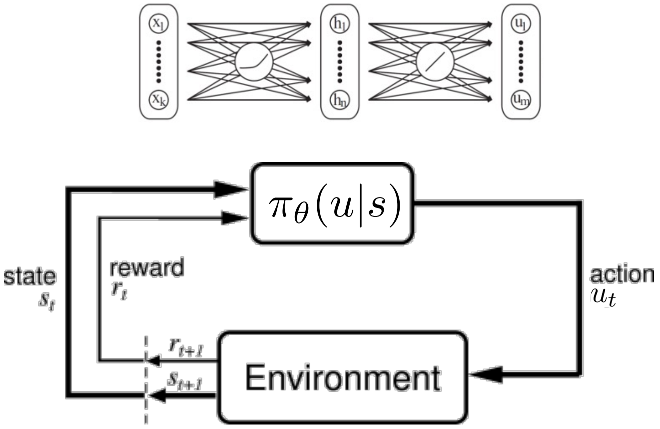
[Figure source: Sutton & Barto, 1998]

Policy Optimization



[Figure source: Sutton & Barto, 1998]

Policy Optimization



- 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

[Figure source: Sutton & Barto, 1998]

Vanilla Policy Gradient

Algorithm 1 "Vanilla" policy gradient algorithm

Initialize policy parameter θ, baseline b
for iteration$=1, 2, \dots$ **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 This Lecture

- Surrogate loss
- Step-sizing and Trust Region Policy Optimization (TRPO)
- Proximal Policy Optimization (PPO)

Outline for This Lecture

- ***Surrogate loss***
- Step-sizing and 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

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

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

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

[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

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

→ Surrogate Loss

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

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

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

[Tang&Abbeel, NeurIPS 2011]

Outline for This Lecture

- Surrogate loss
- ***Step-sizing and 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

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

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

Evaluating the KL

- 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

- 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

- 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:

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

dynamics cancels out! ☺

Evaluating the KL

- 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:

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

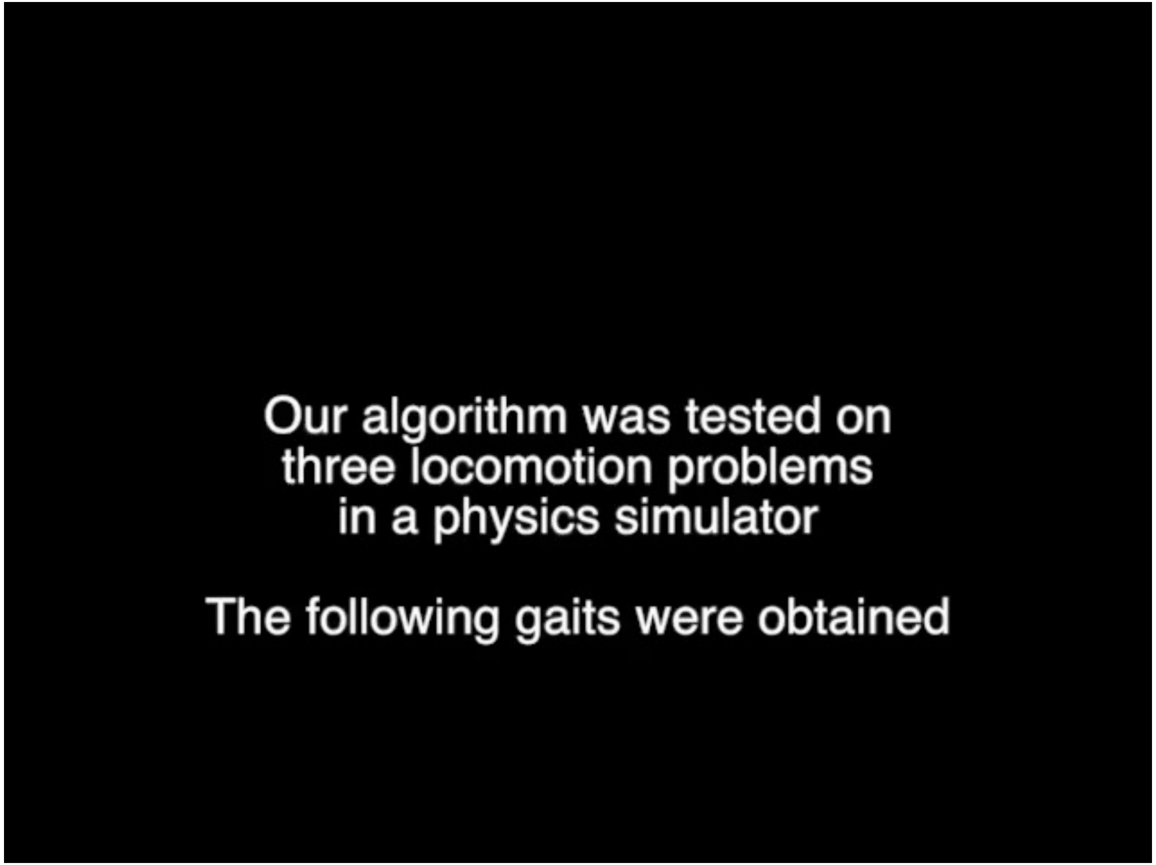
$$= \sum_\tau P(\tau;\theta) \log \frac{P(s_0) \prod_{t=0}^{H-1} \pi_\theta(u_t|s_t) P(s_{t+1}|s_t, u_t)}{P(s_0) \prod_{t=0}^{H-1} \pi_{\theta+\delta\theta}(u_t|s_t) P(s_{t+1}|s_t, u_t)}$$

