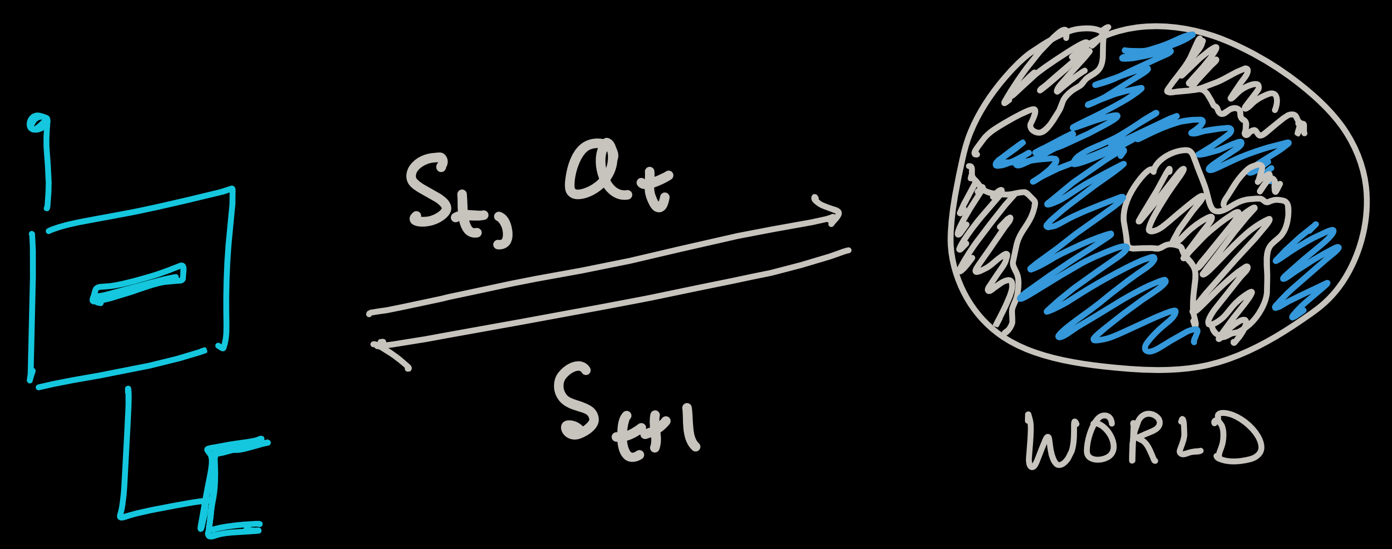
Nightmares of Policy Optimization

Sanjiban Choudhury

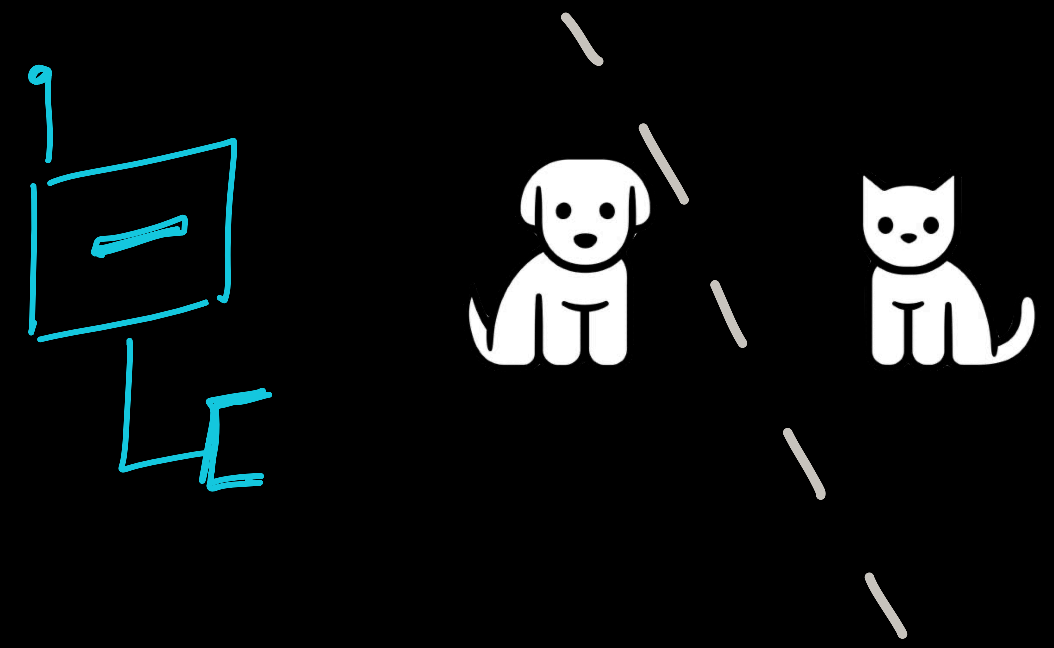
WHAT MAKES



REINFORCEMENT LEARNING

HARDER

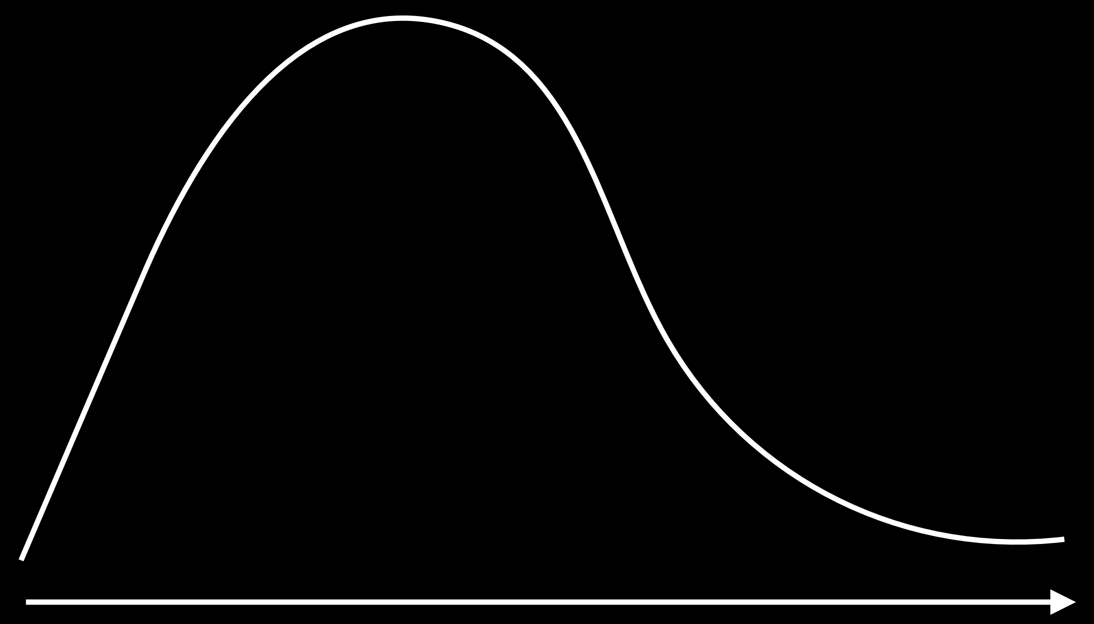
THAN



