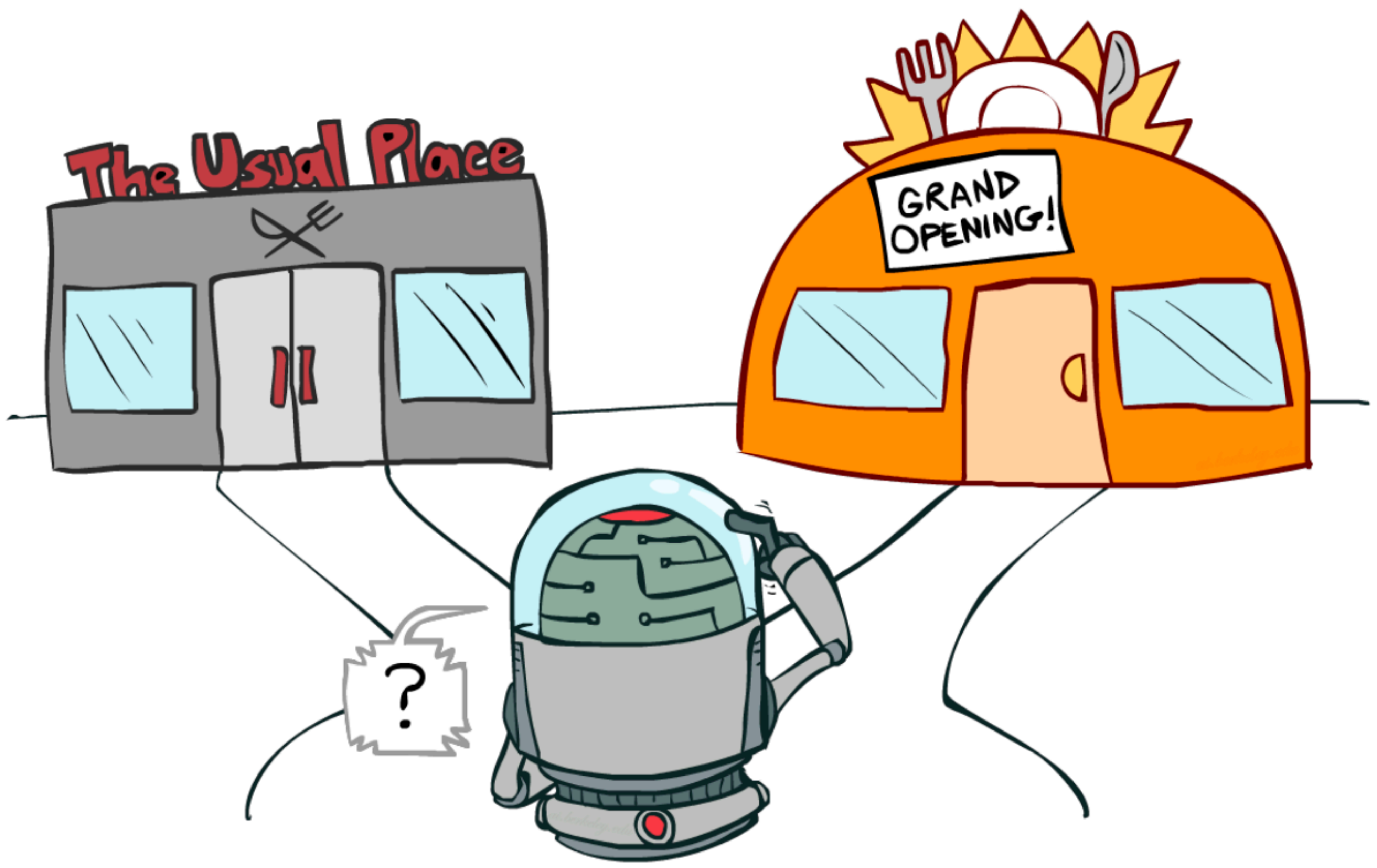
Actor-Critic Methods

Sanjiban Choudhury

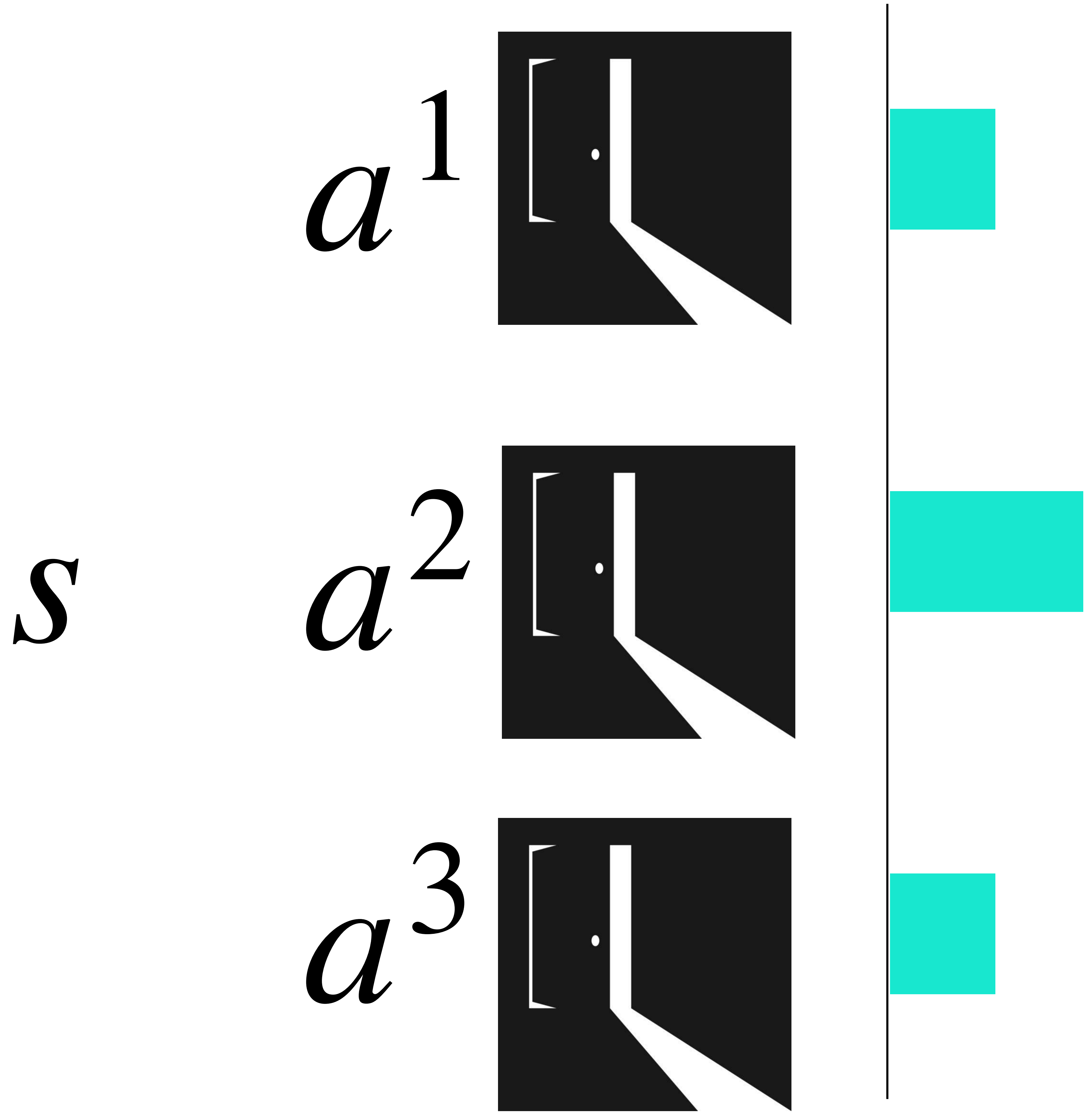
Cornell Bowers CIS
Computer Science

Recap in 60 seconds!

