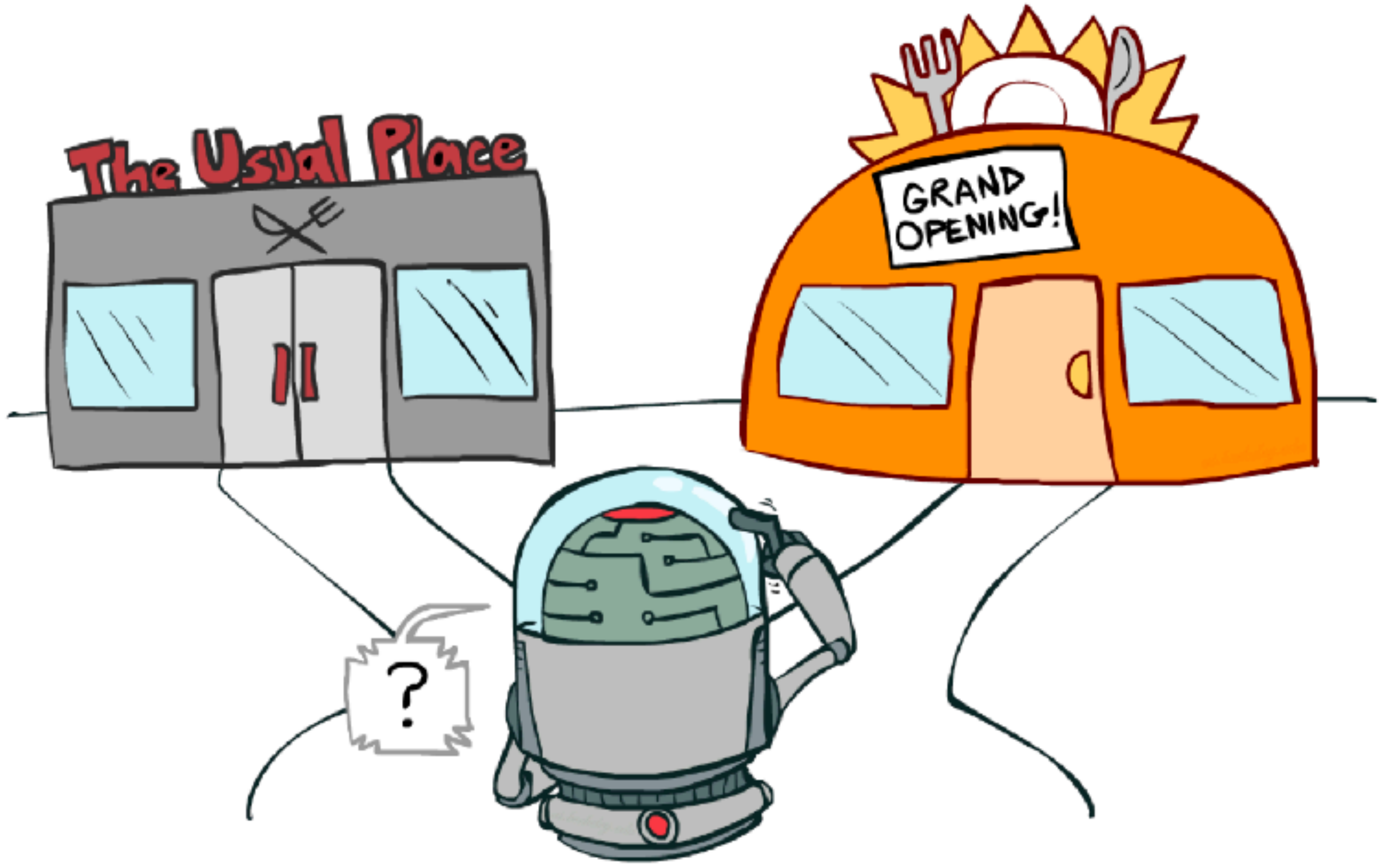
Actor-Critic Methods

Sanjiban Choudhury

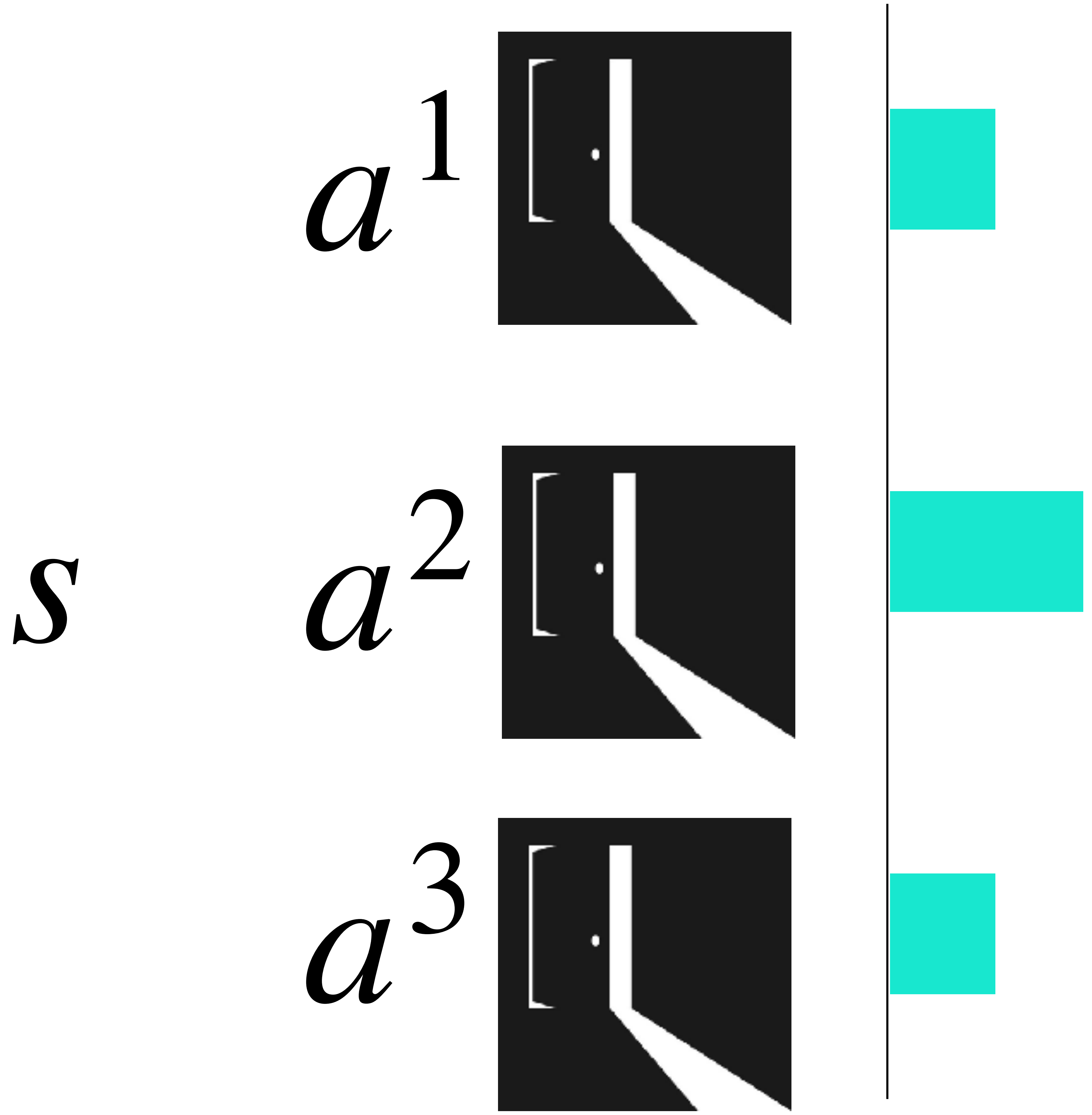
Cornell Bowers C·IS
Computer Science

Recap in
60 seconds!

