Actor Critic Methods

CS 224R

Reminders

Since Wednesday: Homework 1 is out

Next Monday: Project survey due

4/19: Homework 1 due, Homework 2 out

Reminder:

- Send your AWS account ID if you haven’t yet!

The Plan

Policy gradients recap

Variance reduction continued

Policy gradients tricks

Actor-critic

Case studies: robotics & RLHF

Key learning goals:

- Practical policy gradient implementation tricks & case studies
- Understanding a generic actor-critic method

The Plan

Policy gradients recap

Variance reduction continued

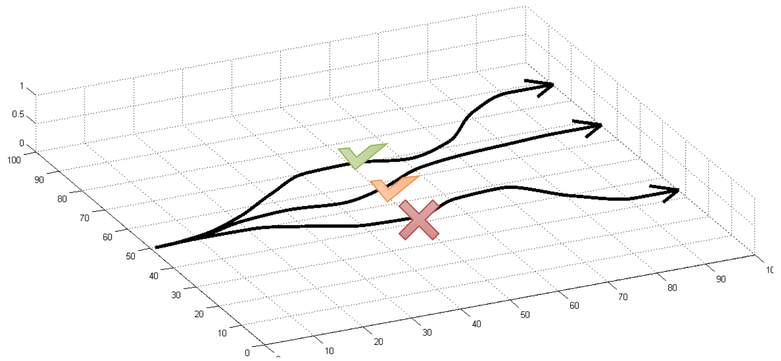
Policy gradients tricks

Actor-critic

Case studies: robotics & RLHF

Evaluating the objective

$$\theta^\star = \arg\max_\theta \underbrace{E_{\tau\sim\pi_\theta(\tau)}\left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t)\right]}_{J(\theta)}$$



$$J(\theta) = E_{\tau\sim\pi_\theta(\tau)}\left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t)\right] \approx \frac{1}{N}\sum_i\sum_t r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t})$$

sum over samples from π_θ

Slide adapted from Sergey Levine

Policy gradients

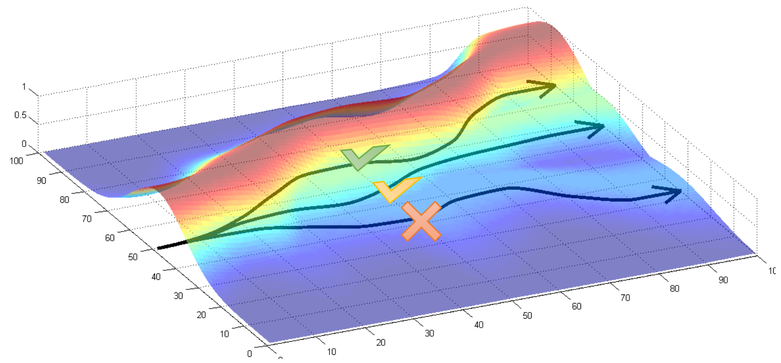
$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t}) \right) \left(\sum_{t=1}^{T} r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) \right)$$

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \underbrace{\nabla_\theta \log \pi_\theta(\tau_i) r(\tau_i)}_{\sum_{t=1}^{T} \nabla_\theta \log_\theta \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})}$$

good stuff is made more likely

bad stuff is made less likely

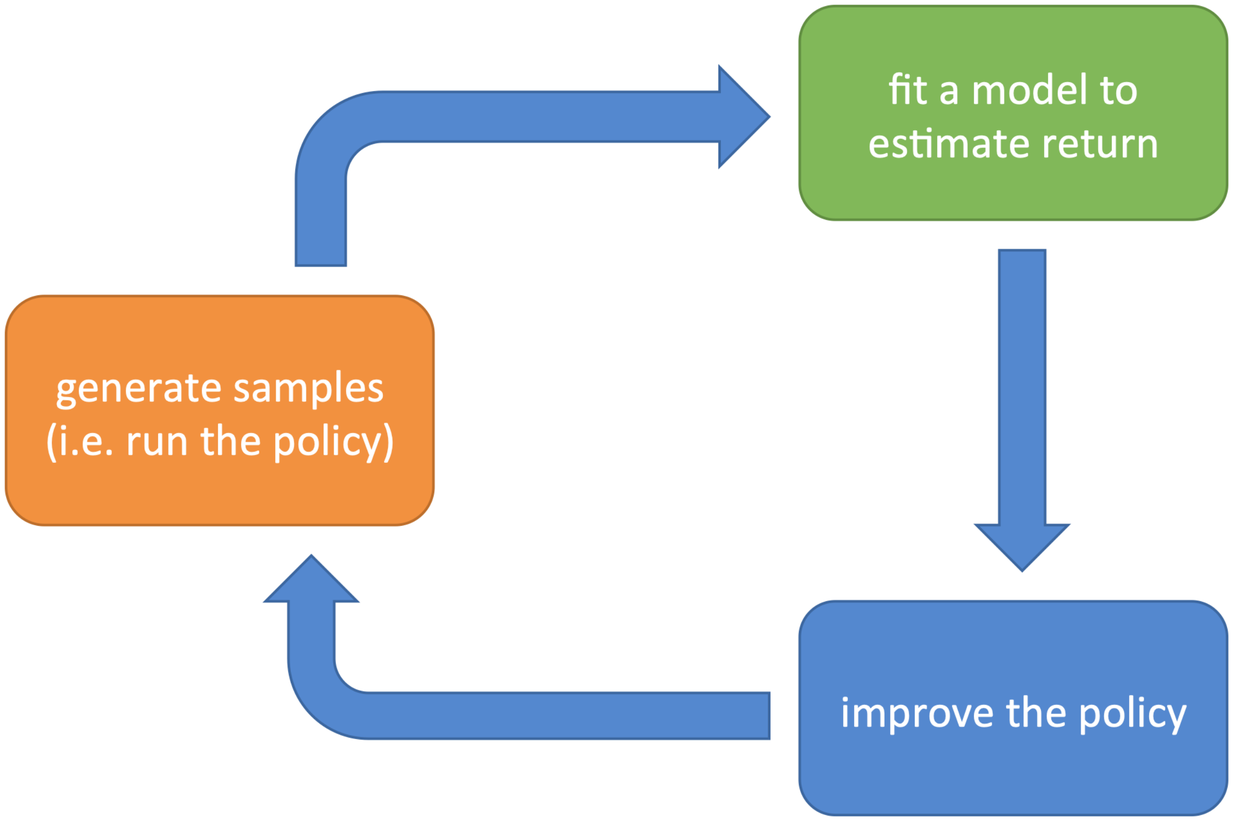
simply formalizes the notion of “trial and error”!



REINFORCE algorithm:

1. sample $\{\tau^i\}$ from $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ (run it on the robot)
2. $\nabla_\theta J(\theta) \approx \sum_i \left(\sum_t \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i|\mathbf{s}_t^i) \right) \left(\sum_t r(\mathbf{s}_t^i, \mathbf{a}_t^i) \right)$
3. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

The anatomy of a reinforcement learning algorithm



compute $\hat{Q} = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$ (MC policy gradient)
fit $Q_\phi(\mathbf{s}, \mathbf{a})$ (actor-critic, Q-learning)
estimate $p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$ (model-based)

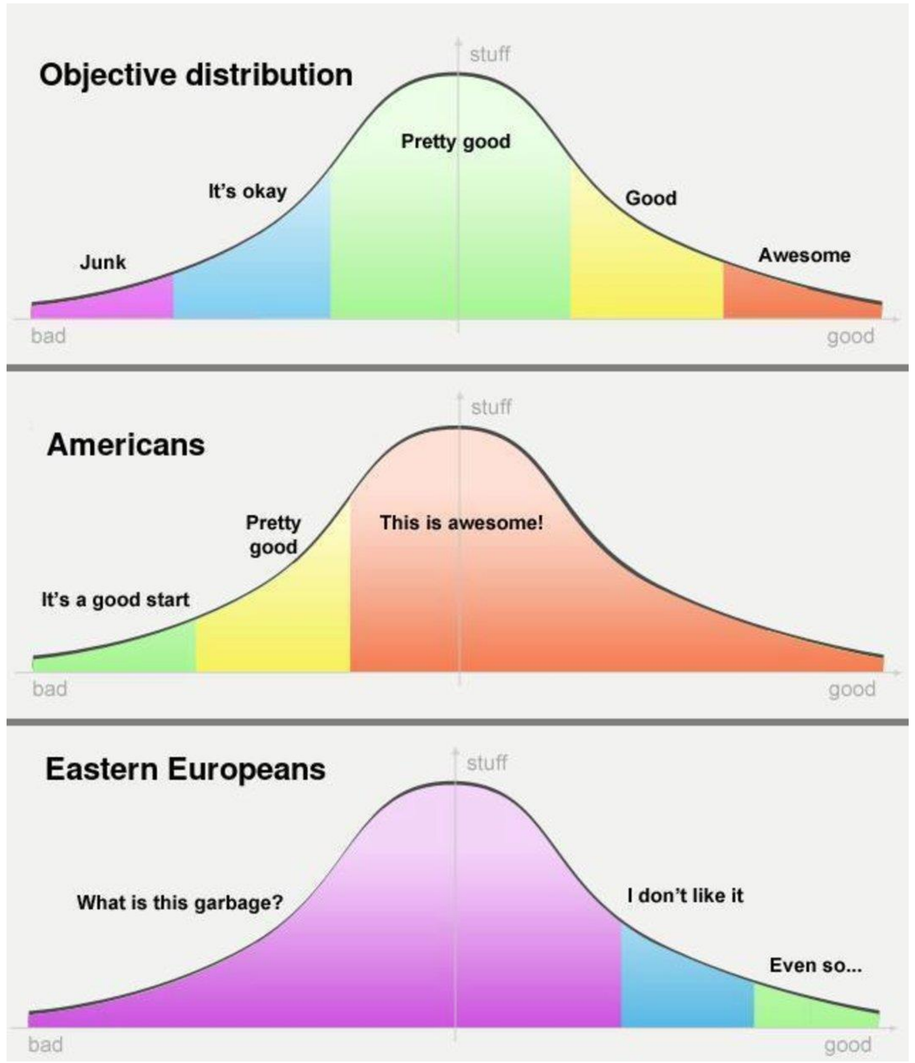
$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$ (policy gradient)
$\pi(\mathbf{s}) = \arg\max Q_\phi(\mathbf{s}, \mathbf{a})$ (Q-learning)
optimize $\pi_\theta(\mathbf{a}|\mathbf{s})$ (model-based)

