# Deep Reinforcement Learning

John Schulman

OpenAI Berkeley[1]

MLSS, May 2016, Cadiz

[1] Berkeley Artificial Intelligence Research Lab

# Agenda

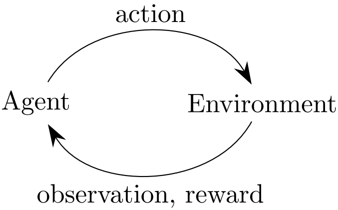


Course materials: `goo.gl/5wsgbJ`

# Introduction and Overview

# What is Reinforcement Learning?

- Branch of machine learning concerned with taking sequences of actions
- Usually described in terms of agent interacting with a previously unknown environment, trying to maximize cumulative reward



# Motor Control and Robotics

Robotics:

- Observations: camera images, joint angles
- Actions: joint torques
- Rewards: stay balanced, navigate to target locations, serve and protect humans

# Business Operations

- Inventory Management
  - Observations: current inventory levels
  - Actions: number of units of each item to purchase
  - Rewards: profit
- Resource allocation: who to provide customer service to first
- Routing problems: in management of shipping fleet, which trucks / truckers to assign to which cargo

# Games

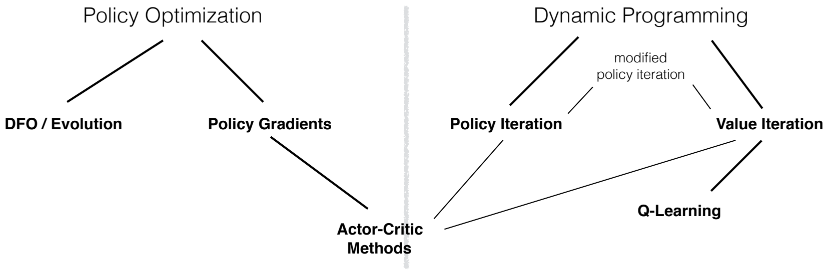
A different kind of optimization problem (min-max) but still considered to be RL.

- Go (complete information, deterministic) – *AlphaGo*[2]
- Backgammon (complete information, stochastic) – *TD-Gammon*[3]
- Stratego (incomplete information, deterministic)
- Poker (incomplete information, stochastic)

[2] David Silver, Aja Huang, et al. "Mastering the game of Go with deep neural networks and tree search". In: *Nature* 529.7587 (2016), pp. 484–489.

[3] Gerald Tesauro. "Temporal difference learning and TD-Gammon". In: *Communications of the ACM* 38.3 (1995), pp. 58–68.

# Approaches to RL

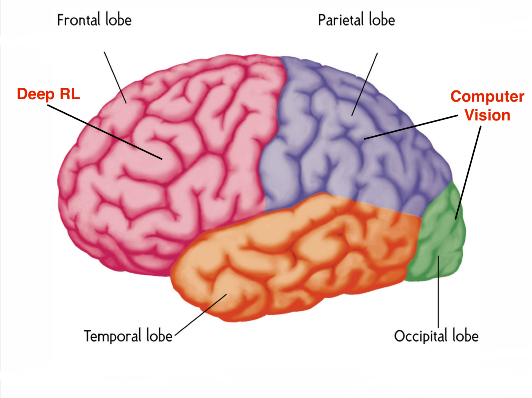


# What is Deep RL?

- RL using nonlinear function approximators
- Usually, updating parameters with stochastic gradient descent

# What's Deep RL?

Whatever the front half of the cerebral cortex does (motor and executive cortices)



# Markov Decision Processes

# Definition

- Markov Decision Process (MDP) defined by $(\mathcal{S}, \mathcal{A}, P)$, where
  - $\mathcal{S}$: **state space**
  - $\mathcal{A}$: **action space**
  - $P(r, s' \mid s, a)$: a transition probability distribution
- Extra objects defined depending on problem setting
  - $\mu$: Initial state distribution
  - $\gamma$: discount factor

# Episodic Setting

- In each episode, the initial state is sampled from $\mu$, and the process proceeds until the *terminal state* is reached. For example:
  - Taxi robot reaches its destination (termination $=$ good)
  - Waiter robot finishes a shift (fixed time)
  - Walking robot falls over (termination $=$ bad)
- Goal: maximize expected reward per episode

# Policies

- Deterministic policies: $a = \pi(s)$
- Stochastic policies: $a \sim \pi(a \mid s)$

- Parameterized policies: $\pi_\theta$

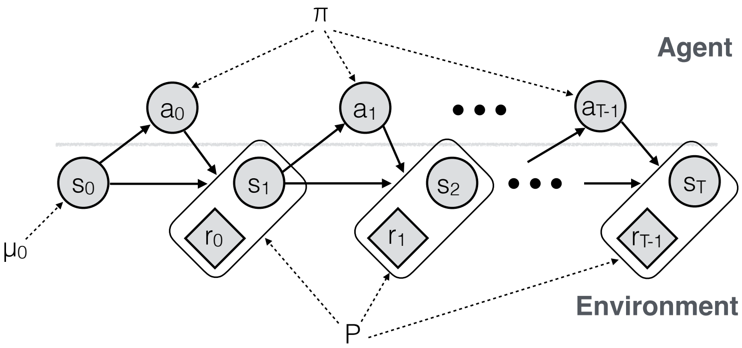
# Episodic Setting

$$
\begin{aligned}
s_0 &\sim \mu(s_0) \\
a_0 &\sim \pi(a_0 \mid s_0) \\
s_1, r_0 &\sim P(s_1, r_0 \mid s_0, a_0) \\
a_1 &\sim \pi(a_1 \mid s_1) \\
s_2, r_1 &\sim P(s_2, r_1 \mid s_1, a_1) \\
&\dots \\
a_{T-1} &\sim \pi(a_{T-1} \mid s_{T-1}) \\
s_T, r_{T-1} &\sim P(s_T \mid s_{T-1}, a_{T-1})
\end{aligned}
$$