Recap: Two Ingredients of RL



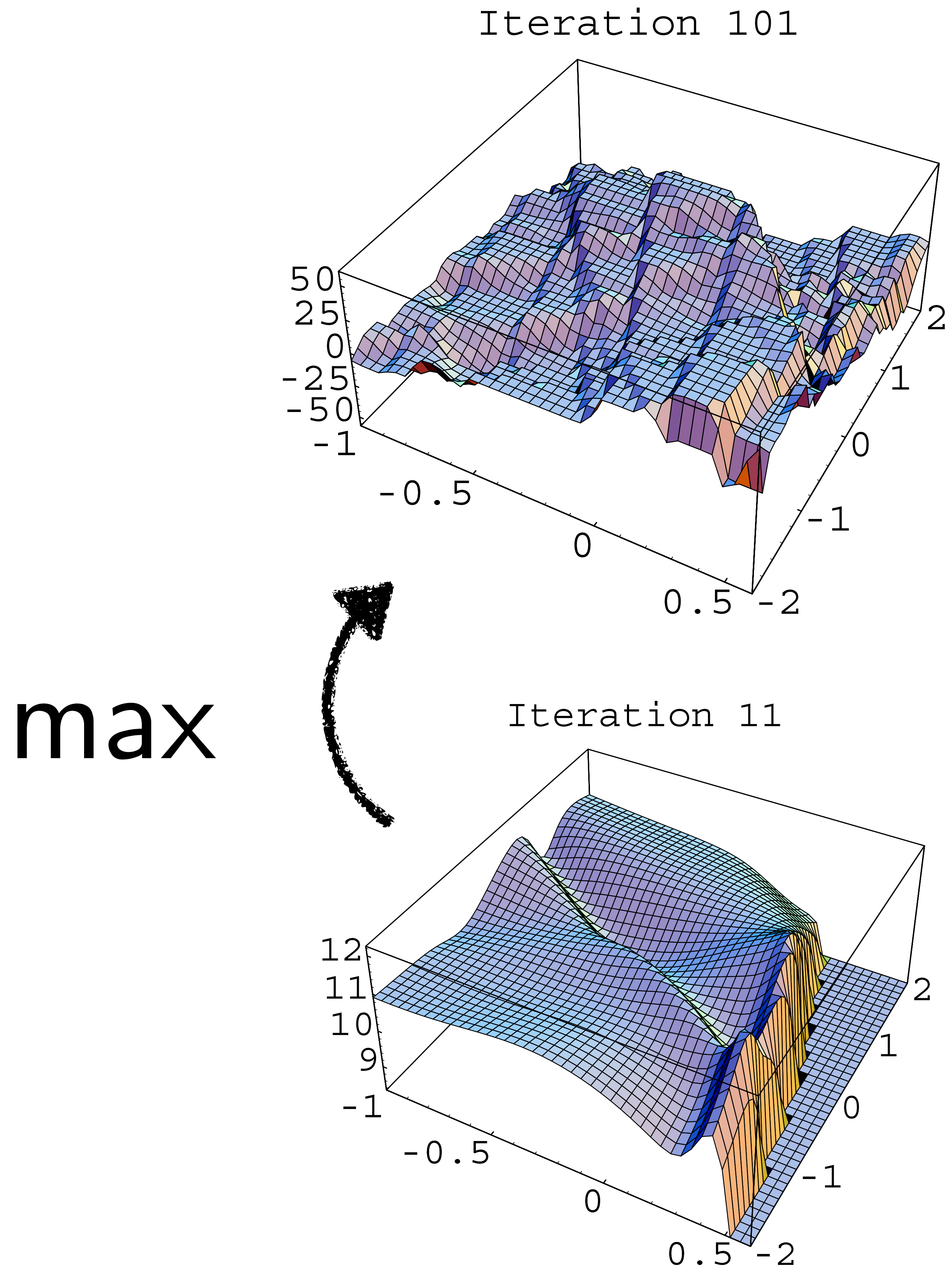
Exploration Exploitation



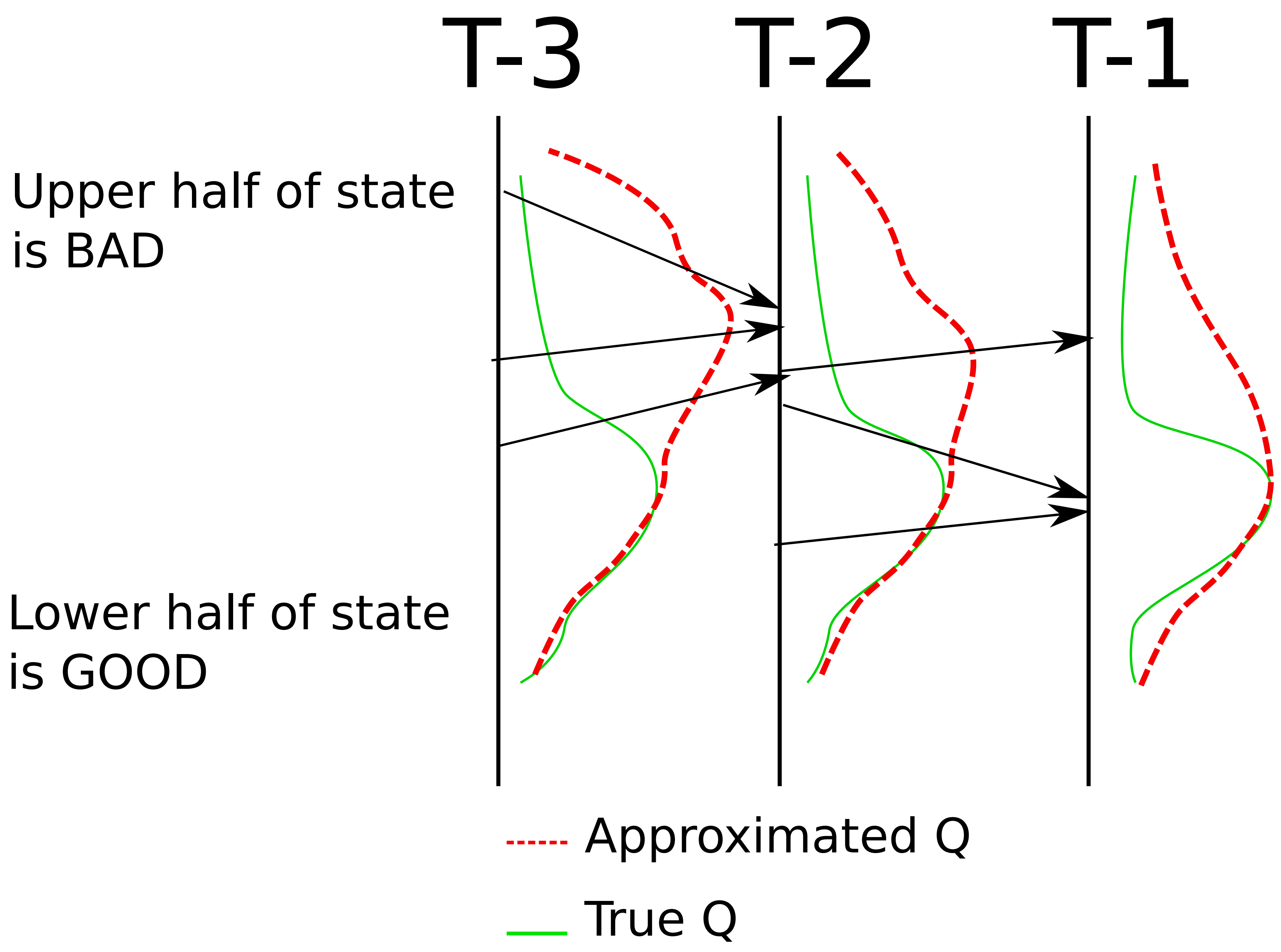
Estimate Values $Q(s, a)$

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 values

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

The Three Nightmares of Policy Optimization

Nightmare 1: High Variance

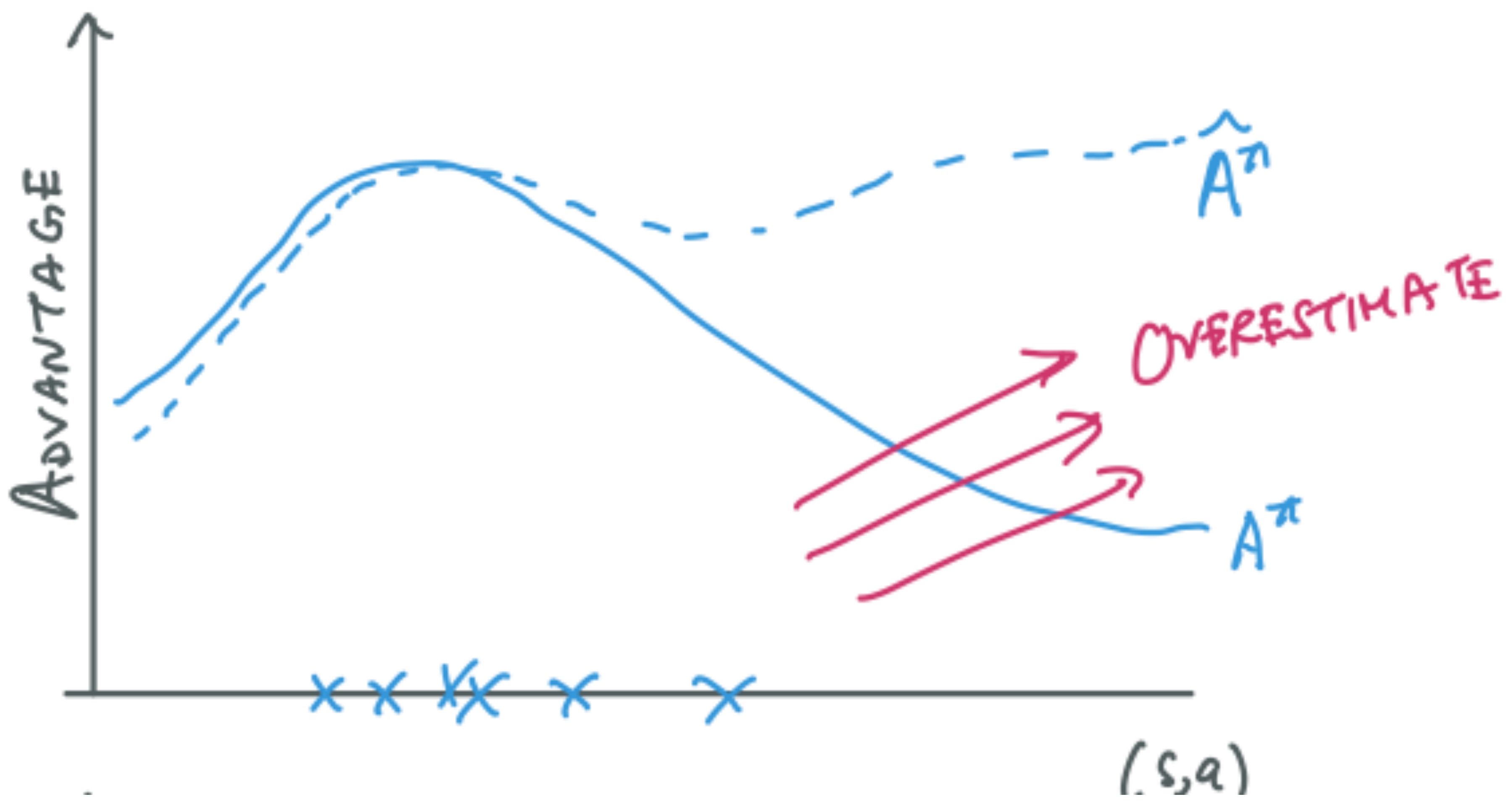
$$\nabla_\theta J = E_{s \sim d^{\pi_\theta}(s), a \sim \pi_\theta(a|s)} [\nabla_\theta \log \pi_\theta(a|s) \boxed{Q^{\pi_\theta}(s,a)}]$$

Solution: Subtract off a baseline!