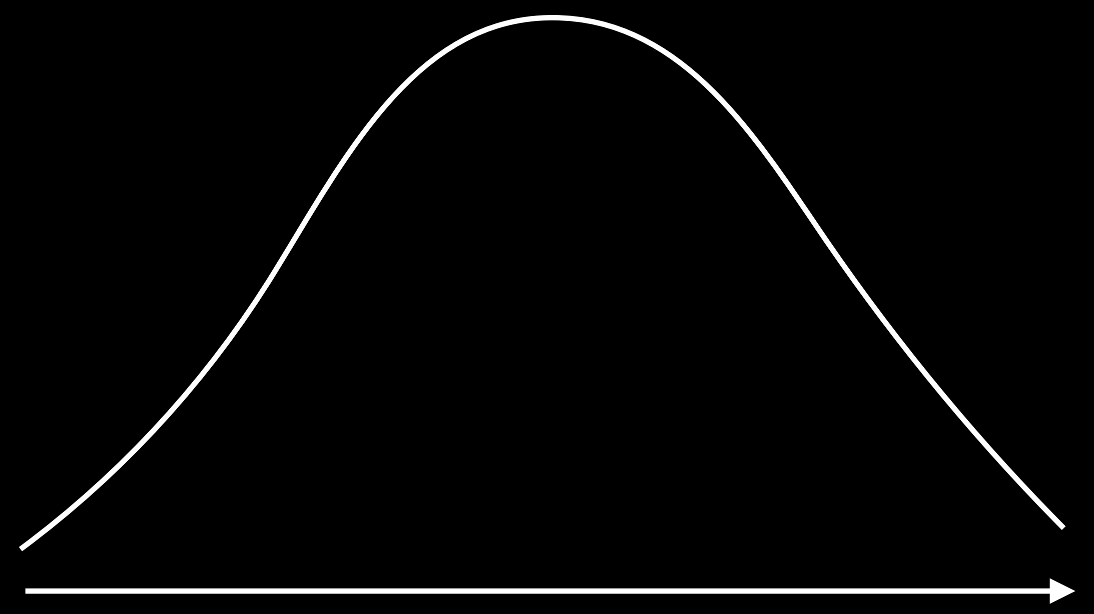
SUPERVISED LEARNING

?

Bellman is Beautiful …

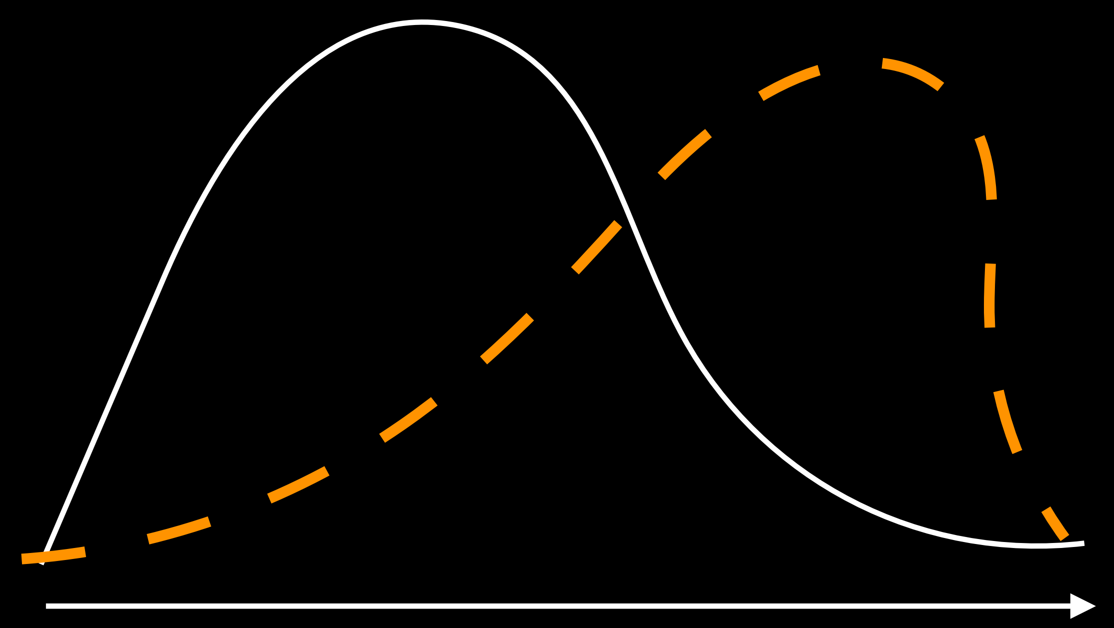


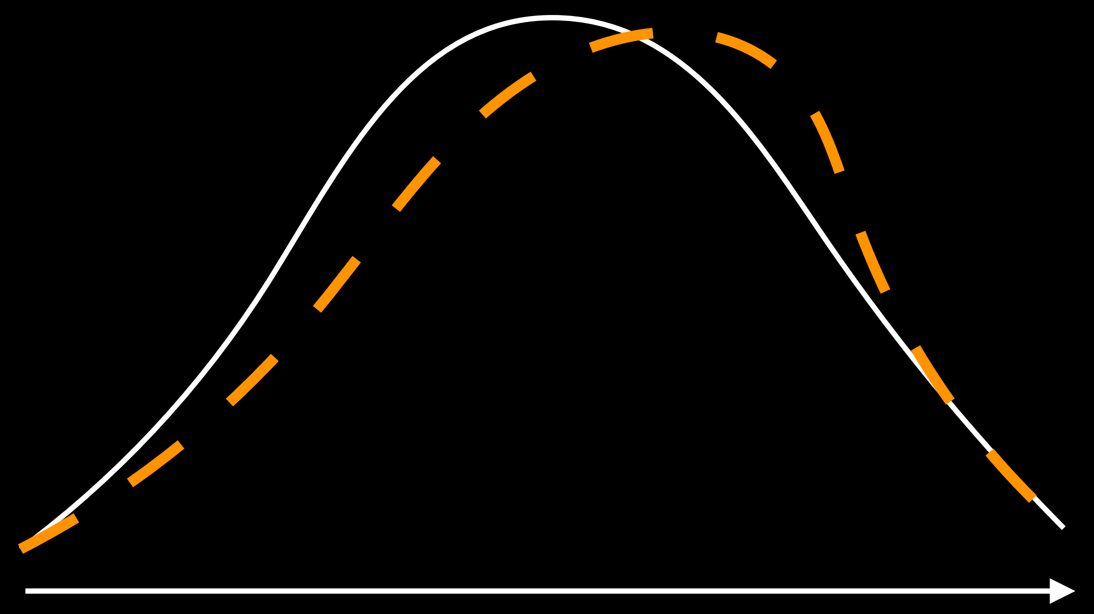
$$V^*(s)$$



$$\max_{a} r(s, a) + V^*(s')$$

But errors in Bellman compound!!!