Recap: Two Ingredients of RL



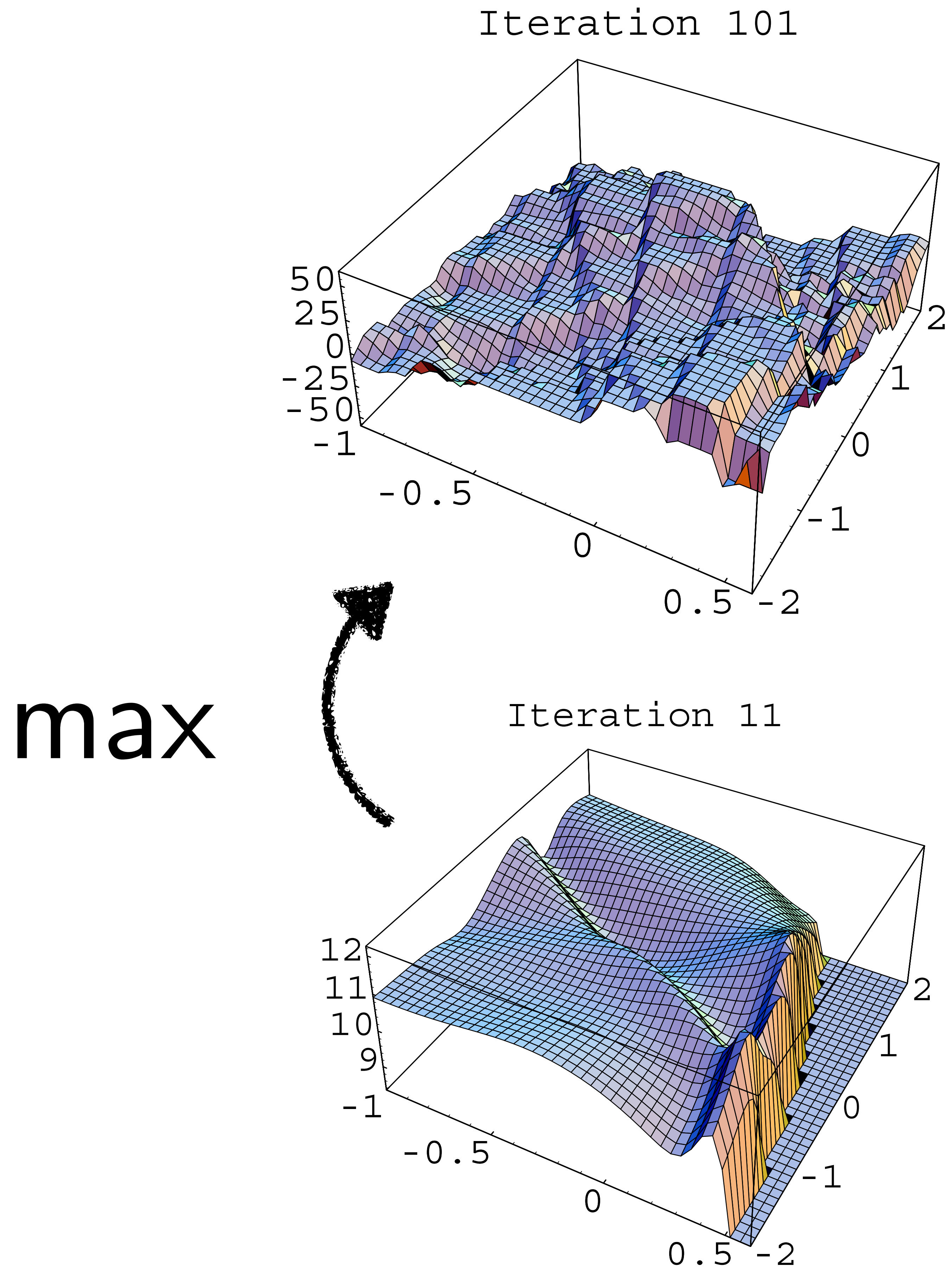
Exploration Exploitation



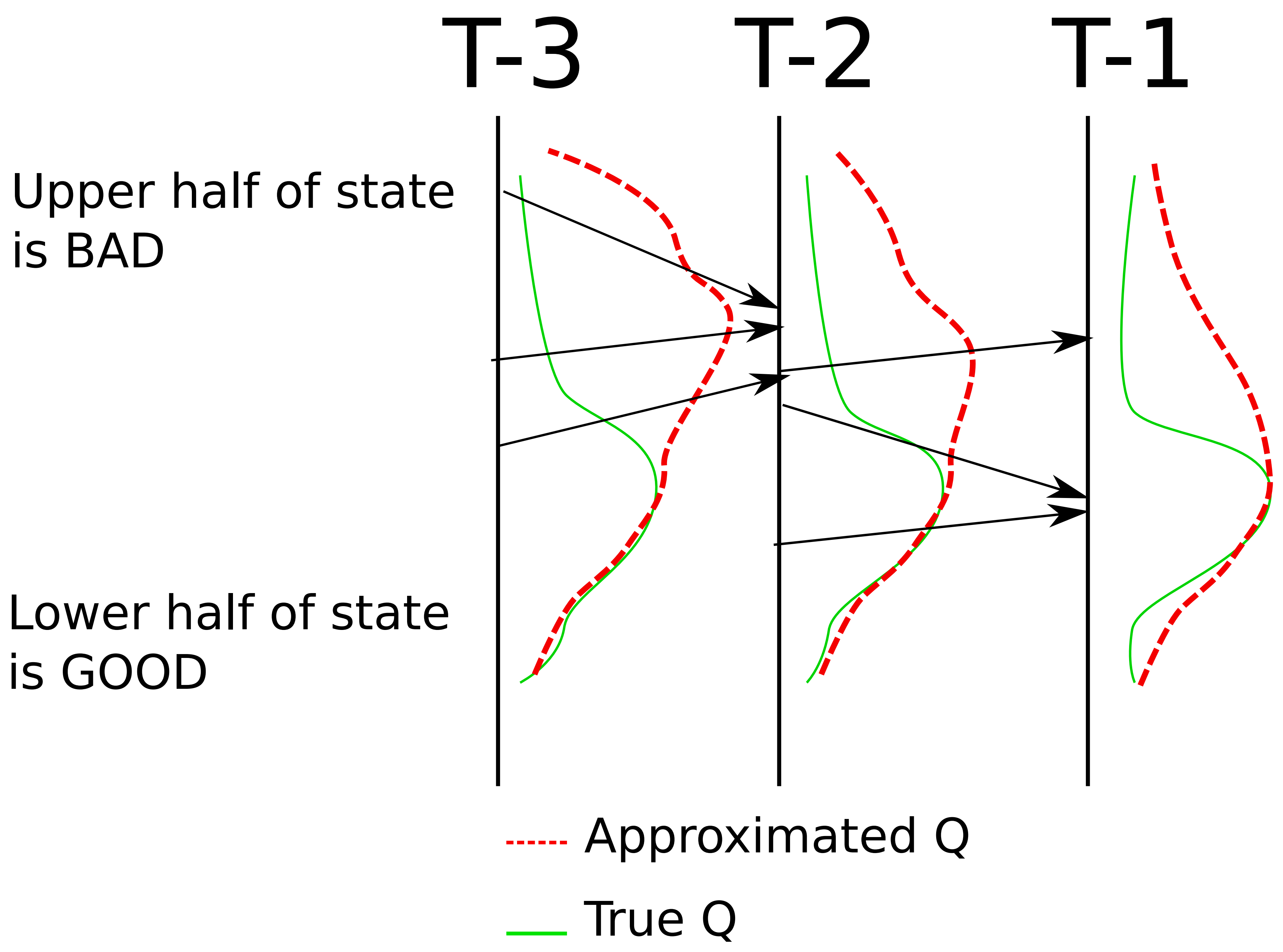
Estimate Values $Q(s, a)$

Curses of Function Approximation

Value Iteration: Bootstrapping



Policy Iteration: Distribution Shift



The Power of a Policy!

All we need at the end of the day is a good policy.