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

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

Nightmare 2: Distribution Shift



Solution: Take small steps!

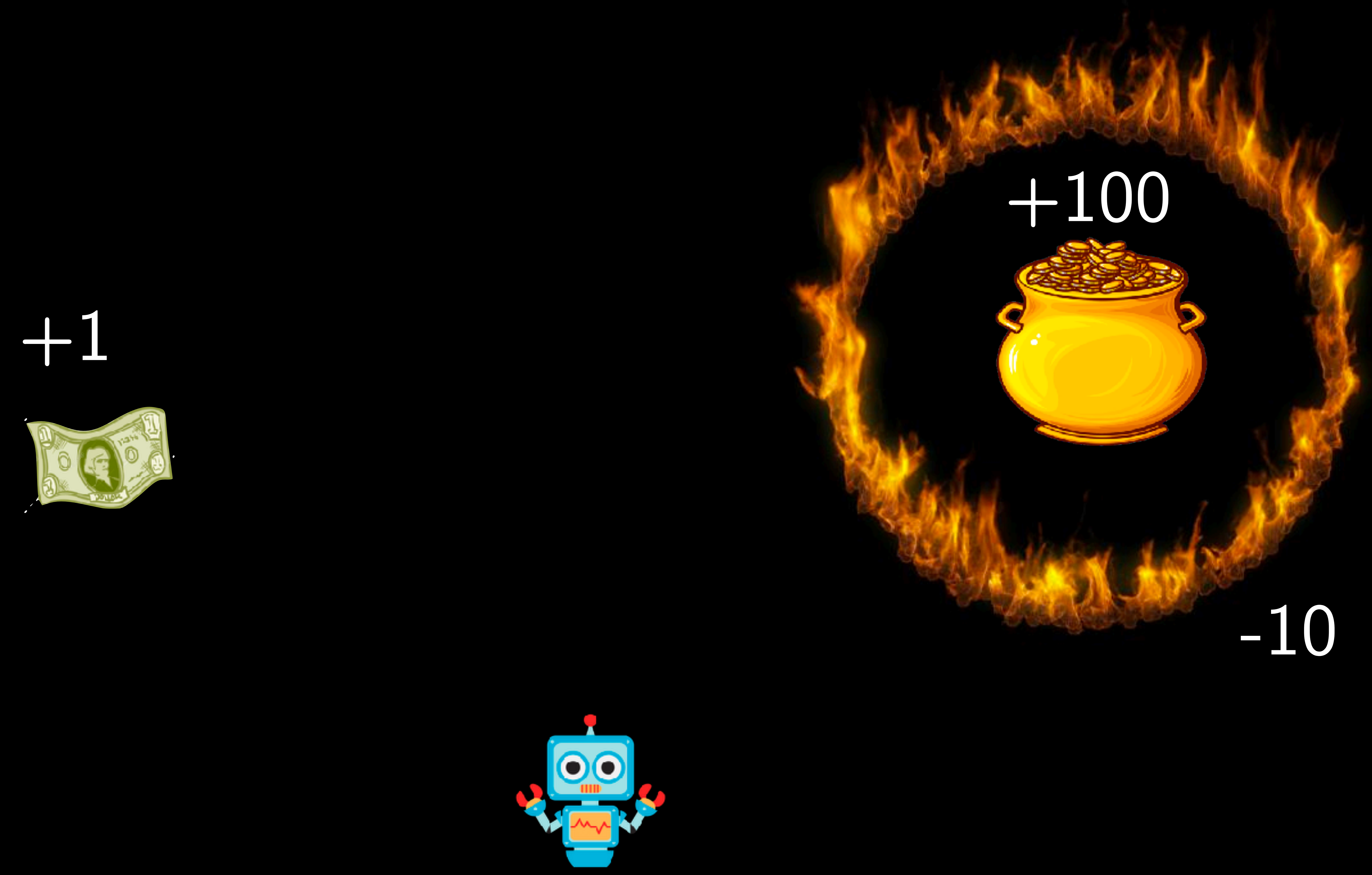
$$\max_{\Delta\theta} J(\theta + \Delta\theta)$$

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

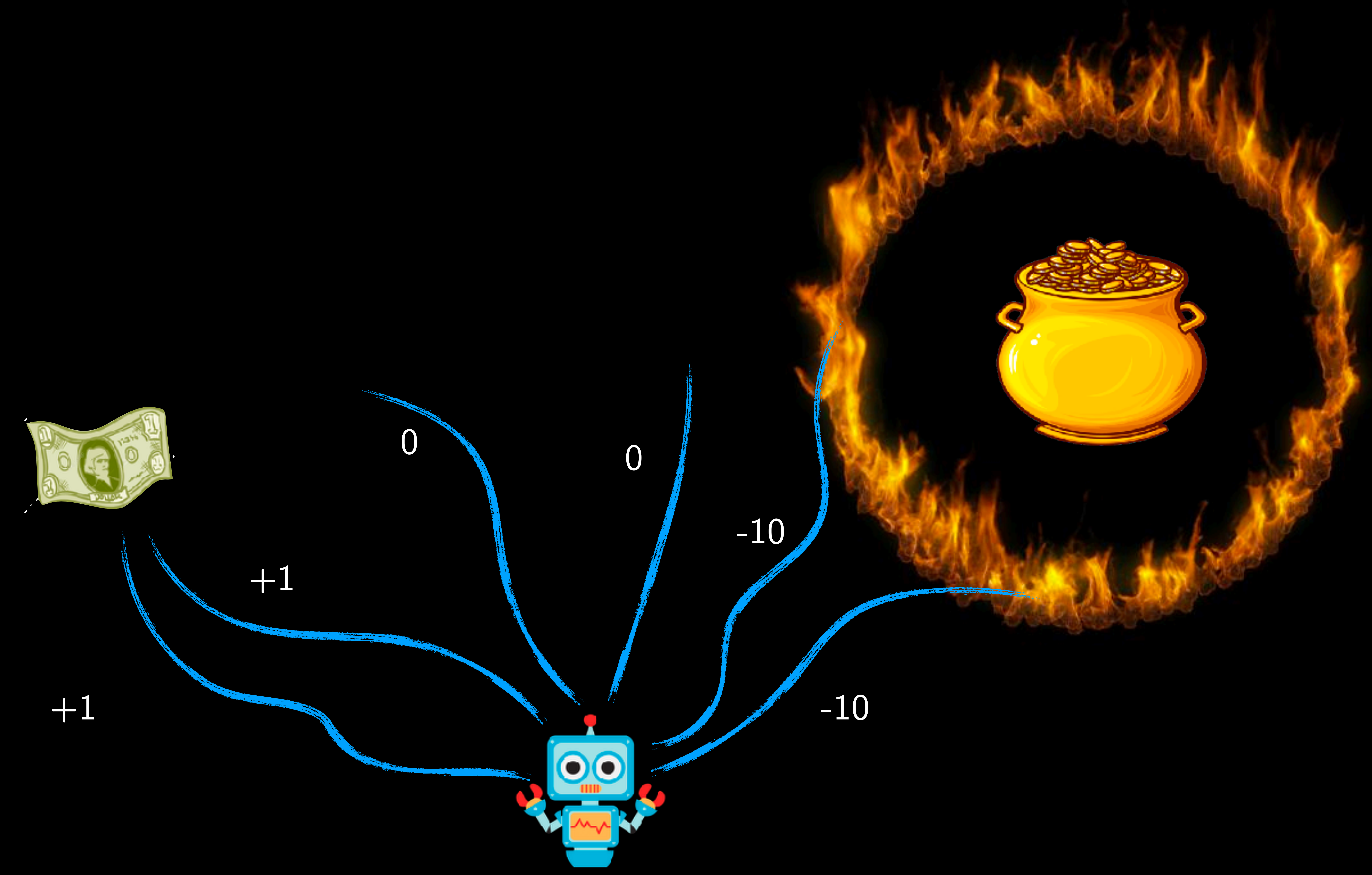
Nightmare 3:

Local Optima

The Ring of Fire

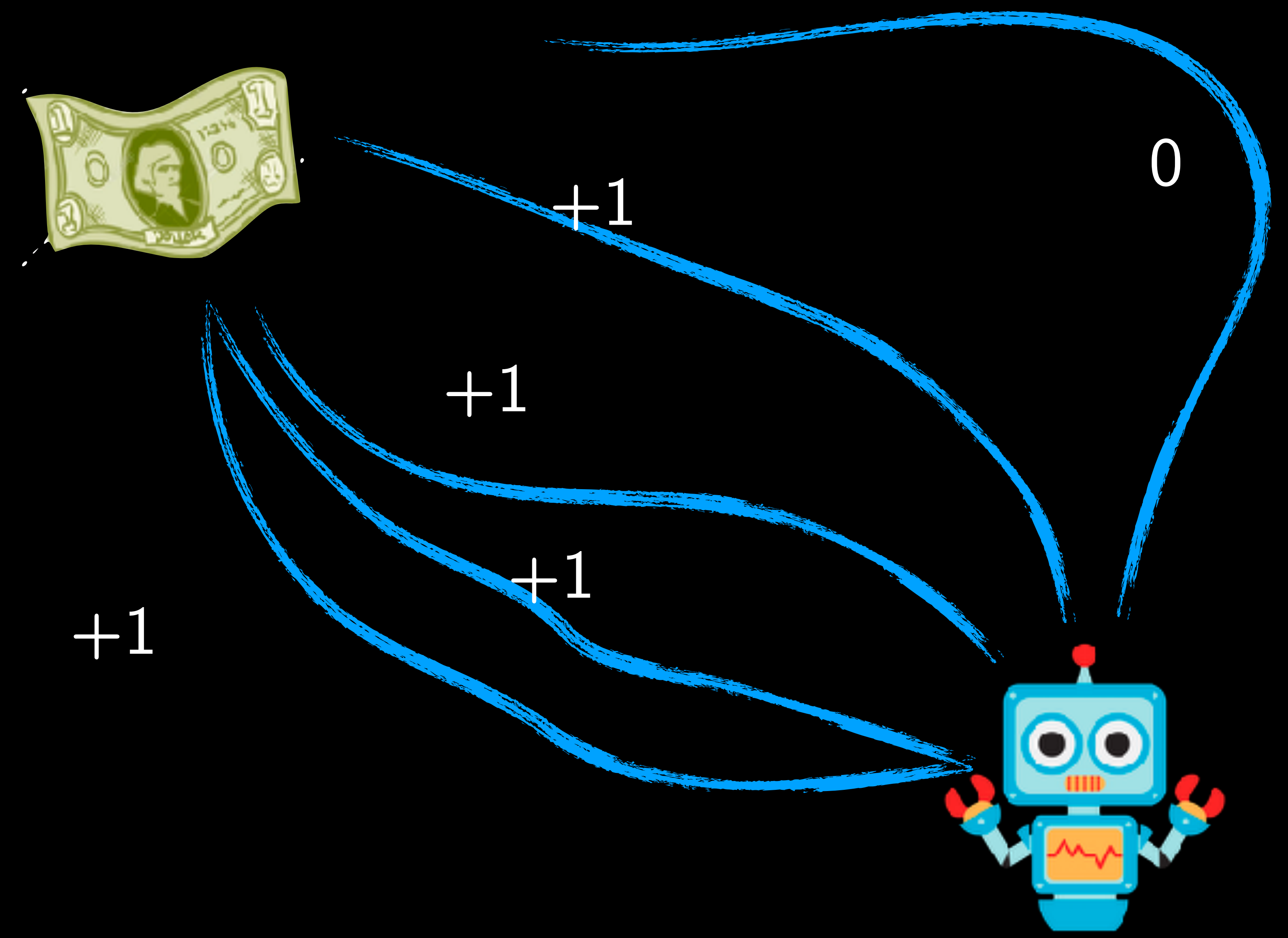


The Ring of Fire



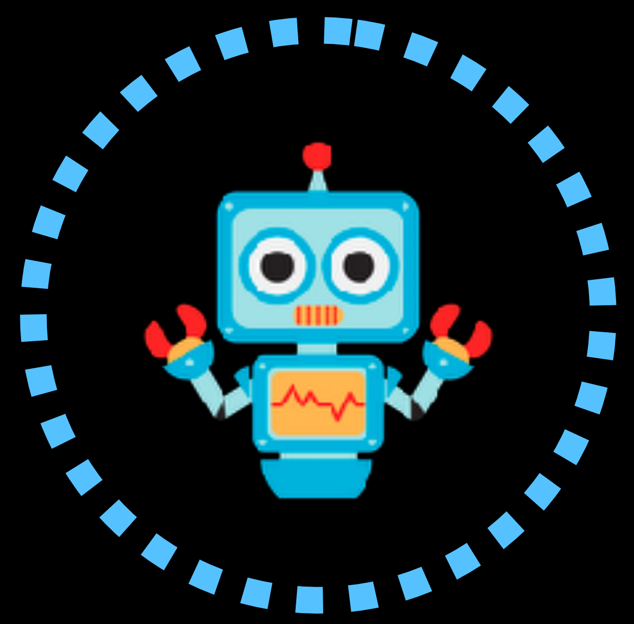
The Ring of Fire

Get's sucked into a local optima!!

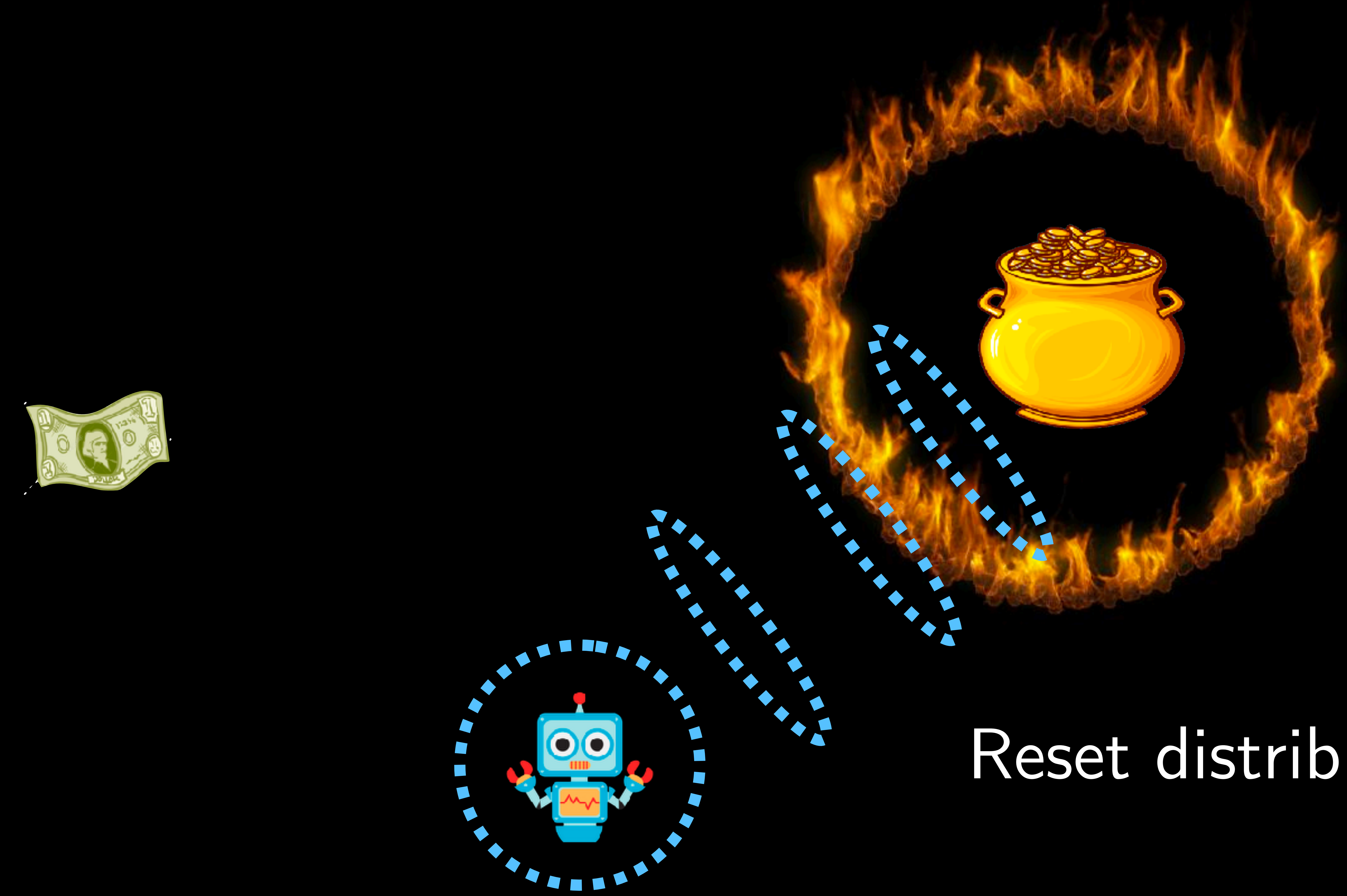


Idea: What if we had a "good reset distribution?"