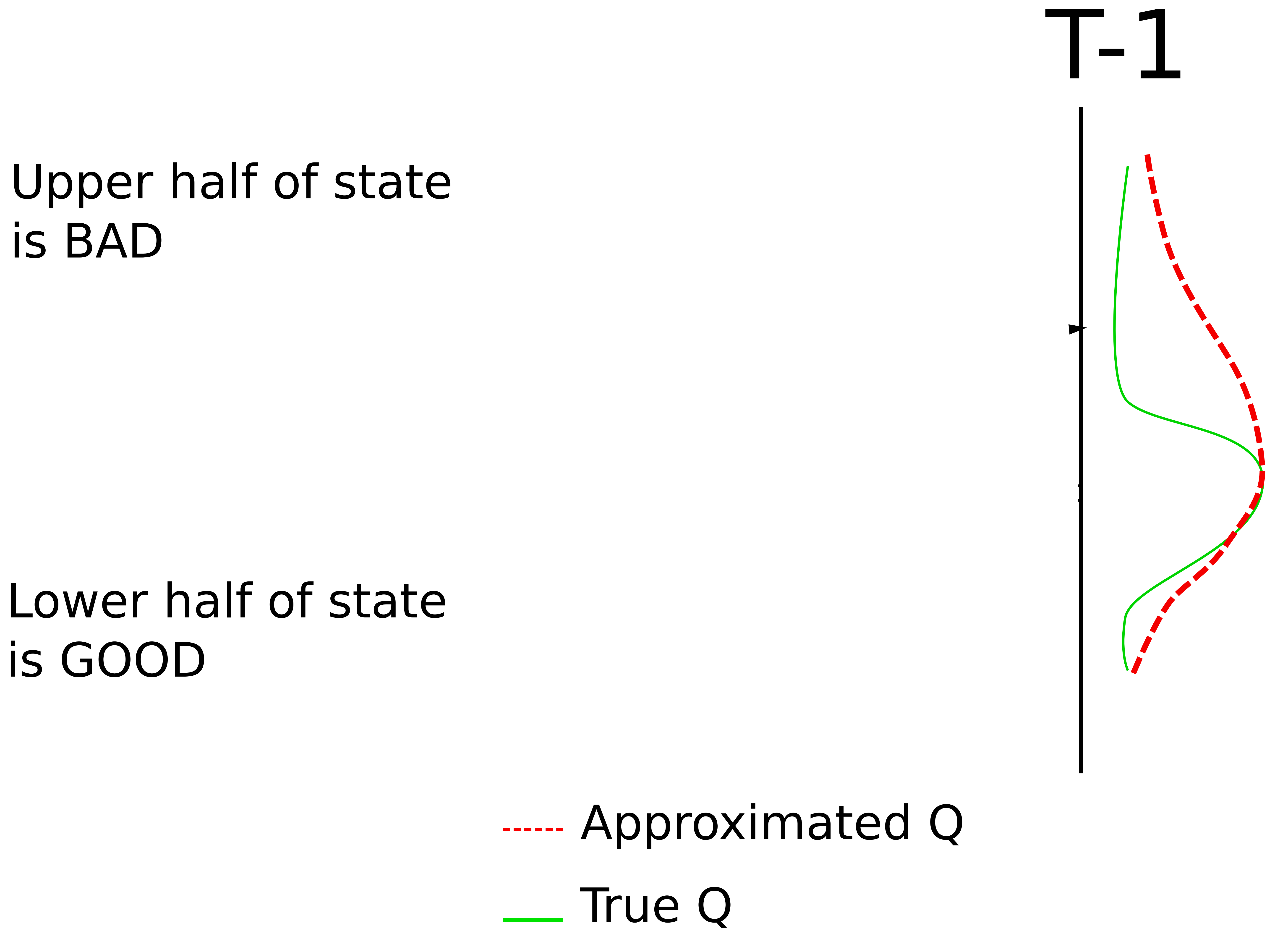
$V^{\cancel{*}}(s)$

$\max_a r(s,\cancel{a}) + V(s')$

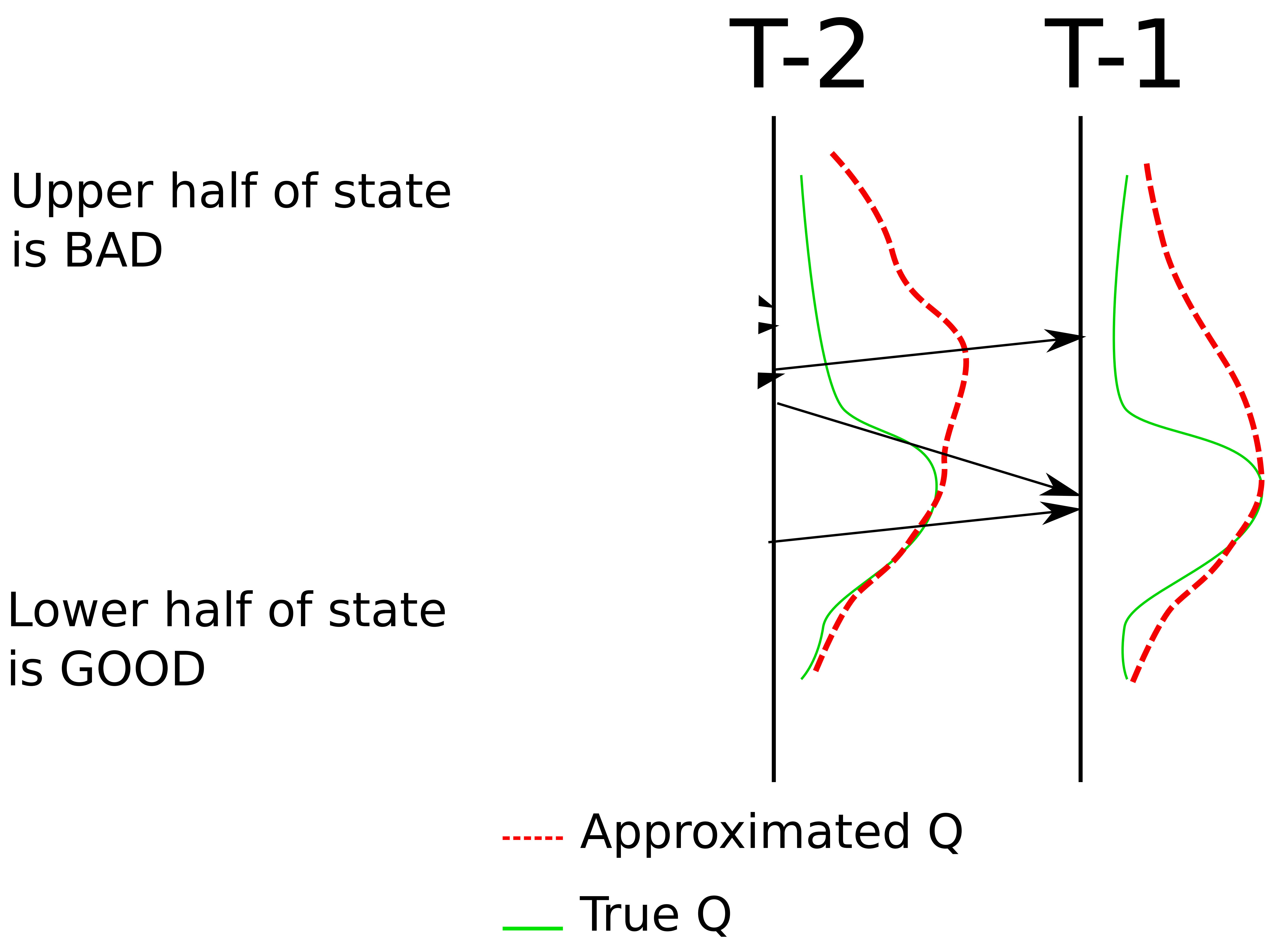
$V_\theta(s)$

$\max_a r(s,a) + V_\theta(s')$