Objective:

$$
\begin{aligned}
&\text{maximize}\, \eta(\pi), \;\; \text{where} \\
&\qquad \eta(\pi) = E[r_0 + r_1 + \cdots + r_{T-1} \mid \pi]
\end{aligned}
$$

# Episodic Setting




Objective:

$$\text{maximize}\, \eta(\pi), \text{ where}$$
$$\eta(\pi) = E[r_0 + r_1 + \cdots + r_{T-1} \mid \pi]$$

# Parameterized Policies

- A family of policies indexed by parameter vector $\theta \in \mathbb{R}^d$
  - Deterministic: $a = \pi(s, \theta)$
  - Stochastic: $\pi(a \mid s, \theta)$
- Analogous to classification or regression with input $s$, output $a$. E.g. for neural network stochastic policies:
  - Discrete action space: network outputs vector of probabilities
  - Continuous action space: network outputs mean and diagonal covariance of Gaussian

# Reinforcement Learning via Black-Box Optimization

# Derivative Free Optimization Approach

- Objective:

$$\text{maximize}\, E[R \mid \pi(\cdot, \theta)]$$

- View $\theta \to \blacksquare \to R$ as a black box
- Ignore all other information other than $R$ collected during episode

# Cross-Entropy Method

- Evolutionary algorithm
- 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**

**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*

Victor Gabillon, Mohammad Ghavamzadeh, and Bruno Scherrer. "Approximate Dynamic Programming Finally Performs Well in the Game of Tetris". In: *Advances in Neural Information Processing Systems*. 2013

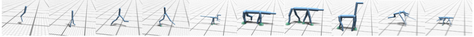
# Cross-Entropy Method

- Evolutionary algorithm
- Works **embarrassingly** well
- A similar algorithm, Covariance Matrix Adaptation, has become standard in graphics:

**Optimal Gait and Form for Animal Locomotion**

Kevin Wampler* Zoran Popović

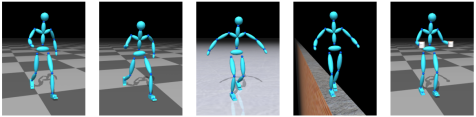
University of Washington



**Optimizing Walking Controllers for Uncertain Inputs and Environments**

Jack M. Wang David J. Fleet Aaron Hertzmann

University of Toronto



# Cross-Entropy Method

Initialize $\mu \in \mathbb{R}^d, \sigma \in \mathbb{R}^d$
**for** iteration $= 1, 2, \ldots$ **do**
Collect n samples of $\theta_i \sim N(\mu, \text{diag}(\sigma))$
Perform a noisy evaluation $R_i \sim \theta_i$
Select the top $p$% of samples (e.g. $p = 20$), which we'll call the **elite set**
Fit a Gaussian distribution, with diagonal covariance, to the elite set, obtaining a new $\mu, \sigma$.
**end for**
Return the final $\mu$.

# Cross-Entropy Method

- Analysis: a very similar algorithm is an minorization-maximization (MM) algorithm, guaranteed to monotonically increase expected reward
- Recall that Monte-Carlo EM algorithm collects samples, reweights them, and them maximizes their logprob
- We can derive MM algorithm where each iteration you maximize $\sum_i \log p(\theta_i) R_i$

# Policy Gradient Methods

# Policy Gradient Methods: Overview

Problem:

$$\text{maximize } E[R \mid \pi_\theta]$$

Intuitions: collect a bunch of trajectories, and ...

1. Make the good trajectories more probable
2. Make the good actions more probable (actor-critic, GAE)
3. Push the actions towards good actions (DPG, SVG)

# Score Function Gradient Estimator

- Consider an expectation $E_{x \sim p(x \mid \theta)}[f(x)]$. Want to compute gradient wrt $\theta$

$$
\begin{aligned}
\nabla_\theta E_x[f(x)] &= \nabla_\theta \int \mathrm{d}x \; p(x \mid \theta) f(x) \\
&= \int \mathrm{d}x \; \nabla_\theta p(x \mid \theta) f(x) \\
&= \int \mathrm{d}x \; p(x \mid \theta) \frac{\nabla_\theta p(x \mid \theta)}{p(x \mid \theta)} f(x) \\
&= \int \mathrm{d}x \; p(x \mid \theta) \nabla_\theta \log p(x \mid \theta) f(x) \\
&= E_x[f(x) \nabla_\theta \log p(x \mid \theta)].
\end{aligned}
$$