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t}) \left(\sum_{t'=t}^{T} r(\mathbf{a}_{i,t'}, \mathbf{s}_{i,t'}) \right)$$

Variance of the gradient estimator

policy gradient: $$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \underbrace{\nabla_\theta \log \pi_\theta(\tau_i) r(\tau_i)}_{T}$$

$$\sum_{t=1} \nabla_\theta \log_\theta \pi_\theta(\mathbf{a}_{i,t} | \mathbf{s}_{i,t})$$



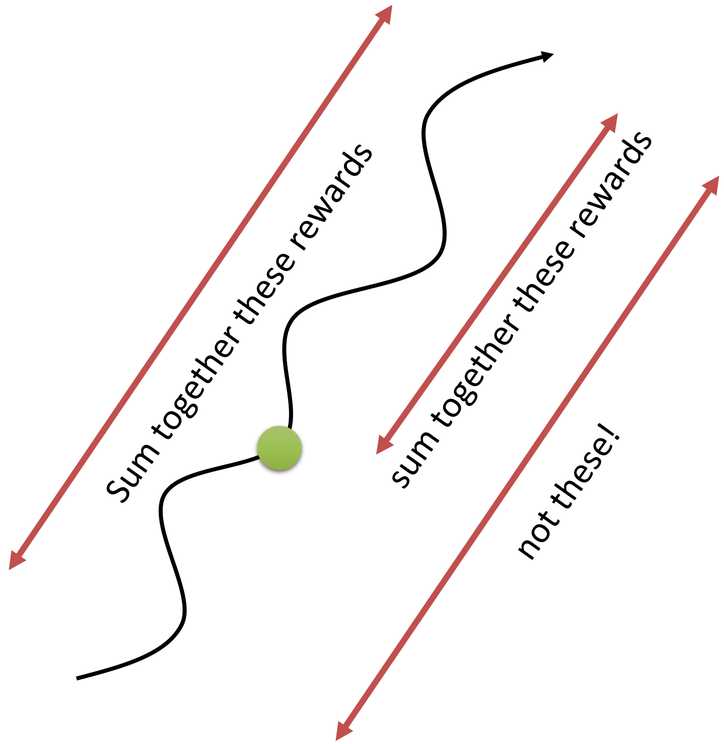
Small way to reduce variance

policy gradient: $$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})\right)\left(\sum_{t=1}^{T} r(\mathbf{s}_{i,t},\mathbf{a}_{i,t})\right)$$

$$\frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})\left(\sum_{t'=1}^{T} r(\mathbf{a}_{i,t'},\mathbf{s}_{i,t'})\right)$$

$$\frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t})\left(\sum_{t'=t}^{T} r(\mathbf{a}_{i,t'},\mathbf{s}_{i,t'})\right)$$

Reward “to go”



The Plan

Policy gradients recap

Variance reduction continued

Policy gradients tricks

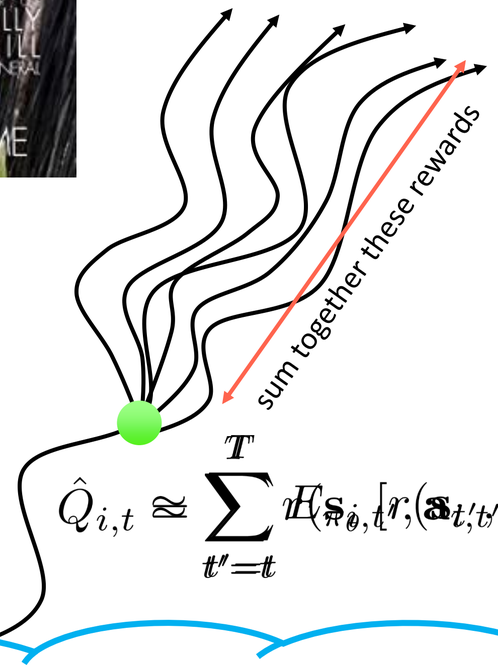
Actor-critic

Case studies: robotics & RLHF

Improving the policy gradient

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t}) \underbrace{\left(\sum_{t'=t}^{T} r(\mathbf{a}_{i,t'}, \mathbf{s}_{i,t'}) \right)}_{\text{Reward "to go"}}$$

$\hat{Q}_{i,t}$



$\hat{Q}_{i,t}$: estimate of expected reward if we take action $\mathbf{a}_{i,t}$ in state $\mathbf{s}_{i,t}$

can we get a better estimate?

$Q(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^{T} E_{\pi_\theta}\left[r(\mathbf{s}_{t'}, \mathbf{a}_{t'})|\mathbf{s}_t, \mathbf{a}_t\right]$: true *expected* reward-to-go

$$\hat{Q}_{i,t} \approx \sum_{t'=t}^{T} E_{\pi_\theta}\left[r(\mathbf{s}_{t'}, \mathbf{a}_{t'})|\mathbf{s}_t, \mathbf{a}_t\right]$$

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t}) Q(\mathbf{s}_{i,t}, \mathbf{a}_{i,t})$$

$$J(\theta) = E_{\tau \sim \pi_\theta(\tau)}\left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t)\right] \approx \frac{1}{N} \sum_i \sum_t r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t})$$

Slide adapted from Sergey Levine

State & state-action value functions

$Q^\pi(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^{T} E_{\pi_\theta}[r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) | \mathbf{s}_t, \mathbf{a}_t]$: total reward from taking $\mathbf{a}_t$ in $\mathbf{s}_t$

$V^\pi(\mathbf{s}_t) = E_{\mathbf{a}_t \sim \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)}[Q^\pi(\mathbf{s}_t, \mathbf{a}_t)]$: total reward from $\mathbf{s}_t$

$A^\pi(\mathbf{s}_t, \mathbf{a}_t) = Q^\pi(\mathbf{s}_t, \mathbf{a}_t) - V^\pi(\mathbf{s}_t)$:

Slide adapted from Sergey Levine

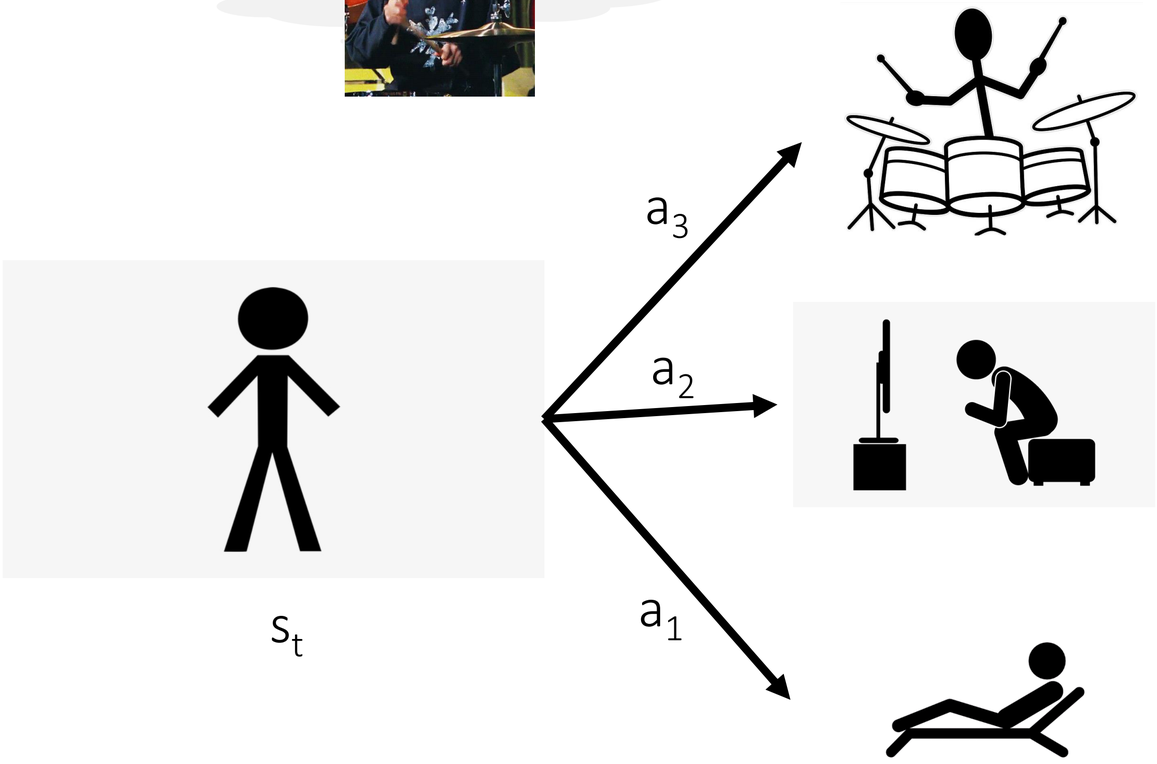
Value-Based RL

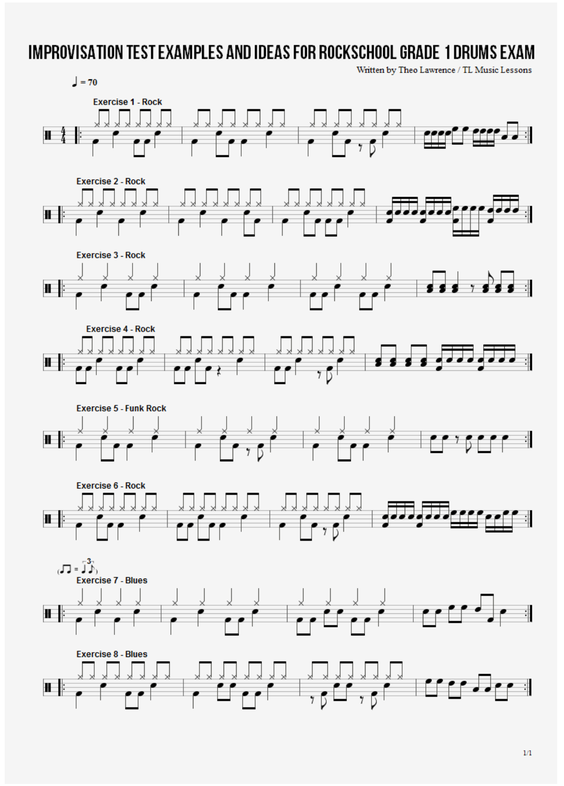
Value function: $V^{\pi}(\mathbf{s}_t) = ?$