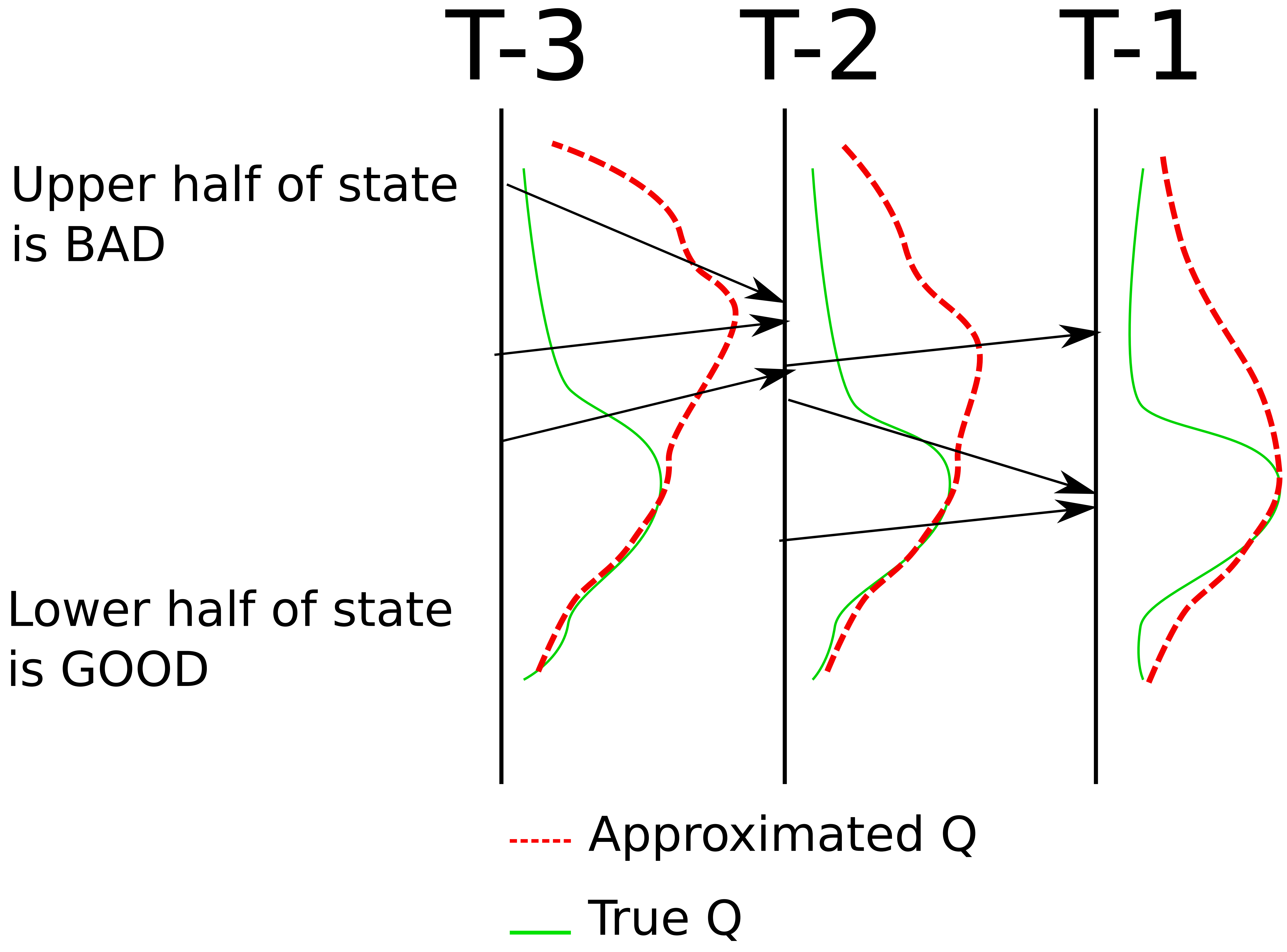
The problem of distribution shift



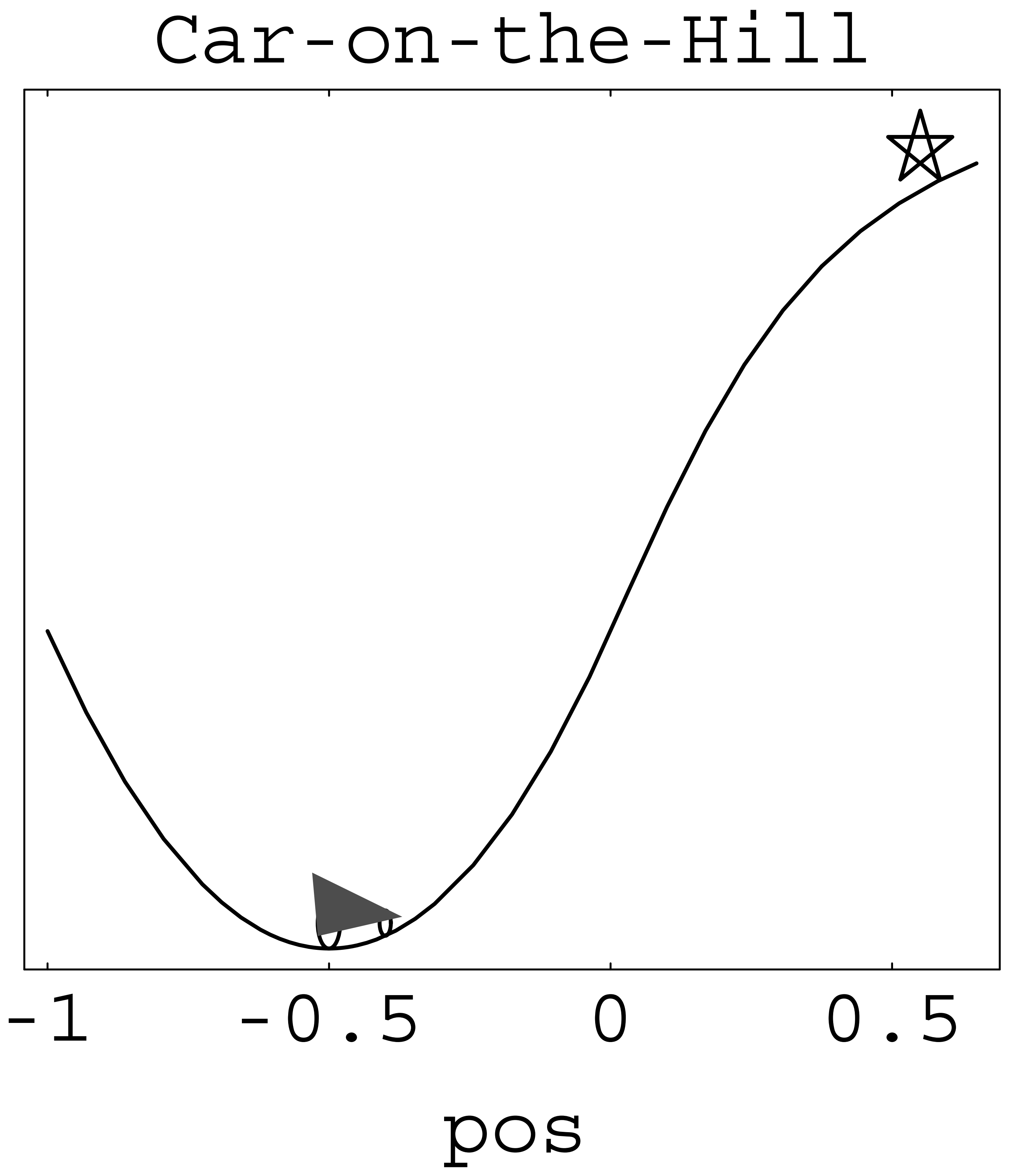
The problem of distribution shift