dynamics cancels out! ☺

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

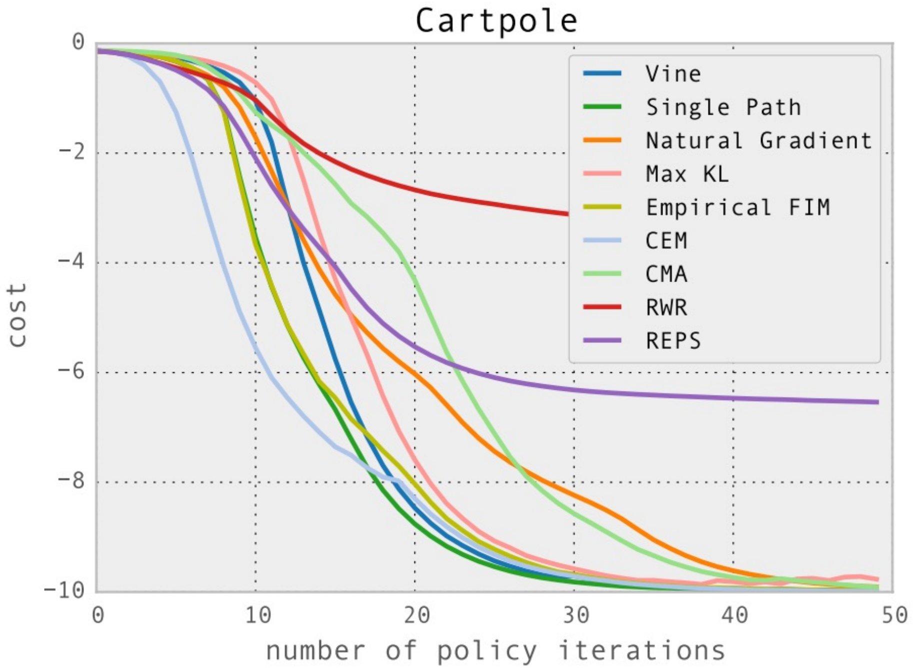
$$\approx \frac{1}{M} \sum_{(s,u) \text{ in roll-outs under } \theta} \log \frac{\pi_\theta(u|s)}{\pi_{\theta+\delta\theta}(u|s)}$$

