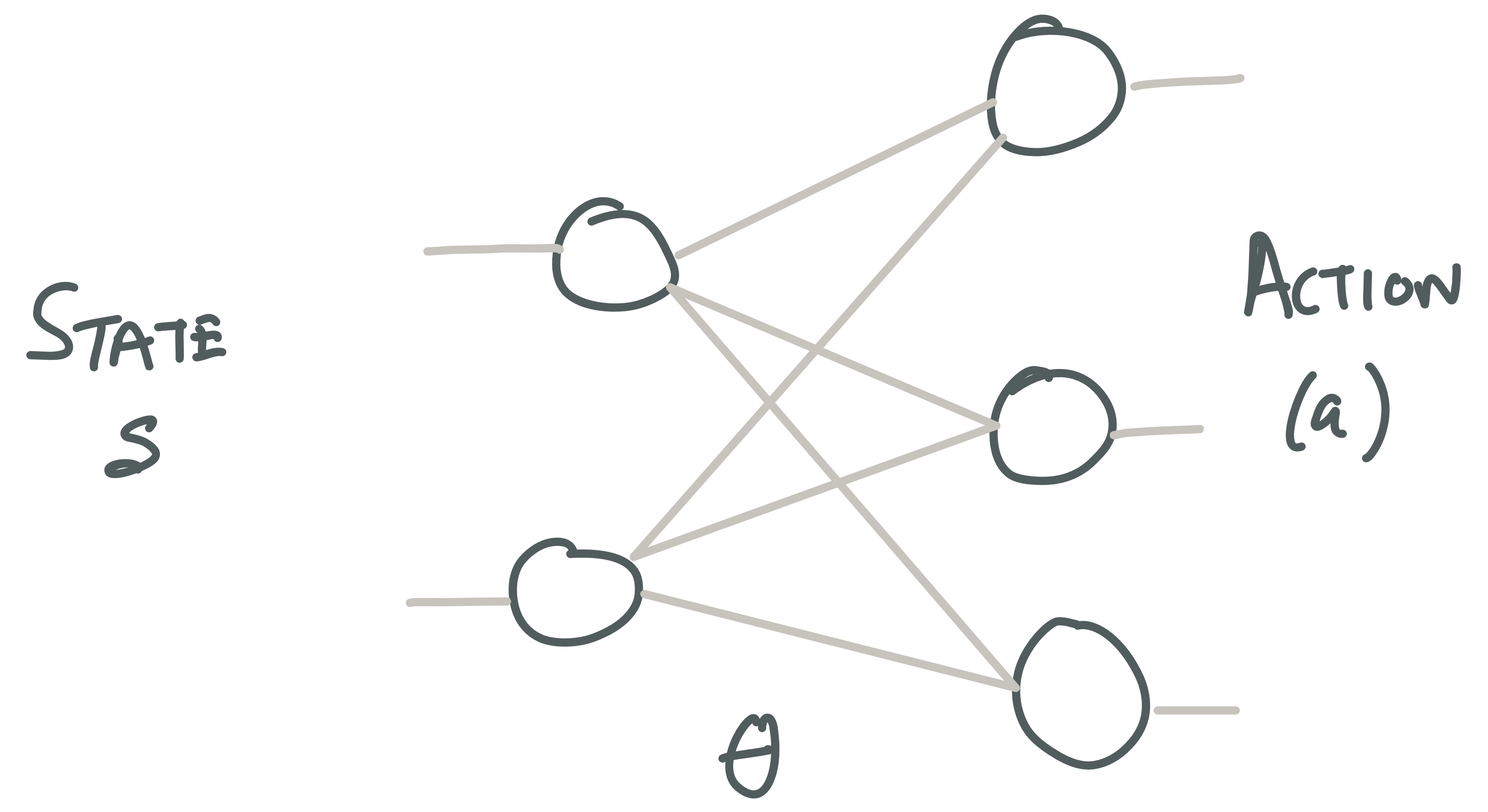
Nightmares of Policy Optimization (contd)

Sanjiban Choudhury

Cornell Bowers CIS
Computer Science

SUPERVISED LEARNING (CLASSIFICATION)



PROB OVER ACTIONS: $\pi_\theta(a|s)$

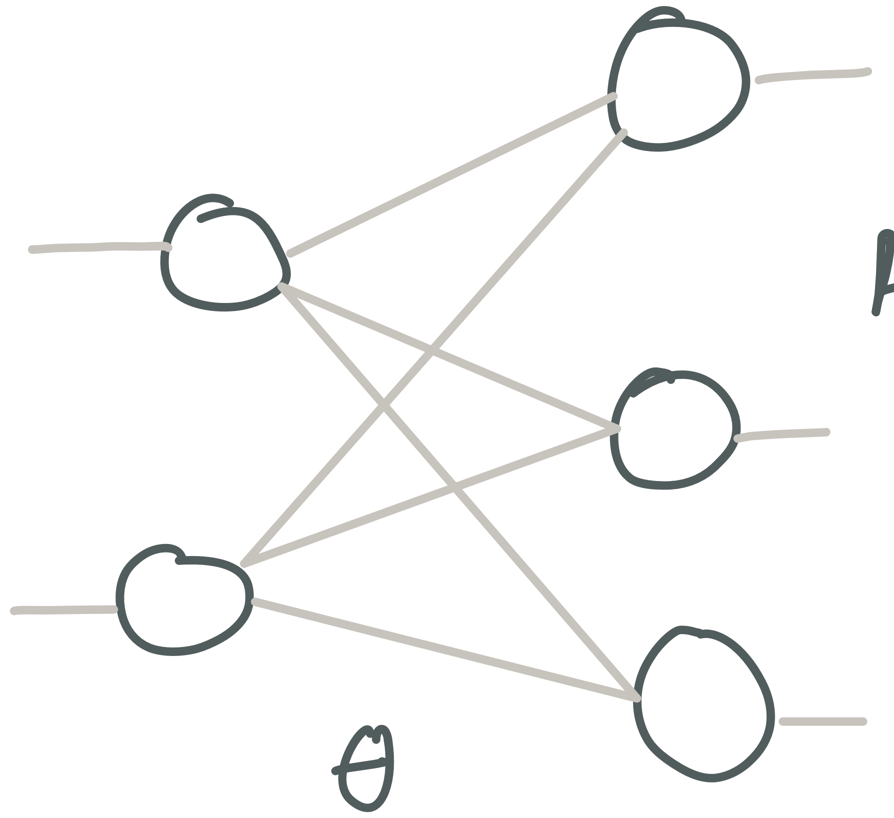
GIVEN A DATAPOINT

$$(s, a^*)$$

WHAT IS THE CLASSIFICATION LOSS?

SUPERVISED LEARNING (CLASSIFICATION)

STATE s



ACTION (a)

θ

PROB OVER ACTIONS: $\pi_\theta(a|s)$

GIVEN A DATAPOINT

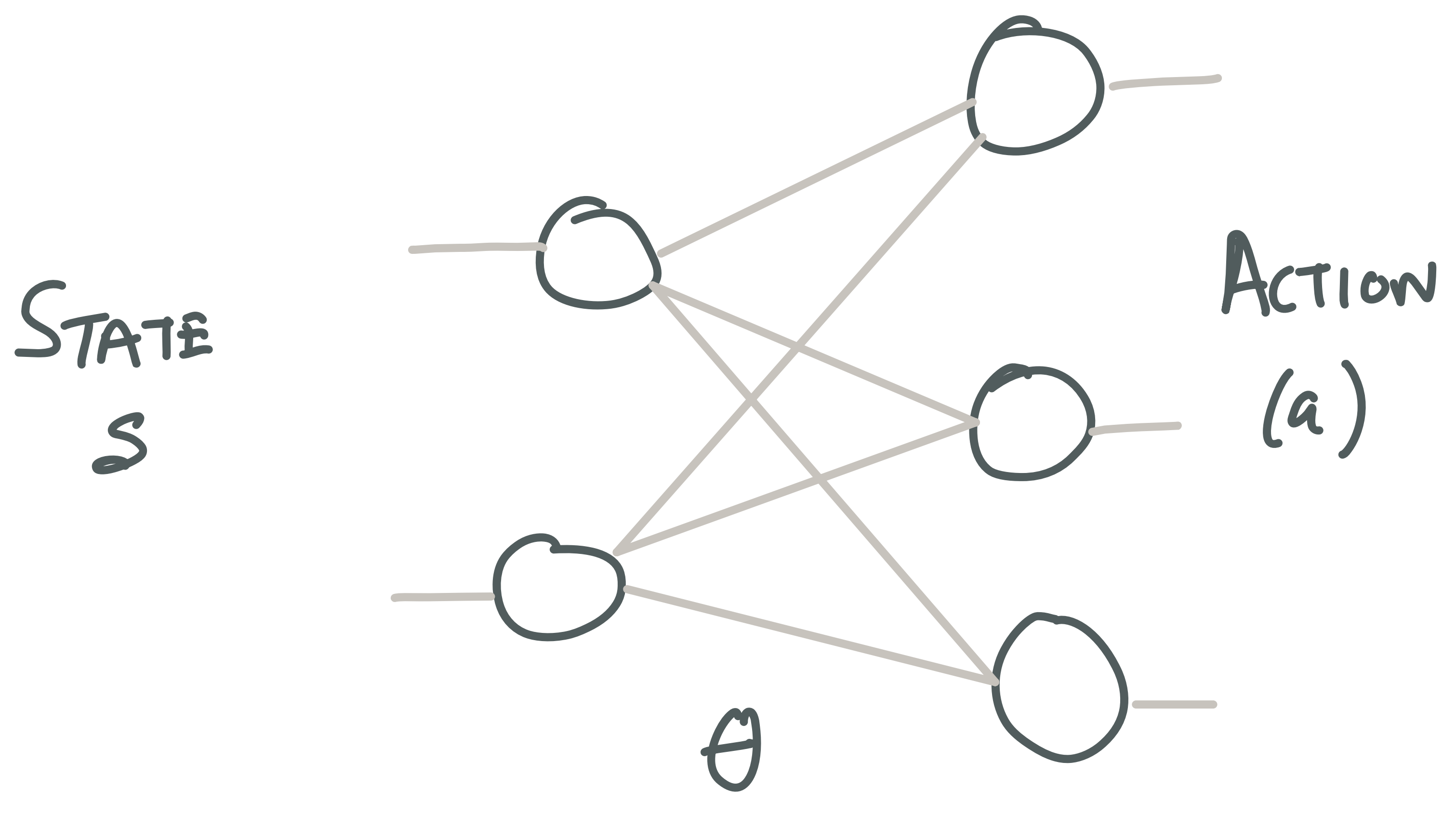
$$(s, a^*)$$

WHAT IS THE CLASSIFICATION LOSS?

$$\mathcal{L}(\theta) = -\log \pi_\theta(a^*|s)$$

GRADIENT: $\nabla_\theta \mathcal{L}(\theta) = -\nabla_\theta \log \pi_\theta(a^*|s)$

Supervised Learning (Weighted Classification)