Q function: $Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t) = ?$

Advantage function: $A^{\pi}(\mathbf{s}_t, \mathbf{a}_t) = ?$

Reward = 1 if I can play it in a month, 0 otherwise





Current $\pi(\mathbf{a}_1|\mathbf{s}) = 1$

State & state-action value functions

$Q^\pi(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^{T} E_{\pi_\theta}\left[r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) | \mathbf{s}_t, \mathbf{a}_t\right]$: total reward from taking $\mathbf{a}_t$ in $\mathbf{s}_t$

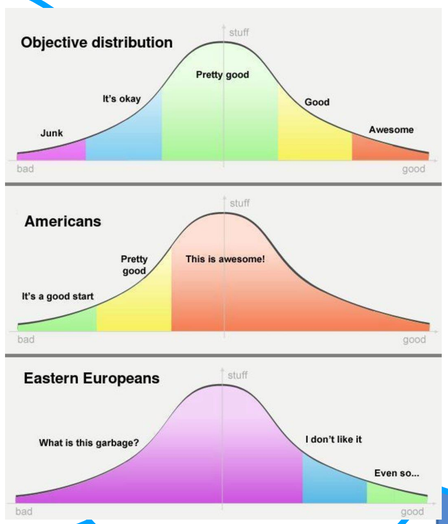
$V^\pi(\mathbf{s}_t) = E_{\mathbf{a}_t \sim \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)}[Q^\pi(\mathbf{s}_t, \mathbf{a}_t)]$: total reward from $\mathbf{s}_t$

$A^\pi(\mathbf{s}_t, \mathbf{a}_t) = Q^\pi(\mathbf{s}_t, \mathbf{a}_t) - V^\pi(\mathbf{s}_t)$: how much better $\mathbf{a}_t$ is

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t}) Q^\pi(\mathbf{s}_{i,t}, \mathbf{a}_{i,t})$$

fit Q^π, V^π, or A^π

odel to
eturn



improve the policy

$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Slide adapted from Sergey Levine

Value function fitting

$$Q^\pi(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^{T} E_{\pi_\theta}\left[r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) | \mathbf{s}_t, \mathbf{a}_t\right]$$

$$V^\pi(\mathbf{s}_t) = E_{\mathbf{a}_t \sim \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)}[Q^\pi(\mathbf{s}_t, \mathbf{a}_t)]$$

$$A^\pi(\mathbf{s}_t, \mathbf{a}_t) = Q^\pi(\mathbf{s}_t, \mathbf{a}_t) - V^\pi(\mathbf{s}_t)$$

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_{i,t}|\mathbf{s}_{i,t}) A^\pi(\mathbf{s}_{i,t}, \mathbf{a}_{i,t})$$

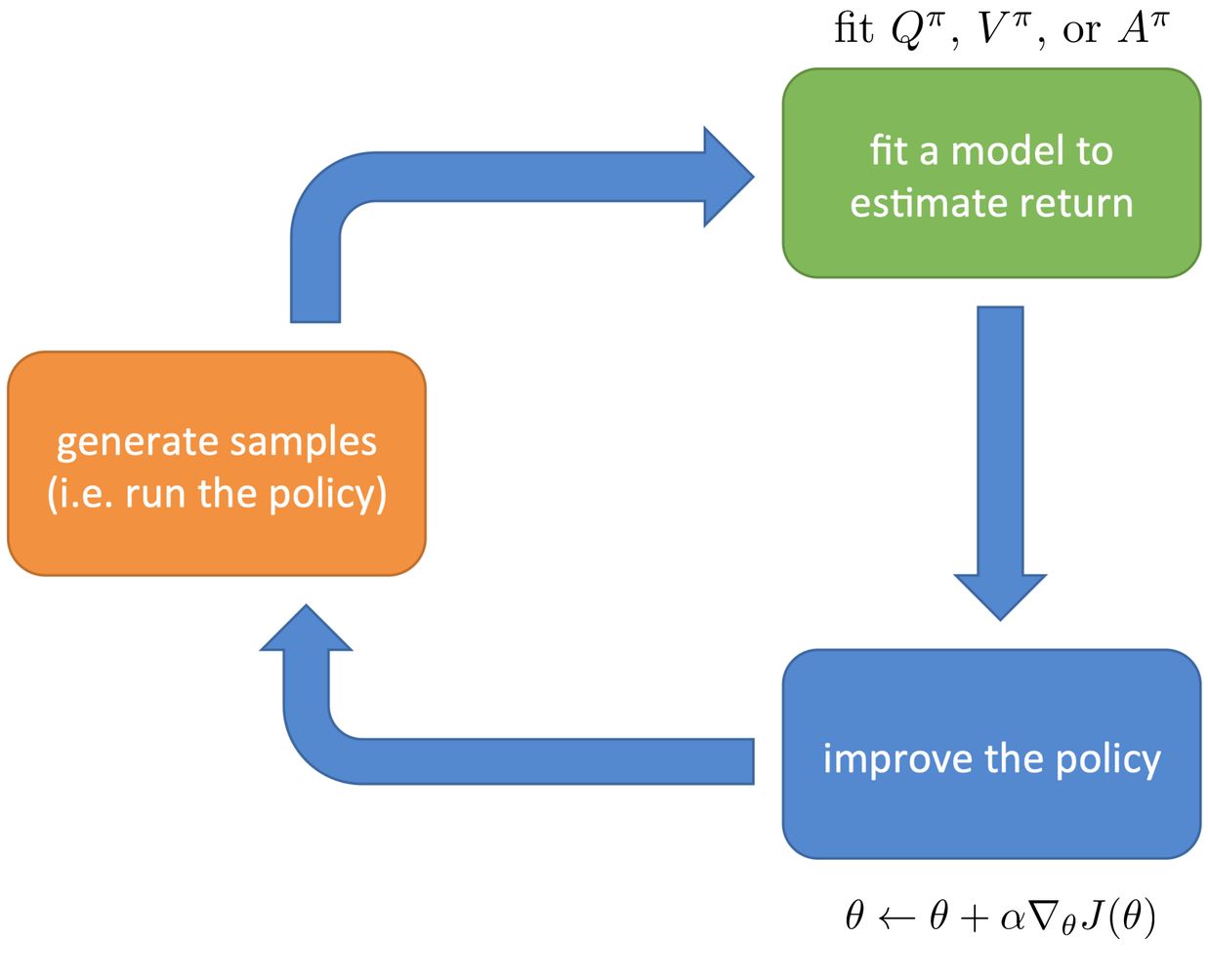
fit *what* to *what*?

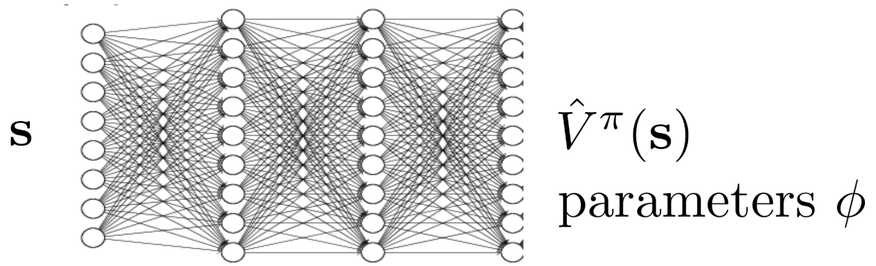
Q^π, V^π, A^π?

$$Q^\pi(\mathbf{s}_t, \mathbf{a}_t) \approx r(\mathbf{s}_t, \mathbf{a}_t) + \sum_{t'=t+1}^{T} E_{\pi_\theta}\left[r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) | \mathbf{s}_{t+1}, \mathbf{a}_t\right]$$

$$A^\pi(\mathbf{s}_t, \mathbf{a}_t) \approx r(\mathbf{s}_t, \mathbf{a}_t) + V^\pi(\mathbf{s}_{t+1}) - V^\pi(\mathbf{s}_t)$$

let's just fit $V^\pi(\mathbf{s})$!