- Last expression gives us an unbiased gradient estimator. Just sample $x_i \sim p(x \mid \theta)$, and compute $\hat{g}_i = f(x_i) \nabla_\theta \log p(x_i \mid \theta)$.
- Need to be able to compute and differentiate density $p(x \mid \theta)$ wrt $\theta$

# Derivation via Importance Sampling

Alternate Derivation Using Importance Sampling

$$
\begin{aligned}
\mathbb{E}_{x\sim\theta}\left[f(x)\right] &= \mathbb{E}_{x\sim\theta_{\mathrm{old}}}\left[\frac{p(x \mid \theta)}{p(x \mid \theta_{\mathrm{old}})} f(x)\right] \\
\nabla_\theta \mathbb{E}_{x\sim\theta}\left[f(x)\right] &= \mathbb{E}_{x\sim\theta_{\mathrm{old}}}\left[\frac{\nabla_\theta p(x \mid \theta)}{p(x \mid \theta_{\mathrm{old}})} f(x)\right] \\
\nabla_\theta \mathbb{E}_{x\sim\theta}\left[f(x)\right]\Big|_{\theta=\theta_{\mathrm{old}}} &= \mathbb{E}_{x\sim\theta_{\mathrm{old}}}\left[\frac{\nabla_\theta p(x \mid \theta)\big|_{\theta=\theta_{\mathrm{old}}}}{p(x \mid \theta_{\mathrm{old}})} f(x)\right] \\
&= \mathbb{E}_{x\sim\theta_{\mathrm{old}}}\left[\nabla_\theta \log p(x \mid \theta)\big|_{\theta=\theta_{\mathrm{old}}} f(x)\right]
\end{aligned}
$$

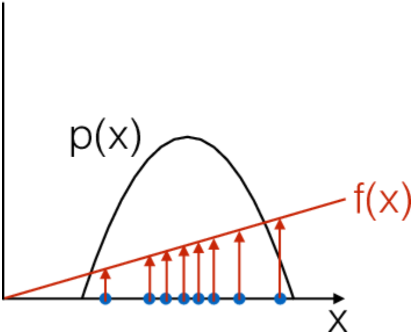
# Score Function Gradient Estimator: Intuition

$$\hat{g}_i = f(x_i)\nabla_\theta \log p(x_i \mid \theta)$$

- Let's say that $f(x)$ measures how good the sample $x$ is.
- Moving in the direction $\hat{g}_i$ pushes up the logprob of the sample, in proportion to how good it is
- *Valid even if $f(x)$ is discontinuous, and unknown, or sample space (containing $x$) is a discrete set*

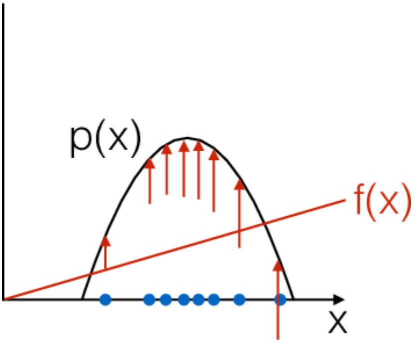
# Score Function Gradient Estimator: Intuition

$$\hat{g}_i = f(x_i)\nabla_\theta \log p(x_i \mid \theta)$$



# Score Function Gradient Estimator: Intuition

$$\hat{g}_i = f(x_i)\nabla_\theta \log p(x_i \mid \theta)$$



# Score Function Gradient Estimator for Policies

- Now random variable $x$ is a whole trajectory $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_{T-1}, a_{T-1}, r_{T-1}, s_T)$
$$\nabla_\theta E_\tau[R(\tau)] = E_\tau[\nabla_\theta \log p(\tau \mid \theta) R(\tau)]$$
- Just need to write out $p(\tau \mid \theta)$:
$$p(\tau \mid \theta) = \mu(s_0) \prod_{t=0}^{T-1} [\pi(a_t \mid s_t, \theta) P(s_{t+1}, r_t \mid s_t, a_t)]$$
$$\log p(\tau \mid \theta) = \log \mu(s_0) + \sum_{t=0}^{T-1} [\log \pi(a_t \mid s_t, \theta) + \log P(s_{t+1}, r_t \mid s_t, a_t)]$$
$$\nabla_\theta \log p(\tau \mid \theta) = \nabla_\theta \sum_{t=0}^{T-1} \log \pi(a_t \mid s_t, \theta)$$
$$\nabla_\theta \mathbb{E}_\tau [R] = \mathbb{E}_\tau \left[ R \nabla_\theta \sum_{t=0}^{T-1} \log \pi(a_t \mid s_t, \theta) \right]$$
- Interpretation: using good trajectories (high $R$) as supervised examples in classification / regression

# Policy Gradient–Slightly Better Formula

- Previous slide:

$$\nabla_\theta \mathbb{E}_\tau [R] = \mathbb{E}_\tau \left[ \left( \sum_{t=0}^{T-1} r_t \right) \left( \sum_{t=0}^{T-1} \nabla_\theta \log \pi(a_t \mid s_t, \theta) \right) \right]$$

- But we can cut trajectory to $t$ steps and derive gradient estimator for one reward term $r_{t'}$.

$$\nabla_\theta \mathbb{E}[r_{t'}] = \mathbb{E}\left[ r_{t'} \sum_{t=0}^{t} \nabla_\theta \log \pi(a_t \mid s_t, \theta) \right]$$