Prob over actions: $\pi_\theta(a|s)$

Given a datapoint

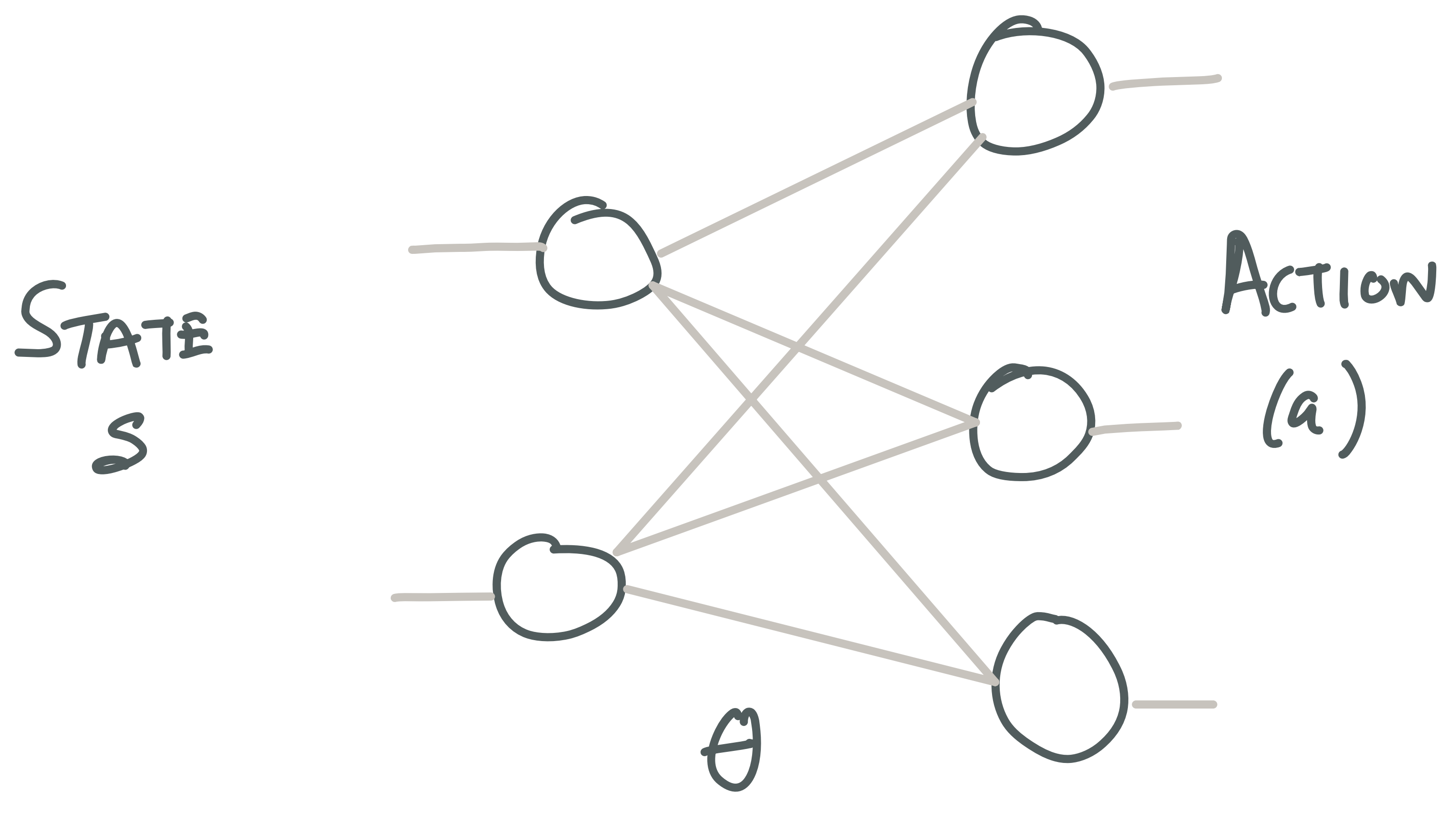
(s_1, a_1, R_1) (s_1, a_1, R_2)

What is the weighted classification loss?

Supervised Learning (Weighted Classification)

Given a datapoint

$(s_1, a_1, R_1) \quad (s_1, a_1, R_2)$



Prob over actions: $\pi_\theta(a|s)$

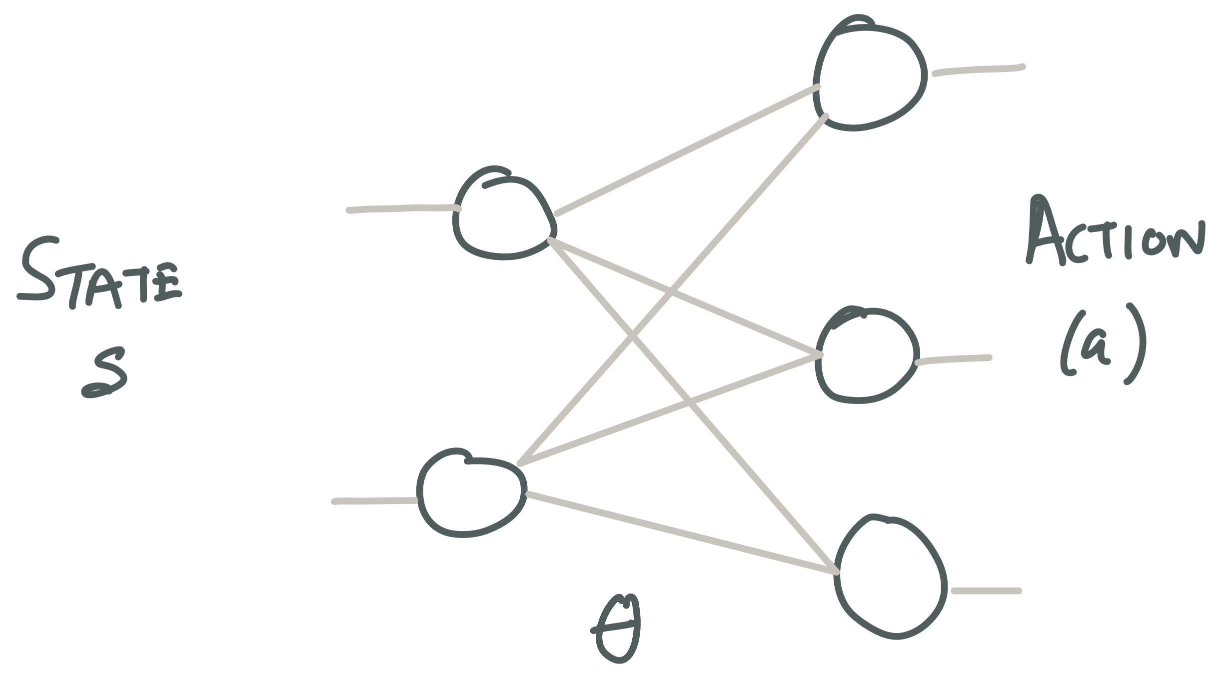
What is the weighted classification loss?

$$\mathcal{L}(\theta) = -R_1 \log \pi_\theta(a_1|s_1)$$
$$-R_2 \log \pi_\theta(a_2|s_2)$$
$$\cdots$$

Gradient: $\nabla_\theta \mathcal{L}(\theta) = -R_1 \nabla_\theta \log \pi_\theta(a_1|s_1)$

$$-R_2 \nabla_\theta \log \pi_\theta(a_2|s_2)$$
$$\cdots$$

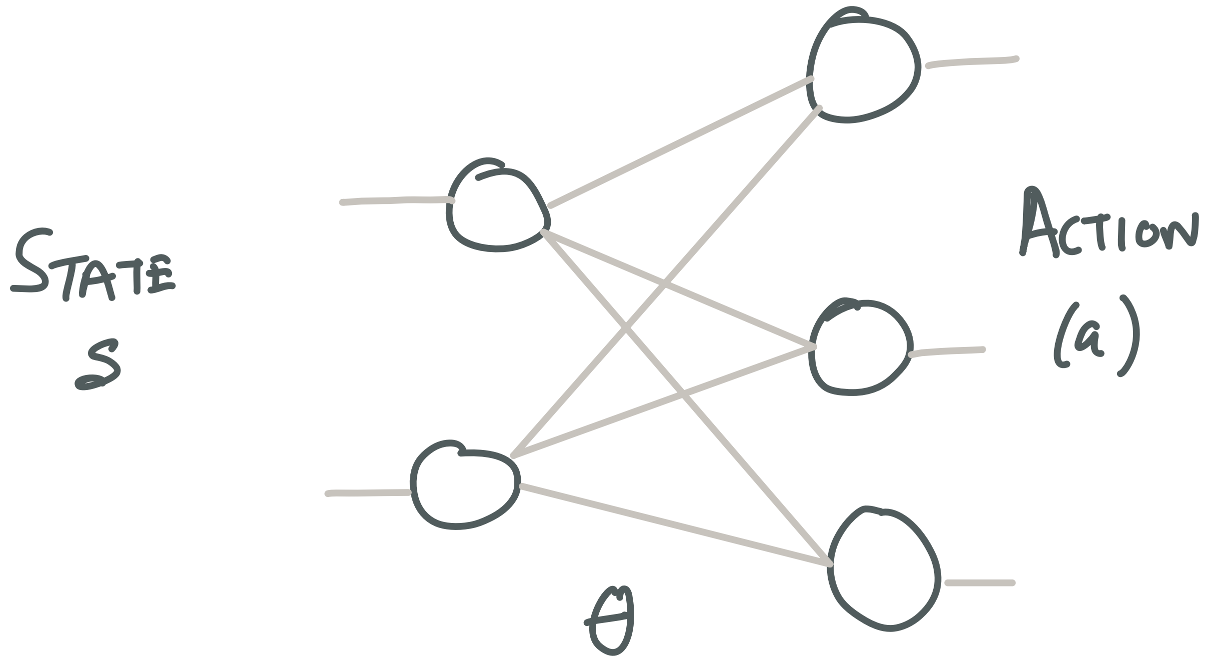
Policy Gradient (Multi-Step Classification)



Prob over actions: $\pi_\theta(a|s)$

Given trajectory $s_1 \; a_1 \; s_2 \; a_2 \; s_3 \; a_3 \; \ldots$ with rewards R_1, R_2, R_3

Policy Gradient (Multi-Step Classification)



Prob over actions: $\pi_\theta(a|s)$

Given Trajectory

$$\overset{R_1}{s_1 \; a_1} \; \overset{R_2}{s_2 \; a_2} \; \overset{R_3}{s_3 \; a_3} \; \dots$$

Convert to Returns

$$(R_1 + R_2 + R_3 + \dots) = \hat{Q}_1$$

$$\boxed{s_1 \; a_1} \; s_2 \; a_2 \; s_3 \; a_3 \; \dots$$

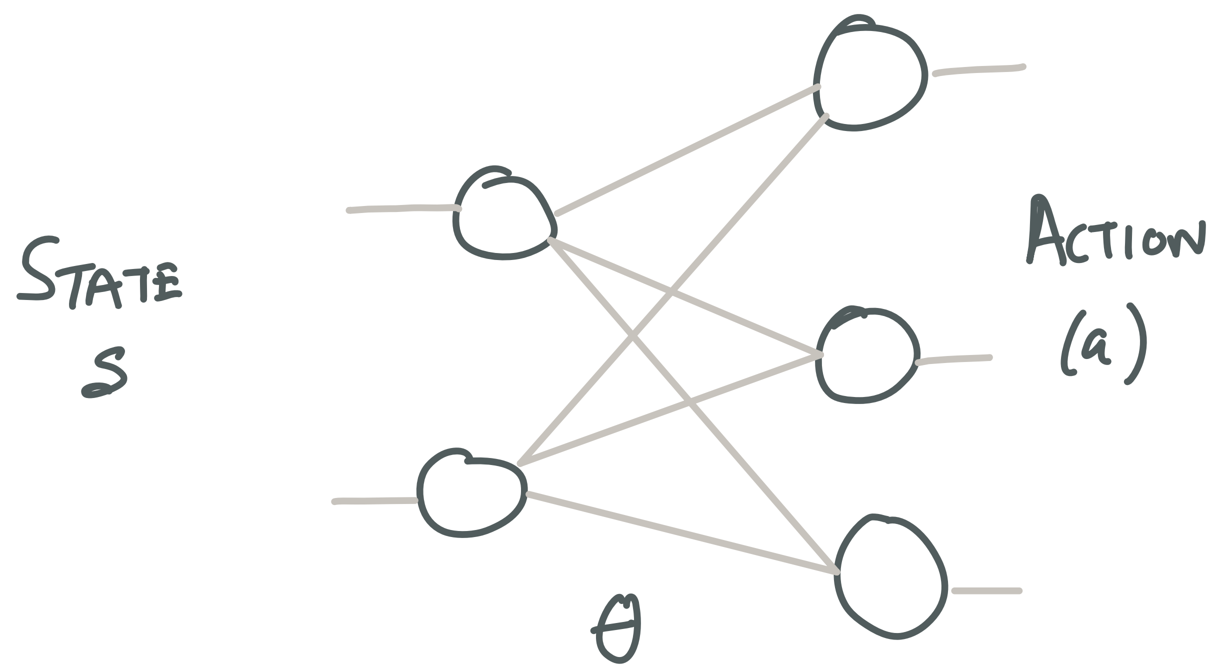
$(s_1, a_1, \hat{Q}_1)$

$$R_1 \quad (R_2 + R_3 + \dots) = \hat{Q}_2$$

$$s_1 \; a_1 \; \boxed{s_2 \; a_2} \; s_3 \; a_3 \; \dots$$

$(s_2, a_2, \hat{Q}_2)$

Policy Gradient (Multi-Step Classification)