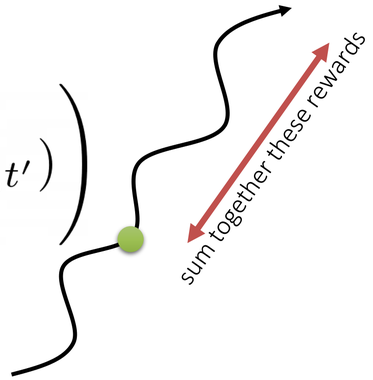




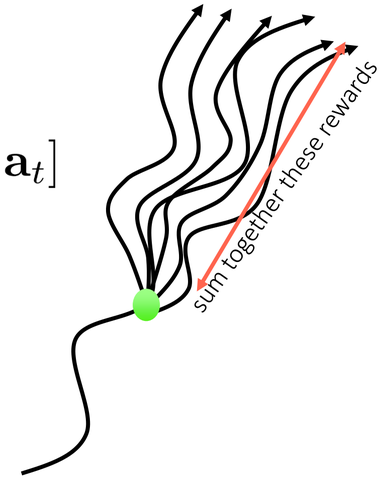
Slide adapted from Sergey Levine

Multi-Step Prediction

$$\hat{Q}_{i,t} \approx \left(\sum_{t'=t}^{T} r(\mathbf{a}_{i,t'}, \mathbf{s}_{i,t'}) \right)$$

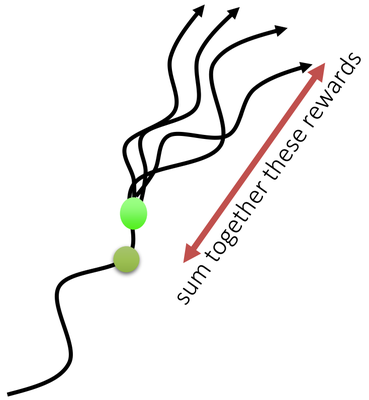


$$Q_{i,t} = \sum_{t'=t}^{T} E_{\pi_\theta} \left[r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) | \mathbf{s}_t, \mathbf{a}_t \right]$$



- How do you update your predictions about winning the game?
- What happens if you don't finish the game?
- Do you always wait till the end?

$$\hat{Q}_{i,t} \approx r(\mathbf{a}_{i,t}, \mathbf{s}_{i,t}) + V^{\pi}(\mathbf{s}_{t+1})$$



How can we use all of this to fit a better estimator?

Goal: fit V^π

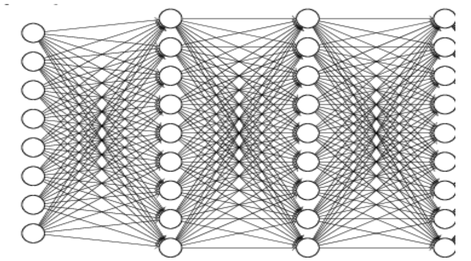
ideal target: $y_{i,t} = \sum_{t'=t}^{T} E_{\pi_\theta}\left[r(\mathbf{s}_{t'}, \mathbf{a}_{t'})|\mathbf{s}_{i,t}\right] \approx r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) + \sum_{t'=t+1}^{T} E_{\pi_\theta}\left[r(\mathbf{s}_{t'}, \mathbf{a}_{t'})|\mathbf{s}_{i,t+1}\right] \approx r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) + \underbrace{\hat{V}^\pi_\phi(\mathbf{s}_{i,t+1})}$

directly use previous fitted value function!

Monte Carlo target: $y_{i,t} = \sum_{t'=t}^{T} r(\mathbf{s}_{i,t'}, \mathbf{a}_{i,t'})$

training data: $\left\{\left(\mathbf{s}_{i,t}, \underbrace{r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) + \hat{V}^\pi_\phi(\mathbf{s}_{i,t+1})}_{y_{i,t}}\right)\right\}$

supervised regression: $\mathcal{L}(\phi) = \frac{1}{2}\sum_i \left\|\hat{V}^\pi_\phi(\mathbf{s}_i) - y_i\right\|^2$



sometimes referred to as a “bootstrapped” estimate

Slide adapted from Sergey Levine

Aside: discount factors

$$y_{i,t} \approx r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) + \hat{V}^{\pi}_{\phi}(\mathbf{s}_{i,t+1})$$

$$\mathcal{L}(\phi) = \frac{1}{2} \sum_i \left\| \hat{V}^{\pi}_{\phi}(\mathbf{s}_i) - y_i \right\|^2$$

what if T (episode length) is ∞?

$\hat{V}^{\pi}_{\phi}$ can get infinitely large in many cases

episodic tasks

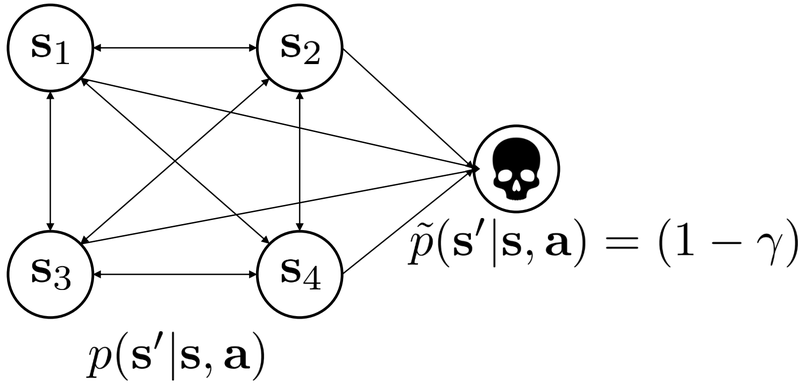
continuous/cyclical tasks

simple trick: better to get rewards sooner than later

$$y_{i,t} \approx r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) + \gamma \hat{V}^{\pi}_{\phi}(\mathbf{s}_{i,t+1})$$

discount factor $\gamma \in [0, 1]$ (0.99 works well)

γ changes the MDP:



N-step returns

$$\hat{A}^\pi_{\mathrm{C}}(\mathbf{s}_t, \mathbf{a}_t) = r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \hat{V}^\pi_\phi(\mathbf{s}_{t+1}) - \hat{V}^\pi_\phi(\mathbf{s}_t)$$

$$\hat{A}^\pi_{\mathrm{MC}}(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^{\infty} \gamma^{t'-t} r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) - \hat{V}^\pi_\phi(\mathbf{s}_t)$$

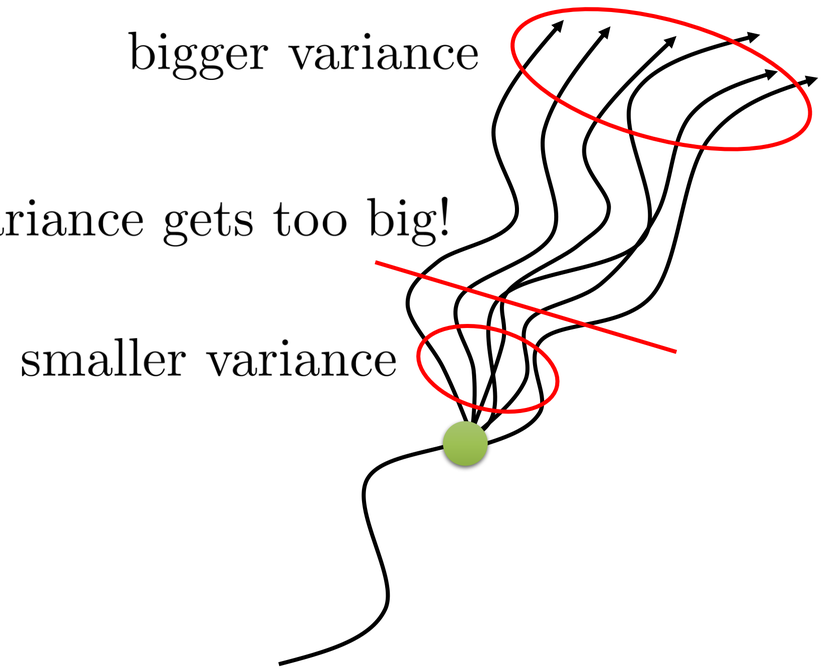
+ lower variance

- higher bias if value is wrong (it always is)

+ no bias

- higher variance (because single-sample estimate)

Can we combine these two, to control bias/variance tradeoff?



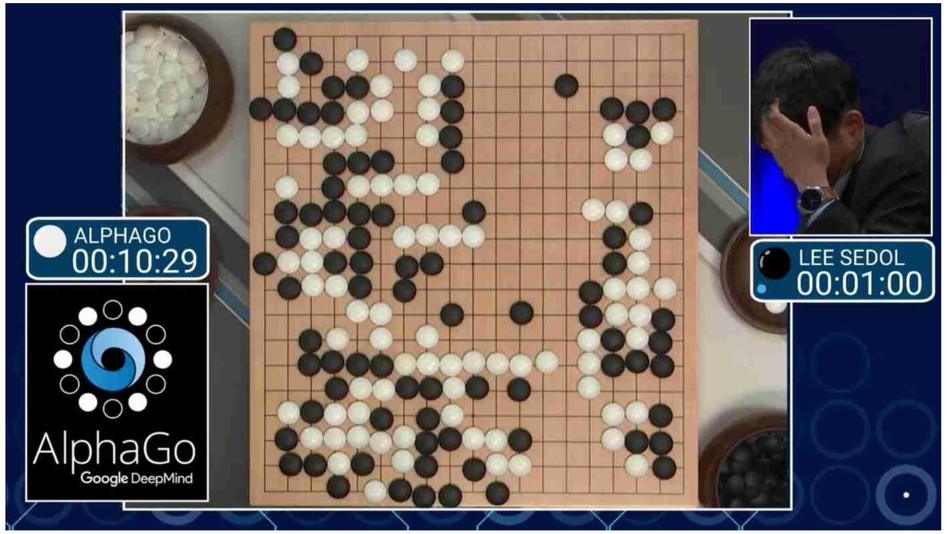
$$\hat{A}^\pi_n(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^{t+n} \gamma^{t'-t} r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) - \hat{V}^\pi_\phi(\mathbf{s}_t) + \gamma^n \hat{V}^\pi_\phi(\mathbf{s}_{t+n})$$

choosing $n > 1$ often works better!