- Sum this formula over $t$, obtaining

$$\begin{aligned} \nabla_\theta \mathbb{E}[R] &= \mathbb{E}\left[ \sum_{t=0}^{T-1} r_{t'} \sum_{t=0}^{t'} \nabla_\theta \log \pi(a_t \mid s_t, \theta) \right] \\ &= \mathbb{E}\left[ \sum_{t=0}^{T-1} \nabla_\theta \log \pi(a_t \mid s_t, \theta) \sum_{t'=t}^{T-1} r_{t'} \right] \end{aligned}$$

# Adding a Baseline

- Suppose $f(x) \geq 0, \quad \forall x$
- Then for every $x_i$, gradient estimator $\hat{g}_i$ tries to push up it's density
- We can derive a new unbiased estimator that avoids this problem, and only pushes up the density for better-than-average $x_i$.

$$\begin{aligned} \nabla_\theta \mathbb{E}_x [f(x)] &= \nabla_\theta \mathbb{E}_x [f(x) - b] \\ &= \mathbb{E}_x [\nabla_\theta \log p(x \mid \theta)(f(x) - b)] \end{aligned}$$

- A near-optimal choice of $b$ is always $\mathbb{E}[f(x)]$ (which must be estimated)

# Policy Gradient with Baseline

- Recall

$$\nabla_\theta \mathbb{E}_\tau [R] = \sum_{t'=0}^{T-1} r_{t'} \sum_{t=t}^{T-1} \nabla_\theta \log \pi(a_t \mid s_t, \theta)$$

- Using the $\mathbb{E}_{a_t} [\nabla_\theta \log \pi(a_t \mid s_t, \theta)] = 0$, we can show

$$\nabla_\theta \mathbb{E}_\tau [R] = \mathbb{E}_\tau \left[ \sum_{t=0}^{T-1} \nabla_\theta \log \pi(a_t \mid s_t, \theta) \left( \sum_{t=t'}^{T-1} r_{t'} - b(s_t) \right) \right]$$

  for any "baseline" function $b : \mathcal{S} \to \mathbb{R}$
- Increase logprob of action $a_t$ proportionally to how much returns $\sum_{t=t'}^{T-1} r_{t'}$ are better than expected
- Later: use *value functions* to further isolate effect of action, at the cost of bias
- For more general picture of score function gradient estimator, see *stochastic computation graphs*[4].

[4] John Schulman, Nicolas Heess, et al. "Gradient Estimation Using Stochastic Computation Graphs". In: *Advances in Neural Information Processing Systems*. 2015, pp. 3510–3522.

# That's all for today

Course Materials: goo.gl/5wsgbJ

# Variance Reduction for Policy Gradients

# Review (I)

- Process for generating trajectory
  $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_{T-1}, a_{T-1}, r_{T-1}, s_T)$

$$
\begin{aligned}
s_0 &\sim \mu(s_0) \\
a_0 &\sim \pi(a_0 \mid s_0) \\
s_1, r_0 &\sim P(s_1, r_0 \mid s_0, a_0) \\
a_1 &\sim \pi(a_1 \mid s_1) \\
s_2, r_1 &\sim P(s_2, r_1 \mid s_1, a_1) \\
&\ldots \\
a_{T-1} &\sim \pi(a_{T-1} \mid s_{T-1}) \\
s_T, r_{T-1} &\sim P(s_T \mid s_{T-1}, a_{T-1})
\end{aligned}
$$

- Given parameterized policy $\pi(a \mid s, \theta)$, the optimization problem is

$$\underset{\theta}{\text{maximize}}\, \mathbb{E}_\tau \left[R \mid \pi(\cdot \mid \cdot, \theta)\right]$$

where $R = r_0 + r_1 + \cdots + r_{T-1}$.

# Review (II)

- In general, we can compute gradients of expectations with the *score function gradient estimator*

$$\nabla_\theta \mathbb{E}_{x \sim p(x \mid \theta)} [f(x)] = \mathbb{E}_x [\nabla_\theta \log p(x \mid \theta) f(x)]$$

- We derived a formula for the policy gradient

$$\nabla_\theta \mathbb{E}_\tau [R] = \mathbb{E}_\tau \left[ \sum_{t=0}^{T-1} \nabla_\theta \log \pi(a_t \mid s_t, \theta) \left( \sum_{t=t'}^{T-1} r_{t'} - b(s_t) \right) \right]$$

# Value Functions

- The *state-value function* $V^\pi$ is defined as:

$$V^\pi(s) = E[r_0 + r_1 + r_2 + \dots \mid s_0 = s]$$

  Measures *expected future return, starting with state $s$*

- The *state-action value function* $Q^\pi$ is defined as

$$Q^\pi(s, a) = E[r_0 + r_1 + r_2 + \dots \mid s_0 = s, a_0 = a]$$

- The *advantage function* $A^\pi$ is

$$A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$$

  Measures *how much better is action $a$ than what the policy $\pi$ would've done.*