Prob over actions: $\pi_\theta(a|s)$

Given Trajectory

$$\overset{R_1}{s_1\ a_1}\ \overset{R_2}{s_2\ a_2}\ \overset{R_3}{s_3\ a_3}\ \ldots$$

Convert to Returns

$$(R_1 + R_2 + R_3 + \cdots\cdots) = \hat{Q}_1$$

$(s_1, a_1, \hat{Q}_1)$ $\quad \boxed{s_1\ a_1}\ s_2\ a_2\ s_3\ a_3\ \ldots$

$$R_1\ (R_2 + R_3 + \cdots\cdots) = \hat{Q}_2$$

$(s_2, a_2, \hat{Q}_2)$ $\quad s_1\ a_1\ \boxed{s_2\ a_2}\ s_3\ a_3\ \ldots$

Loss

$$\mathcal{L}(\theta) = -\hat{Q}_1 \log \pi_\theta(a_1|s_1) - \hat{Q}_2 \log \pi_\theta(a_2|s_2) - \cdots$$

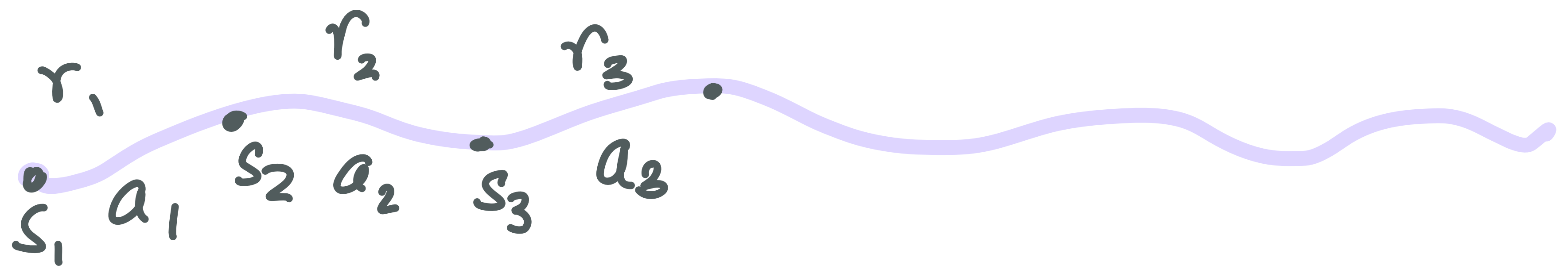
Gradient

$$\nabla_\theta \mathcal{L}(\theta) = -\hat{Q}_1 \nabla_\theta \log \pi_\theta(a_1|s_1) - \hat{Q}_2 \nabla_\theta \log \pi_\theta(a_2|s_2) - \cdots$$

Reinforce

Start with random policy π_θ

While not converged:



Rollout π_θ to collect $(s_1, a_1, r_1, s_2, a_2, r_2, \ldots\ldots)$

Create dataset $\mathcal{D} = (s_1, a_1, \hat{Q}_1, s_2, a_2, \hat{Q}_2, \ldots\ldots)$

Compute gradients $\nabla_\theta J = \sum_{(s_t, a_t, \hat{Q}_t) \in \mathcal{D}} \nabla_\theta \log \pi_\theta(a_t | s_t)\, \hat{Q}_t$

Update $\theta \leftarrow \theta + \alpha \nabla_\theta J$

Life is good!

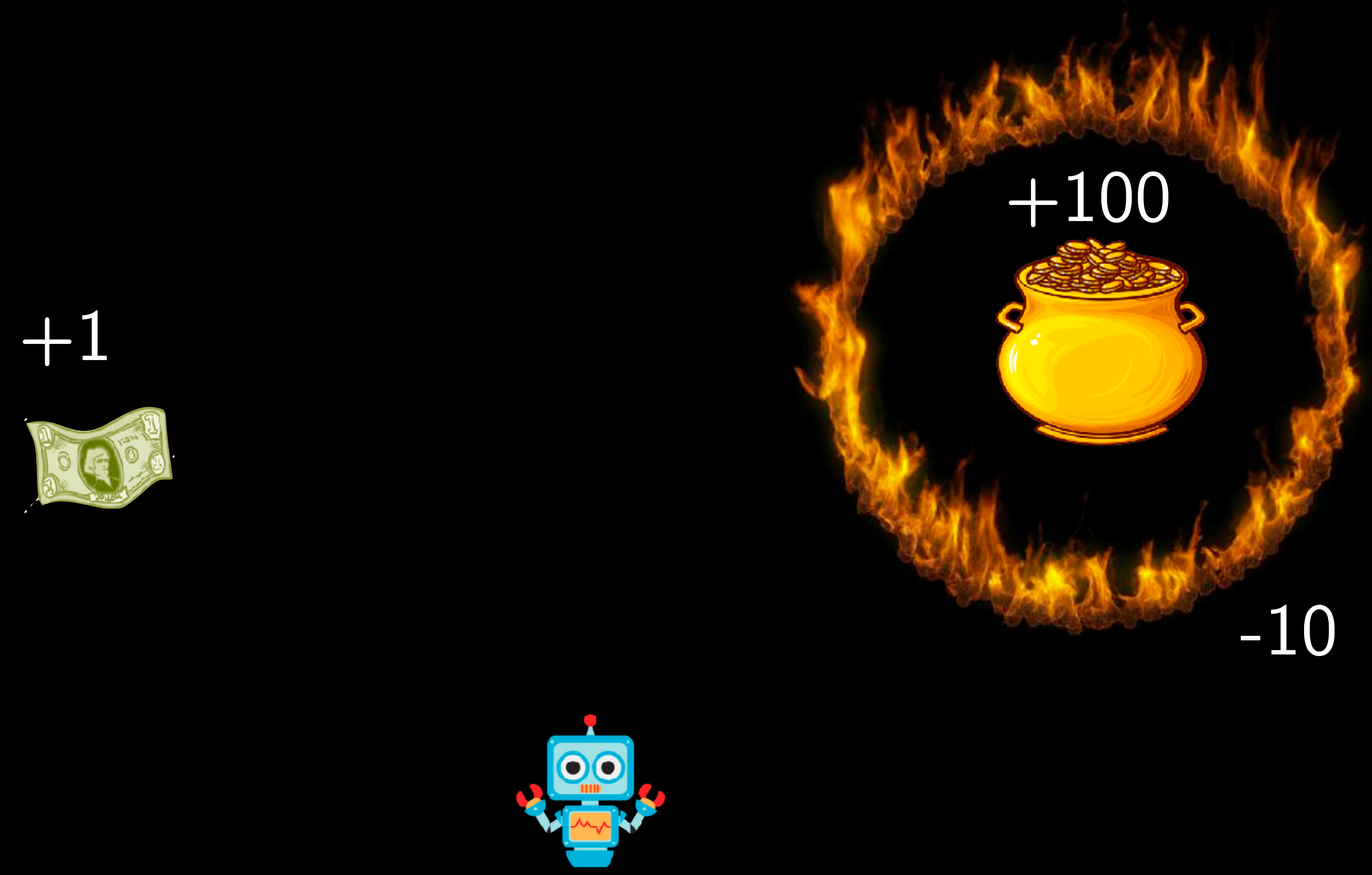
This solves everything ...

The Three Nightmares of Policy Optimization

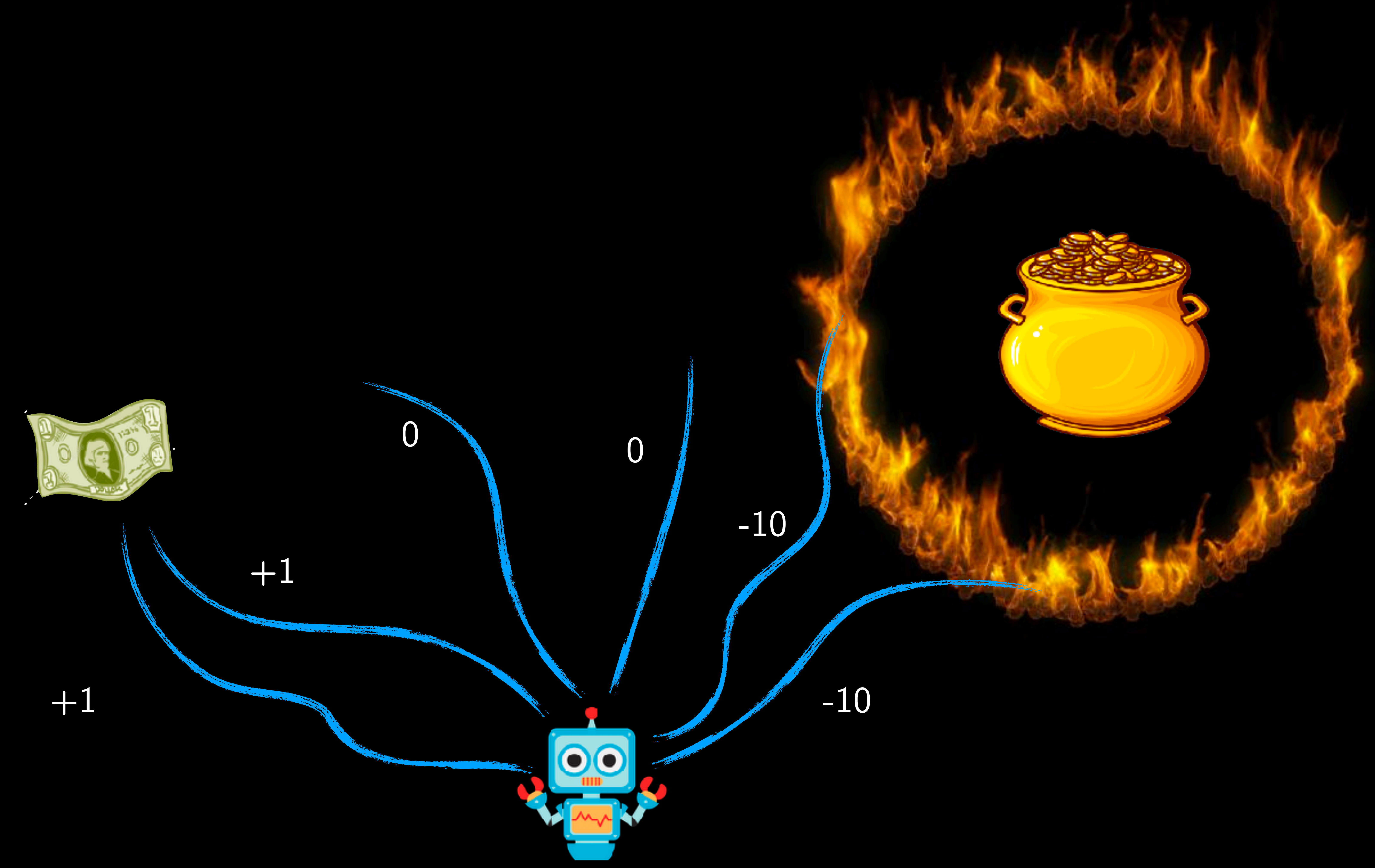
Nightmare 1:

Local Optima

The Ring of Fire

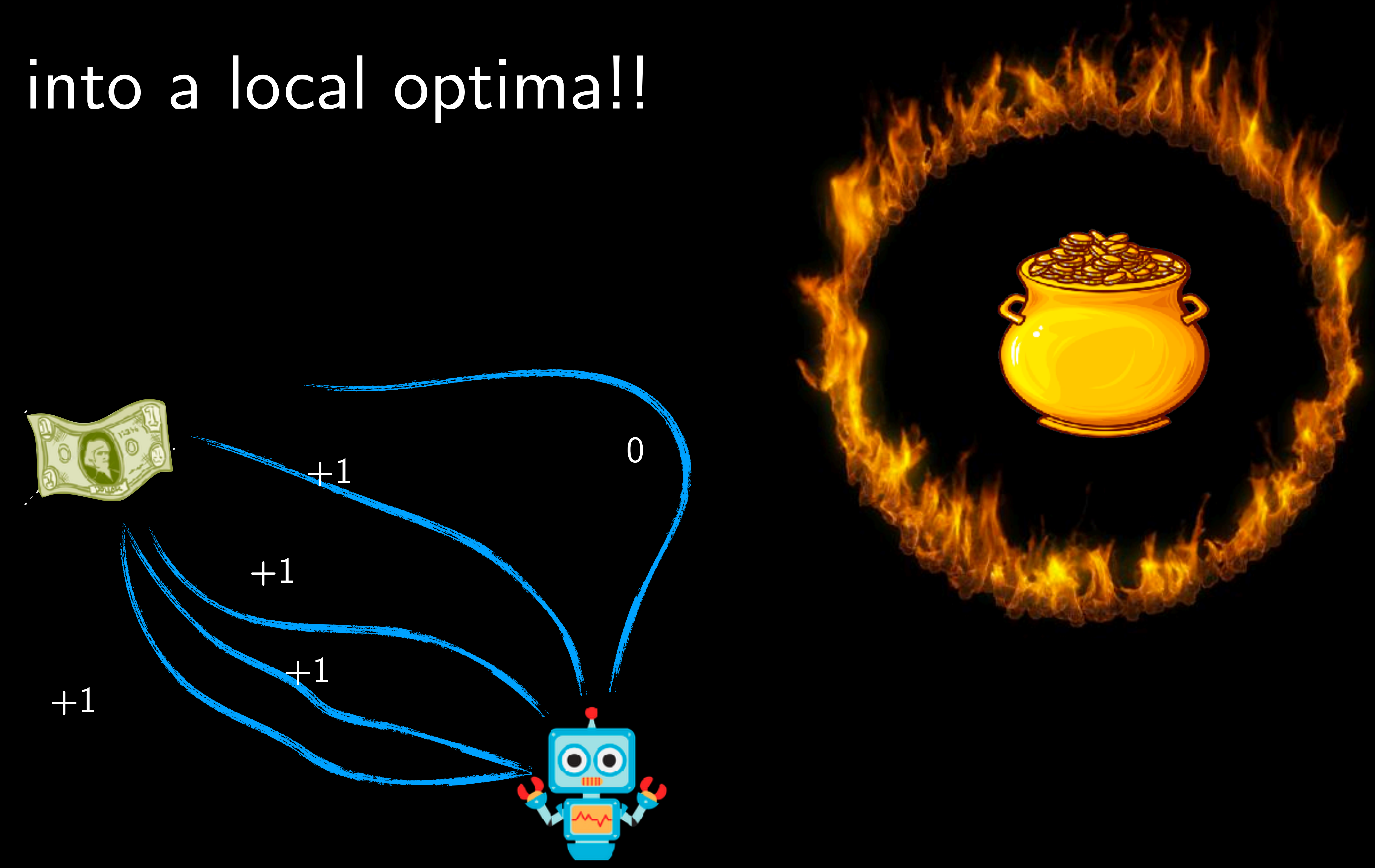


The Ring of Fire



The Ring of Fire

Get's sucked into a local optima!!

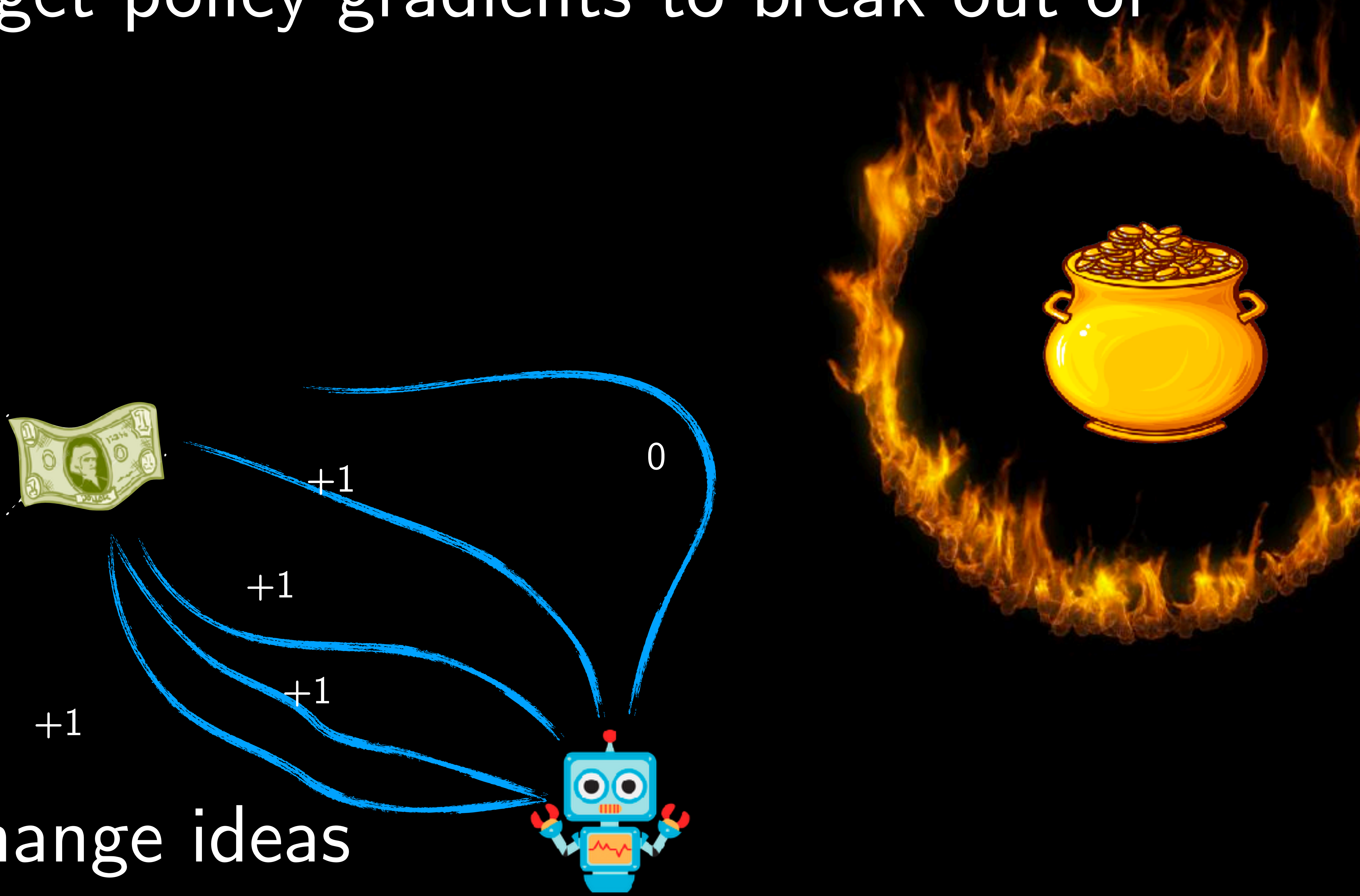


Activity!

Think-Pair-Share

Think (30 sec): How do we get policy gradients to break out of local optima?