Slide adapted from Sergey Levine

Policy evaluation example

AlphaGo, Silver et al. 2016



reward: game outcome

value function $\hat{V}_\phi^\pi(\mathbf{s}_t)$:
expected outcome given board state

The Plan

Policy gradients recap

Variance reduction continued

Policy gradients tricks

Actor-critic

Case studies: robotics & RLHF

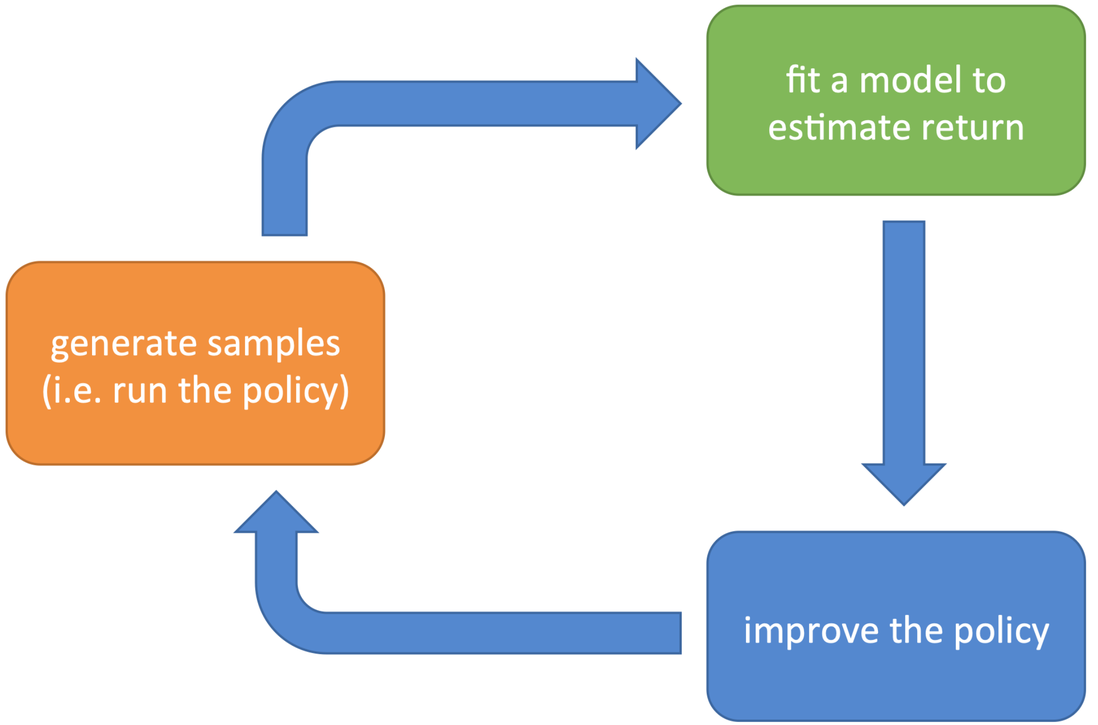
Why is there so many RL algorithms?

Different tradeoffs:

- Continuous vs discrete actions
- Is it easier to learn the environment or the policy?
- Sample complexity

Off or on policy algorithms:

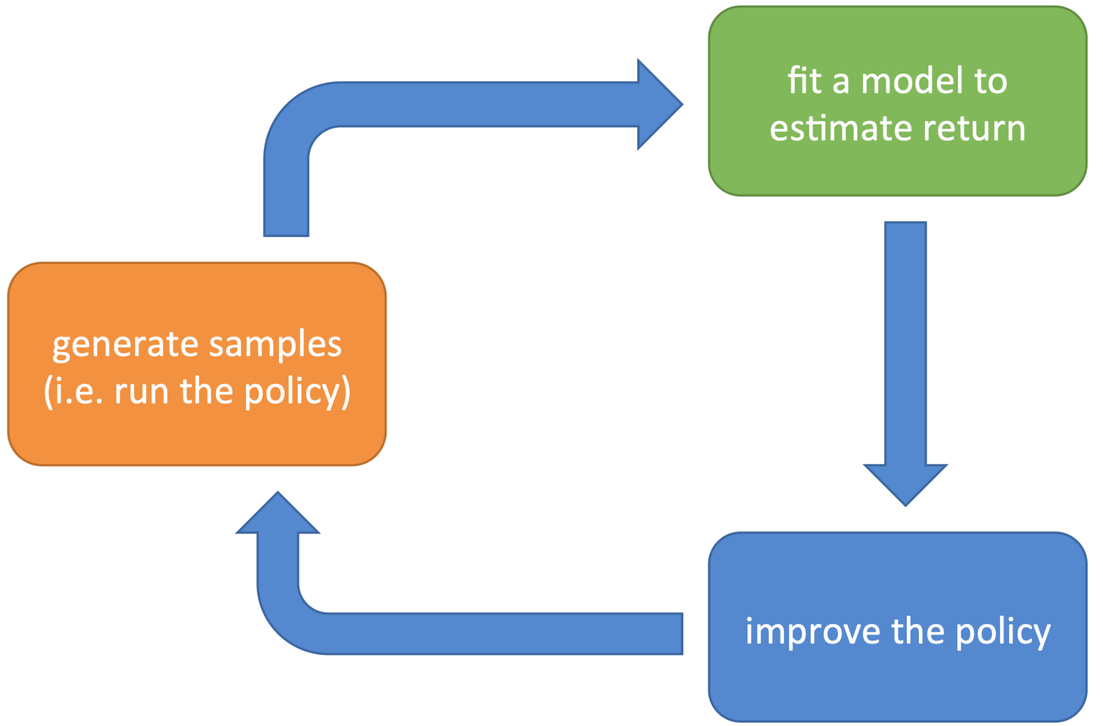
- **Off policy**: able to improve the policy without generating new samples from that policy
- **On policy**: each time the policy is changed, even a little bit, we need to generate new samples



$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$$

Can policy gradients reuse old data?

$$\text{policy gradient:} \quad \nabla_\theta J(\theta) = E_{\tau \sim \pi_\theta(\tau)} \left[\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_t | \mathbf{s}_t) \right) \left(\sum_{t=1}^{T} r(\mathbf{s}_t, \mathbf{a}_t) \right) \right]$$



$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$$

Importance sampling

$$\theta^\star = \arg\max_\theta E_{\tau\sim\pi_\theta(\tau)}\left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t)\right]$$

$$J(\theta) = E_{\tau\sim\pi_\theta(\tau)}\left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t)\right]$$

Importance sampling

$$E_{x\sim p(x)}[f(x)] = \int p(x)f(x)dx$$

$$\int p(x)\frac{q(x)}{q(x)}f(x)dx = \int q(x)\frac{p(x)}{q(x)}f(x)dx$$

$$E_{x\sim q(x)}[\frac{p(x)}{q(x)}f(x)]$$

what if we don't have samples from $\pi_\theta(\tau)$?

we have samples from $\bar{\pi}(\tau)$

$$J(\theta) = E_{\tau\sim\bar{\pi}(\tau)}[\frac{\pi_\theta(\tau)}{\bar{\pi}(\tau)}r(\tau)]$$

Importance sampling in policy gradient

$$\text{policy gradient:} \quad \nabla_\theta J(\theta) = E_{\tau \sim \pi_\theta(\tau)} \left[\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_t | \mathbf{s}_t) \right) \left(\sum_{t=1}^{T} r(\mathbf{s}_t, \mathbf{a}_t) \right) \right]$$

$$\theta^\star = \arg\max_\theta E_{\tau \sim \pi_\theta(\tau)} \left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) \right]$$

$$J(\theta) = E_{\tau \sim \pi_\theta(\tau)} \left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) \right]$$

a convenient identity

$$\pi_\theta(\tau) \nabla_\theta \log \pi_\theta(\tau) = \pi_\theta(\tau) \frac{\nabla_\theta \pi_\theta(\tau)}{\pi_\theta(\tau)} = \nabla_\theta \pi_\theta(\tau)$$

$$J(\theta') = E_{\tau \sim \pi_\theta(\tau)} \left[\frac{\pi_{\theta'}(\tau)}{\pi_\theta(\tau)} r(\tau) \right]$$

$$\frac{\pi_{\theta'}(\tau)}{\pi_\theta(\tau)} = \frac{p(\mathbf{s}_1) \prod_{t=1}^{T} \pi_{\theta'}(\mathbf{a}_t | \mathbf{s}_t) p(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t)}{p(\mathbf{s}_1) \prod_{t=1}^{T} \pi_\theta(\mathbf{a}_t | \mathbf{s}_t) p(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t)}$$

$$\nabla_{\theta'} J(\theta') = E_{\tau \sim \pi_\theta(\tau)} \left[\frac{\nabla_{\theta'} \pi_{\theta'}(\tau)}{\pi_\theta(\tau)} r(\tau) \right] = E_{\tau \sim \pi_\theta(\tau)} \left[\frac{\pi_{\theta'}(\tau)}{\pi_\theta(\tau)} \nabla_{\theta'} \log \pi_{\theta'}(\tau) r(\tau) \right]$$

$$= E_{\tau \sim \pi_\theta(\tau)} \left[\left(\frac{\prod_{t=1}^{T} \pi_{\theta'}(\mathbf{a}_t | \mathbf{s}_t)}{\prod_{t=1}^{T} \pi_\theta(\mathbf{a}_t | \mathbf{s}_t)} \right) \left(\sum_{t=1}^{T} \nabla_{\theta'} \log \pi_{\theta'}(\mathbf{a}_t | \mathbf{s}_t) \right) \left(\sum_{t=1}^{T} r(\mathbf{s}_t, \mathbf{a}_t) \right) \right]$$

Problem with importance sampling in (policy gradient)

$$\nabla_{\theta'} J(\theta') = E_{\tau \sim \pi_\theta(\tau)} \left[\left(\frac{\prod_{t=1}^{T} \pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t)}{\prod_{t=1}^{T} \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)} \right) \left(\sum_{t=1}^{T} \nabla_{\theta'} \log \pi_{\theta'}(\mathbf{a}_t|\mathbf{s}_t) \right) \left(\sum_{t=1}^{T} r(\mathbf{s}_t, \mathbf{a}_t) \right) \right]$$

Let's try it in code!

Importance sampling

$$E_{x \sim p(x)}[f(x)] = \int p(x) f(x) dx$$

$$\int p(x) \frac{q(x)}{q(x)} f(x) dx = \int q(x) \frac{p(x)}{q(x)} f(x) dx$$

$$E_{x \sim q(x)}[\frac{p(x)}{q(x)} f(x)]$$

Solution?