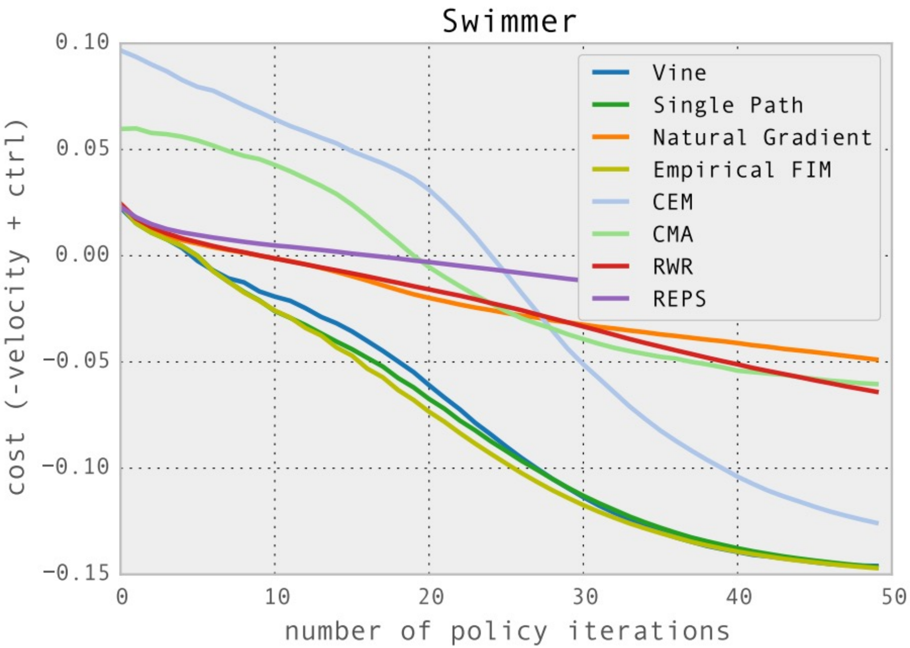
Experiments in Locomotion



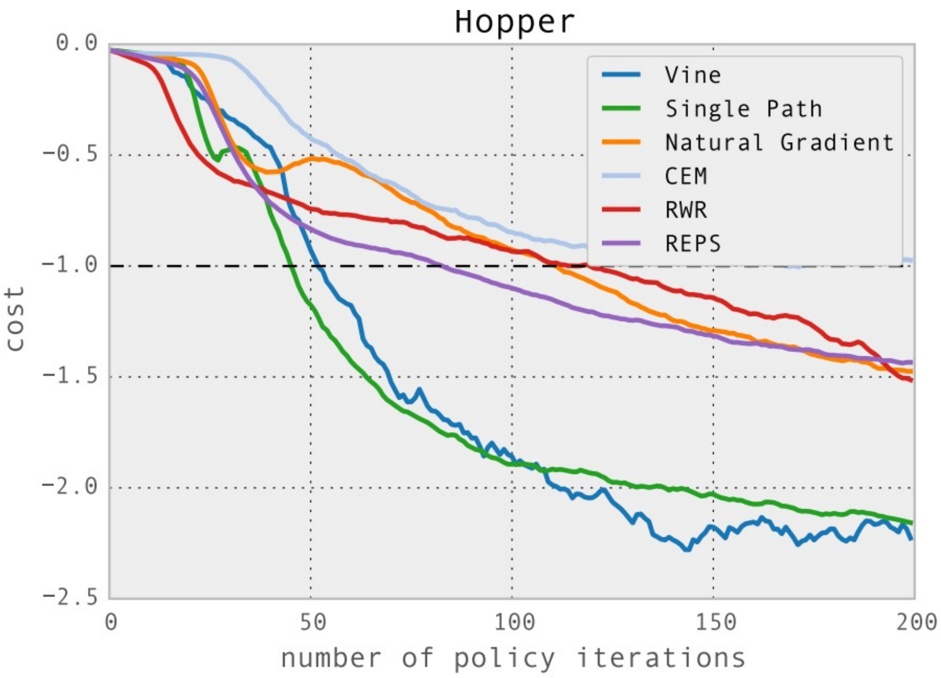
[Schulman, Levine, Moritz, Jordan, Abbeel, 2014]