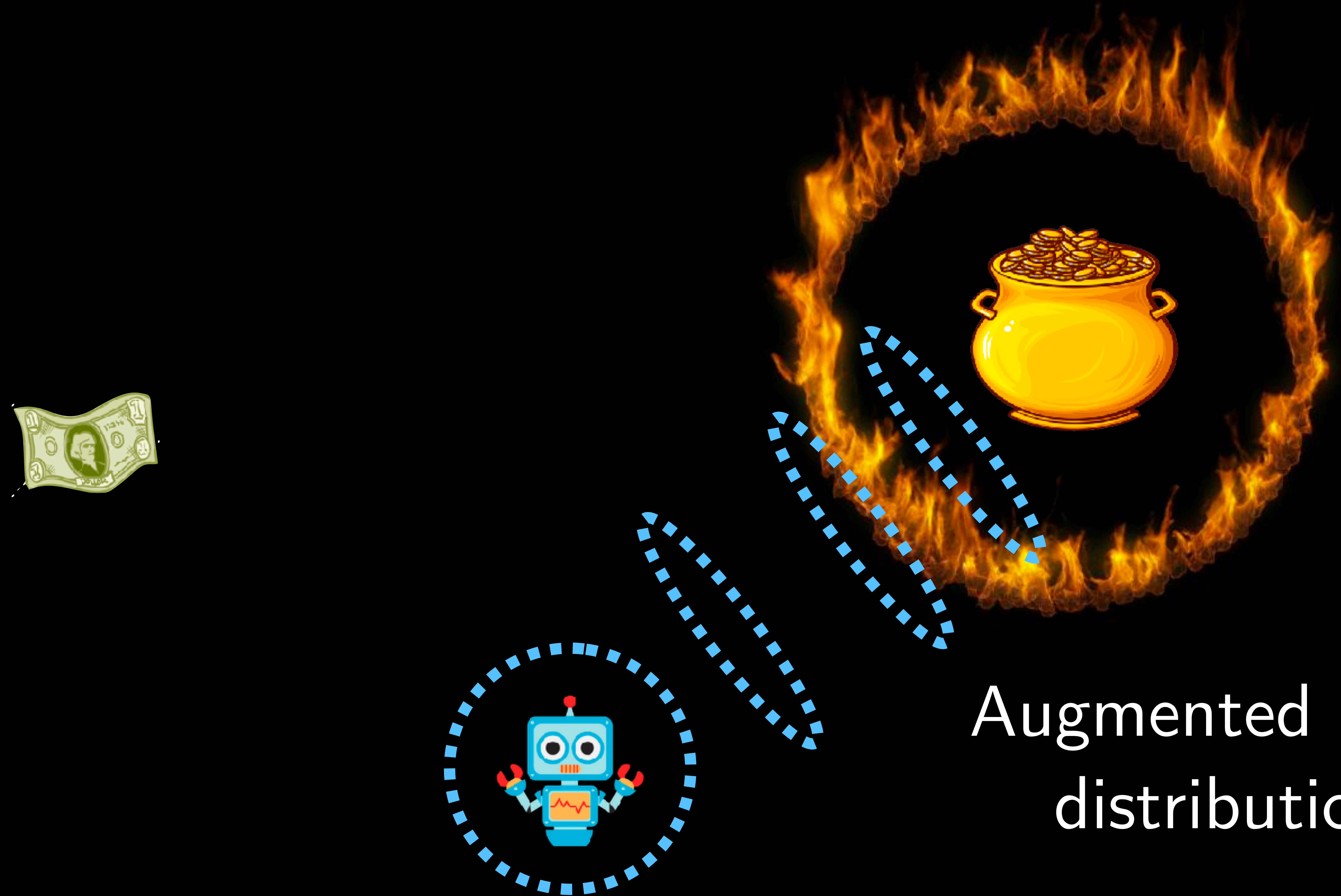
Pair: Find a partner



Share (45 sec): Partners exchange ideas

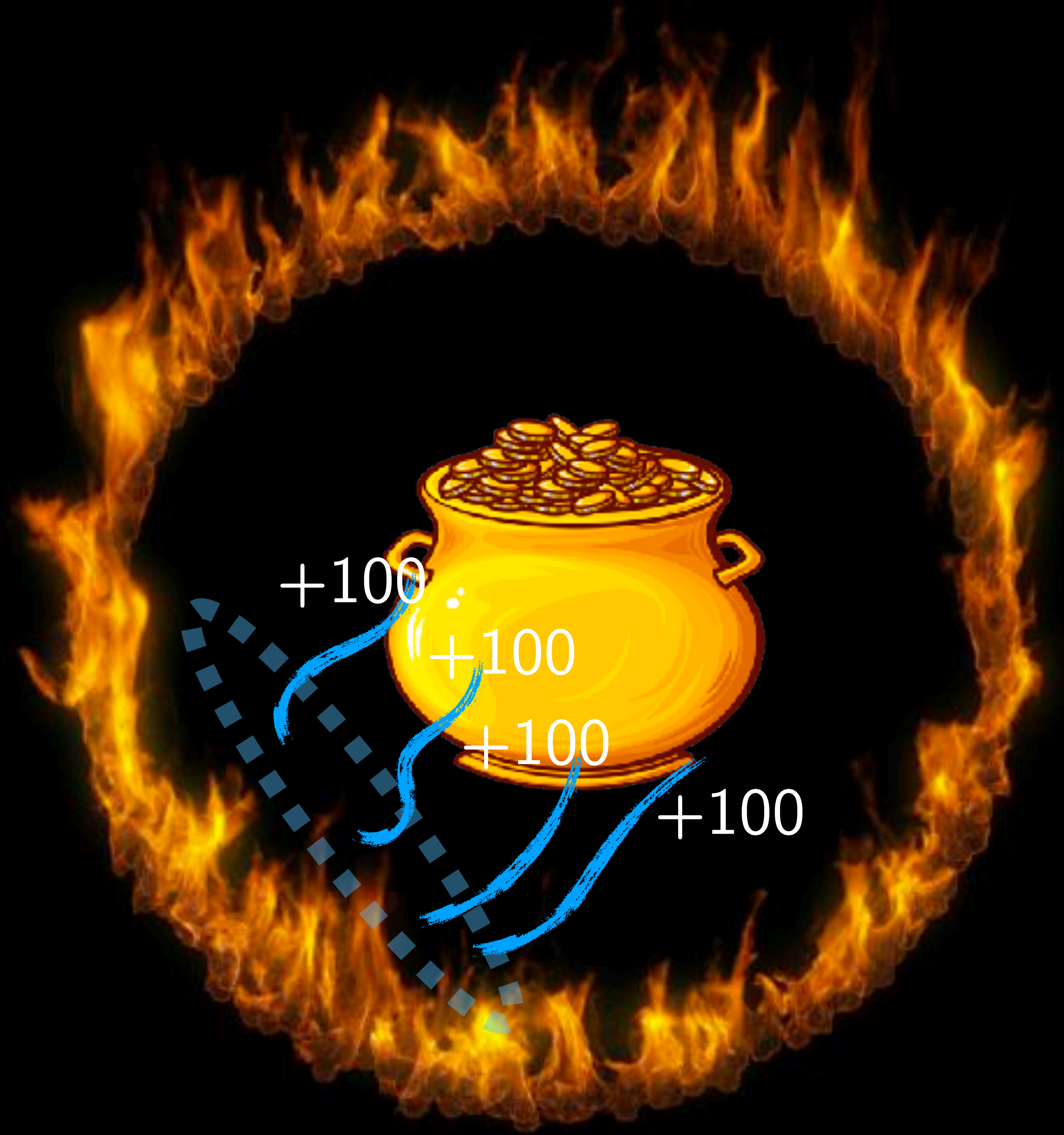
Idea: What if we had a “good reset distribution?”

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



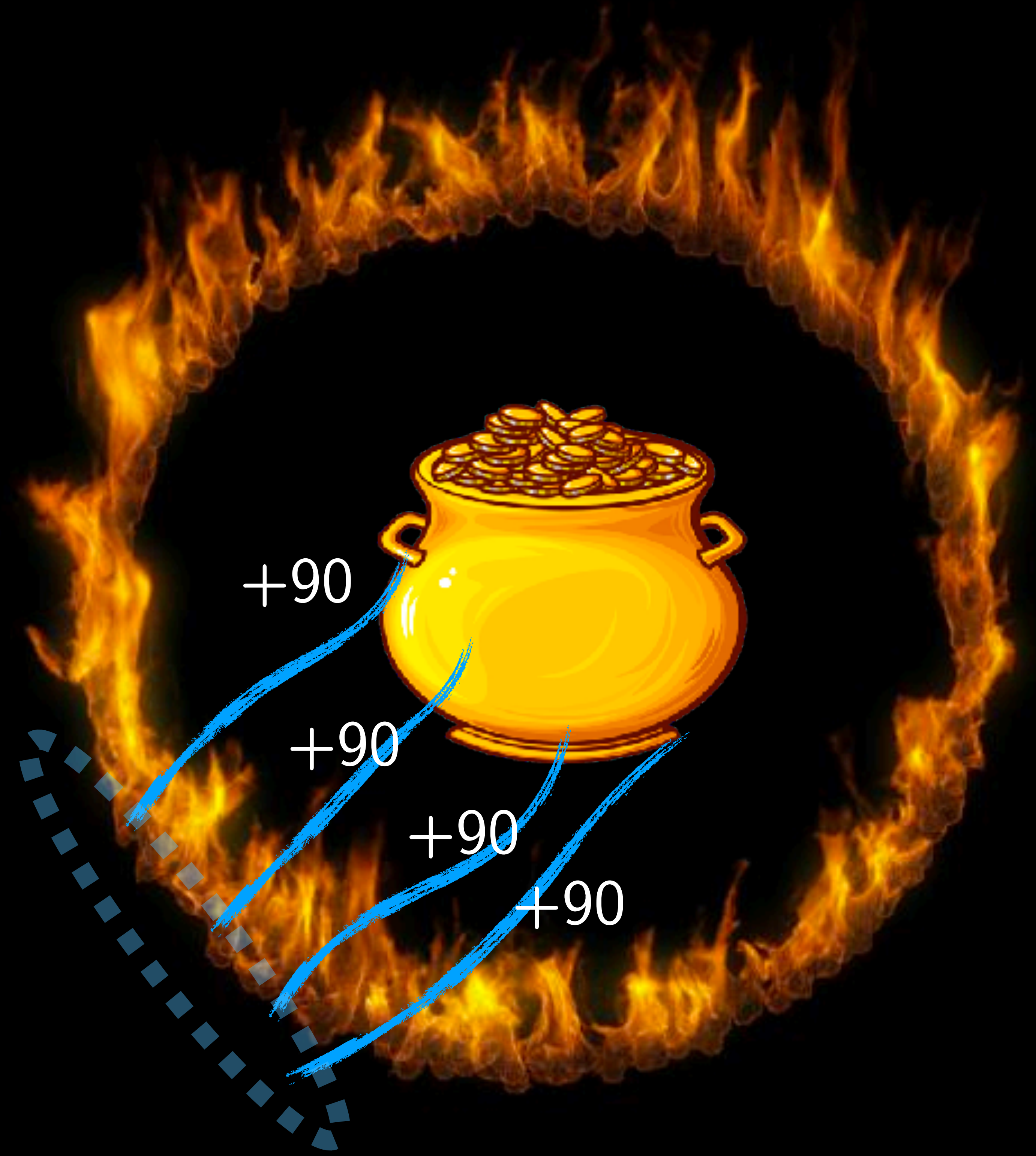
Augmented reset distribution

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



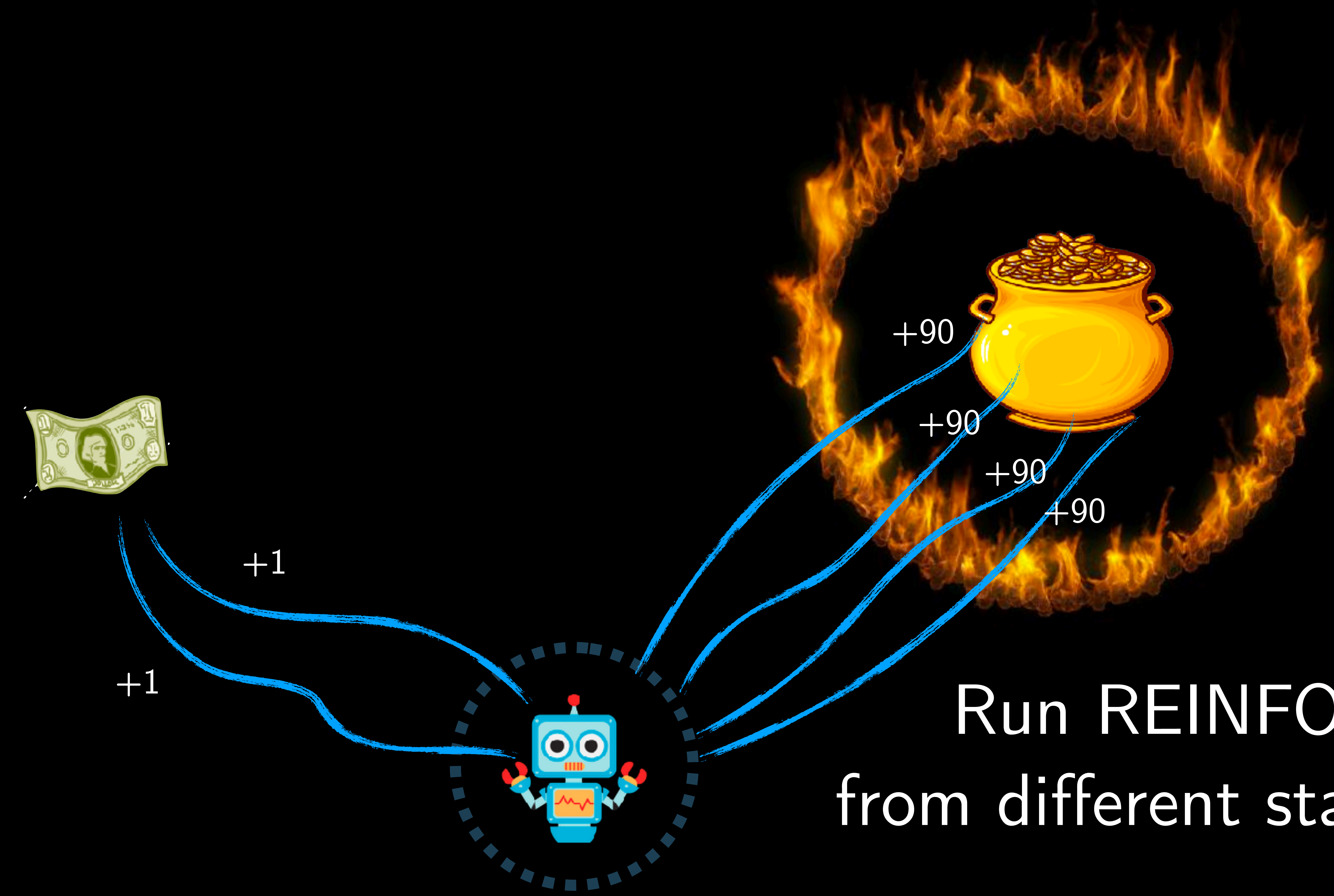
Run REINFORCE
from different start states