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$$

Stay close to the previous policy!

$$\theta' \leftarrow \arg\max(\theta' - \theta)\nabla_\theta J(\theta) \quad s.t. ||\theta' - \theta||^2 \leq \epsilon$$

Policy not parameters

$$\theta' \leftarrow \arg\max(\theta' - \theta)\nabla_\theta J(\theta) \quad s.t. D_{KL}(\pi_{\theta'}, \pi_\theta) \leq \epsilon$$

Trust region policy optimization (TRPO)

Apply all the tricks:

- Use advantage function to reduce the variance
- Use importance sampling to take multiple gradient steps
- Constrain the optimization objective in the policy space

Trust Region Policy Optimization

John Schulman JOSCHU@EECS.BERKELEY.EDU
Sergey Levine SLEVINE@EECS.BERKELEY.EDU
Philipp Moritz PCMORITZ@EECS.BERKELEY.EDU
Michael Jordan JORDAN@CS.BERKELEY.EDU
Pieter Abbeel PABBEEL@CS.BERKELEY.EDU
University of California, Berkeley, Department of Electrical Engineering and Computer Sciences

Our optimization problem in Equation (13) is exactly equivalent to the following one, written in terms of expectations:

$$\underset{\theta}{\text{maximize}}\ \mathbb{E}_{s\sim\rho_{\theta_{\text{old}}},a\sim q}\left[\frac{\pi_\theta(a|s)}{q(a|s)}Q_{\theta_{\text{old}}}(s,a)\right] \qquad (14)$$

$$\text{subject to } \mathbb{E}_{s\sim\rho_{\theta_{\text{old}}}}\left[D_{\text{KL}}(\pi_{\theta_{\text{old}}}(\cdot|s)\ \|\ \pi_\theta(\cdot|s))\right]\le\delta.$$

Proximal policy optimization (PPO)

Apply all the tricks:

- Use advantage function to reduce the variance
- Use importance sampling to take multiple gradient steps
- Constrain the optimization objective in the policy space

Proximal Policy Optimization Algorithms

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov
OpenAI
{joschu, filip, prafulla, alec, oleg}@openai.com

Let $r_t(\theta)$ denote the probability ratio $r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}$, so $r(\theta_{\text{old}}) = 1$. TRPO maximizes a "surrogate" objective

$$L^{CPI}(\theta) = \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\hat{A}_t\right] = \hat{\mathbb{E}}_t\left[r_t(\theta)\hat{A}_t\right]. \tag{6}$$

The superscript CPI refers to conservative policy iteration [KL02], where this objective was proposed. Without a constraint, maximization of L^{CPI} would lead to an excessively large policy update; hence, we now consider how to modify the objective, to penalize changes to the policy that move $r_t(\theta)$ away from 1.

The main objective we propose is the following:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right] \tag{7}$$

Examples

TRPO applied to continuous control

PPO applied to Dota

The Plan

Policy gradients recap

Variance reduction continued

Policy gradients tricks

Actor-critic

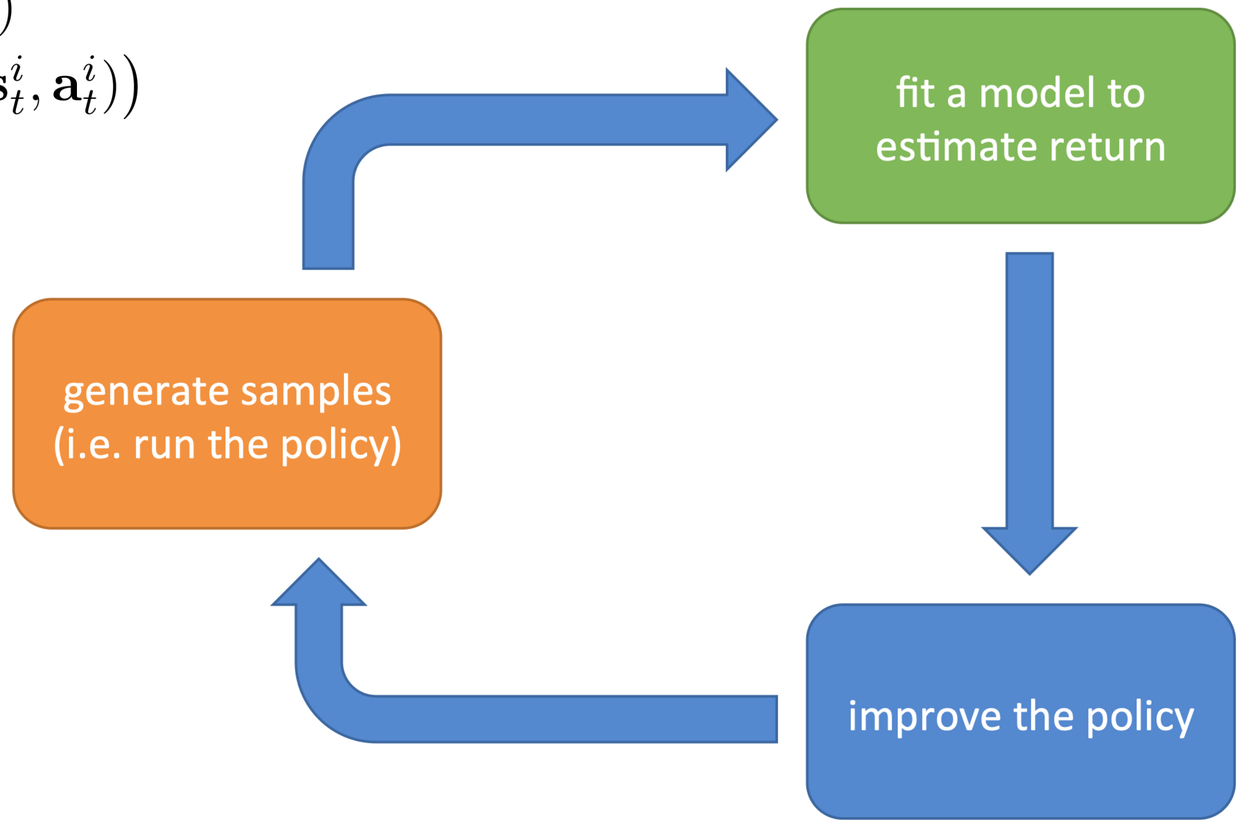
Case studies: robotics & RLHF

REINFORCE algorithm:

1. sample $\{\tau^i\}$ from $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ (run the policy)
2. $\nabla_\theta J(\theta) \approx \sum_i \left(\sum_t \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i|\mathbf{s}_t^i)\right)\left(\sum_t r(\mathbf{s}_t^i, \mathbf{a}_t^i)\right)$
3. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

online actor-critic algorithm:

1. take action $\mathbf{a} \sim \pi_\theta(\mathbf{a}|\mathbf{s})$, get $(\mathbf{s}, \mathbf{a}, \mathbf{s}', r)$
2. update $\hat{V}_\phi^\pi$ using target $r + \gamma \hat{V}_\phi^\pi(\mathbf{s}')$
3. evaluate $\hat{A}^\pi(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \gamma \hat{V}_\phi^\pi(\mathbf{s}') - \hat{V}_\phi^\pi(\mathbf{s})$
4. $\nabla_\theta J(\theta) \approx \nabla_\theta \log \pi_\theta(\mathbf{a}|\mathbf{s}) \hat{A}^\pi(\mathbf{s}, \mathbf{a})$
5. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$



Can we make it more off-policy friendly?

Slide adapted from Sergey Levine

The Plan

Policy gradients recap

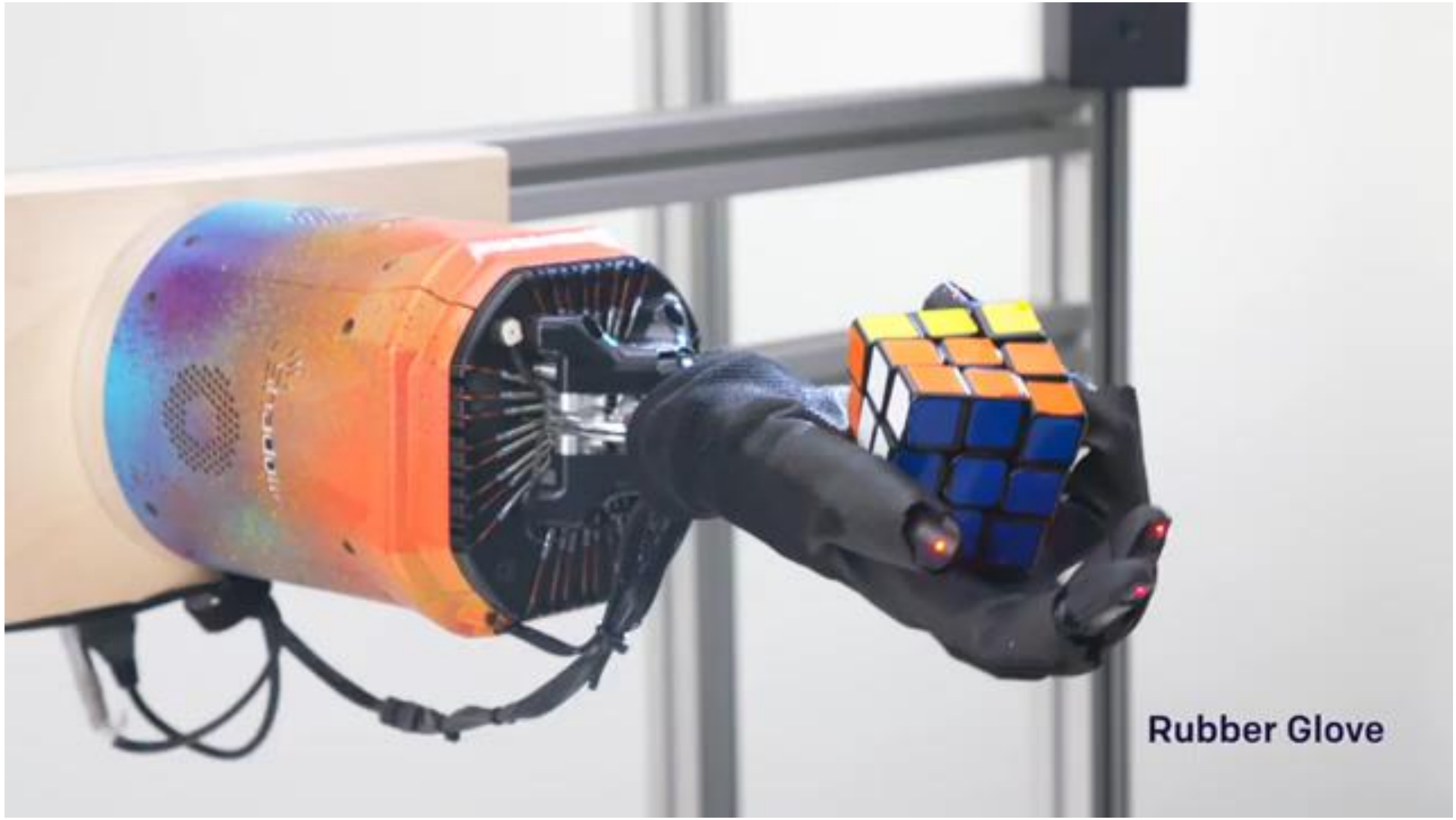
Variance reduction continued

Policy gradients tricks

Actor-critic

Case studies: robotics & RLHF

Case study: PPO applied to robotics



Case study: PPO applied to robotics

Solving the Rubik's Cube with a robot hand is still not easy. Our method currently solves the Rubik's