Start distribution

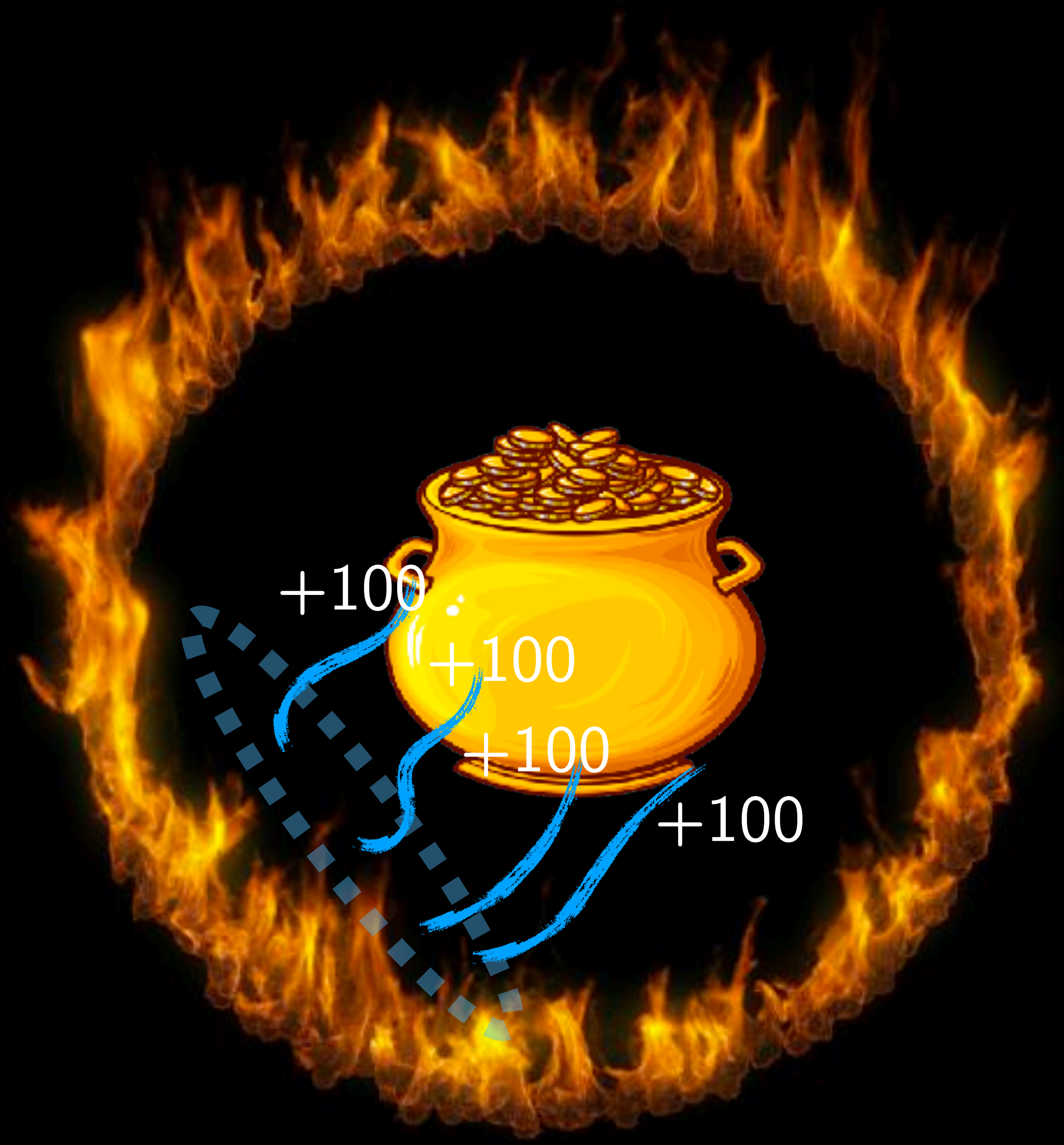


Idea: What if we had a "good reset distribution?"



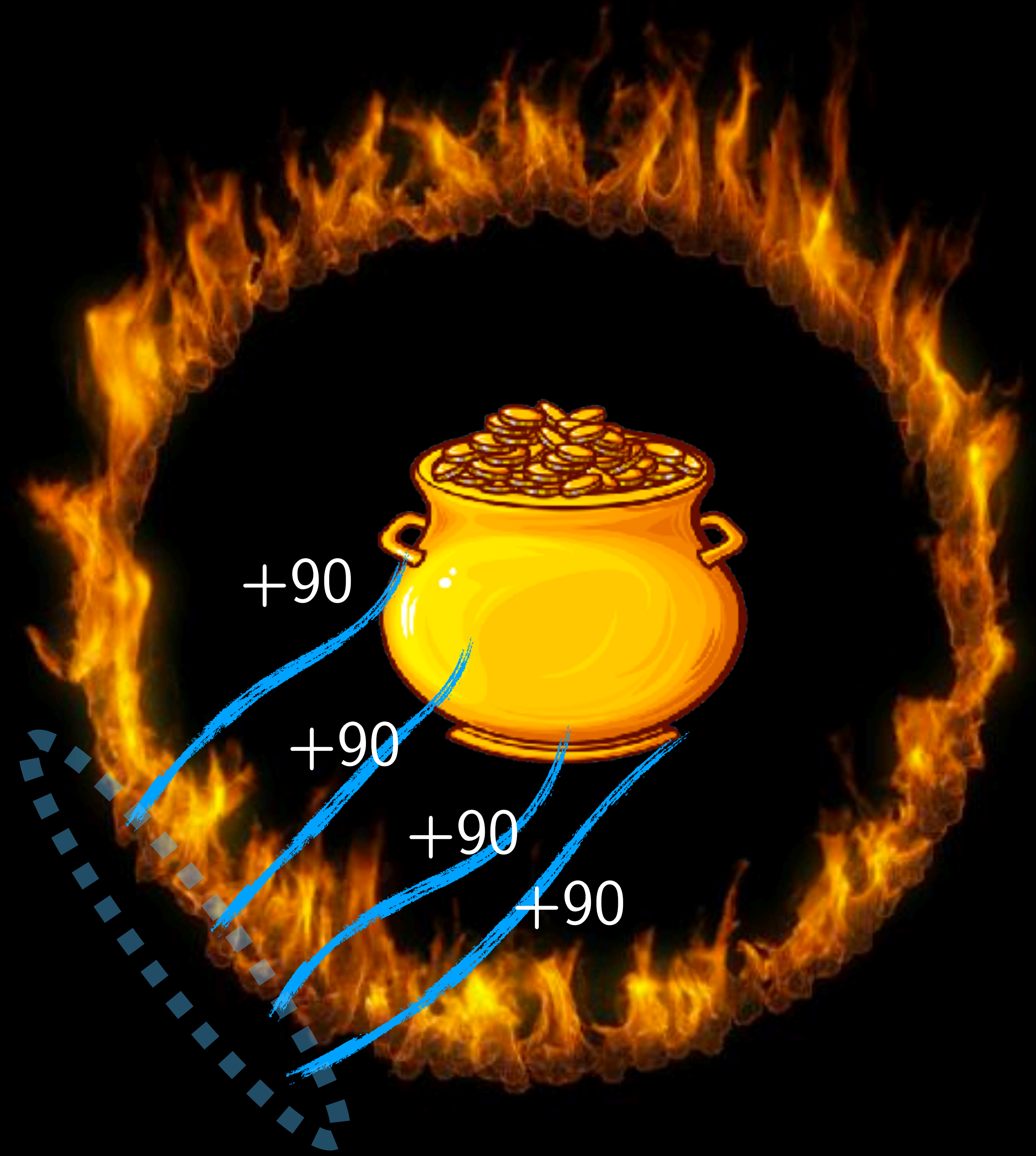
Reset distribution

Idea: What if we had a “good reset distribution?”



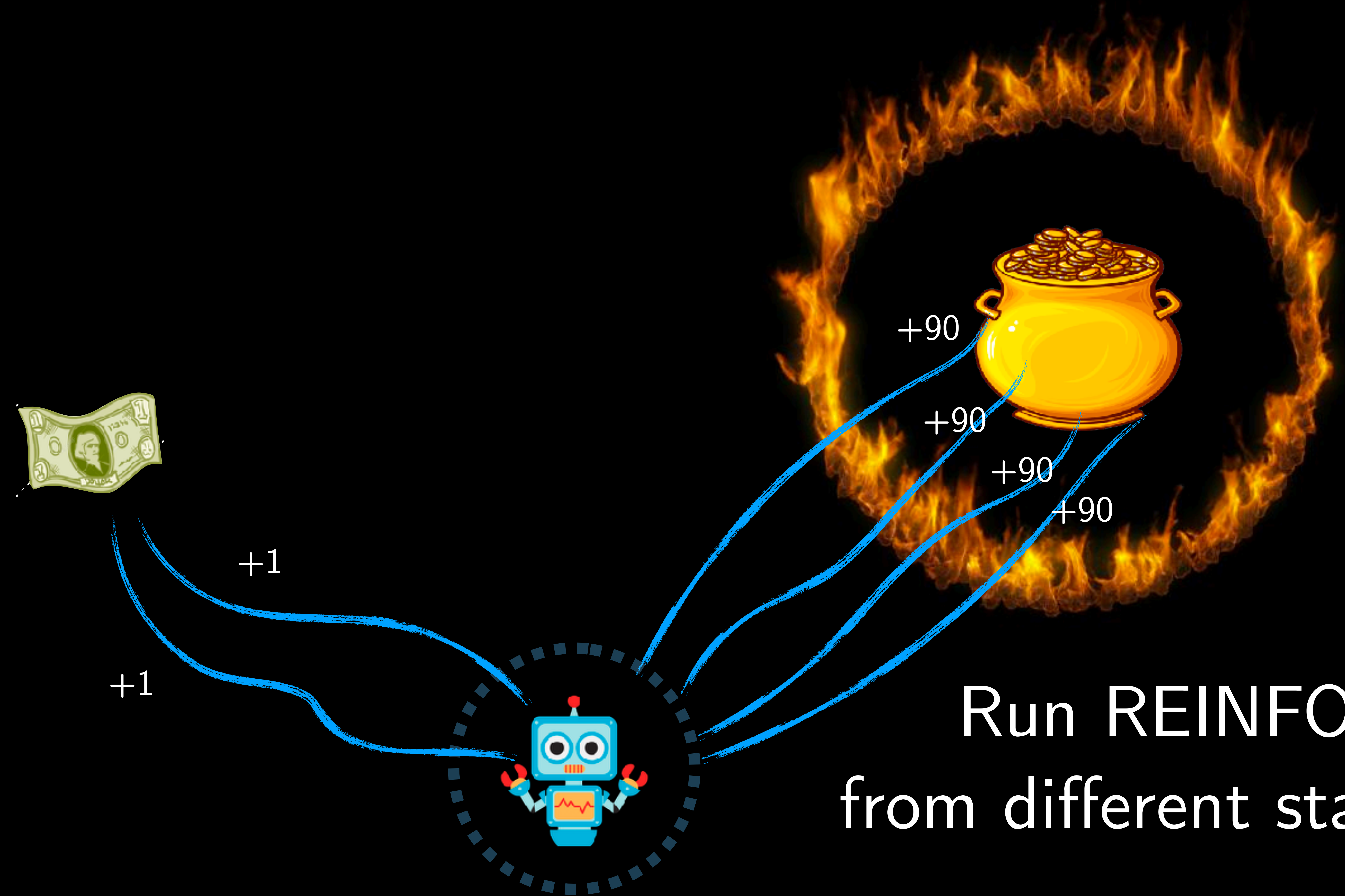
Run REINFORCE
from different start states