Black box: Try different policies and pick the best one

Gray box: Be smarter, push probability mass on actions that lead to high rewards

$$\nabla_\theta J = E_{p(\xi|\theta)}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t)\, Q^{\pi_\theta}(s_t, a_t)\right]$$

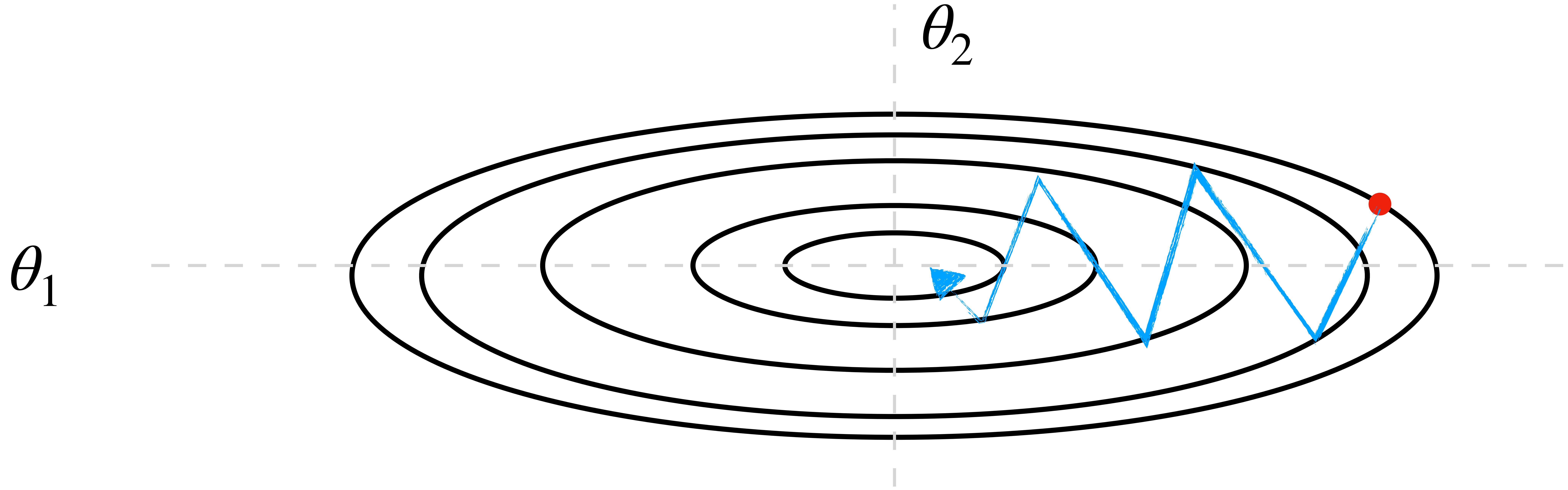
Wait ... how did we get around the distribution shift problems?

The Policy Gradient Theorem

$$\nabla_\theta J = E_{p(\xi|\theta)}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t)\, Q^{\pi_\theta}(s_t, a_t)\right]$$

Is this gradient the best descent direction?

What would gradient descent do here?



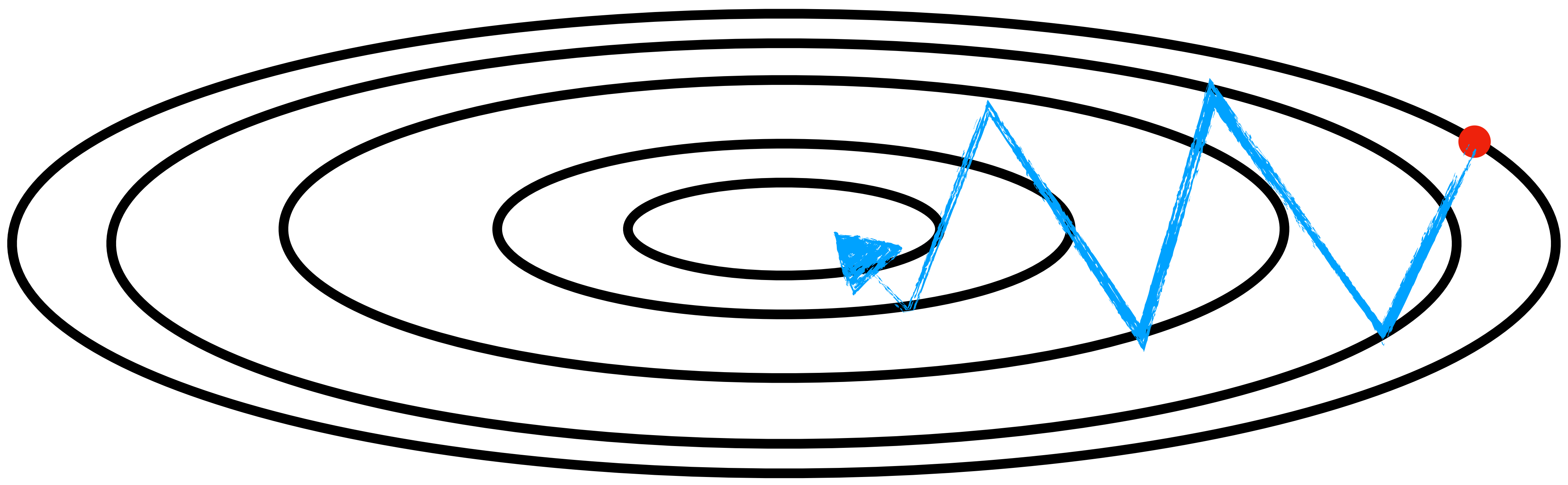
How can we get it to converge better?

Activity!

Think-Pair-Share

Think (30 sec): How can we get gradient descent to converge better in the example below?

Pair: Find a partner



Share (45 sec): Partners exchange ideas

Gradient Descent as Steepest Descent

Gradient Descent is simply Steepest Descent with L2 norm

$$\max_{\Delta\theta} J(\theta + \Delta\theta) \qquad s.t. \qquad \|\Delta\theta\| \leq \epsilon \qquad \longrightarrow \qquad \Delta\theta = \nabla_\theta J(\theta)$$

What would update look like for another norm?

$$\max_{\Delta\theta} J(\theta + \Delta\theta) \qquad s.t. \qquad \Delta\theta^\top G(\theta)\Delta\theta \leq \epsilon \qquad \longrightarrow \qquad \Delta\theta = \frac{1}{2\lambda} G^{-1}(\theta)\,\nabla_\theta J(\theta)$$

What's a good norm for distributions?

What is a good norm for distributions?

$$\max_{\Delta\theta} J(\theta + \Delta\theta)$$

$$\text{s.t. } KL(P(\theta + \Delta\theta) || P(\theta)) \leq \epsilon$$

What is a good norm for distributions?

$$\max_{\Delta\theta} J(\theta + \Delta\theta)$$

~~s.t. $KL(P(\theta + \Delta\theta) || P(\theta)) \leq \epsilon$~~

s.t. $\Delta\theta^T G(\theta) \Delta\theta \leq \epsilon$

Fischer Information Matrix

$$G(\theta) = E_{p_\theta}\left[\nabla_\theta \log(p_\theta)\, \nabla_\theta \log(p_\theta)^\top\right]$$

"Natural" Gradient Descent

Start with an arbitrary initial policy π_θ
while *not converged* **do**
Run simulator with π_θ to collect $\{\xi^{(i)}\}_{i=1}^{N}$
Compute estimated gradient

$$\tilde{\nabla}_\theta J = \frac{1}{N}\sum_{i=1}^{N}\left[\left(\sum_{t=0}^{T-1}\nabla_\theta \log \pi_\theta\left(a_t^{(i)}|s_t^{(i)}\right)\right)R(\xi^{(i)})\right]$$

$$\tilde{G}(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left[\nabla_\theta \log \pi_\theta(a_i|s_i)\nabla_\theta \log \pi_\theta(a_i|s_i)^\top\right]$$