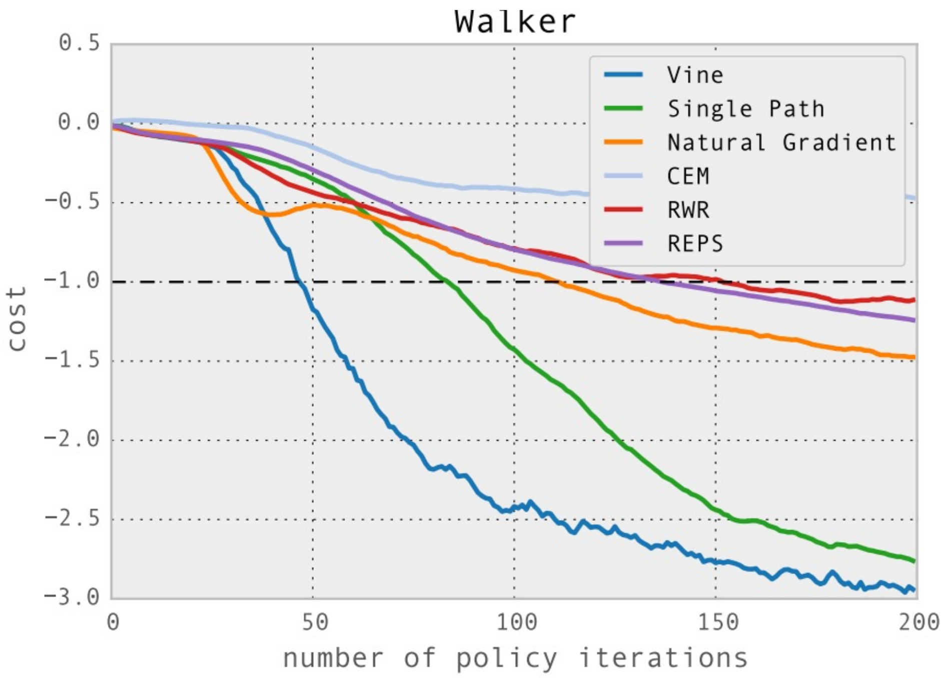
Learning Curves -- Comparison



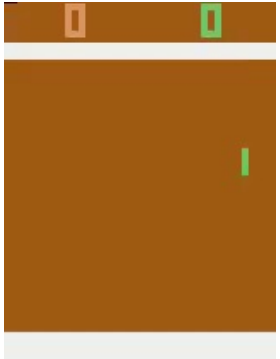


Learning Curves -- Comparison

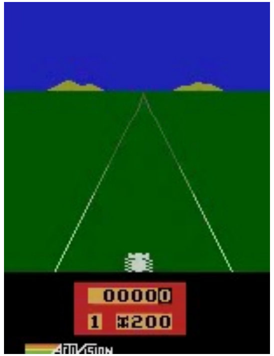




Atari Games

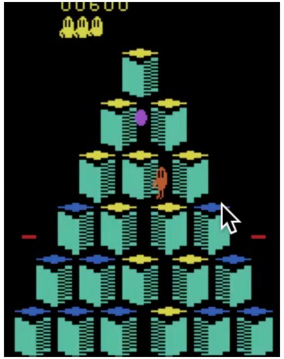


Pong



Enduro

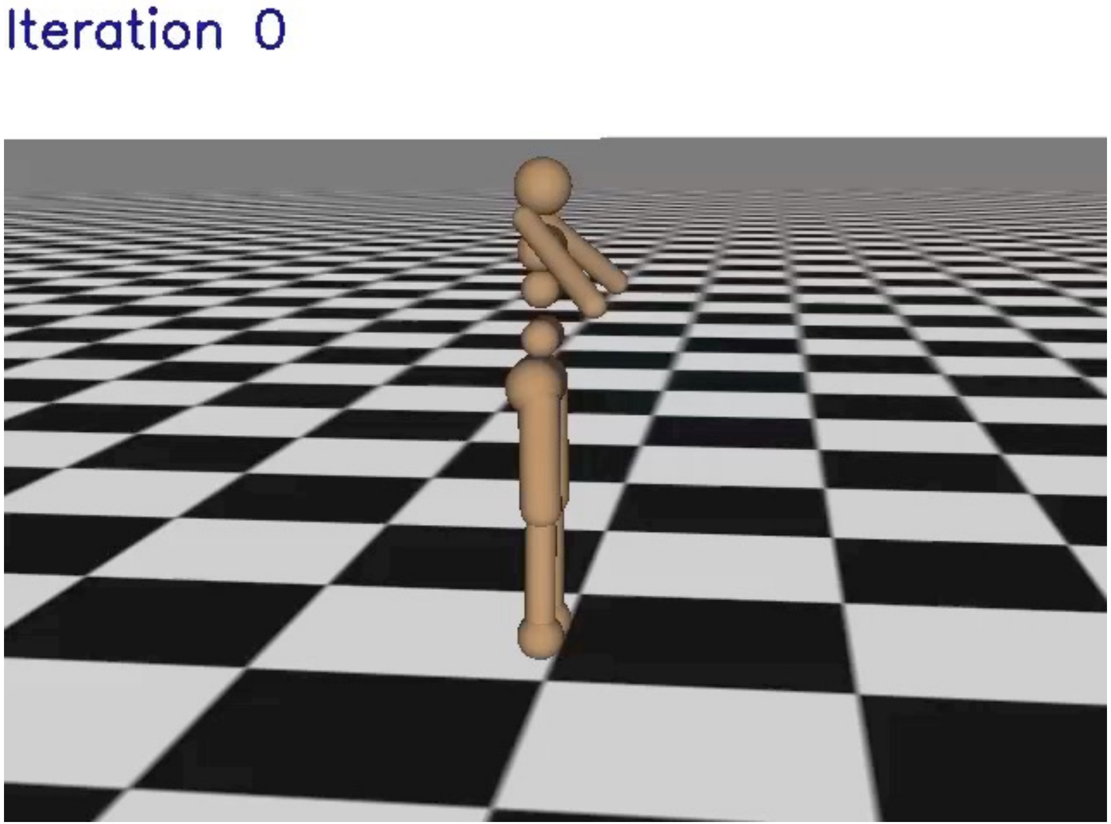
Beamrider