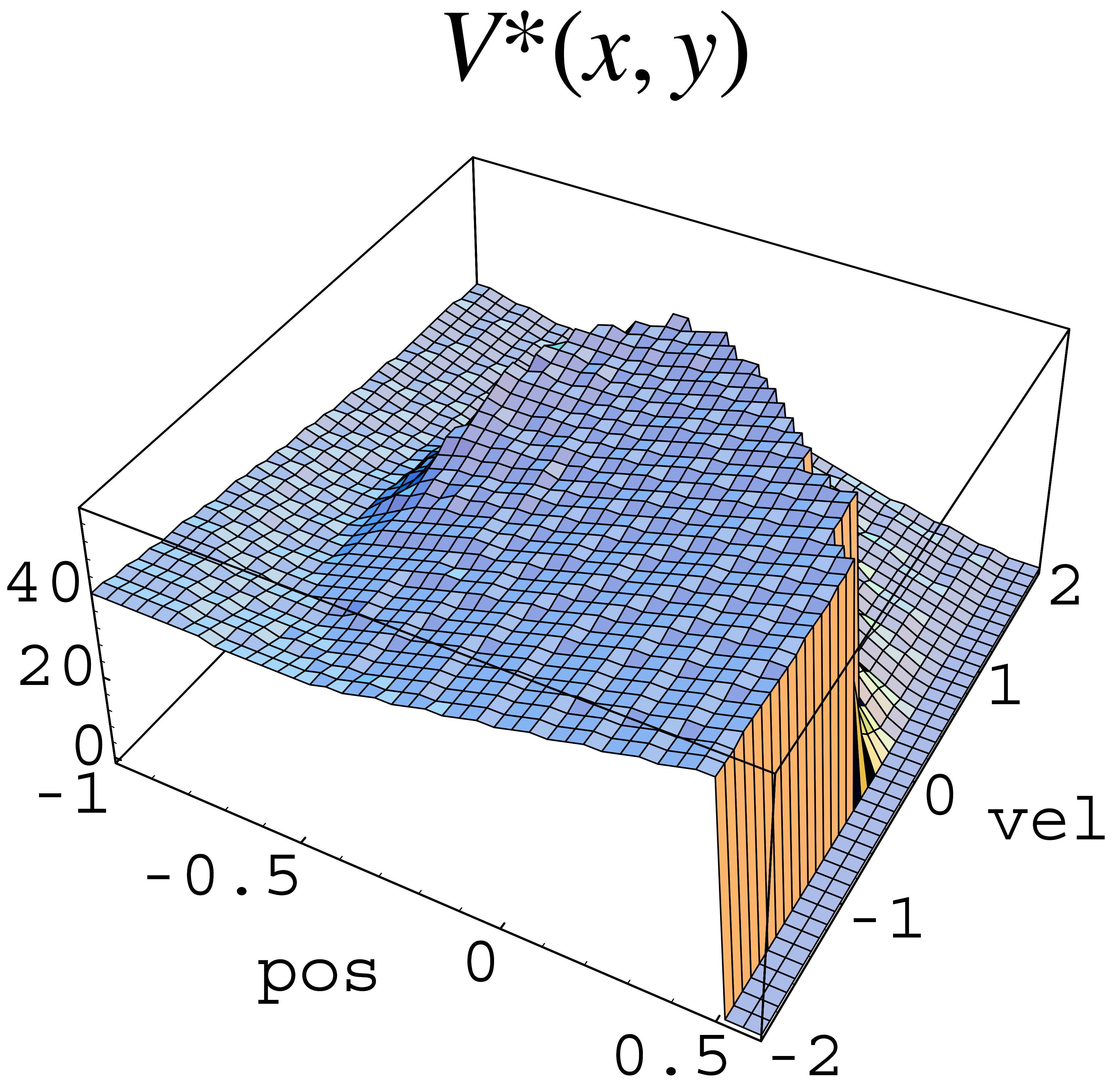


The problem of distribution shift

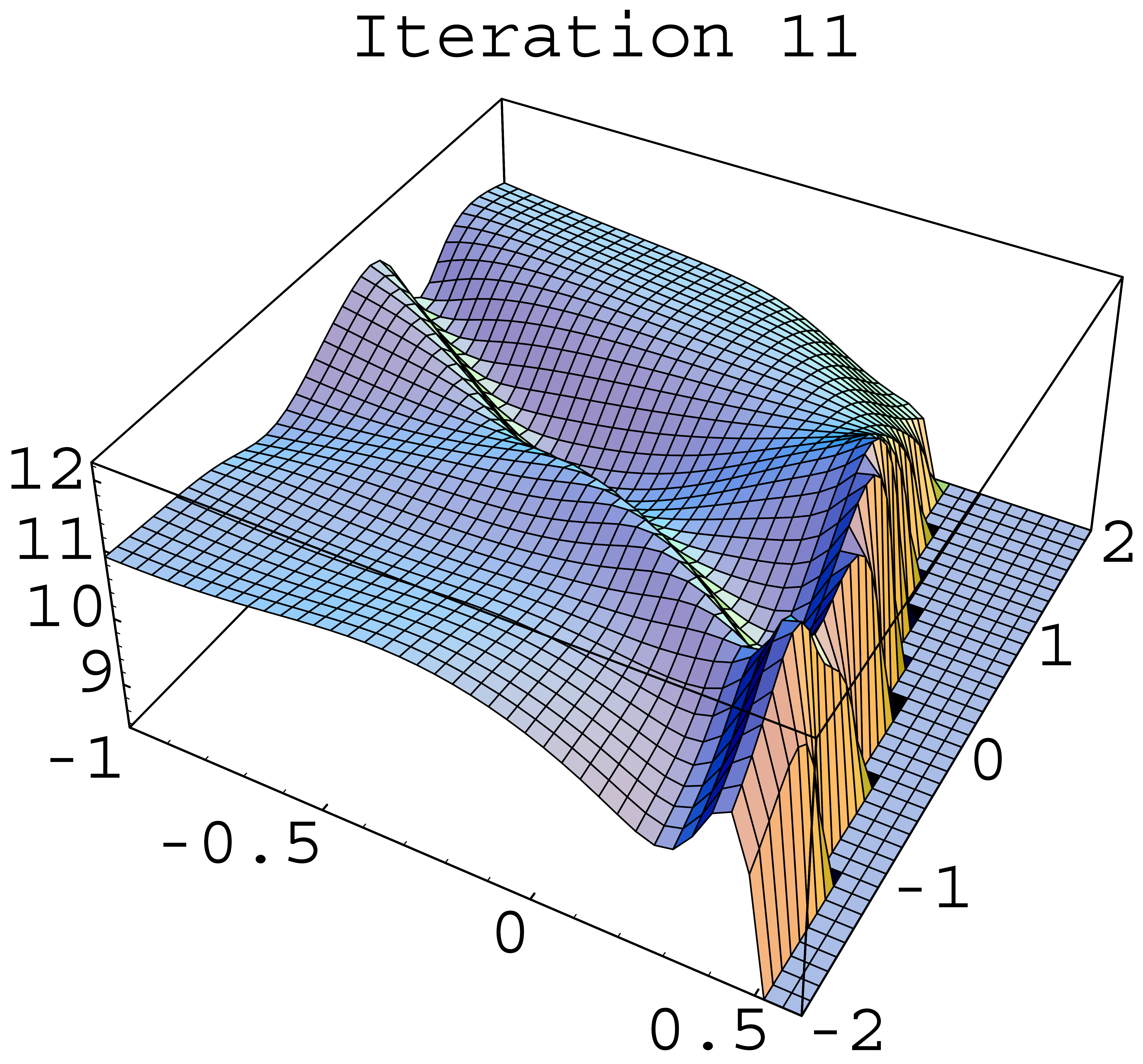


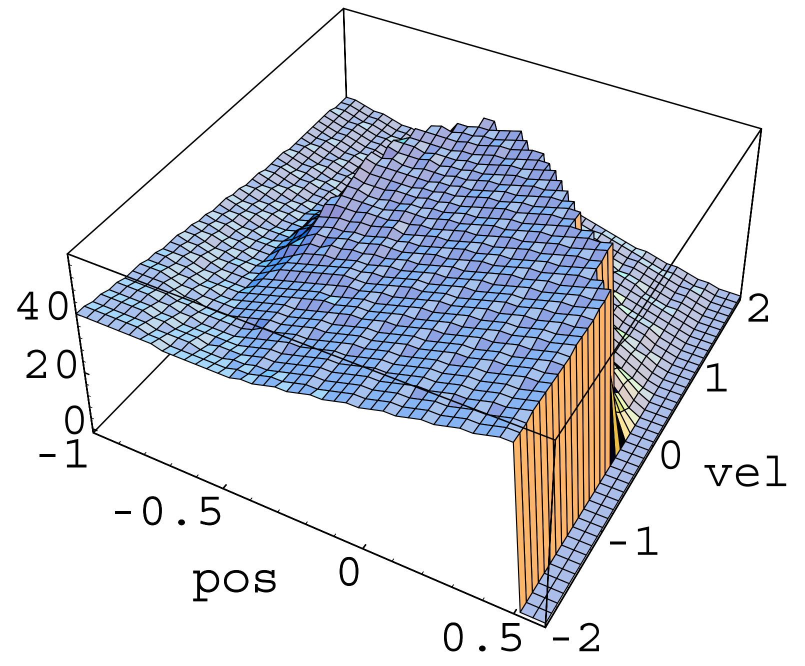