# Refining the Policy Gradient Formula

- Recall

$$\nabla_\theta \mathbb{E}_\tau[R] = \mathbb{E}_\tau\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi(a_t \mid s_t, \theta)\left(\sum_{t=t'}^{T-1} r_{t'} - b(s_t)\right)\right]$$

$$= \sum_{t=0}^{T-1} \mathbb{E}_\tau\left[\nabla_\theta \log \pi(a_t \mid s_t, \theta)\left(\sum_{t=t'}^{T-1} r_{t'} - b(s_t)\right)\right]$$

$$= \sum_{t=0}^{T-1} \mathbb{E}_{s_0 \ldots a_t}\left[\nabla_\theta \log \pi(a_t \mid s_t, \theta)\mathbb{E}_{r_t s_{t+1} \ldots s_T}\left[\left(\sum_{t=t'}^{T-1} r_{t'} - b(s_t)\right)\right]\right]$$

$$= \sum_{t=0}^{T-1} \mathbb{E}_{s_0 \ldots a_t}\left[\nabla_\theta \log \pi(a_t \mid s_t, \theta)\mathbb{E}_{r_t s_{t+1} \ldots s_T}\left[Q^\pi(s_t, a_t) - b(s_t)\right]\right]$$

- Where the last equality used the fact that

$$\mathbb{E}_{r_t s_{t+1} \ldots s_T}\left[\sum_{t=t'}^{T-1} r_{t'}\right] = Q^\pi(s_t, a_t)$$

# Refining the Policy Gradient Formula

- From the previous slide, we've obtained

$$\nabla_\theta \mathbb{E}_\tau [R] = \mathbb{E}_\tau \left[ \sum_{t=0}^{T-1} \nabla_\theta \log \pi(a_t \mid s_t, \theta)(Q^\pi(s_t, a_t) - b(s_t)) \right]$$

- Now let's define $b(s) = V^\pi(s)$, which turns out to be near-optimal[5]. We get

$$\nabla_\theta \mathbb{E}_\tau [R] = \mathbb{E}_\tau \left[ \sum_{t=0}^{T-1} \nabla_\theta \log \pi(a_t \mid s_t, \theta) A^\pi(s_t, a_t) \right]$$

- Intuition: increase the probability of good actions (positive advantage) decrease the probability of bad ones (negative advantage)

[5] Evan Greensmith, Peter L Bartlett, and Jonathan Baxter. "Variance reduction techniques for gradient estimates in reinforcement learning". In: *The Journal of Machine Learning Research* 5 (2004), pp. 1471–1530.

# Variance Reduction

- Now, we have the following policy gradient formula:

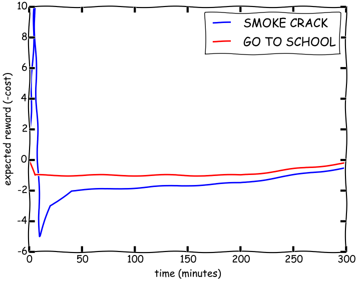
$$\nabla_\theta \mathbb{E}_\tau [R] = \mathbb{E}_\tau \left[ \sum_{t=0}^{T-1} \nabla_\theta \log \pi(a_t \mid s_t, \theta) A^\pi(s_t, a_t) \right]$$

- $A^\pi$ is not known, but we can plug in a random variable $\hat{A}_t$, an *advantage estimator*
- Previously, we showed that taking

$$\hat{A}_t = r_t + r_{t+1} + r_{t+2} + \cdots - b(s_t)$$

for any function $b(s_t)$, gives an unbiased policy gradient estimator. $b(s_t) \approx V^\pi(s_t)$ gives variance reduction.

# The Delayed Reward Problem

- One reason RL is difficult is the long delay between action and reward
- 

# The Delayed Reward Problem

- With policy gradient methods, we are confounding the effect of multiple actions:

  $$\hat{A}_t = r_t + r_{t+1} + r_{t+2} + \cdots - b(s_t)$$

  mixes effect of $a_t, a_{t+1}, a_{t+2}, \ldots$
- SNR of $\hat{A}_t$ scales roughly as $1/T$
  - Only $a_t$ contributes to *signal* $A^{\pi}(s_t, a_t)$, but $a_{t+1}, a_{t+2}, \ldots$ contribute to noise.

# Var. Red. Idea 1: Using Discounts

- Discount factor $\gamma$, $0 < \gamma < 1$, downweights the effect of rewars that are far in the future—ignore long term dependencies
- We can form an advantage estimator using the *discounted return*:

$$\hat{A}_t^{\gamma} = \underbrace{r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots}_{\text{discounted return}} - b(s_t)$$

  reduces to our previous estimator when $\gamma = 1$.
- So advantage has expectation zero, we should fit baseline to be *discounted value function*

$$V^{\pi,\gamma}(s) = \mathbb{E}_{\tau}\left[r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots \mid s_0 = s\right]$$

- $\hat{A}_t^{\gamma}$ is a biased estimator of the advantage function

# Var. Red. Idea 2: Value Functions in the Future

- Another approach for variance reduction is to use the value function to estimate future rewards

$$
\begin{aligned}
&r_t + r_{t+1} + r_{t+2} + \dots && \text{use empirical rewards} \\
&\Rightarrow && \\
&r_t + V(s_{t+1}) && \text{cut off at one timestep} \\
&r_t + r_{t+1} + V(s_{t+2}) && \text{cut off at two timesteps} \\
&\dots &&
\end{aligned}
$$