Idea: What if we had a "good reset distribution?"



Run REINFORCE
from different start states

Idea: What if we had a "good reset distribution?"



Run REINFORCE
from different start states

Solution: Use a good “reset” distribution

Choose a reset distribution $\mu(s)$ instead of start state distribution

Try your best to “cover” states the expert will visit

Justify using the PDL!

Vanilla Policy Gradient (REINFORCE)

Start with an arbitrary initial policy $\pi_\theta(a \,|\, s)$

while *not converged* **do**

Roll-out $\pi_\theta(a \,|\, s)$ to collect trajectories $D = \{s_0^i, a_0^i, r_0^i, \ldots, s_{T-1}^i, a_{T-1}^i, r_{T-1}^i\}_{i=1}^N$

Compute reward-to-go for each timestep for each trajectory

$$\hat{Q}^{\pi_\theta}(s_t^i, a_t^i) = \sum_{t'=t}^{T-1} r(s_{t'}^i, a_{t'}^i)$$

Compute gradient

$$\nabla_\theta J(\theta) = \frac{1}{N} \left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t^i \,|\, s_t^i) \; \hat{Q}^{\pi_\theta}(s_t^i, a_t^i) \right]$$

Update parameters $\quad \theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Let's apply the fixes to the nightmares!

Start with an arbitrary initial policy $\pi_\theta(a \mid s)$

while *not converged* **do**

Roll-out $\pi_\theta(a \mid s)$ to collect trajectories $D = \{s_0^i, a_0^i, r_0^i, \ldots, s_{T-1}^i, a_{T-1}^i, r_{T-1}^i\}_{i=1}^N$

Compute reward-to-go for each timestep for each trajectory $\hat{Q}^{\pi_\theta}(s_t^i, a_t^i) = \sum_{t'=t}^{T-1} r(s_{t'}^i, a_{t'}^i)$

Compute gradient

$$\nabla_\theta J(\theta) = \frac{1}{N}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t^i \mid s_t^i)\ \hat{Q}^{\pi_\theta}(s_t^i, a_t^i)\right]$$

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

Fix #1: Subtract baseline

Start with an arbitrary initial policy $\pi_\theta(a \mid s)$

while *not converged* **do**

Roll-out $\pi_\theta(a \mid s)$ to collect trajectories $D = \{s_0^i, a_0^i, r_0^i, \ldots, s_{T-1}^i, a_{T-1}^i, r_{T-1}^i\}_{i=1}^N$

Compute reward-to-go for each timestep for each trajectory $\hat{Q}^{\pi_\theta}(s_t^i, a_t^i) = \sum_{t'=t}^{T-1} r(s_{t'}^i, a_{t'}^i)$

Fit value function $\hat{V}^{\pi_\theta}(s_t^i) \approx \sum_{t'=t}^{T-1} r(s_{t'}^i, a_{t'}^i)$

How??

Compute advantage $\hat{A}^{\pi_\theta}(s_t^i, a_t^i) = \hat{Q}^{\pi_\theta}(s_t^i, a_t^i) - \hat{V}^{\pi_\theta}(s_t^i)$

Compute gradient

$$\nabla_\theta J(\theta) = \frac{1}{N}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t^i \mid s_t^i)\, \hat{A}^{\pi_\theta}(s_t^i, a_t^i)\right]$$

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

Fitting values!

Monte-Carlo

$$V(s) \leftarrow V(s) + \alpha(G_t - V(s))$$

Needs full time-horizon trajectories

Temporal Difference

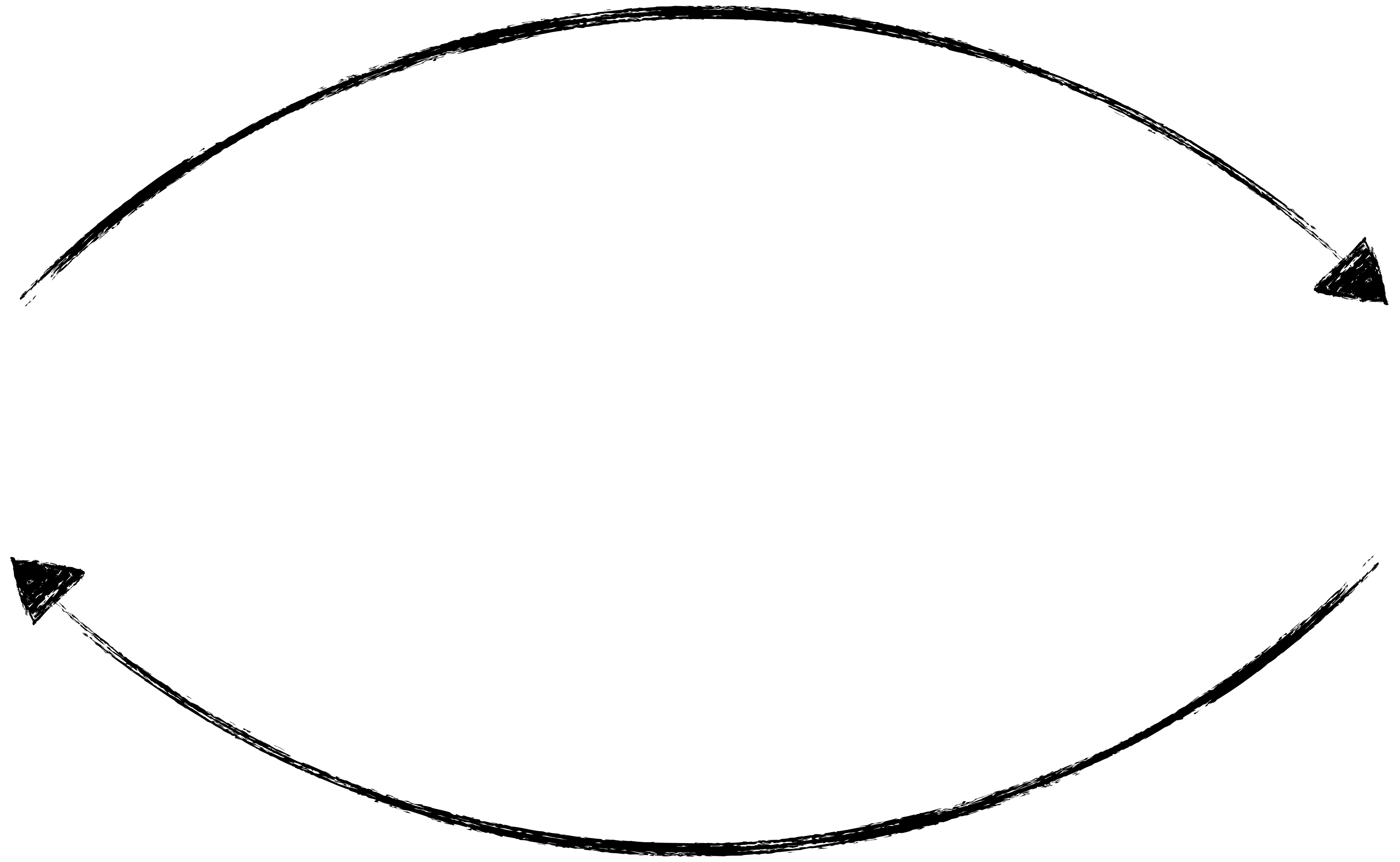
$$V(s) \leftarrow V(s) + \alpha(c + \gamma V(s') - V(s))$$

Works with partial segments!
(s,a,r,s')

Actor-Critic Framework

Actor

Critic



Policy improvement of π

Estimates value functions A^{π}

Actor-Critic Framework

Start with an arbitrary initial policy $\pi_\theta(a \mid s)$

while *not converged* **do**

Roll-out $\pi_\theta(a \mid s)$ to collect trajectories $D = \{s^i, a^i, r^i, s^i_+\}_{i=1}^N$

Fit value function $\hat{V}^{\pi_\theta}(s^i)$ using TD, i.e. minimize $(r^i + \gamma \hat{V}^{\pi_\theta}(s^i_+) - \hat{V}^{\pi_\theta}(s^i))^2$