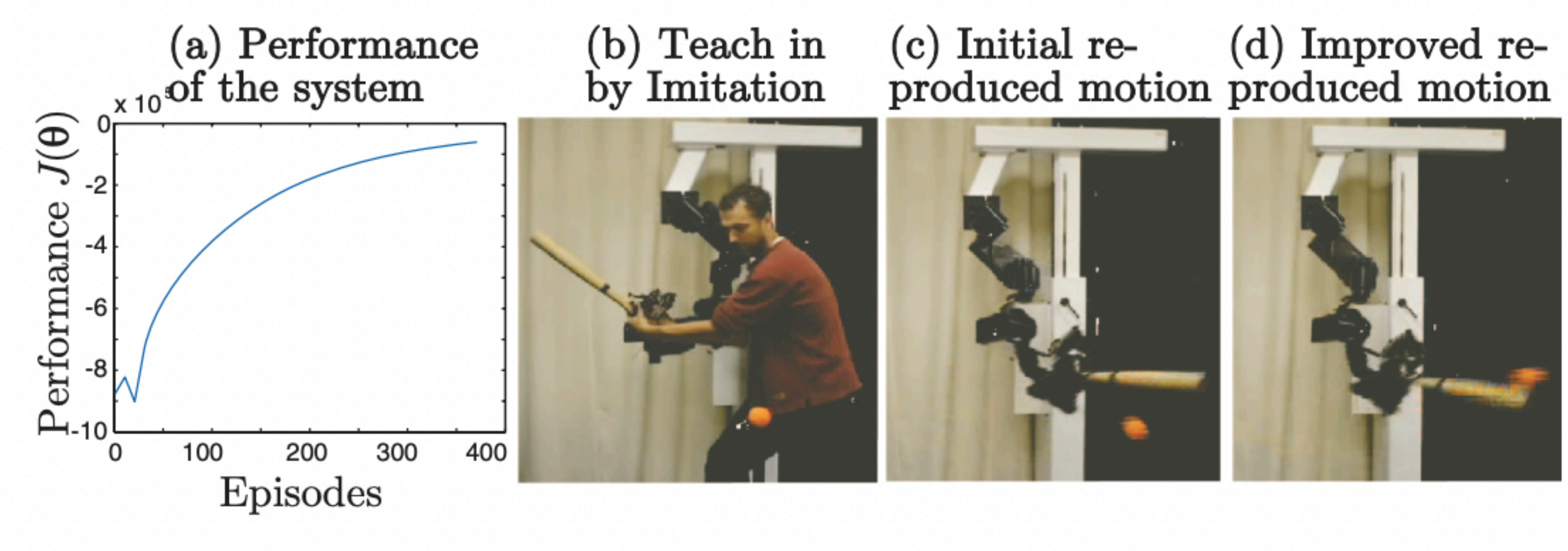
Update parameters $\theta \leftarrow \theta + \alpha \tilde{G}^{-1}(\theta)\tilde{\nabla}_\theta J$.
return π_θ

Modern variants are TRPO, PPO, etc

But does this work on *real robots?*

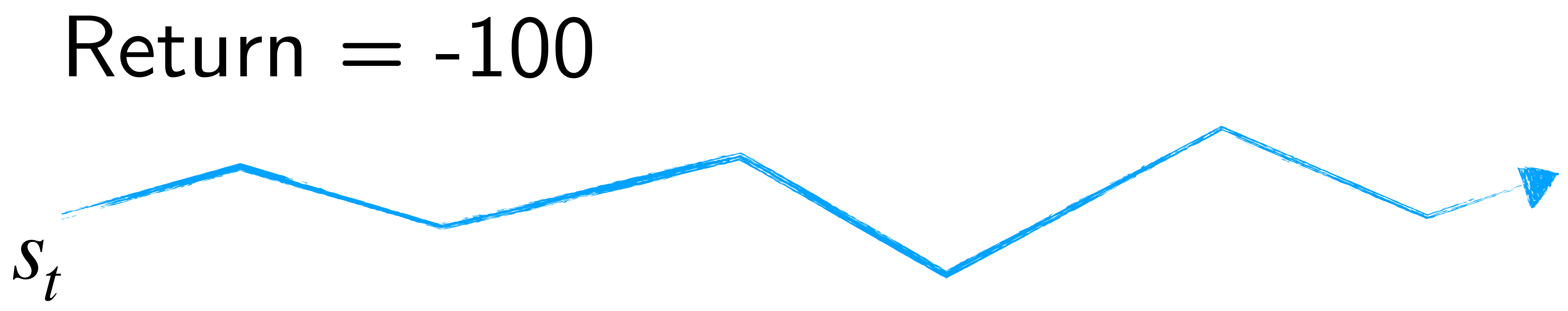
Policy Gradient Methods for Robotics

[Peters and Schaal, 2006]



Initially, we teach a rudimentary stroke by supervised learning as can be seen in Figure 3 (b); however, it fails to reproduce the behavior as shown in (c); subsequently, we improve the performance using the episodic Natural Actor-Critic which yields the performance shown in (a) and the behavior in (d). After approximately 200-300 trials, the ball can be hit properly by the robot.

Consider the following single roll-out

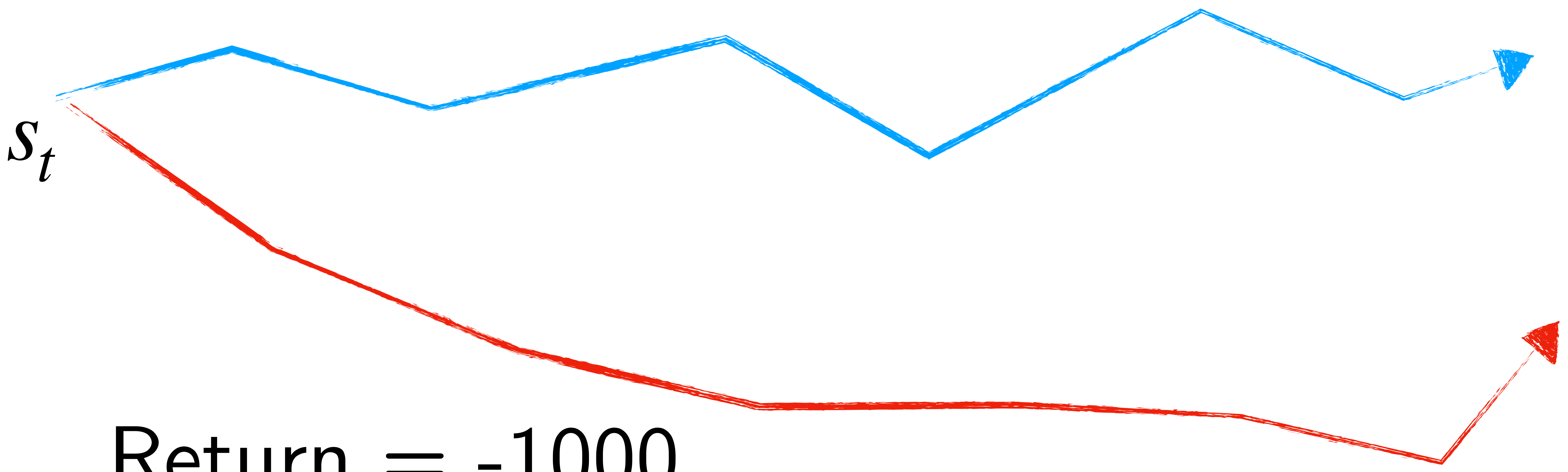


What would the gradient at s_t be?

Is this a good roll-out or a bad roll out?

It depends on other trajectories!

Return = -100



Return = -1000

How can we incorporate *relative* information?

Problem: High Variance

$$\nabla_\theta J = E_{p(\xi|\theta)}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) \boxed{Q^{\pi_\theta}(s_t, a_t)}\right]$$

One of the reasons for the high variance is that the algorithm does not know how well the trajectories perform compared to other trajectories.

Solution: Subtract a baseline!

$$\nabla_\theta J = E_{d^{\pi_\theta}(s)} E_{\pi_\theta(a|s)} \left[\nabla_\theta \log(\pi_\theta(a|s) \left(Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s) \right) \right].$$

Prove this does not change the gradient!

$$= E_{d^{\pi_\theta}(s)} E_{\pi_\theta(a|s)} \left[\nabla_\theta \log(\pi_\theta(a|s) A^{\pi_\theta}(s,a) \right]$$

Recap (again) in 60 seconds!