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

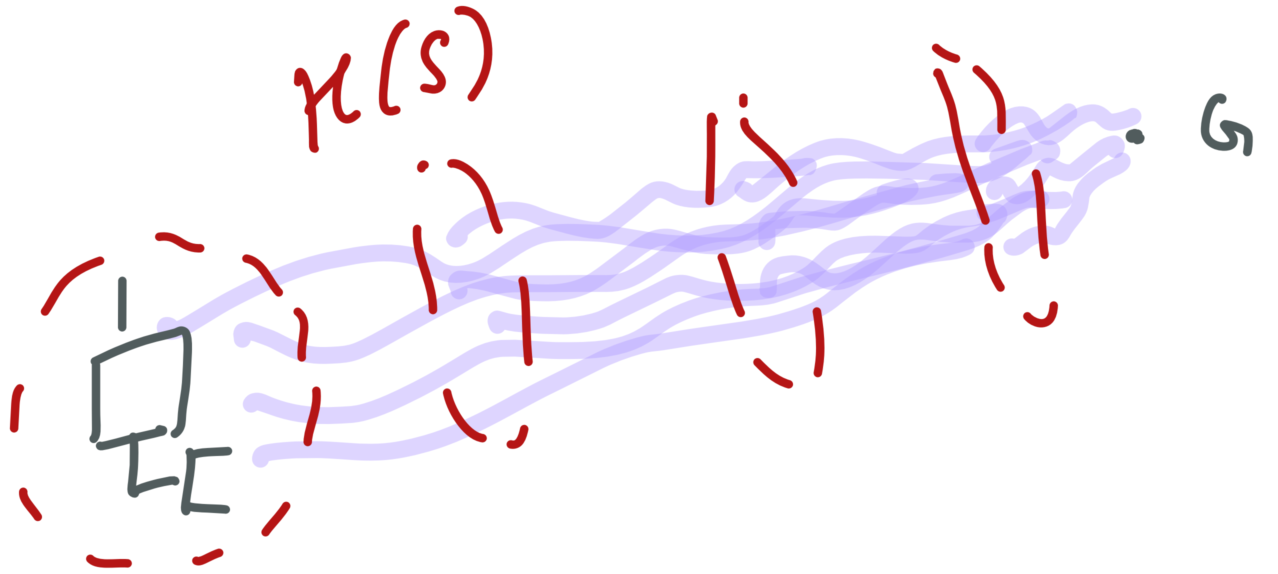
Solution: Use a good "reset" distribution

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

Try your best to "cover" states the expert will visit

Suffer at most a penalty of $\|\frac{d_{\pi^*}}{\mu}\|_\infty$

REINFORCE (WITH RESETS)

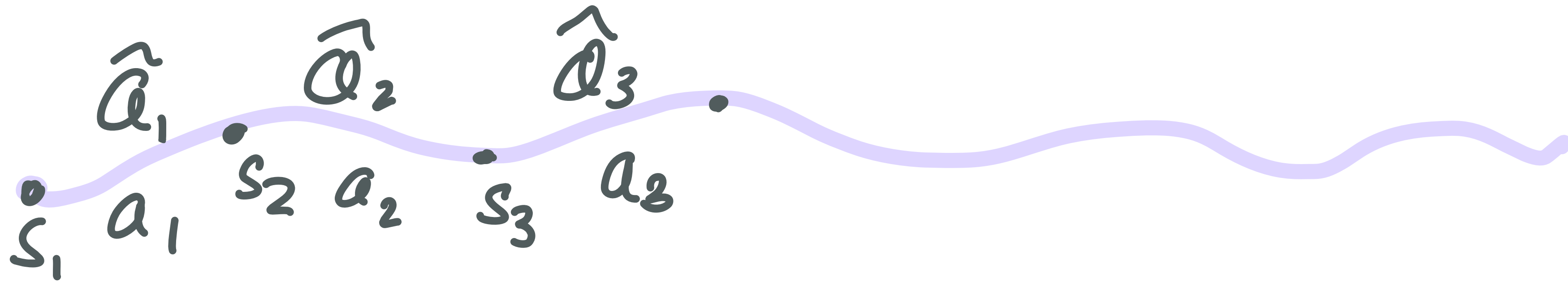


START WITH RANDOM POLICY π_θ

WHILE NOT CONVERGED:

ROLLOUT π_θ TO COLLECT $(s_1, a_1, r_1, s_2, a_2, r_2, \dots)$

(STARTING FROM INITIAL STATE $s_1 \sim \mu(s)$)



CREATE DATASET $D = (s_1, a_1, \hat{Q}_1, s_2, a_2, \hat{Q}_2, \dots)$

COMPUTE GRADIENTS $\nabla_\theta J = \sum_{(s_t, a_t, \hat{Q}_t) \in D} \nabla_\theta \log \pi_\theta(a_t | s_t)\, \hat{Q}_t$

UPDATE $\theta \leftarrow \theta + \alpha \nabla_\theta J$

Nightmare 2:

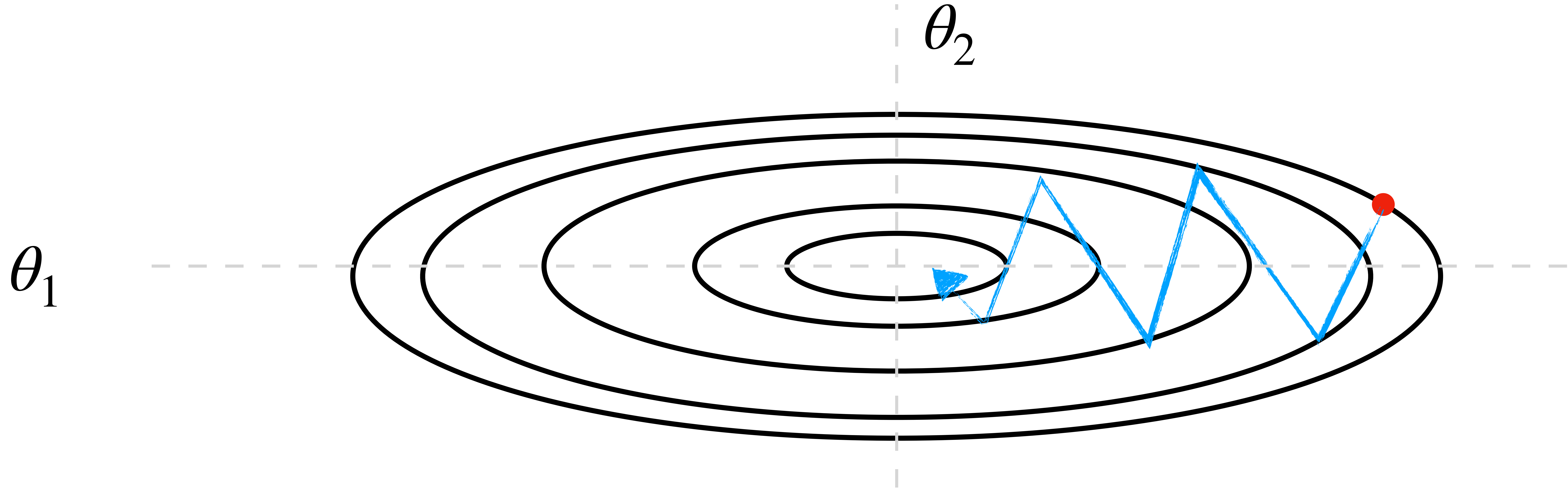
Distribution Shift

Is gradient descent the best direction?

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

Note all the terms in the above equation that depend on theta.
If we change theta by a small amount, how do these terms change?

What would gradient descent do here?



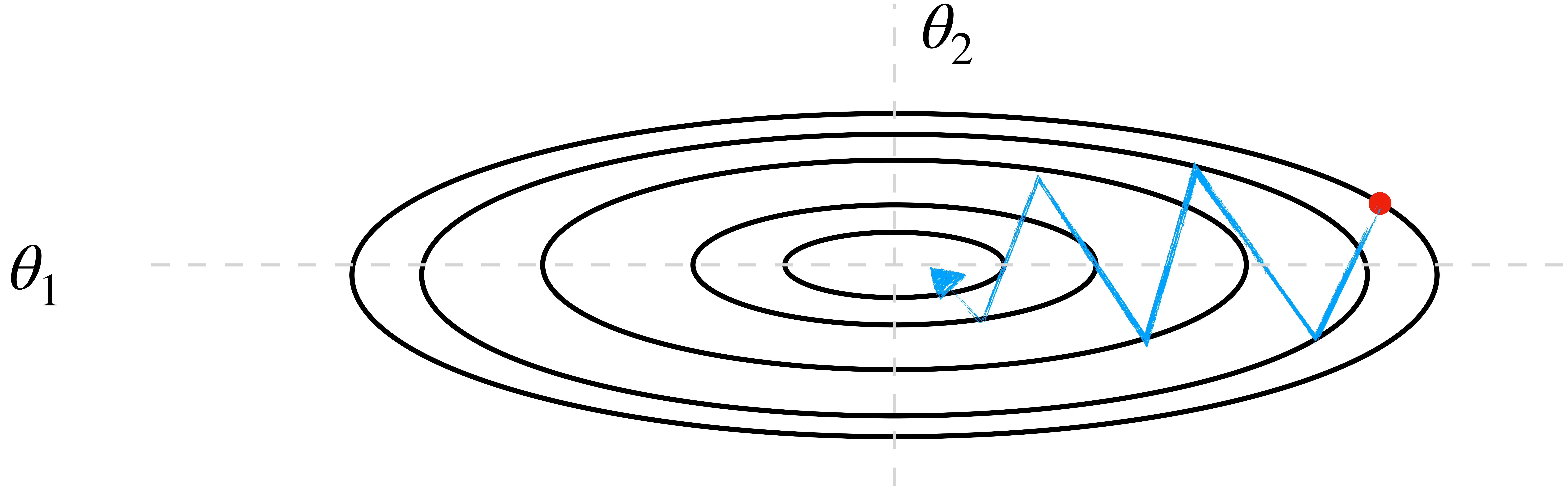
What assumption does it make that is breaking?

How can we make it choose a better direction?

Gradient Descent as Steepest Descent

Gradient Descent is simply Steepest Descent with L2 norm

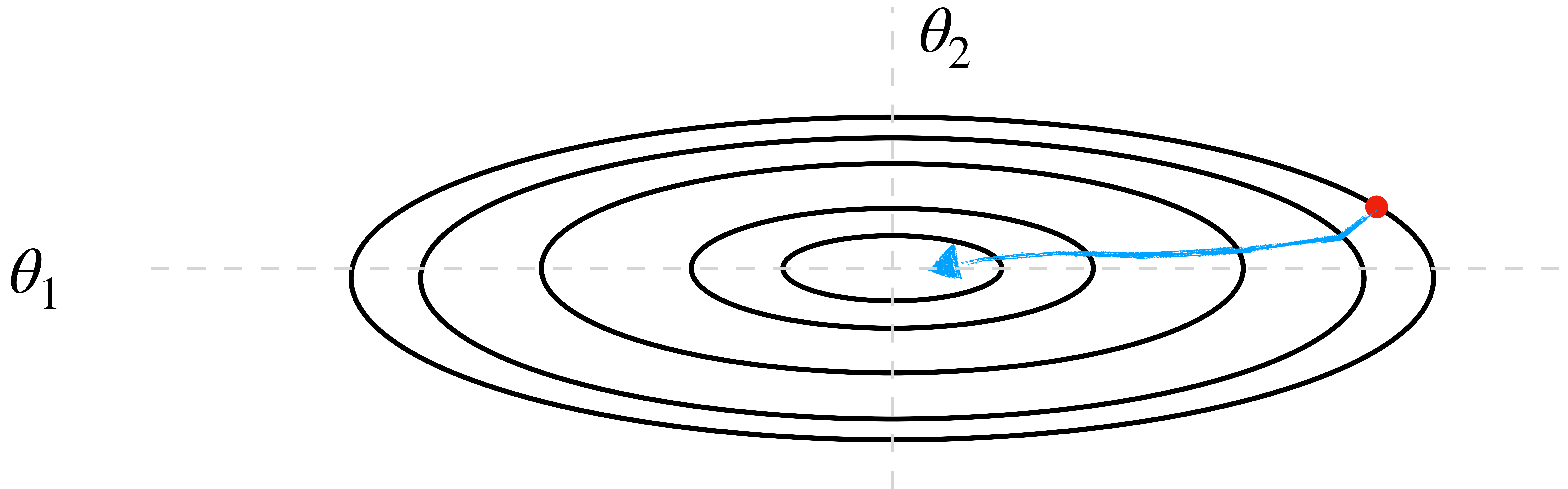
$$\min_{\Delta\theta} J(\theta + \Delta\theta) \ \text{ s.t } \ ||\Delta\theta|| \leq \epsilon \quad \longrightarrow \quad \Delta\theta = -\nabla_{\theta} J(\theta)$$



Steepest Descent with a different norm

A different norm G means a different notion of "small step"

$$\min_{\Delta\theta} J(\theta + \Delta\theta) \;\; \text{s.t} \;\; \Delta\theta^T G \Delta\theta \leq \epsilon \longrightarrow \Delta\theta = -G^{-1}\nabla_\theta J(\theta)$$



What is the best norm for policy gradient?