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]
- ...

Learning Locomotion (TRPO + GAE)



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

Outline for This Lecture

- Surrogate loss
- Step-sizing and 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?

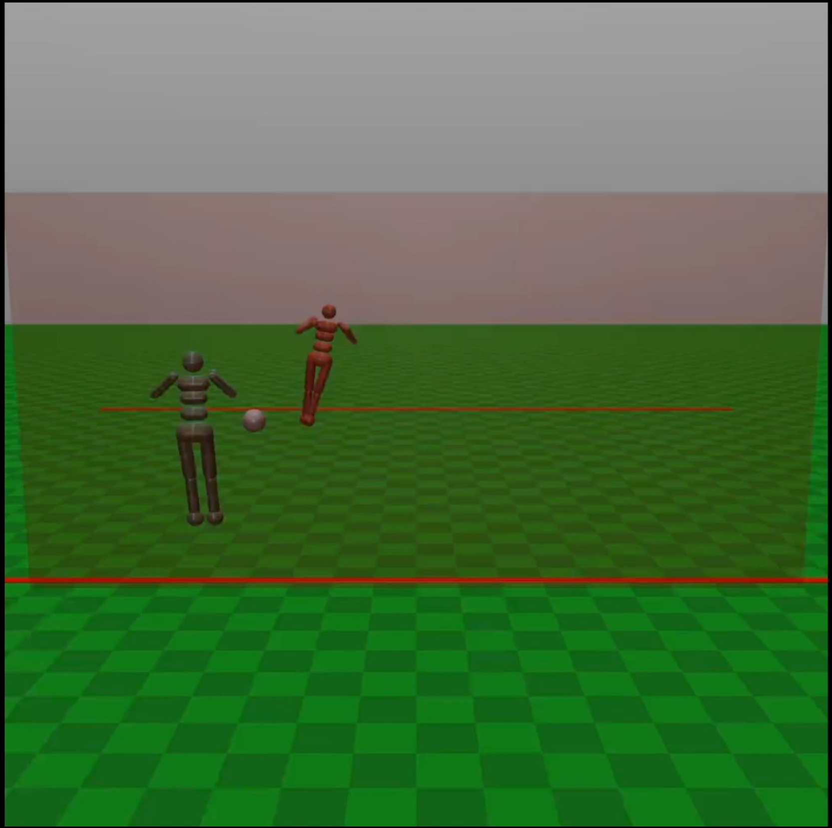
Proximal Policy Optimization V2 – “Clipped Surrogate Loss”