Compounding Errors in Mountain Car



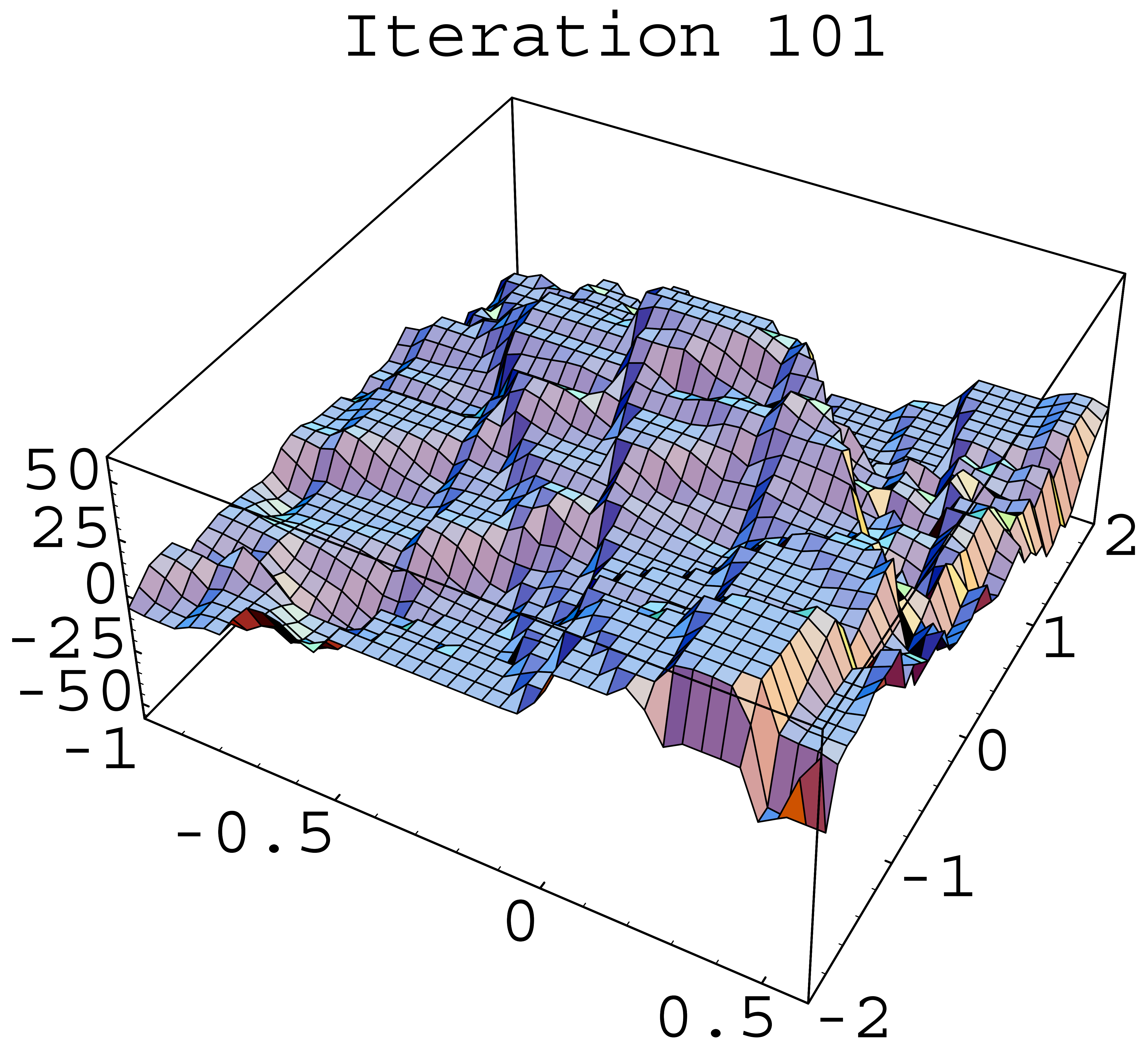


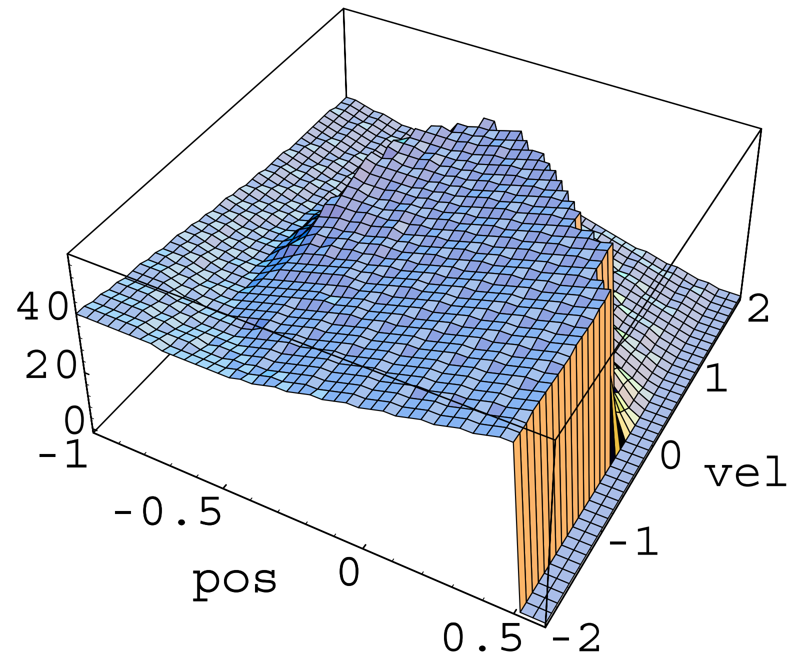
What happens when we run value iteration with a *2 Layer MLP?*