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

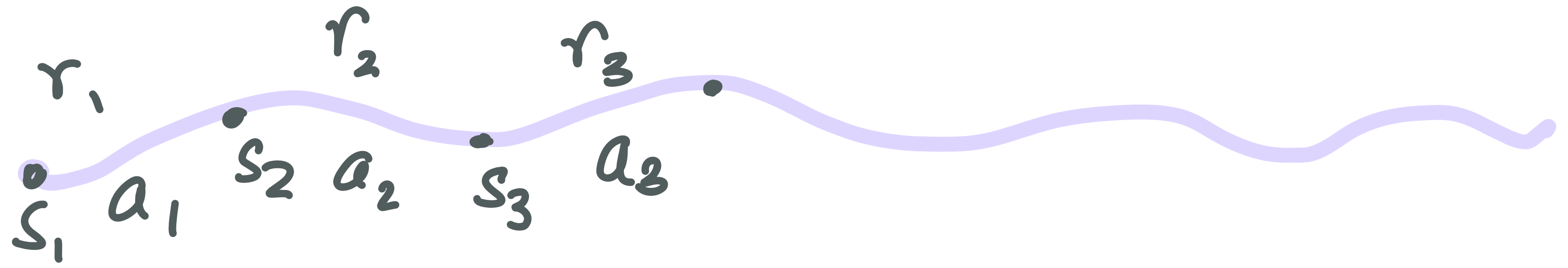
Don't make small changes in θ, make small changes in the "distribution $\pi_\theta(a\,|\,s)$"

$$\min_{\Delta\theta} J(\theta + \Delta\theta) \qquad \text{s.t. } KL(\pi_{(\theta+\Delta\theta)} \,||\, \pi_\theta) \le \epsilon$$

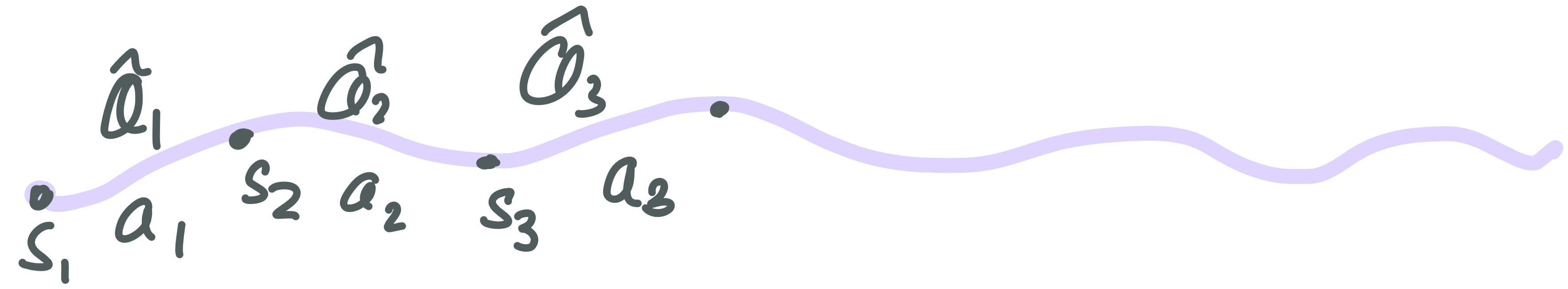
REINFORCE (WITH NATURAL GRADIENTS)

START WITH RANDOM POLICY π_θ

WHILE NOT CONVERGED:



ROLLOUT π_θ TO COLLECT $(s_1, a_1, r_1, s_2, a_2, r_2, \ldots\ldots)$



CREATE DATASET $D = (s_1, a_1, \hat{Q}_1, s_2, a_2, \hat{Q}_2, \ldots\ldots)$

COMPUTE GRADIENTS $\nabla_\theta J = \sum_{(s_t, a_t, \hat{Q}_t) \in D} \nabla_\theta \log \pi_\theta (a_t | s_t) \hat{Q}_t$

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

$G(\theta) = \sum \left[\nabla_\theta \log \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s)^T \right]$

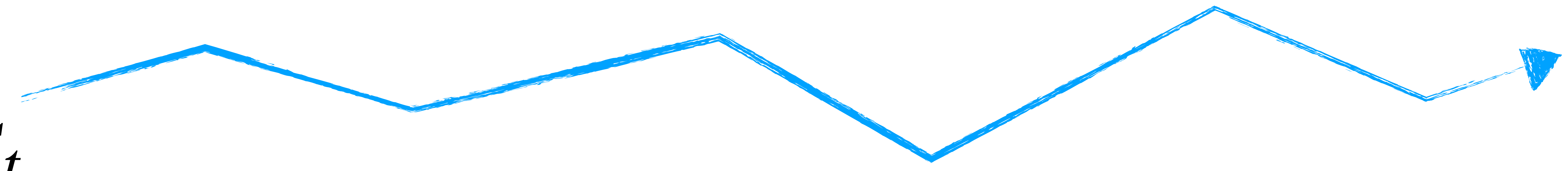
FISCHER INFORMATION MATRIX

Nightmare 3:

High Variance

Consider the following single roll-out

Return = -100

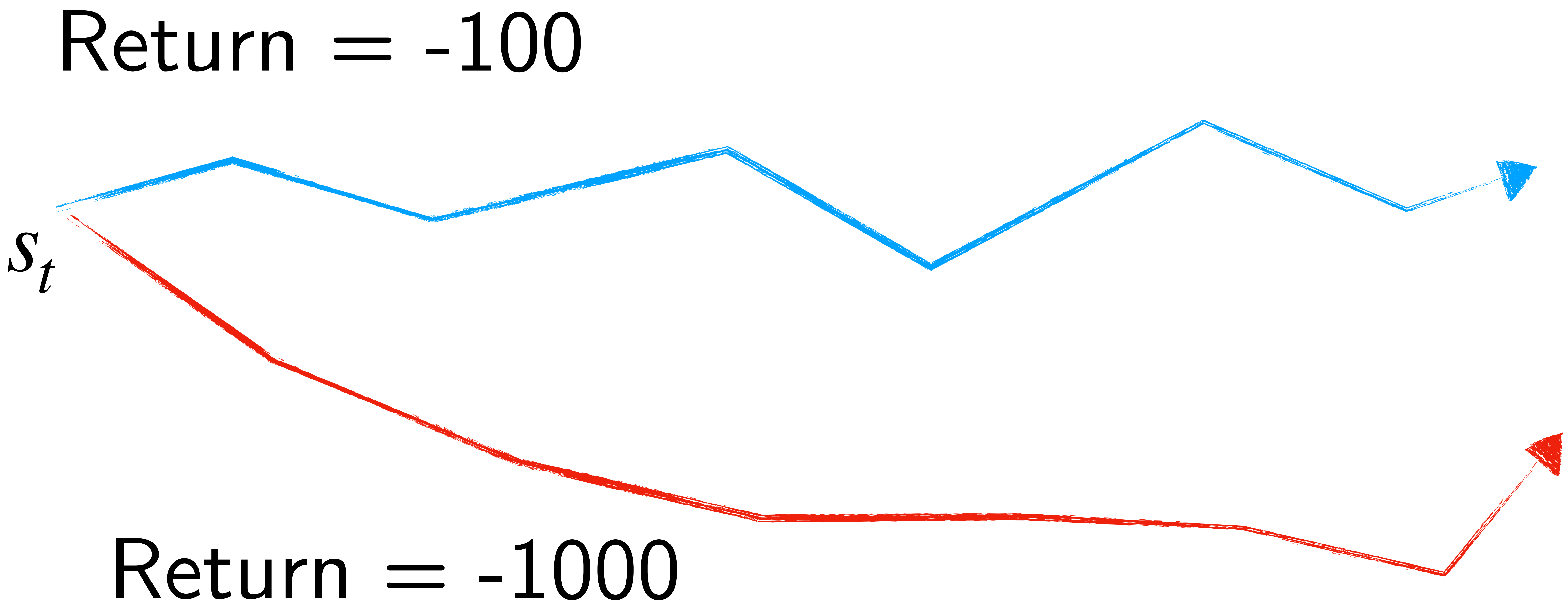


s_t

What would the gradient at s_t be?

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

It depends on other trajectories!



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!

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

Does this bias the gradient ??

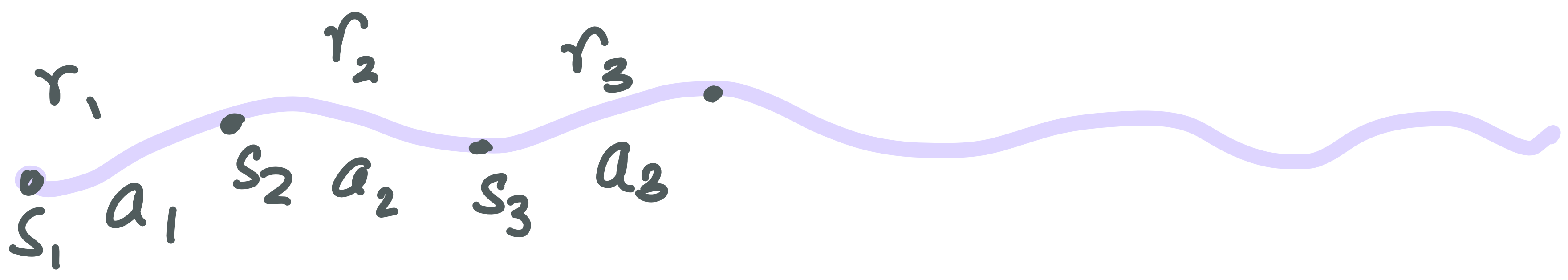
$$= E_{d^{\pi_\theta}(s)} E_{\pi_\theta(a|s)} \left[\nabla_\theta \log(\pi_\theta(a|s) A^{\pi_\theta}(s,a) \right]$$

(Advantage)

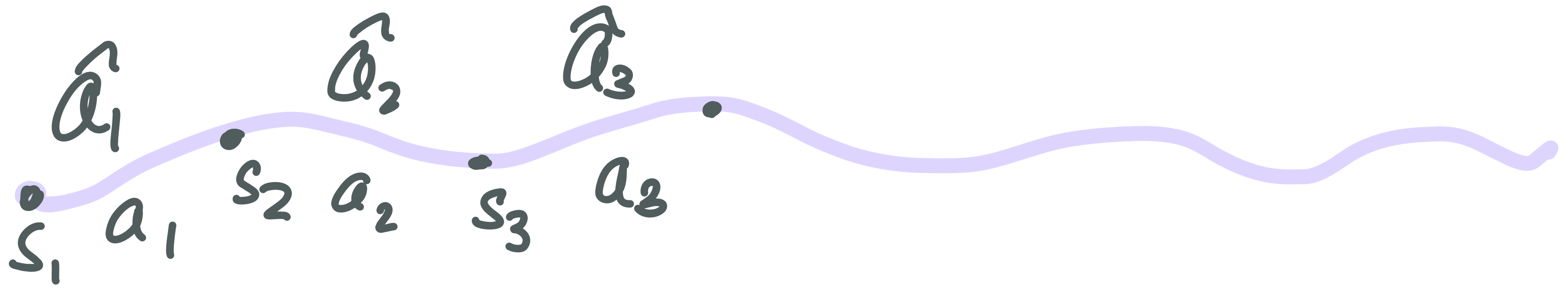
Reinforce (with Baseline)