Compute advantage $\hat{A}^{\pi_\theta}(s^i, a^i) = r(s^i, a^i) + \gamma \hat{V}^{\pi_\theta}(s^i_+) - \hat{V}^{\pi_\theta}(s^i)$

Compute gradient

$$\nabla_\theta J(\theta) = \frac{1}{N}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a^i_t \mid s^i_t)\, \hat{A}^{\pi_\theta}(s^i, a^i)\right]$$

Update parameters $\qquad \theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Fix #2: Take small steps

Start with an arbitrary initial policy $\pi_\theta(a \,|\, s)$

while *not converged* **do**

Roll-out $\pi_\theta(a \,|\, s)$ to collect trajectories $D = \{s^i, a^i, r^i, s^i_+\}_{i=1}^N$

Fit value function $\hat{V}^{\pi_\theta}(s^i)$ using TD, i.e. minimize $(r^i + \gamma \hat{V}^{\pi_\theta}(s^i_+) - \hat{V}^{\pi_\theta}(s^i))^2$

Compute advantage $\hat{A}^{\pi_\theta}(s^i, a^i) = r(s^i, a^i) + \gamma \hat{V}^{\pi_\theta}(s^i_+) - \hat{V}^{\pi_\theta}(s^i)$

Compute gradient

$$\nabla_\theta J(\theta) = \frac{1}{N}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a^i_t \,|\, s^i_t)\, \hat{A}^{\pi_\theta}(s^i, a^i)\right]$$

s.t. $KL(\pi(\theta + \Delta\theta) \,||\, \pi(\theta)) \leq \epsilon$

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

How??

Natural Gradient Descent (rediscovered as TRPO)

Start with an arbitrary initial policy $\pi_\theta(a \mid s)$

while *not converged* **do**

Roll-out $\pi_\theta(a \mid s)$ to collect trajectories $D = \{s^i, a^i, r^i, s^i_+\}_{i=1}^N$

Fit value function $\hat{V}^{\pi_\theta}(s^i)$ using TD, i.e. minimize $(r^i + \gamma \hat{V}^{\pi_\theta}(s^i_+) - \hat{V}^{\pi_\theta}(s^i))^2$

Compute advantage $\hat{A}^{\pi_\theta}(s^i, a^i) = r(s^i, a^i) + \gamma \hat{V}^{\pi_\theta}(s^i_+) - \hat{V}^{\pi_\theta}(s^i)$

Compute gradient

$$\nabla_\theta J(\theta) = \frac{1}{N}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a^i_t \mid s^i_t)\, \hat{A}^{\pi_\theta}(s^i, a^i)\right]$$

$$\cancel{\text{s.t. } KL(\pi(\theta + \Delta\theta) \,||\, \pi(\theta)) \le \epsilon}$$

$$\approx \Delta\theta^T G(\theta) \Delta\theta \le \epsilon$$

Update parameters $\quad \theta \leftarrow \theta + \alpha G(\theta)^{-1} \nabla_\theta J(\theta)$

$G(\theta)$ is Fischer Information Matrix

$$G(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta \nabla_\theta \log \pi_\theta^T\right]$$

Natural Gradient Descent (rediscovered as TRPO!)

Start with an arbitrary initial policy $\pi_\theta(a \mid s)$

while *not converged* **do**

Roll-out $\pi_\theta(a \mid s)$ to collect trajectories $D = \{s^i, a^i, r^i, s_+^i\}_{i=1}^N$

Fit value function $\hat{V}^{\pi_\theta}(s^i)$ using TD, i.e. minimize $(r^i + \gamma \hat{V}^{\pi_\theta}(s_+^i) - \hat{V}^{\pi_\theta}(s^i))^2$

Compute advantage $\hat{A}^{\pi_\theta}(s^i, a^i) = r(s^i, a^i) + \gamma \hat{V}^{\pi_\theta}(s_+^i) - \hat{V}^{\pi_\theta}(s^i)$

Compute gradient

$$\nabla_\theta J(\theta) = \frac{1}{N}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t^i \mid s_t^i)\, \hat{A}^{\pi_\theta}(s^i, a^i)\right]$$

~~s.t.~~ $\cancel{KL(\pi(\theta + \Delta\theta) \,||\, \pi(\theta)) \leq \epsilon}$

$$\approx \Delta\theta^T G(\theta) \Delta\theta \leq \epsilon$$

Update parameters $\quad \theta \leftarrow \theta + \alpha G(\theta)^{-1} \nabla_\theta J(\theta)$

Don't directly compute the inverse,
use conjugate gradient to solve $G(\theta)x = \nabla_\theta J(\theta)$

$G(\theta)$ is Fischer Information Matrix

$$G(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta \nabla_\theta \log \pi_\theta^T\right]$$

Proximal Policy Optimization (PPO)

Computing Fischer Information matrix is expensive and slow!

Idea: Instead of taking small steps, change the loss function so there is no benefit in taking large steps!

Proximal Policy Optimization (PPO)

Computing Fischer Information matrix is expensive and slow!

Idea: Instead of taking small steps, change the loss function so there is no benefit in taking large steps!

Instead of defining gradient, we will define a surrogate loss function (Lets say we are at iteration k)

$$\mathscr{L}(\theta) = \mathbb{E}_{s,a\sim\pi_{\theta_k}}\left[\frac{\pi_\theta}{\pi_{\theta_k}} A^{\pi_{\theta_k}}(s,a)\right]$$

Proximal Policy Optimization (PPO)

Computing Fischer Information matrix is expensive and slow!

Idea: Instead of taking small steps, change the loss function so there is no benefit in taking large steps!

Clip the loss if the policy π_θ deviates too much from π_{θ_k}

$$\mathscr{L}(\theta) = \mathbb{E}_{s,a\sim\pi_{\theta_k}}\left[\min\left(\frac{\pi_\theta}{\pi_{\theta_k}} A^{\pi_{\theta_k}}(s,a), \text{clip}\left(\frac{\pi_\theta}{\pi_{\theta_k}}, 1-\epsilon, 1+\epsilon\right) A^{\pi_{\theta_k}}(s,a)\right)\right]$$

Fix #3: Use a reset distribution

Start with an arbitrary initial policy $\pi_\theta(a \mid s)$

while *not converged* **do**

Roll-out $\pi_\theta(a \mid s)$ to collect trajectories $D = \{s^i, a^i, r^i, s^i_+\}_{i=1}^N$

Fit value function $\hat{V}^{\pi_\theta}(s^i)$ using TD, i.e. minimize $(r^i + \gamma \hat{V}^{\pi_\theta}(s^i_+) - \hat{V}^{\pi_\theta}(s^i))^2$

Compute advantage $\hat{A}^{\pi_\theta}(s^i, a^i) = r(s^i, a^i) + \gamma \hat{V}^{\pi_\theta}(s^i_+) - \hat{V}^{\pi_\theta}(s^i)$

Compute gradient

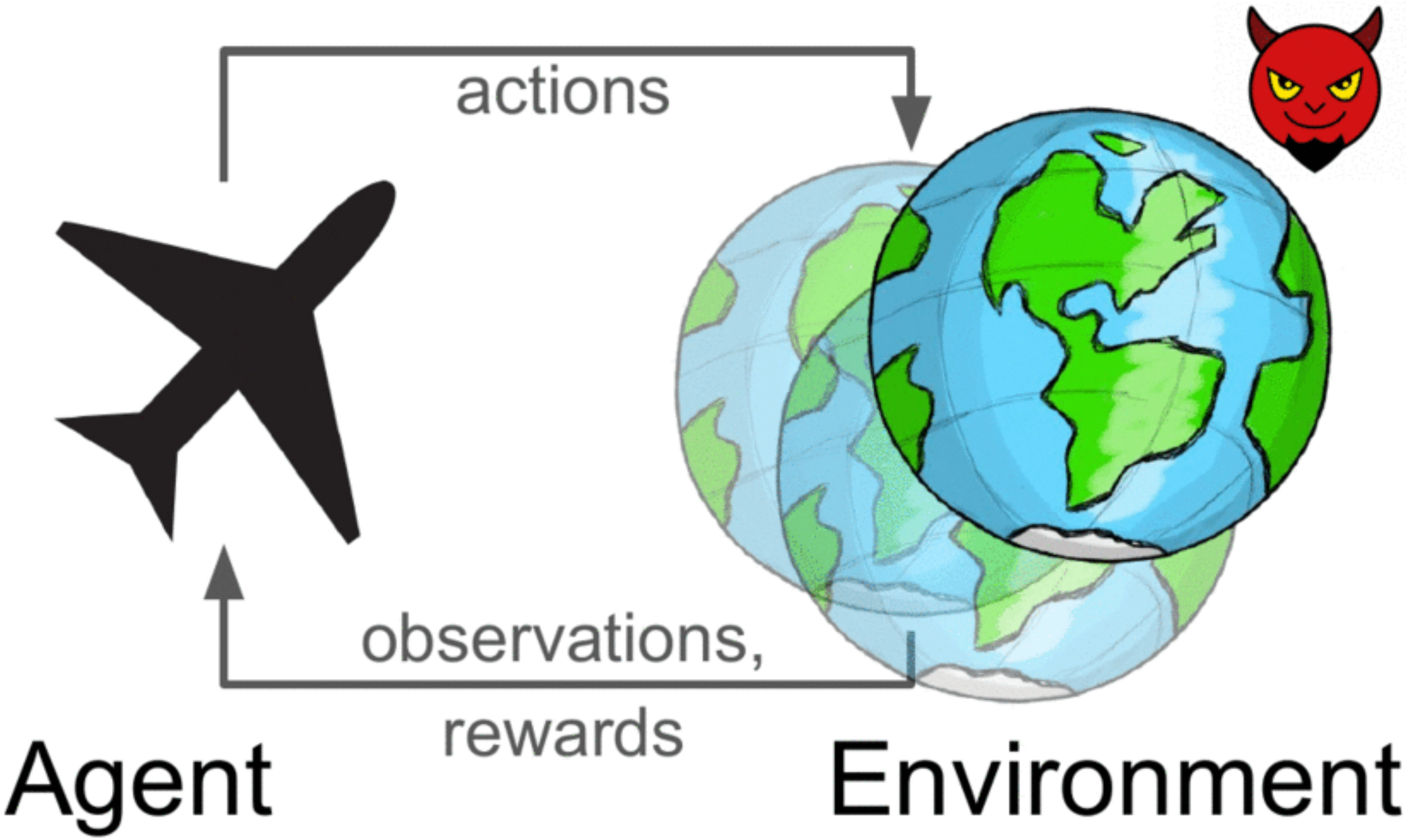
$$\nabla_\theta J(\theta) = \frac{1}{N}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a^i_t \mid s^i_t)\, \hat{A}^{\pi_\theta}(s^i, a^i)\right] \text{ s.t. } KL(\pi(\theta + \Delta\theta) \,||\, \pi(\theta)) \leq \epsilon$$