1. Local Optima: Use Exploration Distribution

2. Distribution Shift: *Natural* Gradient Descent

3. High Variance: Subtract baseline

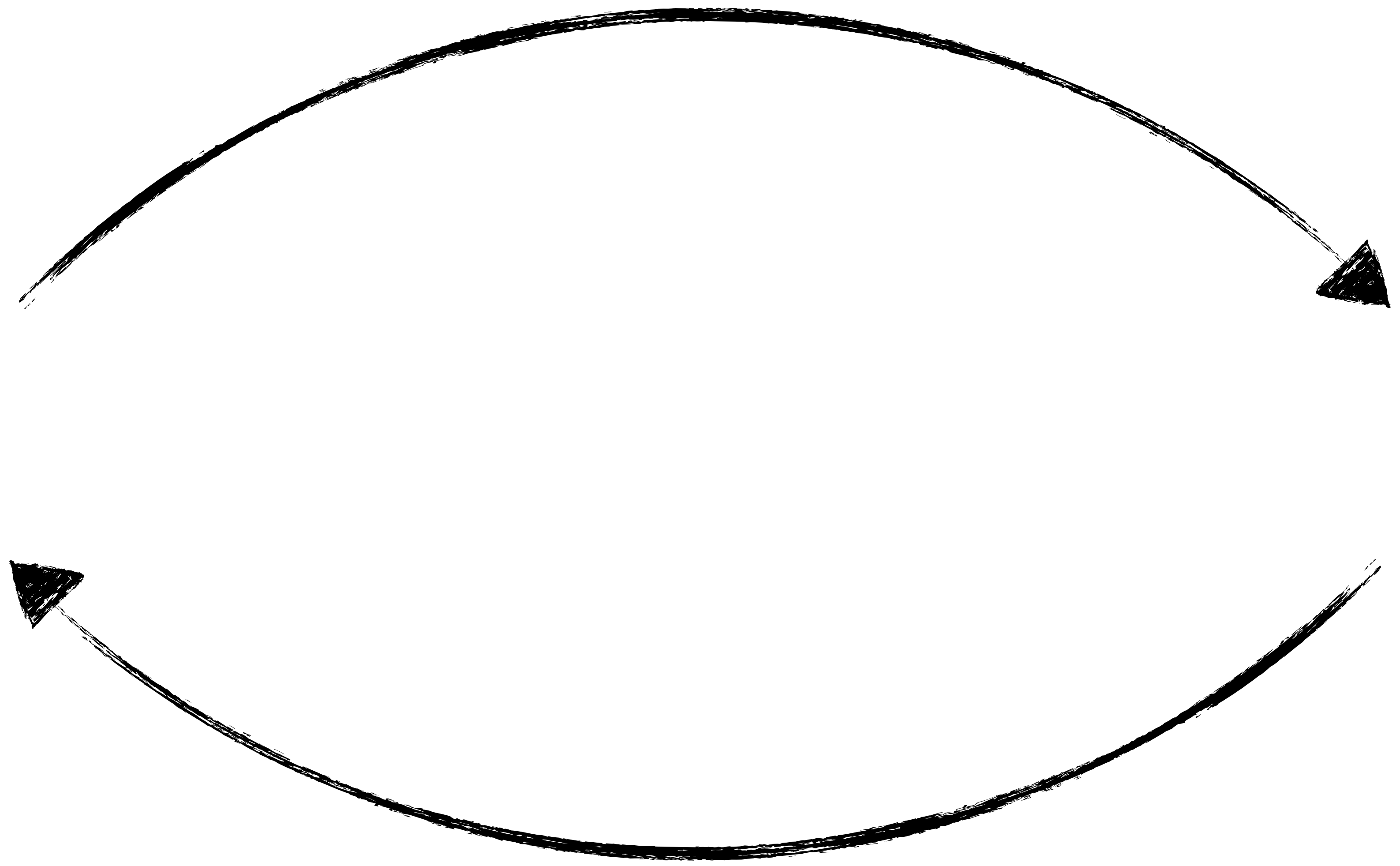
If we are estimating values … can we bring back MC and TD?

Monte-Carlo	Temporal Difference
$V(s) \leftarrow V(s) + \alpha(G_t - V(s))$	$V(s) \leftarrow V(s) + \alpha(c + \gamma V(s') - V(s))$
Zero Bias	Can have bias
High Variance	Low Variance
Always convergence (Just have to wait till heat death of the universe)	May *not* converge if using function approximation

Actor-Critic Algorithms

Actor

Critic



Policy improvement of π

Estimates value functions $Q^\pi_\phi / V^\pi_\phi / A^\pi_\phi$

Natural Gradient Descent

TD, MC

The General Actor Critic Framework

batch actor-critic algorithm:

1. sample $\{\mathbf{s}_i, \mathbf{a}_i\}$ from $\pi_\theta(\mathbf{a}|\mathbf{s})$ (run it on the robot)
2. fit $\hat{V}^\pi_\phi(\mathbf{s})$ to sampled reward sums (TD, MC)
3. evaluate $\hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i) = r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \hat{V}^\pi_\phi(\mathbf{s}'_i) - \hat{V}^\pi_\phi(\mathbf{s}_i)$
4. $\nabla_\theta J(\theta) \approx \sum_i \nabla_\theta \log \pi_\theta(\mathbf{a}_i|\mathbf{s}_i) \hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i)$
5. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Credit: Sergey Levine

Practical Issues and Fixes

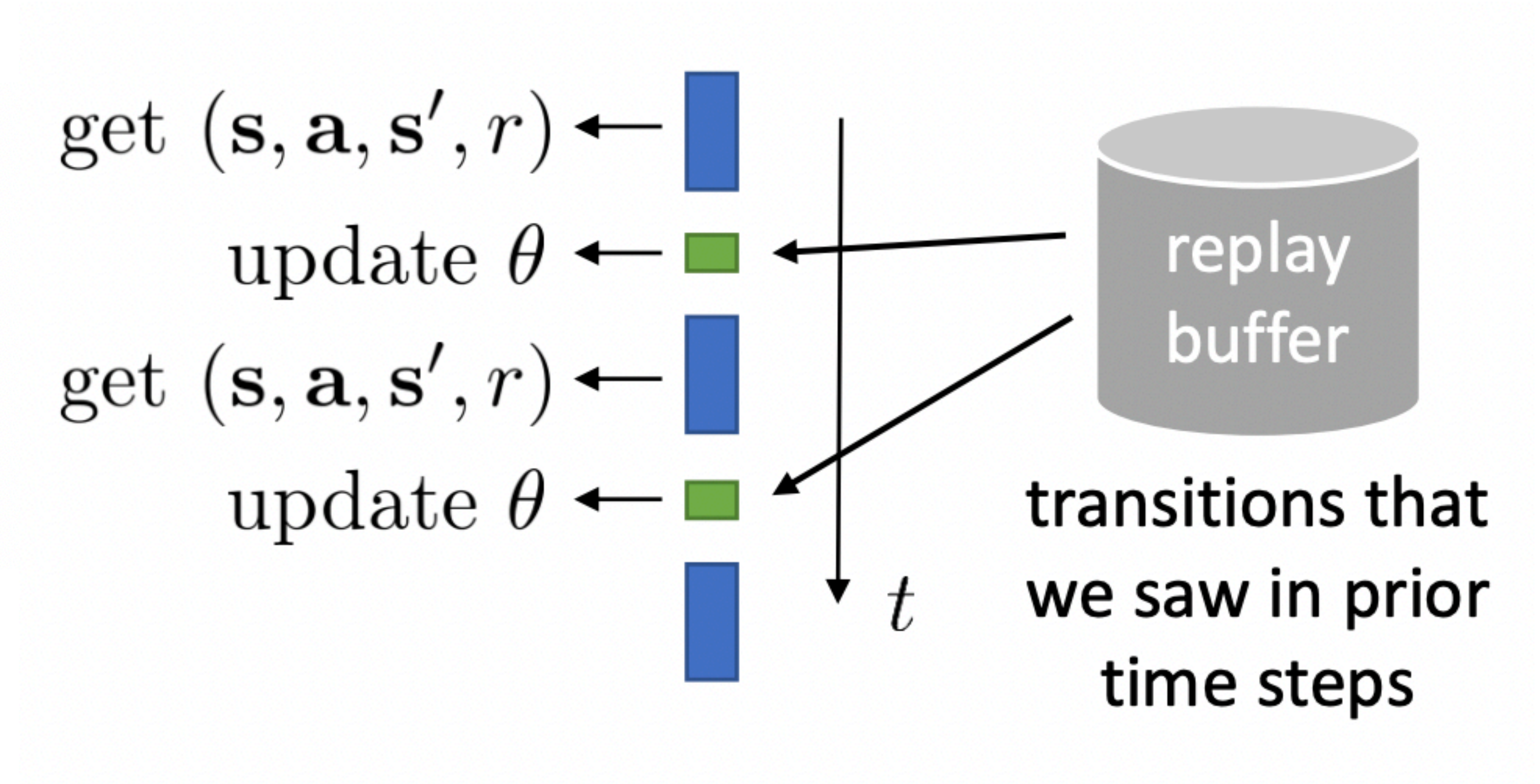
Problem 1: How do we make Actor Critic off-policy?

batch actor-critic algorithm:

1. sample $\{\mathbf{s}_i, \mathbf{a}_i\}$ from $\pi_\theta(\mathbf{a}|\mathbf{s})$ (run it on the robot)
2. fit $\hat{V}_\phi^\pi(\mathbf{s})$ to sampled reward sums
3. evaluate $\hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i) = r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \hat{V}_\phi^\pi(\mathbf{s}'_i) - \hat{V}_\phi^\pi(\mathbf{s}_i)$
4. $\nabla_\theta J(\theta) \approx \sum_i \nabla_\theta \log \pi_\theta(\mathbf{a}_i|\mathbf{s}_i) \hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i)$
5. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Credit: Sergey Levine

Problem 1: How do we make Actor Critic off-policy?