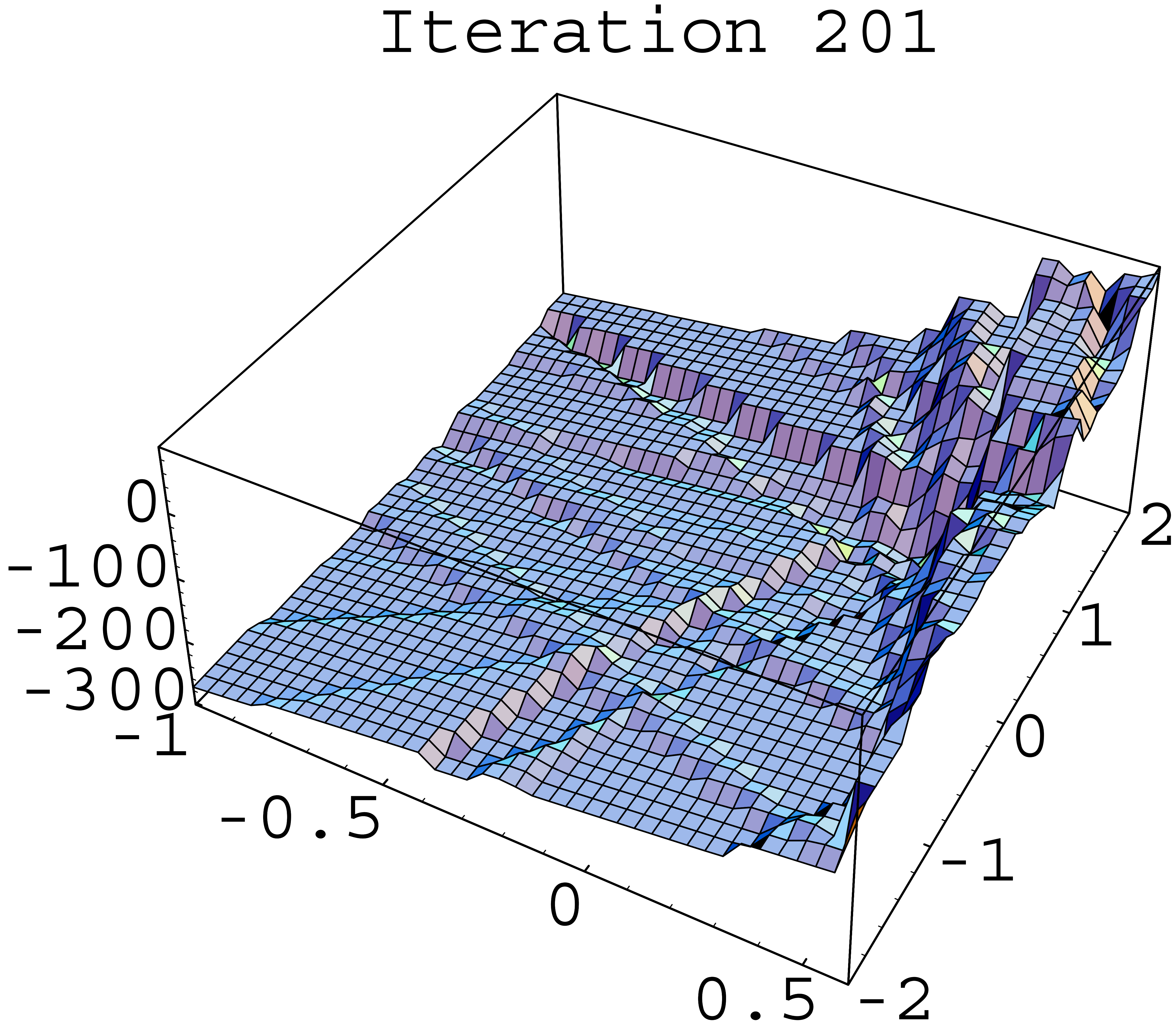


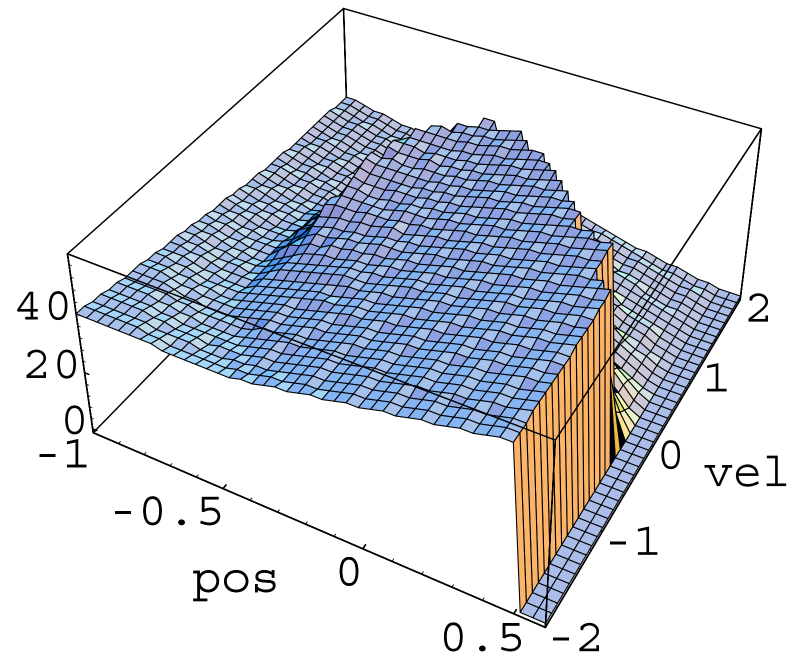
What happens when we run value iteration with a *2 Layer MLP?*