Start with random policy π_θ

While not converged:

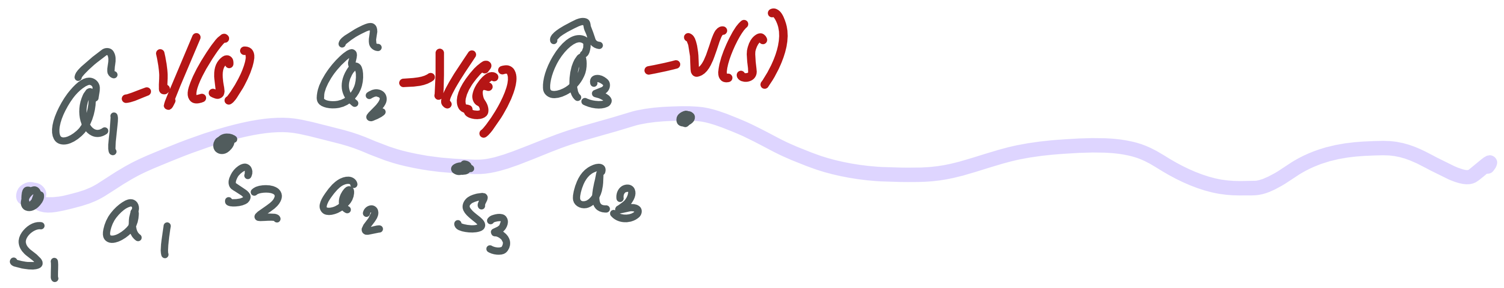


Rollout π_θ to collect $(s_1, a_1, r_1, s_2, a_2, r_2, \ldots\ldots)$



Create dataset $D = (s_1, a_1, \hat{Q}_1, s_2, a_2, \hat{Q}_2, \ldots\ldots)$

Train $V_\phi(s) = \sum_{(s_t, \hat{Q}) \in D} \| V_\phi(s) - \hat{Q}_1 \|$



Compute gradients $\nabla_\theta J = \sum_{\substack{(s_t, a_t, \hat{Q}_t) \\ \in D}} \nabla_\theta \log \pi_\theta(a_t \mid s_t)\left(\hat{Q}_t - V_\phi(s_t)\right)$

Update $\theta \leftarrow \theta + \alpha \nabla_\theta J$

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

Policy improvement of π

Natural Gradient Descent

Critic

Estimates value functions $Q^{\pi}_{\phi}/V^{\pi}_{\phi}/A^{\pi}_{\phi}$

TD, MC

The General Actor Critic Framework

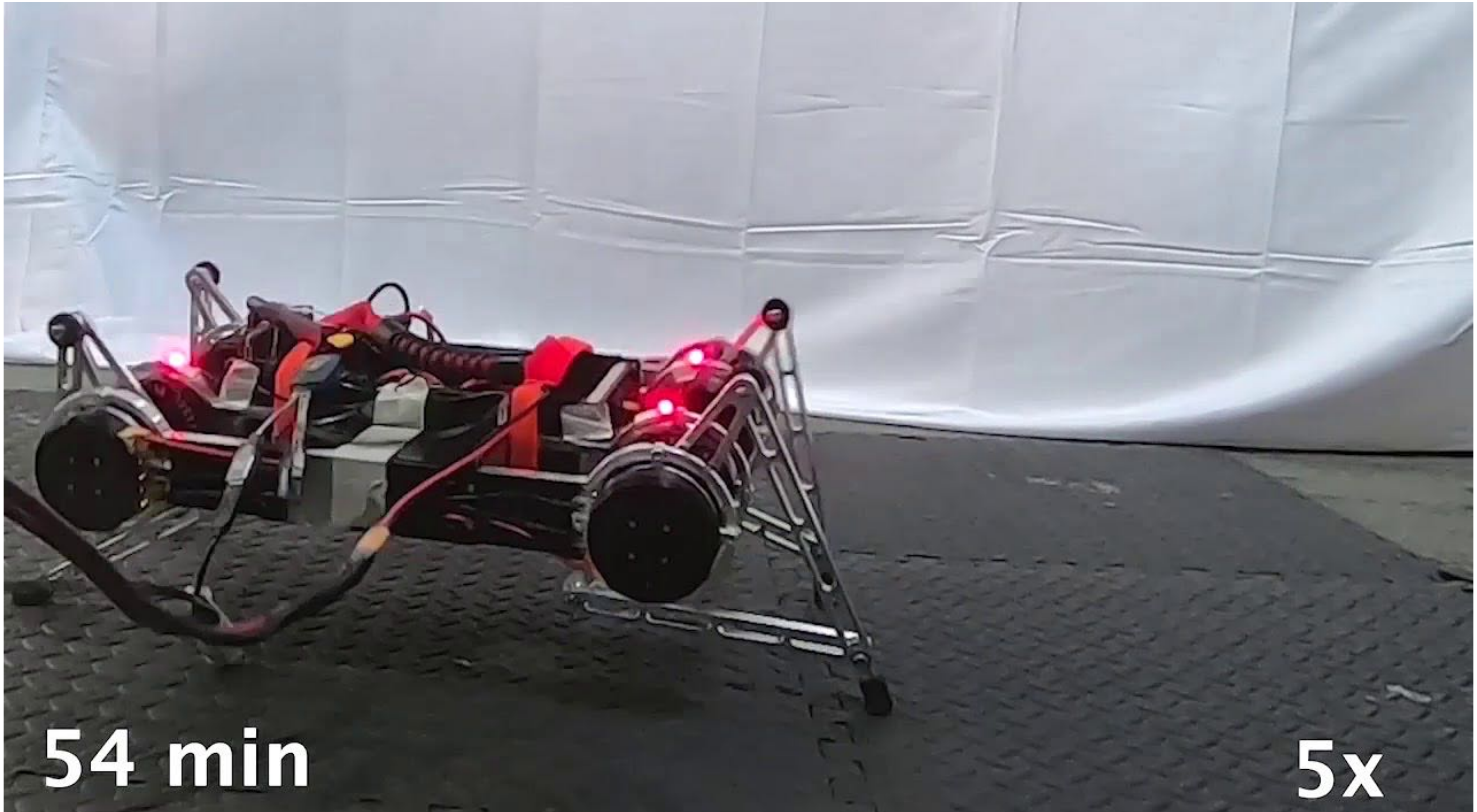
batch actor-critic algorithm:

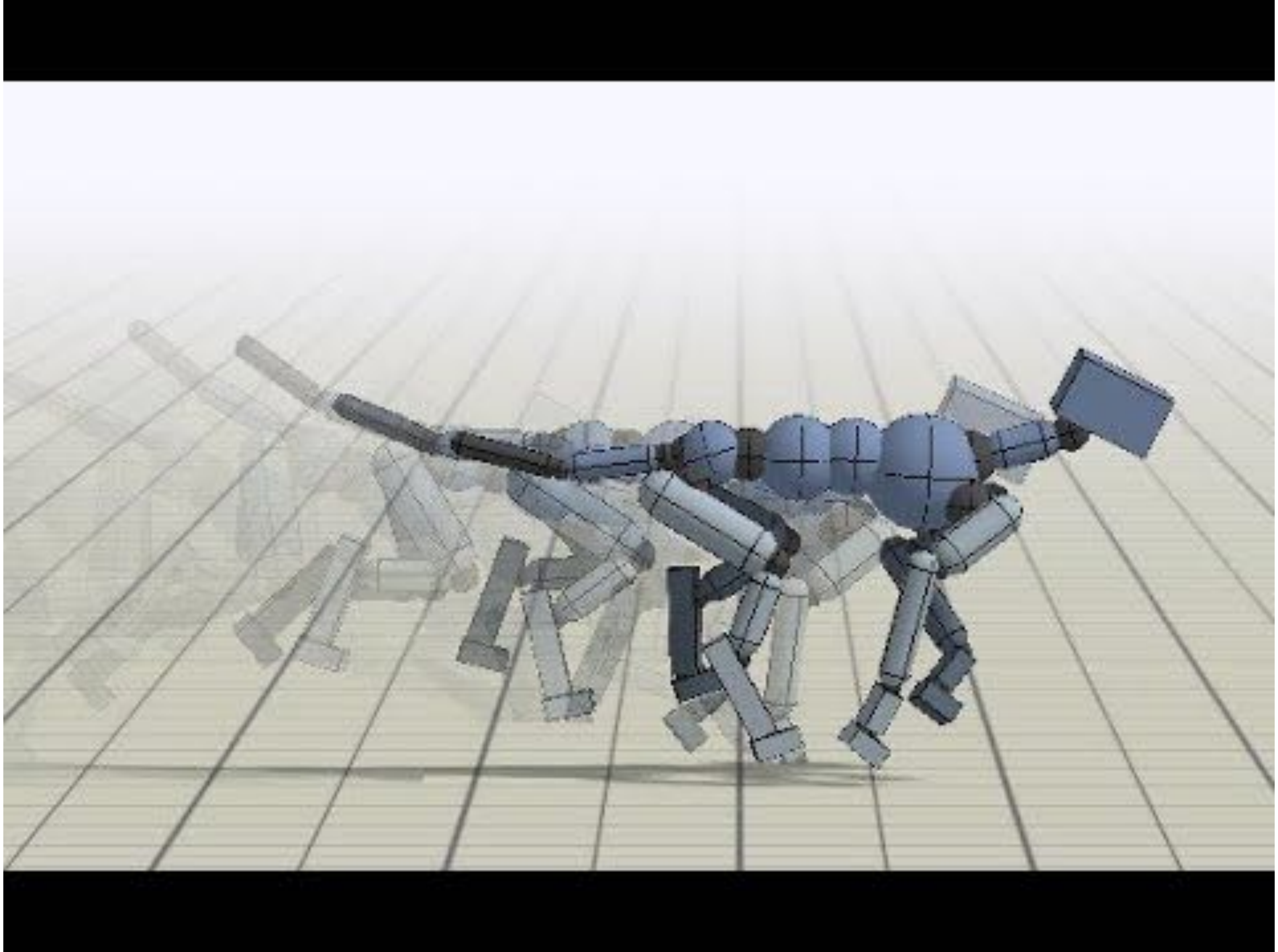
1. sample $\{\mathbf{s}_i, \mathbf{a}_i\}$ from $\pi_\theta(\mathbf{a}|\mathbf{s})$ (run it on the robot)
2. fit $\hat{V}_\phi^\pi(\mathbf{s})$ to sampled reward sums (TD, MC)
3. evaluate $\hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i) = r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \hat{V}_\phi^\pi(\mathbf{s}_i')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

Haarnoja 2018

"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

The Policy Gradient Theorem

$$\nabla_\theta J = E_{p(\xi|\theta)}\left[\sum_{t=0}^{T-1}\left(\nabla_\theta \log \pi_\theta(a_t|s_t)\left(\sum_{t'=0}^{t-1} r(s_{t'},a_{t'}) + \sum_{t'=t}^{T-1} r(s_{t'},a_{t'})\right)\right)\right]$$

$$= E_{p(\xi|\theta)}\left[\sum_{t=0}^{T-1}\left(\nabla_\theta \log \pi_\theta(a_t|s_t)\sum_{t'=t}^{T-1} r(s_{t'},a_{t'})\right)\right],$$

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

15

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