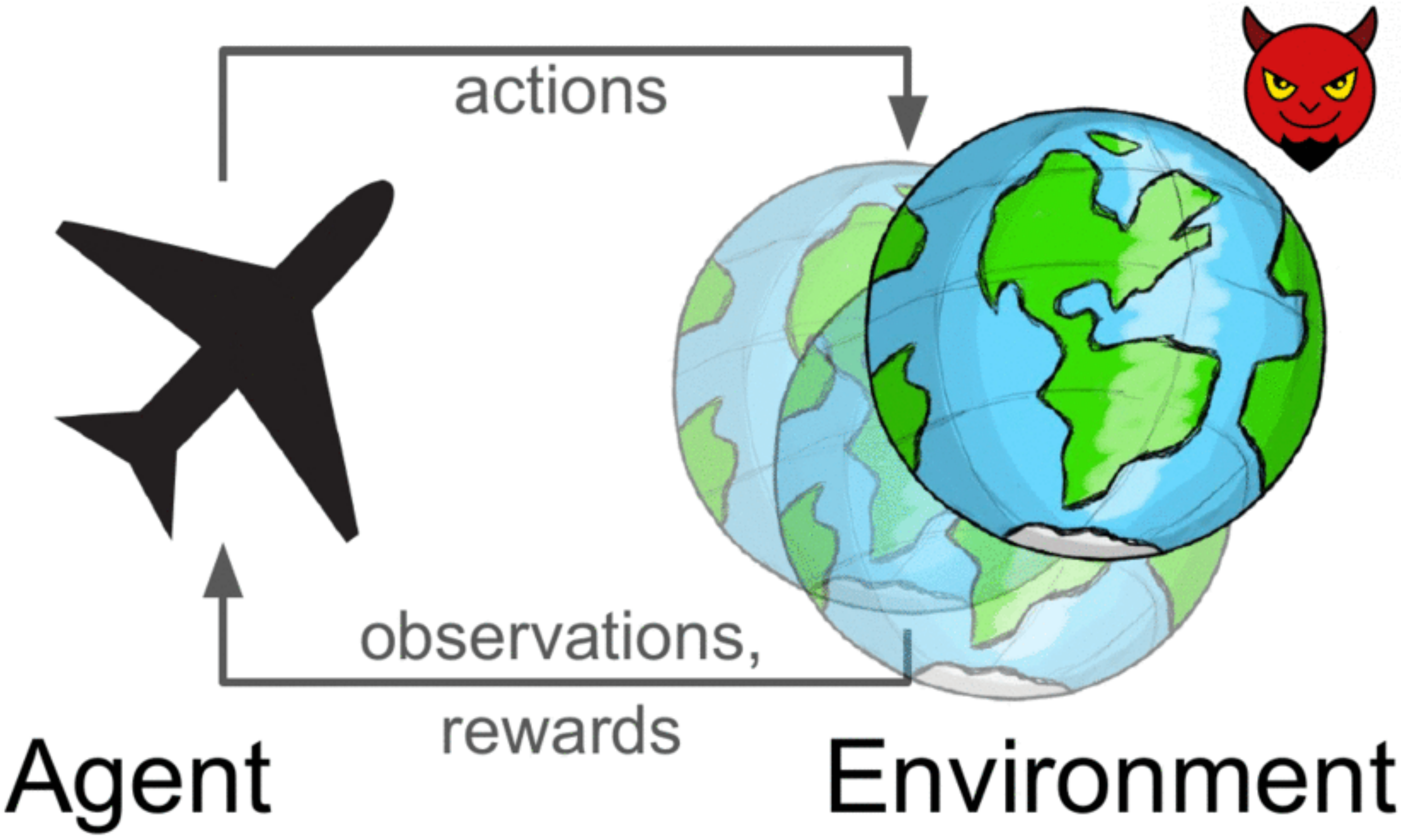


Credit: Sergey Levine

Solution: Carefully assign credit to correct actions!

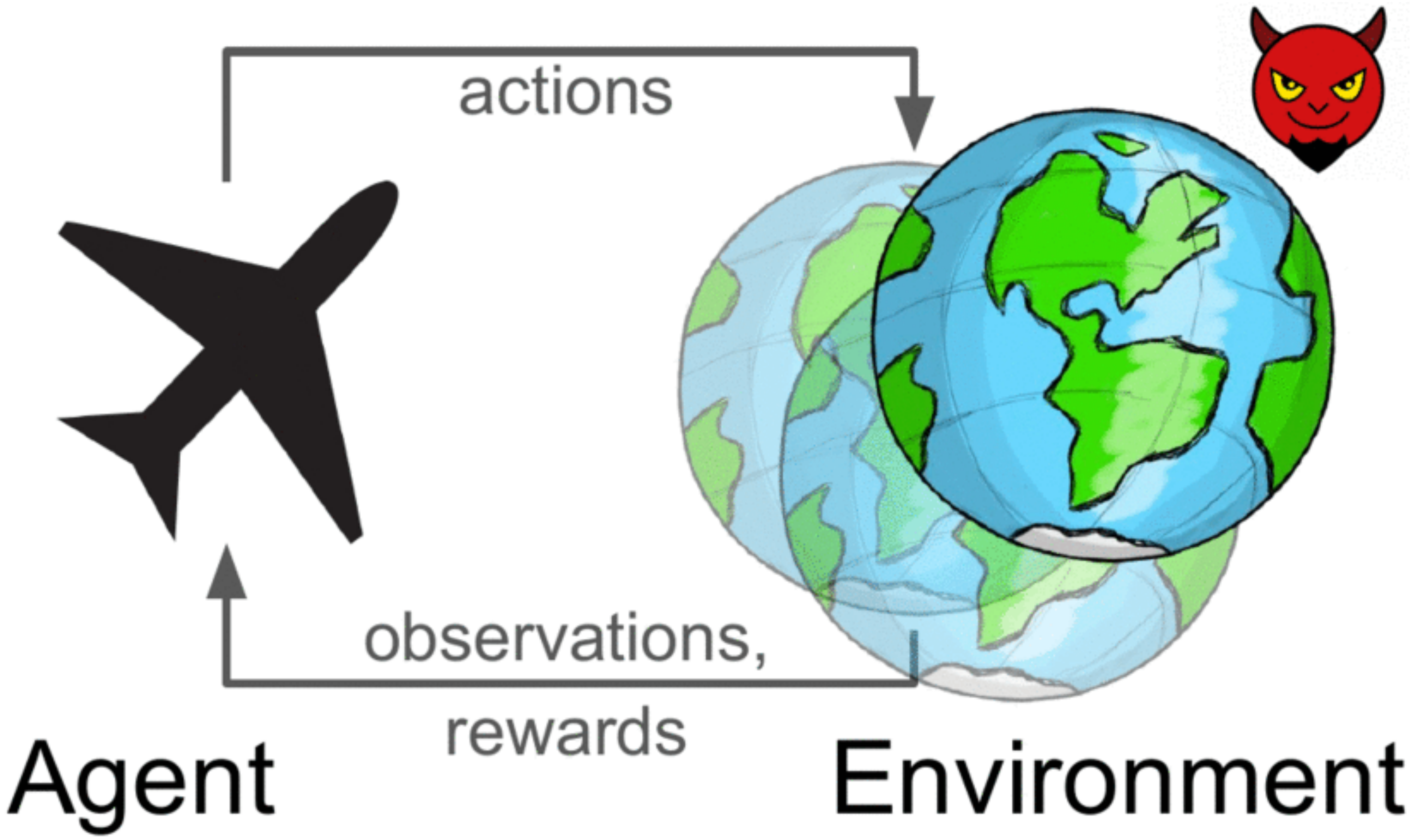
1. take action $\mathbf{a} \sim \pi_\theta(\mathbf{a}|\mathbf{s})$, get $(\mathbf{s}, \mathbf{a}, \mathbf{s}', r)$, store in $\mathcal{R}$
2. sample a batch $\{\mathbf{s}_i, \mathbf{a}_i, r_i, \mathbf{s}'_i\}$ from buffer $\mathcal{R}$
3. update $\hat{Q}^\pi_\phi$ using targets $y_i = r_i + \gamma \hat{Q}^\pi_\phi(\mathbf{s}'_i, \mathbf{a}'_i)$ for each $\mathbf{s}_i, \mathbf{a}_i$
4. $\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_i \nabla_\theta \log \pi_\theta(\mathbf{a}^\pi_i|\mathbf{s}_i) \hat{Q}^\pi(\mathbf{s}_i, \mathbf{a}^\pi_i)$ where $\mathbf{a}^\pi_i \sim \pi_\theta(\mathbf{a}|\mathbf{s}_i)$
5. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Problem 2: How can we be robust to changes in the environment?