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

Instead of rolling out from the start state, rollout from states expert visits

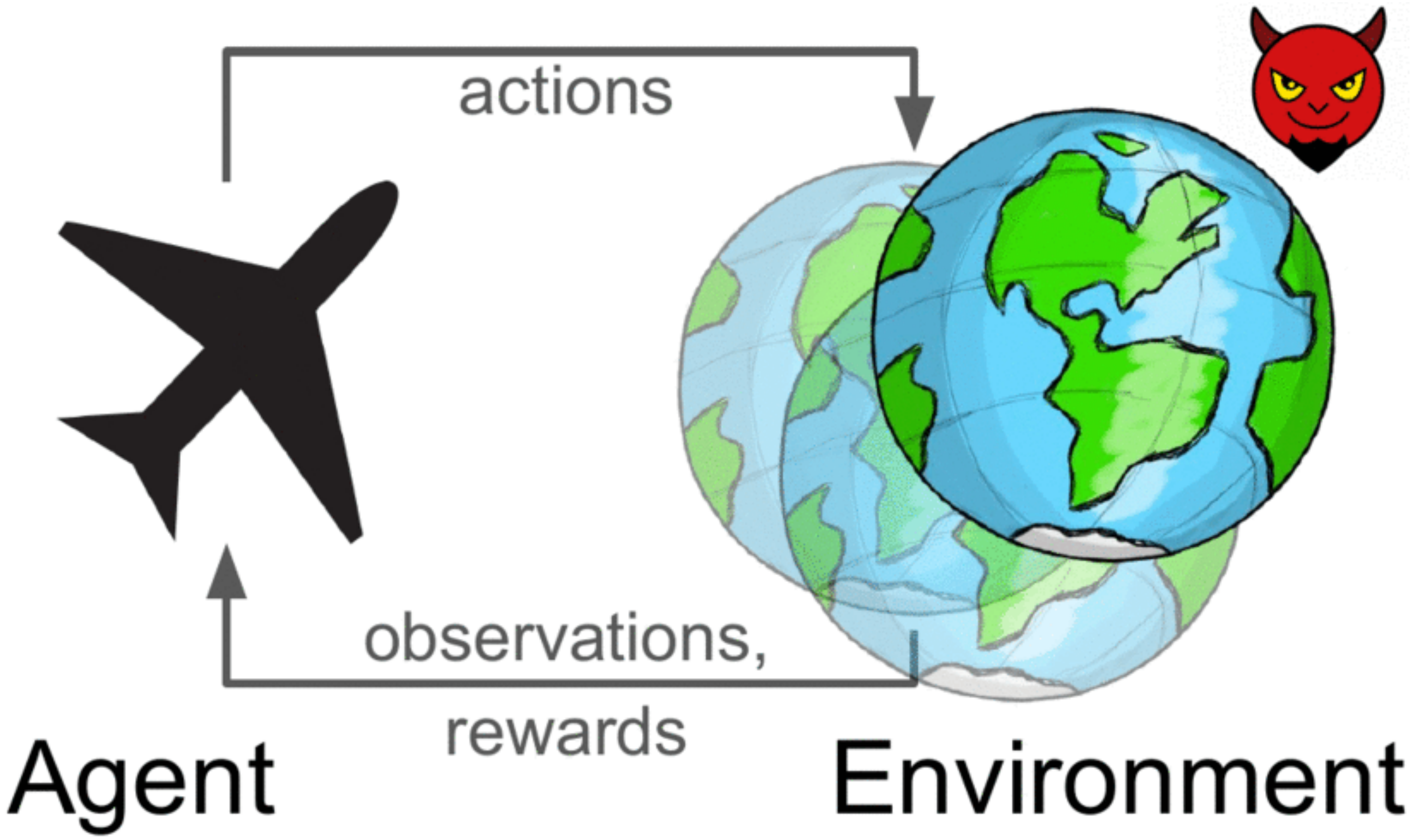
How do we make Actor-Critic more robust to randomness of the environment?

We never see the actual environment in RL



Credit: Ben Eyesenbach

We want our policy to be robust against all possible environments that can explain the data



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

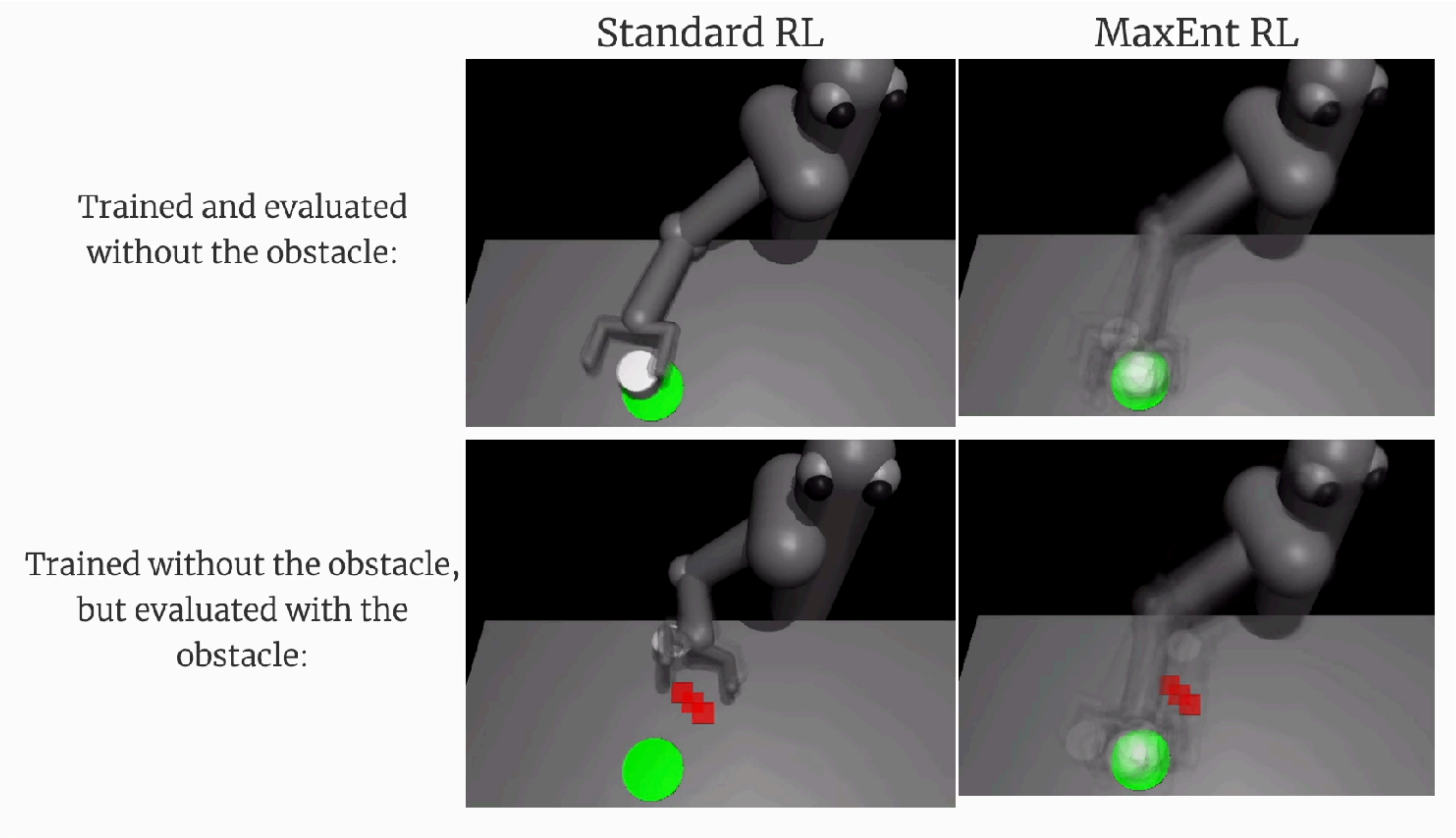
Credit: Ben Eyesenbach

Solution: Use Maximum Entropy RL!

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

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

Haarnoja 2018

“Soft” Actor Critic