We train neural networks to solve the Rubik's Cube in simulation using reinforcement learning and Kociemba's algorithm for picking the solution steps.[A] Domain randomization enables networks trained solely in simulation to transfer to a real robot.

:k

into the hand and continue solving.

Simulator physics. We randomize simulator physics parameters such as geometry, friction, gravity, etc. See Section B.1 for details of their ADR parameterization.

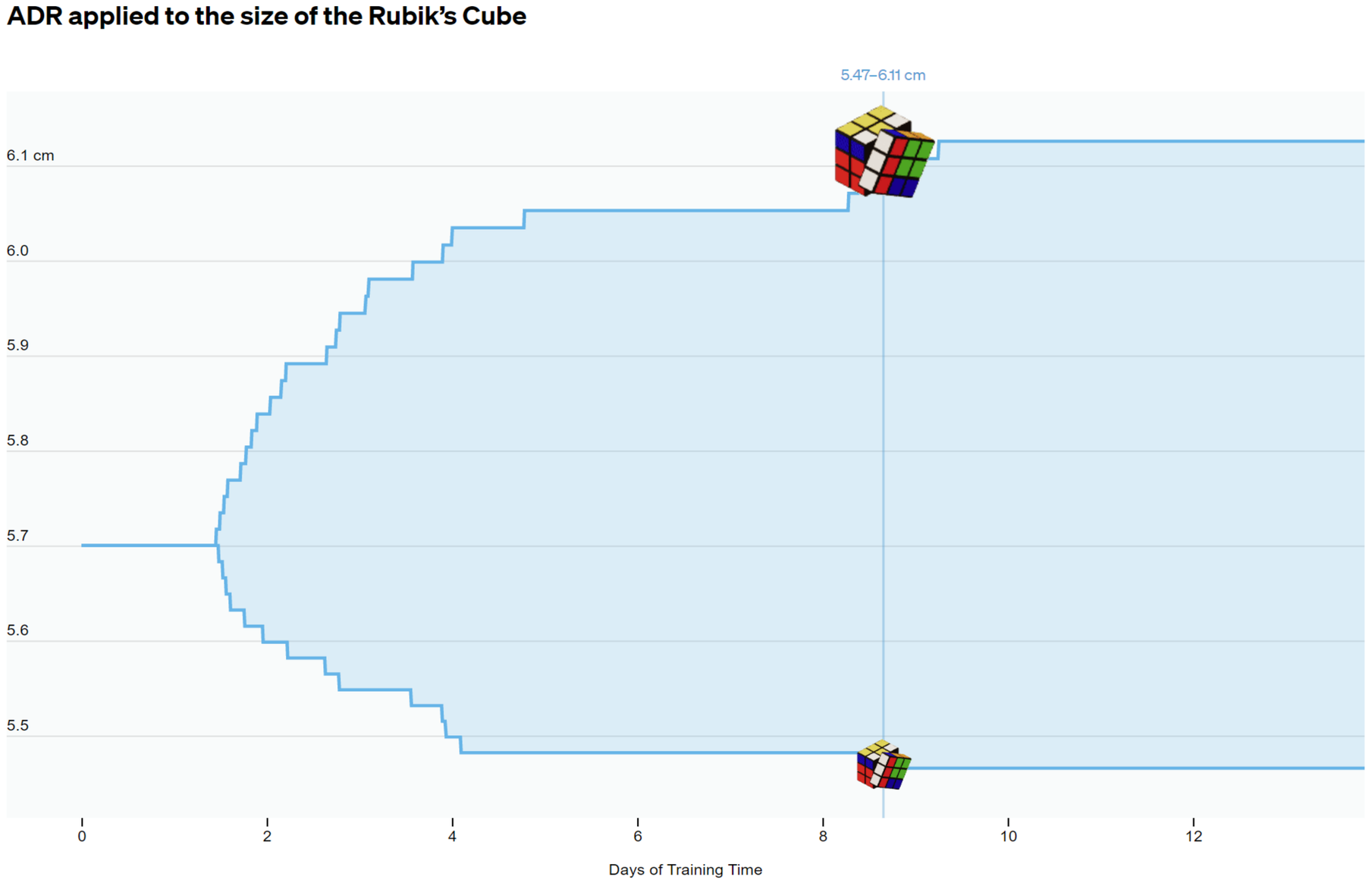
Custom physics. We model additional physical robot effects that are not modelled by the simulator, for example, action latency or motor backlash. See [77, Appendix C.2] for implementation details of these models. We randomize the parameters in these models in a similar way to simulator physics randomizations.

Adversarial. We use an adversarial approach similar to [82, 83] to capture any remaining unmodeled physical effects in the target domain. However, we use random networks instead of a trained adversary. See Section B.3 for details on implementation and ADR parameterization.

Observation. We add Gaussian noise to policy observations to better approximate observation conditions in reality. We apply both correlated noise, which is sampled once at the start of an episode and uncorrelated noise, which is sampled at each time step. We randomize the parameters of the added noise. See Section B.4 for details of their ADR parameterization.

Vision. We randomize several aspects in ORRB [16] to control the rendered scene, including lighting conditions, camera positions and angles, materials and appearances of all the objects, the texture of the background, and the post-processing effects on the rendered images. See Section B.5 for details.

Case study: PPO applied to robotics



Case study: PPO applied to robotics

Table 6: Performance of different policies on the Rubik's cube for a fixed fair scramble goal sequence. We evaluate each policy on the real robot (N=10 trials) and report the mean ± standard error and median number of successes (meaning the total number of successful rotations and flips). We also report two success rates for applying half of a fair scramble ("half") and the other one for fully applying it ("full"). For ADR policies, we report the entropy in nats per dimension (npd). For "Manual DR", we obtain an upper bound on its ADR entropy by running ADR with the policy fixed and report the entropy once the distribution stops changing (marked with an "*").

Policy	Sensing		ADR Entropy	Successes (Real)		Success Rate	
	Pose	Face Angles		Mean	Median	Half	Full
Manual DR	Vision	Giiker	-0.569^* npd	1.8 ± 0.4	2.0	0 %	0 %
ADR	Vision	Giiker	-0.084 npd	3.8 ± 1.0	3.0	0 %	0 %
ADR (XL)	Vision	Giiker	0.467 npd	17.8 ± 4.2	12.5	30 %	10 %
ADR (XXL)	Vision	Giiker	**0.479 npd**	$\mathbf{26.8 \pm 4.9}$	**22.0**	**60 %**	**20 %**
ADR (XXL)	Vision	Vision	**0.479 npd**	12.8 ± 3.4	10.5	20 %	0 %