Credit: Ben Eyesenbach

Problem 2: How can we be robust to changes in the environment?



$$\max_{\pi} \min_{\tilde{p} \in \tilde{\mathcal{P}}, \tilde{r} \in \tilde{\mathcal{R}}} \mathbb{E}_{\tilde{p}(\mathbf{s_{t+1}} | \mathbf{s_t}, \mathbf{a_t}), \pi(\mathbf{a_t} | \mathbf{s_t})} \left[\sum_{t=1}^{T} \tilde{r}(\mathbf{s_t}, \mathbf{a_t}) \right].$$

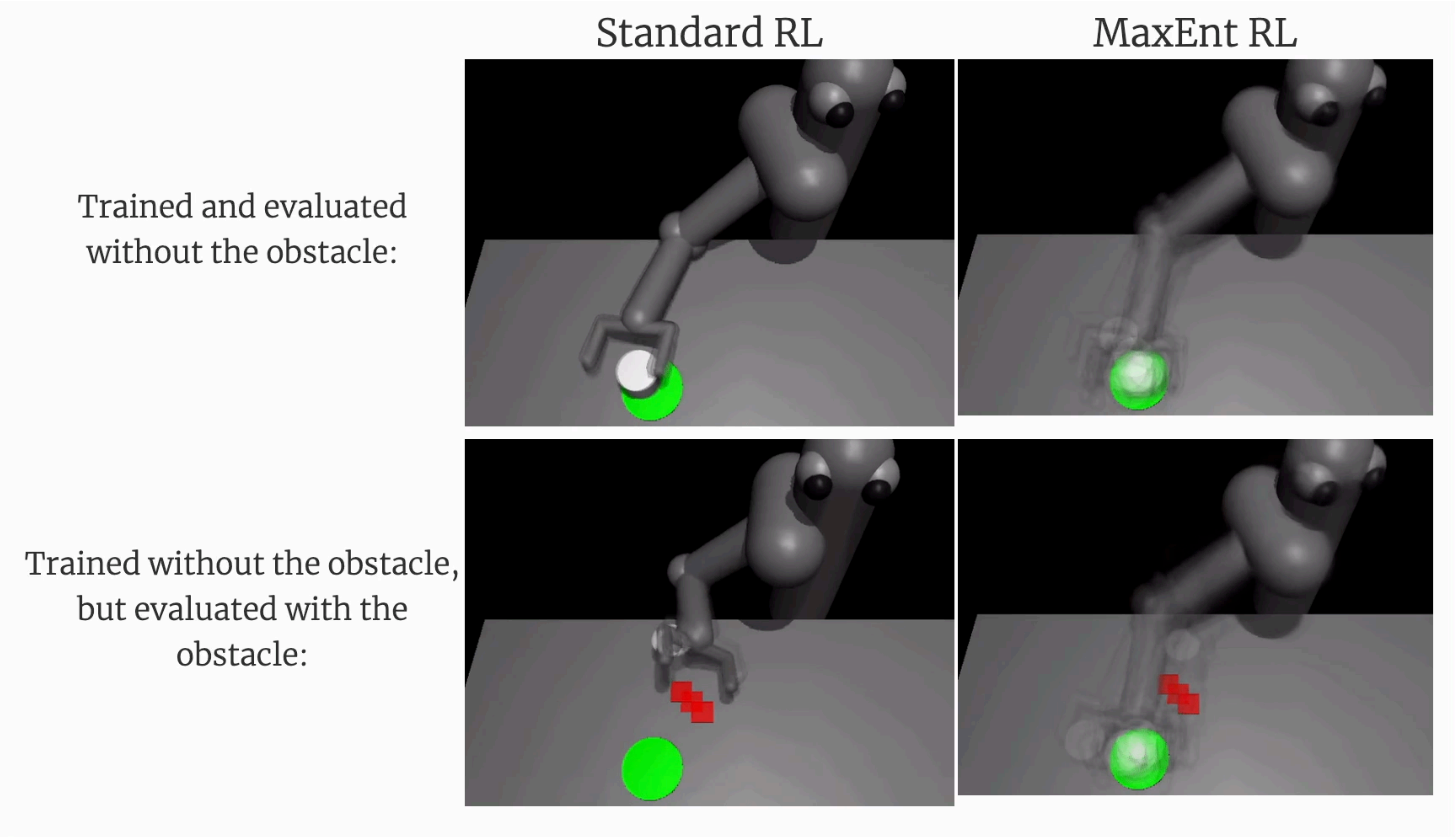
Credit: Ben Eyesenbach

Solution: Use Maximum Entropy RL!

$$J_{\text{MaxEnt}}(\pi; p, r) \triangleq \mathbb{E}_{\mathbf{a_t} \sim \pi(\mathbf{a_t}|\mathbf{s_t}), \mathbf{s_{t+1}} \sim p(\mathbf{s_{t+1}}|\mathbf{s_t}, \mathbf{a_t})} \left[\sum_{t=1}^{T} r(\mathbf{s_t}, \mathbf{a_t}) + \alpha \mathcal{H}_\pi[\mathbf{a_t} \mid \mathbf{s_t}] \right]$$

Intuition: There are many policies that can achieve the same cumulative rewards. MaxEntRL keeps alive all of those policies. Learns many different ways to solve the same task.

Solution: Use Maximum Entropy RL!



Haarnoja 2018

"Soft" Actor Critic

Actor

$$\pi_{\text{new}} = \arg\min_{\pi' \in \Pi} \mathrm{D}_{\mathrm{KL}}\left(\pi'(\,\cdot\,|\mathbf{s}_t) \,\middle\|\, \frac{\exp\left(Q^{\pi_{\text{old}}}(\mathbf{s}_t, \,\cdot\,)\right)}{Z^{\pi_{\text{old}}}(\mathbf{s}_t)}\right)$$