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

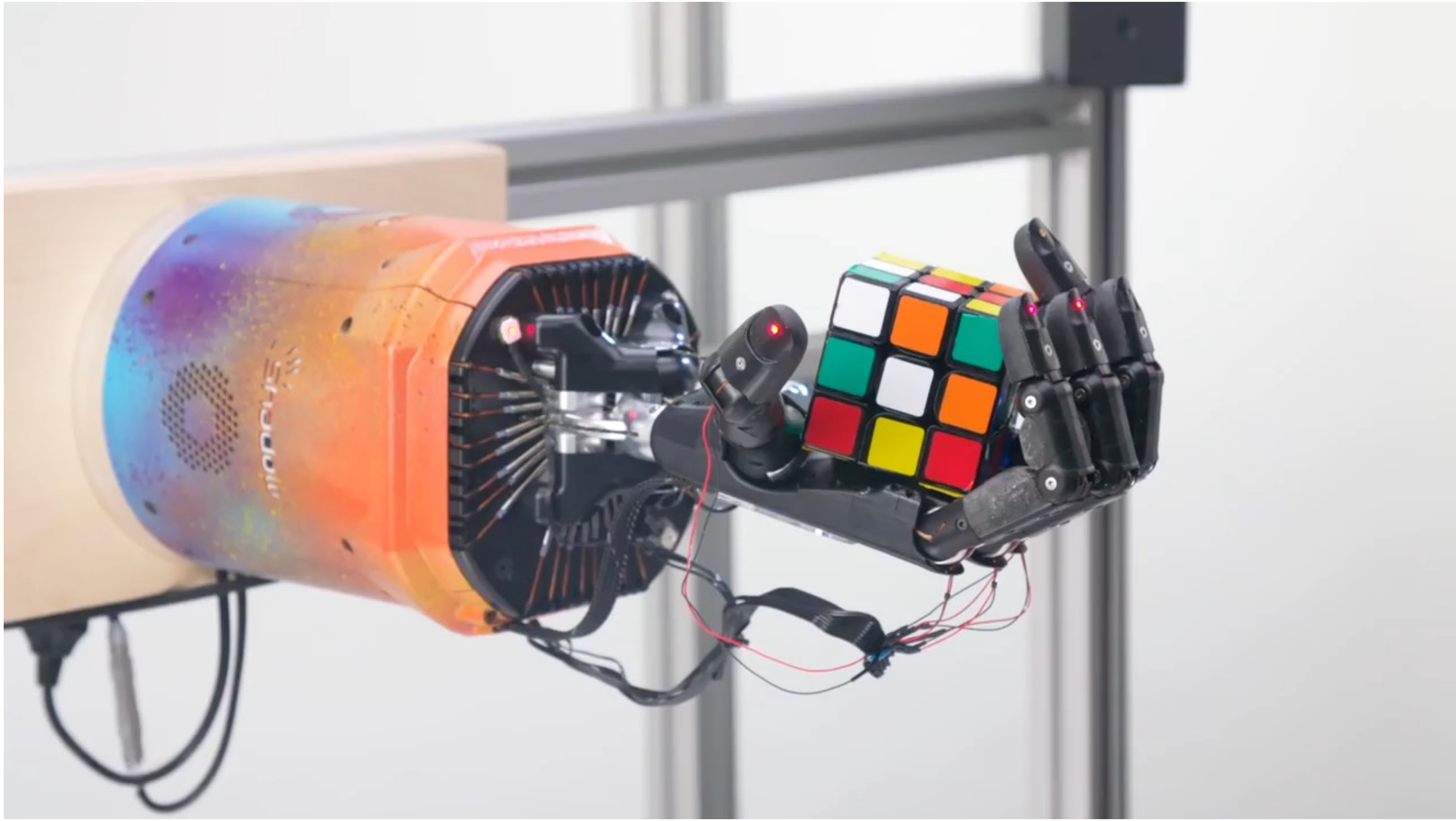


[Bansal et al, 2017]

OpenAI-5 was trained with PPO

OpenAI In-Hand Re-Orientation

OpenAI Rubik's Cube



Summary of This Lecture

- Surrogate loss
- Step-sizing and Trust Region Policy Optimization (TRPO)
- Proximal Policy Optimization (PPO)