Adding the baseline again, we get the advantage estimators

$$
\begin{aligned}
&\hat{A}_t = r_t + V(s_{t+1}) - V(s_t) && \text{cut off at one timestep} \\
&\hat{A}_t = r_t + r_{t+1} + V(s_{t+2}) - V(s_t) && \text{cut off at two timesteps} \\
&\dots &&
\end{aligned}
$$

# Combining Ideas 1 and 2

- Can combine discounts and value functions in the future, e.g., $\hat{A}_t = r_t + \gamma V(s_{t+1}) - V(s_t)$, where $V$ approximates discounted value function $V^{\pi,\gamma}$.
- The above formula is called an *actor-critic* method, where *actor* is the policy $\pi$, and *critic* is the value function $V$.[6]
- Going further, the *generalized advantage estimator*[7]

$$\begin{aligned} \hat{A}_t^{\gamma,\lambda} =& \delta_t + (\gamma\lambda)\delta_{t+1} + (\gamma\lambda)^2\delta_{t+2} + \dots \\ &\text{where} \quad \delta_t = r_t + \gamma V(s_{t+1}) - V(s_t) \end{aligned}$$

- Interpolates between two previous estimators:

$$\begin{aligned} \lambda = 0: &\quad r_t + \gamma V(s_{t+1}) - V(s_t) & \text{(low v, high b)} \\ \lambda = 1: &\quad r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots - V(s_t) & \text{(low b, high v)} \end{aligned}$$

[6] Vijay R Konda and John N Tsitsiklis. "Actor-Critic Algorithms." In: *Advances in Neural Information Processing Systems*. Vol. 13. Citeseer. 1999, pp. 1008–1014.

[7] John Schulman, Philipp Moritz, et al. "High-dimensional continuous control using generalized advantage estimation". In: *arXiv preprint arXiv:1506.02438* (2015).

# Alternative Approach: Reparameterization

- Suppose problem has continuous action space, $a \in \mathbb{R}^d$
- Then $\frac{d}{da} Q^{\pi}(s, a)$ tells use how to improve our action
- We can use reparameterization trick, so $a$ is a deterministic function $a = f(s, z)$, where $z$ is noise. Then,

$$\nabla_{\theta} \mathbb{E}_{\tau}[R] = \nabla_{\theta} Q^{\pi}(s_0, a_0) + \nabla_{\theta} Q^{\pi}(s_1, a_1) + \dots$$

- This method is called the deterministic policy gradient[8]
- A generalized version, which also uses a dynamics model, is described as the stochastic value gradient[9]

[8] David Silver, Guy Lever, et al. "Deterministic policy gradient algorithms". In: *ICML*. 2014; Timothy P Lillicrap et al. "Continuous control with deep reinforcement learning". In: *arXiv preprint arXiv:1509.02971* (2015).

[9] Nicolas Heess et al. "Learning continuous control policies by stochastic value gradients". In: *Advances in Neural Information Processing Systems*. 2015, pp. 2926–2934.

# Trust Region and Natural Gradient Methods

# Optimization Issues with Policy Gradients

- Hard to choose reasonable stepsize that works for the whole optimization
  - we have a gradient estimate, no objective for line search
  - statistics of data (observations and rewards) change during learning
- They make inefficient use of data: each experience is only used to compute one gradient.
  - Given a batch of trajectories, what's the most we can do with it?

# Policy Performance Function

- Let $\eta(\pi)$ denote the performance of policy $\pi$

$$\eta(\pi) = \mathbb{E}_{\tau}\left[R|\pi\right]$$

- The following neat identity holds:

$$\eta(\tilde{\pi}) = \eta(\pi) + \mathbb{E}_{\tau\sim\tilde{\pi}}\left[A^{\pi}(s_0, a_0) + A^{\pi}(s_1, a_1) + A^{\pi}(s_2, a_2) + \dots\right]$$

- Proof: consider nonstationary policy $\pi_0\pi_1\pi_2, \dots$

$$\begin{aligned}
\eta(\tilde{\pi}\tilde{\pi}\tilde{\pi}\cdots) = \eta(\pi\pi\pi\cdots) \\
&+ \eta(\tilde{\pi}\pi\pi\cdots) - \eta(\pi\pi\pi\cdots) \\
&+ \eta(\tilde{\pi}\tilde{\pi}\pi\cdots) - \eta(\tilde{\pi}\pi\pi\cdots) \\
&+ \eta(\tilde{\pi}\tilde{\pi}\tilde{\pi}\cdots) - \eta(\tilde{\pi}\tilde{\pi}\pi\cdots) \\
&+ \dots
\end{aligned}$$

- $t^{\text{th}}$ difference term equals $A^{\pi}(s_t, a_t)$

# Local Approximation

- We just derived an expression for the performance of a policy $\tilde{\pi}$ relative to $\pi$

$$\begin{aligned}\eta(\tilde{\pi}) &= \eta(\pi) + \mathbb{E}_{\tau\sim\tilde{\pi}}\left[A^\pi(s_0,a_0) + A^\pi(s_1,a_1) + \dots\right] \\ &= \eta(\pi) + \mathbb{E}_{s_{0:\infty}\sim\tilde{\pi}}\left[\mathbb{E}_{a_{0:\infty}\sim\tilde{\pi}}\left[A^\pi(s_0,a_0) + A^\pi(s_1,a_1) + \dots\right]\right]\end{aligned}$$