"Soft" Policy Improvement

Critic

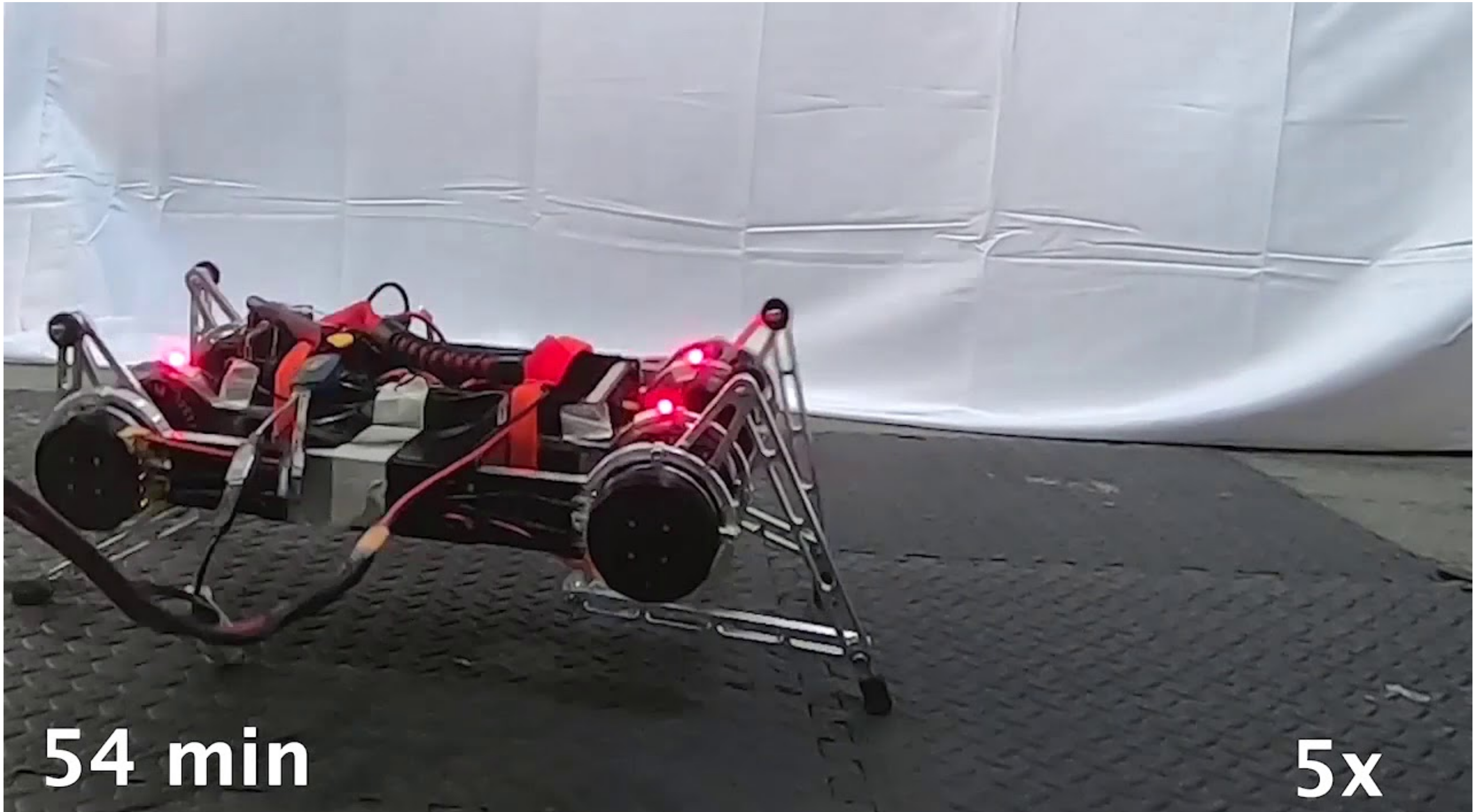
$$\mathcal{T}^{\pi} Q(\mathbf{s}_t, \mathbf{a}_t) \triangleq r(\mathbf{s}_t, \mathbf{a}_t) + \gamma\, \mathbb{E}_{\mathbf{s}_{t+1} \sim p}\left[V(\mathbf{s}_{t+1})\right],$$

where

$$V(\mathbf{s}_t) = \mathbb{E}_{\mathbf{a}_t \sim \pi}\left[Q(\mathbf{s}_t, \mathbf{a}_t) - \log \pi(\mathbf{a}_t|\mathbf{s}_t)\right]$$

"Soft" Value Evaluation

"Soft" Actor Critic



From Policy Gradient to Policy Search

Algorithm 1 Advantage-Weighted Regression

$\pi_1 \leftarrow$ random policy
$\mathcal{D} \leftarrow \emptyset$
for iteration $k = 1, ..., k_{\max}$ **do**
    add trajectories $\{\tau_i\}$ sampled via π_k to $\mathcal{D}$
    $V_k^{\mathcal{D}} \leftarrow \arg\min_V \ \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mathcal{D}}\left[\left|\left|\mathcal{R}^{\mathcal{D}}_{\mathbf{s},\mathbf{a}} - V(\mathbf{s})\right|\right|^2\right]$ Supervised Learning!
    $\pi_{k+1} \leftarrow \arg\max_\pi \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\mathcal{D}}\left[\log \pi(\mathbf{a}|\mathbf{s}) \exp\left(\frac{1}{\beta}\left(\mathcal{R}^{\mathcal{D}}_{\mathbf{s},\mathbf{a}} - V_k^{\mathcal{D}}(\mathbf{s})\right)\right)\right]$ Supervised Learning!
end for

Peng et al, 2019

tl;dr

"Natural" Gradient Descent

Start with an arbitrary initial policy π_θ
while *not converged* **do**
Run simulator with π_θ to collect $\{\xi^{(i)}\}_{i=1}^N$
Compute estimated gradient

$$\tilde{\nabla}_\theta J = \frac{1}{N}\sum_{i=1}^{N}\left[\left(\sum_{t=0}^{T-1}\nabla_\theta \log \pi_\theta\left(a_t^{(i)}|s_t^{(i)}\right)\right) R(\xi^{(i)})\right]$$

$$\tilde{G}(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left[\nabla_\theta \log \pi_\theta(a_i|s_i)\nabla_\theta \log \pi_\theta(a_i|s_i)^\top\right]$$

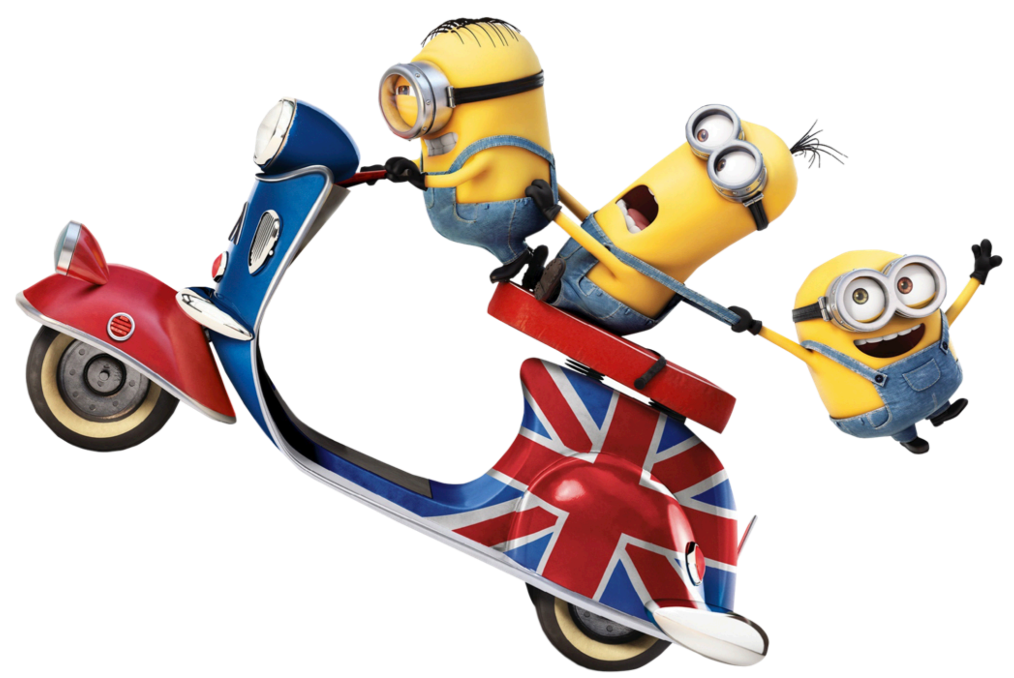
Update parameters $\theta \leftarrow \theta + \alpha \tilde{G}^{-1}(\theta)\tilde{\nabla}_\theta J$.
return π_θ

Modern variants are TRPO, PPO, etc

15

Recap (again) in 60 seconds!

1. Local Optima: Use Exploration Distribution
2. Distribution Shift: *Natural* Gradient Descent
3. High Variance: Subtract baseline



23

The General Actor Critic Framework

batch actor-critic algorithm:

1. sample $\{\mathbf{s}_i, \mathbf{a}_i\}$ from $\pi_\theta(\mathbf{a}|\mathbf{s})$ (run it on the robot)
2. fit $\hat{V}^\pi_\phi(\mathbf{s})$ to sampled reward sums (TD, MC)
3. evaluate $\hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i) = r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \hat{V}^\pi_\phi(\mathbf{s}'_i) - \hat{V}^\pi_\phi(\mathbf{s}_i)$
4. $\nabla_\theta J(\theta) \approx \sum_i \nabla_\theta \log \pi_\theta(\mathbf{a}_i|\mathbf{s}_i)\hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i)$
5. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Credit: Sergey Levine 26