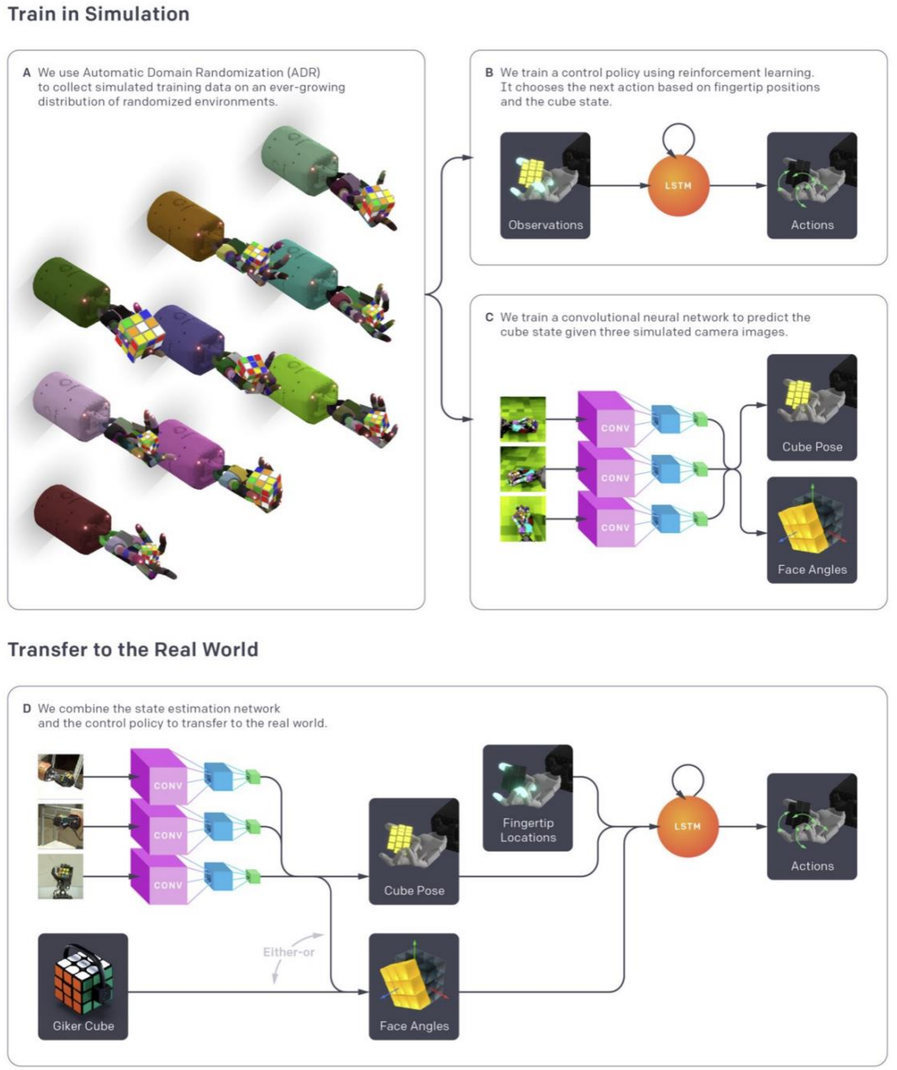
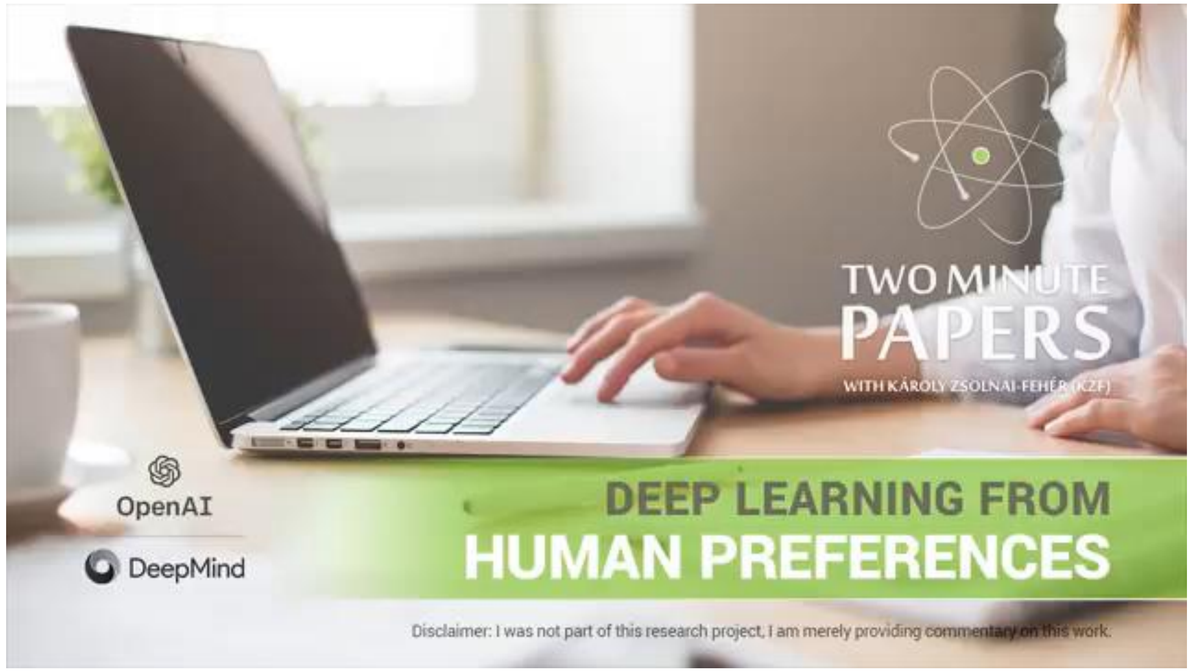
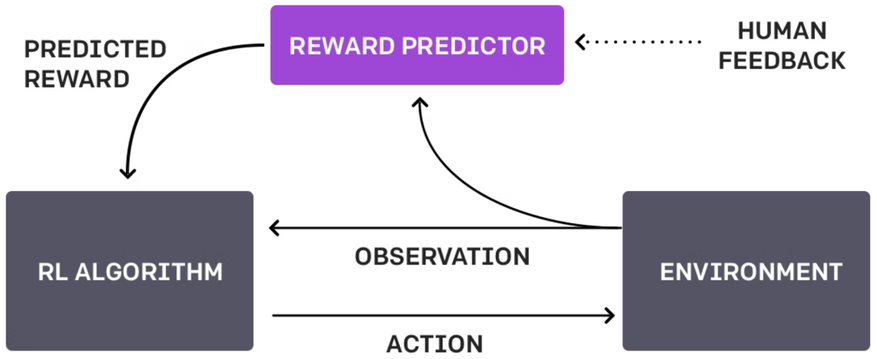


Figure 2: System Overview. (a) We use automatic domain randomization (ADR) to generate a growing distribution of simulations with randomized parameters and appearances. We use this data for both the control policy and vision-based state estimator. (b) The control policy receives observed robot states and rewards from the randomized simulations and learns to solve them using a recurrent neural network and reinforcement learning. (c) The vision-based state estimator uses rendered scenes collected from the randomized simulations and learns to predict the pose as well as face angles of the Rubik's cube using a convolutional neural network (CNN), trained separately from the control policy. (d) To transfer to the real world, we predict the Rubik's cube's pose from 3 real camera feeds with the CNN and measure the robot fingertip locations using a 3D motion capture system. The face angles that describe the internal rotational state of the Rubik's cube are provided by either the same vision state estimator *or* the Giiker cube, a custom cube with embedded sensors and feed it into the policy network.

Case study: PPO applied to LLMs (**speculations**)

RL from human feedback (RLHF)





Reward model

- Imagine a reward function: $R(s; p) \in \mathbb{R}$ for any output s to prompt p
- The reward is higher when humans prefer the output

SAN FRANCISCO, California (CNN) -- A magnitude 4.2 earthquake shook the San Francisco ... overturn unstable objects.

An earthquake hit San Francisco. There was minor property damage, but no injuries.

s_1

$R(s_1; p) = 0.8$

The Bay Area has good weather but is prone to earthquakes and wildfires.

s_2

$R(s_2; p) = 1.2$

[Slide credit: Jesse Mu]

Regularizing pre-trained model

- **Challenge:** how do we ensure that $R(s;p)$ prefer natural language generations?
- Since $R(s;p)$ is trained on natural language inputs, it might fail to assign low scores to unnatural s.
- **Solution:** add regularization term to $R(s;p)$ that penalizes outputs that deviate from natural language.

$$\hat{R}(s;p) \coloneqq R(s;p) - \beta \log\left(\frac{p^{RL}(s)}{p^{PT}(s)}\right)$$

pay a price when $p^{RL}(s) < p^{PT}(s)$

- This is a penalty which prevents us from diverging too far from the pretrained model.

RLHF + PPO (**speculations**)

Step 1

Collect demonstration data, and train a supervised policy.

30k tasks!

A prompt is sampled from our prompt dataset.

A labeler demonstrates the desired output behavior.

This data is used to fine-tune GPT-3 with supervised learning.

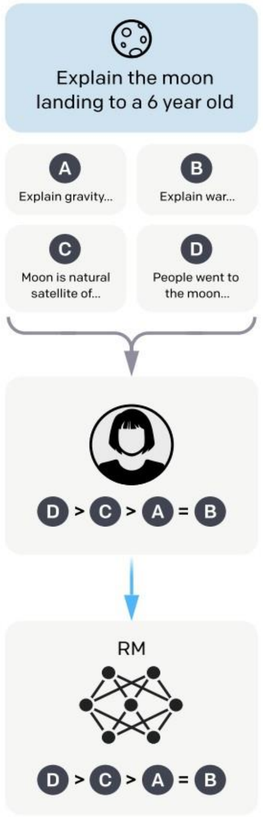
Step 2

Collect comparison data, and train a reward model.

A prompt and several model outputs are sampled.

A labeler ranks the outputs from best to worst.

This data is used to train our reward model.



Step 3

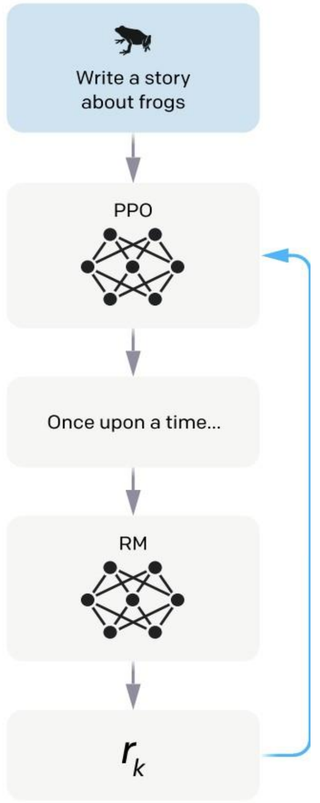
Optimize a policy against the reward model using reinforcement learning.

A new prompt is sampled from the dataset.

The policy generates an output.

The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.



Case study: PPO applied to LLMs (**speculations**)

- What would you try next?
- Is HF more important than RL?
- What do you think the challenges are?

PROMPT *Write a short poem about a wise frog.*

COMPLETION

GPT-3

```
Write a short story in which a character has two different names.

Write a short story in which you try to get something back that you
have lost.

Write a short story in which a character has a bad dream.
```

InstructGPT

```
The frog is a symbol of wisdom
He knows all the secrets of the world
He is a master of disguise
And a great teacher of life
He is a symbol of transformation
And the bringer of change
He is the frog who has seen it all
And knows the meaning of it all
```

Recap

Key learning goals:

- Practical policy gradient implementation tricks & case studies
- Understanding a generic actor-critic method

PG implementation:

- Variance reduction
- Importance sampling and trust region
- RLHF and robotics applications

AC method:

- Uses the advantage fn

Next

Do we even need a policy?

Can we be even more off-policy?

Q learning and its applications