- Can't use this to optimize $\tilde{\pi}$ because state distribution has complicated dependence.
- Let's define $L_\pi$ the *local approximation*, which ignores change in state distribution—can be estimated by sampling from $\pi$

$$\begin{aligned}L_\pi(\tilde{\pi}) &= \mathbb{E}_{s_{0:\infty}\sim\pi}\left[\mathbb{E}_{a_{0:\infty}\sim\tilde{\pi}}\left[A^\pi(s_0,a_0) + A^\pi(s_1,a_1) + \dots\right]\right] \\ &= \mathbb{E}_{s_{0:\infty}}\left[\sum_{t=0}^{T-1}\mathbb{E}_{a\sim\tilde{\pi}}\left[A^\pi(s_t,a_t)\right]\right] \\ &= \mathbb{E}_{s_{0:\infty}}\left[\sum_{t=0}^{T-1}\mathbb{E}_{a\sim\pi}\left[\frac{\tilde{\pi}(a_t \mid s_t)}{\pi(a_t \mid s_t)}A^\pi(s_t,a_t)\right]\right] \\ &= \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{T-1}\frac{\tilde{\pi}(a_t \mid s_t)}{\pi(a_t \mid s_t)}A^\pi(s_t,a_t)\right]\end{aligned}$$

# Local Approximation

- Now let's consider parameterized policy, $\pi(a \mid s, \theta)$. Sample with $\theta_{\text{old}}$, now write local approximation in terms of $\theta$.

$$L_\pi(\tilde{\pi}) = \mathbb{E}_{s_{0:\infty}}\left[\sum_{t=0}^{T-1} \mathbb{E}_{a\sim\pi}\left[\frac{\tilde{\pi}(a_t \mid s_t)}{\pi(a_t \mid s_t)} A^\pi(s_t, a_t)\right]\right]$$

$$\Rightarrow L_{\theta_{\text{old}}}(\theta) = \mathbb{E}_{s_{0:\infty}}\left[\sum_{t=0}^{T-1} \mathbb{E}_{a\sim\theta}\left[\frac{\pi(a_t \mid s_t, \theta)}{\pi(a_t \mid s_t, \theta_{\text{old}})} A^\theta(s_t, a_t)\right]\right]$$

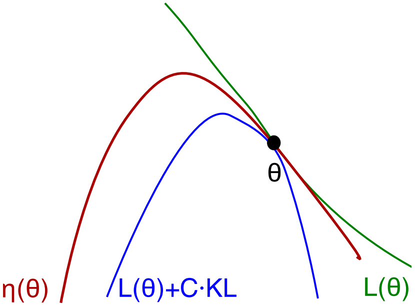
- $L_{\theta_{\text{old}}}(\theta)$ matches $\eta(\theta)$ to first order around $\theta_{\text{old}}$.

$$\begin{aligned}
\nabla_\theta L_{\theta_{\text{old}}}(\theta)\big|_{\theta=\theta_0} &= \mathbb{E}_{s_{0:\infty}}\left[\sum_{t=0}^{T-1} \mathbb{E}_{a\sim\theta}\left[\frac{\nabla_\theta \pi(a_t \mid s_t, \theta)}{\pi(a_t \mid s_t, \theta_{\text{old}})} A^\theta(s_t, a_t)\right]\right] \\
&= \mathbb{E}_{s_{0:\infty}}\left[\sum_{t=0}^{T-1} \mathbb{E}_{a\sim\theta}\left[\nabla_\theta \log \pi(a_t \mid s_t, \theta) A^\theta(s_t, a_t)\right]\right] \\
&= \nabla_\theta \eta(\theta)\big|_{\theta=\theta_{\text{old}}}
\end{aligned}$$

# MM Algorithm

- Theorem (ignoring some details)[10]

$$\eta(\theta) \geq \underbrace{L_{\theta_{\text{old}}}(\theta)}_{\text{local approx. to } \eta} - \underbrace{C \max_s D_{KL}\left[\pi(\cdot \mid \theta_{\text{old}}, s) \parallel \pi(\cdot \mid \theta, s)\right]}_{\text{penalty for changing policy}}$$

- 


- If $\theta_{\text{old}} \rightarrow \theta_{\text{new}}$ improves lower bound, it's guaranteed to improve $\eta$

[10] John Schulman, Sergey Levine, et al. "Trust Region Policy Optimization". In: *arXiv preprint arXiv:1502.05477* (2015).

# Review

- Want to optimize $\eta(\theta)$. Collected data with policy parameter $\theta_{\text{old}}$, now want to do update
- Derived local approximation $L_{\theta_{\text{old}}}(\theta)$
- Optimizing KL penalized local approximation givesn guaranteed improvement to $\eta$
- More approximations gives practical algorithm, called TRPO

# TRPO—Approximations

- Steps:
  - Instead of max over state space, take mean
  - Linear approximation to $L$, quadratic approximation to KL divergence
  - Use hard constraint on KL divergence instead of penalty
- Solve the following problem approximately

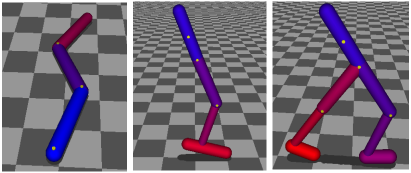
$$\begin{aligned} &\text{maximize } L_{\theta_{\text{old}}}(\theta) \\ &\text{subject to} \quad \overline{D}_{KL}[\theta_{\text{old}} \parallel \theta] \leq \delta \end{aligned}$$

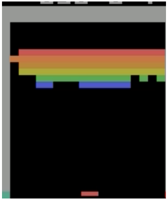
- Solve approximately through line search in the *natural gradient* direction $s = F^{-1}g$
- Resulting algorithm is a refined version of *natural policy gradient*[11]

[11] Sham Kakade. "A Natural Policy Gradient." In: *NIPS*. vol. 14. 2001, pp. 1531–1538.

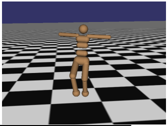
# Empirical Results: TRPO + GAE

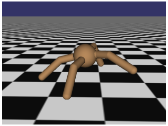
- TRPO, with neural network policies, was applied to learn controllers for 2D robotic swimming, hopping, and walking, and playing Atari games[12]





- Used TRPO along with generalized advantage estimation to optimize locomotion policies for 3D simulated robots[13]





[12] John Schulman, Sergey Levine, et al. "Trust Region Policy Optimization". In: *arXiv preprint arXiv:1502.05477* (2015).

[13] John Schulman, Philipp Moritz, et al. "High-dimensional continuous control using generalized advantage estimation". In: *arXiv preprint arXiv:1506.02438* (2015).

# Putting In Perspective

Quick and incomplete overview of recent results with deep RL algorithms

- Policy gradient methods
  - TRPO + GAE
  - Standard policy gradient (no trust region) + deep nets + parallel implementation[14]
  - Repar trick[15]
- Q-learning[16] and modifications[17]
- Combining search + supervised learning[18]

[14] V. Mnih et al. "Playing Atari with Deep Reinforcement Learning". In: *arXiv preprint arXiv:1312.5602* (2013).

[15] Nicolas Heess et al. "Learning continuous control policies by stochastic value gradients". In: *Advances in Neural Information Processing Systems*. 2015, pp. 2926–2934; Timothy P Lillicrap et al. "Continuous control with deep reinforcement learning". In: *arXiv preprint arXiv:1509.02971* (2015).

[16] V. Mnih et al. "Playing Atari with Deep Reinforcement Learning". In: *arXiv preprint arXiv:1312.5602* (2013).

[17] Ziyu Wang, Nando de Freitas, and Marc Lanctot. "Dueling Network Architectures for Deep Reinforcement Learning". In: *arXiv preprint arXiv:1511.06581* (2015); Hado V Hasselt. "Double Q-learning". In: *Advances in Neural Information Processing Systems*. 2010, pp. 2613–2621.

[18] X. Guo et al. "Deep learning for real-time Atari game play using offline Monte-Carlo tree search planning". In: *Advances in Neural Information Processing Systems*. 2014, pp. 3338–3346; Sergey Levine et al. "End-to-end training of deep visuomotor policies". In: *arXiv preprint arXiv:1504.00702* (2015); Igor Mordatch et al. "Interactive Control of Diverse Complex Characters with Neural Networks". In: *Advances in Neural Information Processing Systems*. 2015, pp. 3114–3122.

# Open Problems

# What's the Right Core Model-Free Algorithm?

- Policy gradients (score function vs. reparameterization, natural vs. not natural) vs. Q-learning vs. derivative-free optimization vs others
- Desiderata
  - scalable
  - sample-efficient
  - robust
  - learns from *off-policy* data

# Exploration

- Exploration: actively encourage agent to reach unfamiliar parts of state space, avoid getting stuck in local maximum of performance
- Can solve finite MDPs in polynomial time with exploration[19]
  - optimism about new states and actions
  - maintain distribution over possible models, and plan with them (Bayesian RL, Thompson sampling)
- How to do exploration in deep RL setting? Thompson sampling[20], novelty bonus[21]

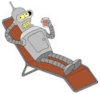
---

[19] Alexander L Strehl et al. "PAC model-free reinforcement learning". In: *Proceedings of the 23rd international conference on Machine learning*. ACM. 2006, pp. 881–888.

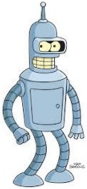
[20] Ian Osband et al. "Deep Exploration via Bootstrapped DQN". . In: *arXiv preprint arXiv:1602.04621* (2016).

[21] Bradly C Stadie, Sergey Levine, and Pieter Abbeel. "Incentivizing Exploration In Reinforcement Learning With Deep Predictive Models". In: *arXiv preprint arXiv:1507.00814* (2015).

# Hierarchy



task 1 … task 2 … task 3 … task 4 … 10 timesteps / day



walk to x … fetch object y … say z … .01 hz: $10^3$ time steps per day

footstep planning: 1hz: $10^5$ timesteps / day

torque control: 100hz: $10^7$ timesteps /day

# More Open Problems

- Using learned models
- Learning from demonstrations

# The End

Questions?