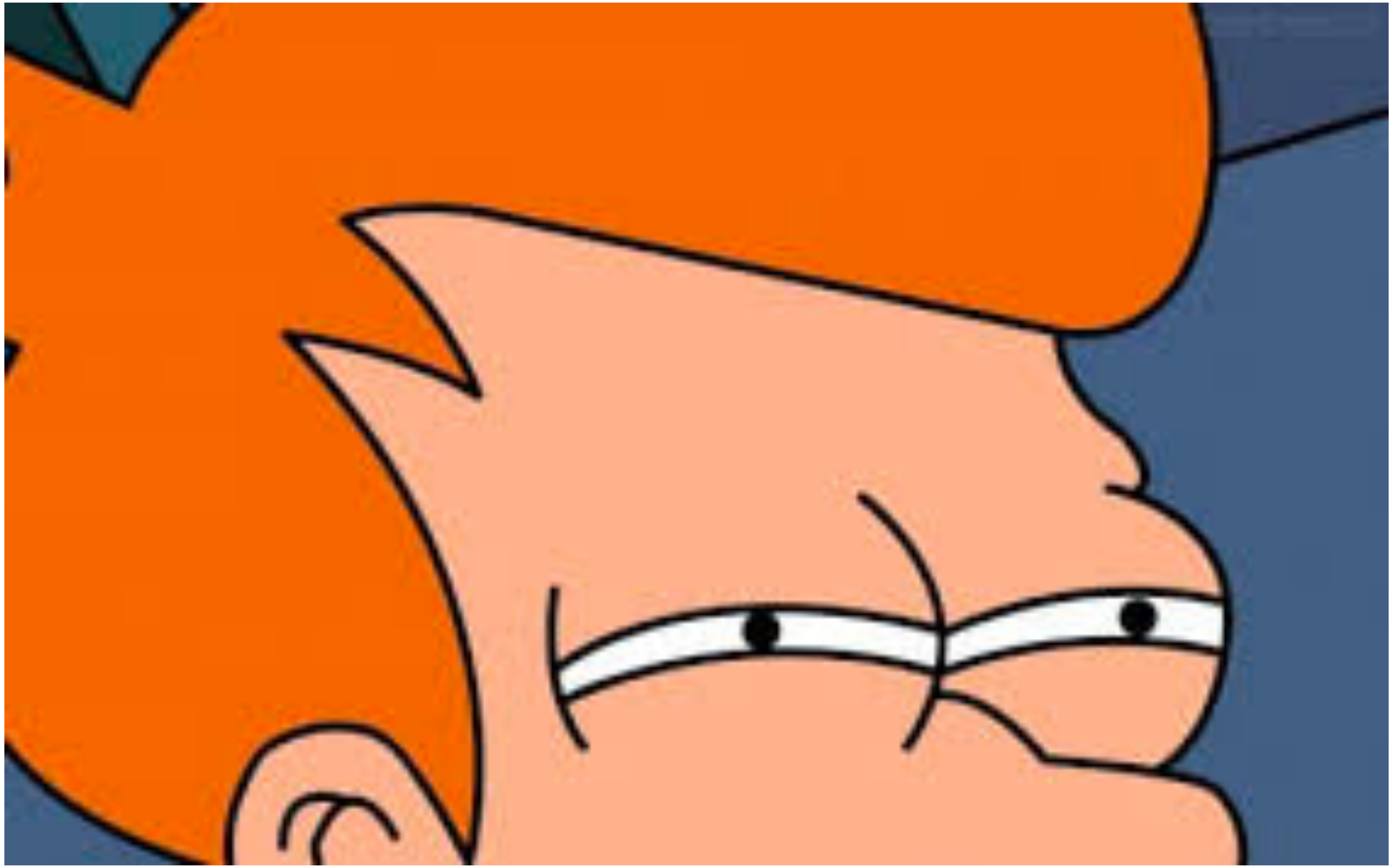


What happens when we run value iteration with a *2 Layer MLP?*





To hell with Value Estimates!



Trust ONLY actual Returns

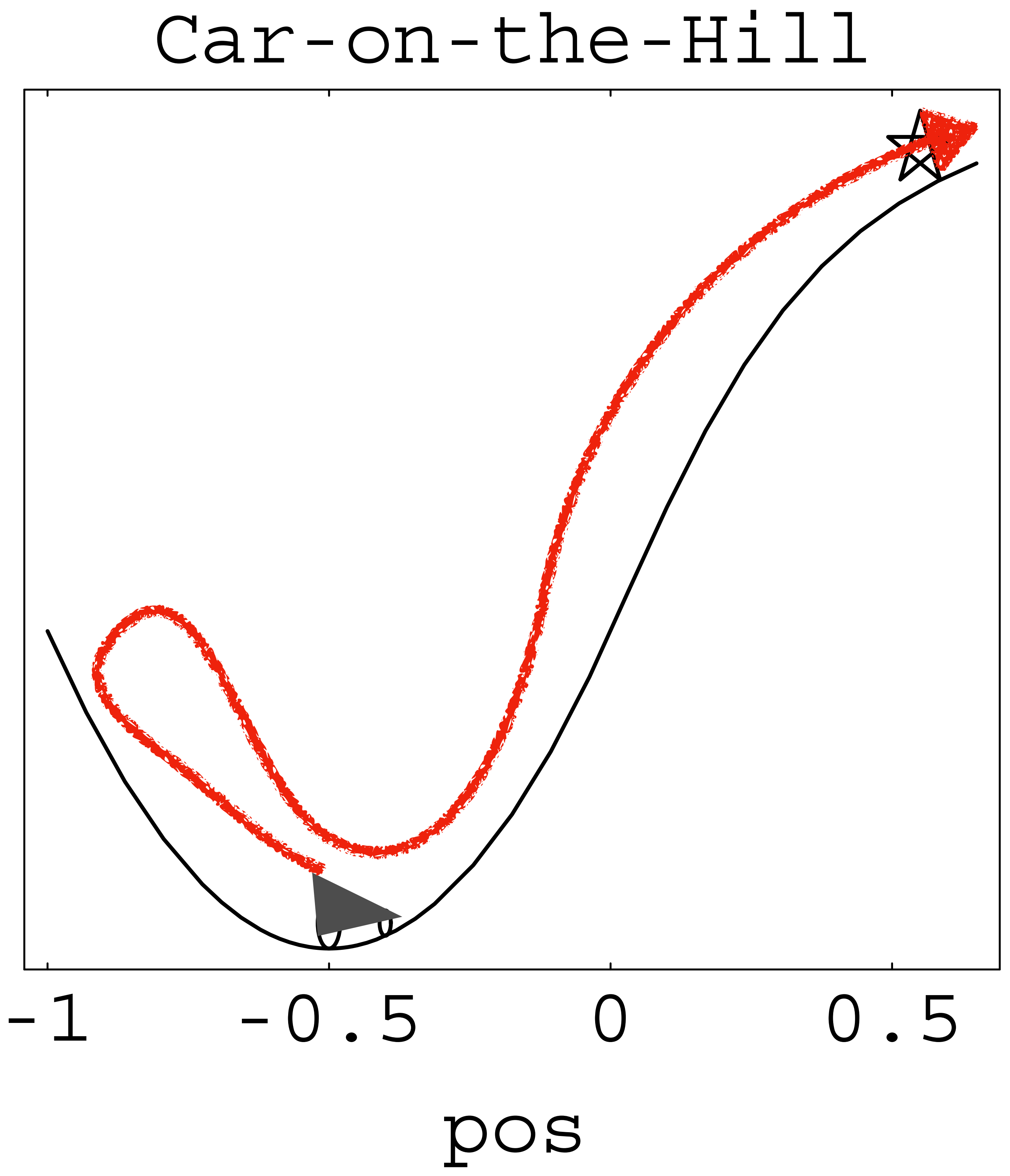
Bye Bye Bellman ...

"not to be blinded by the beauty of the Bellman equation"

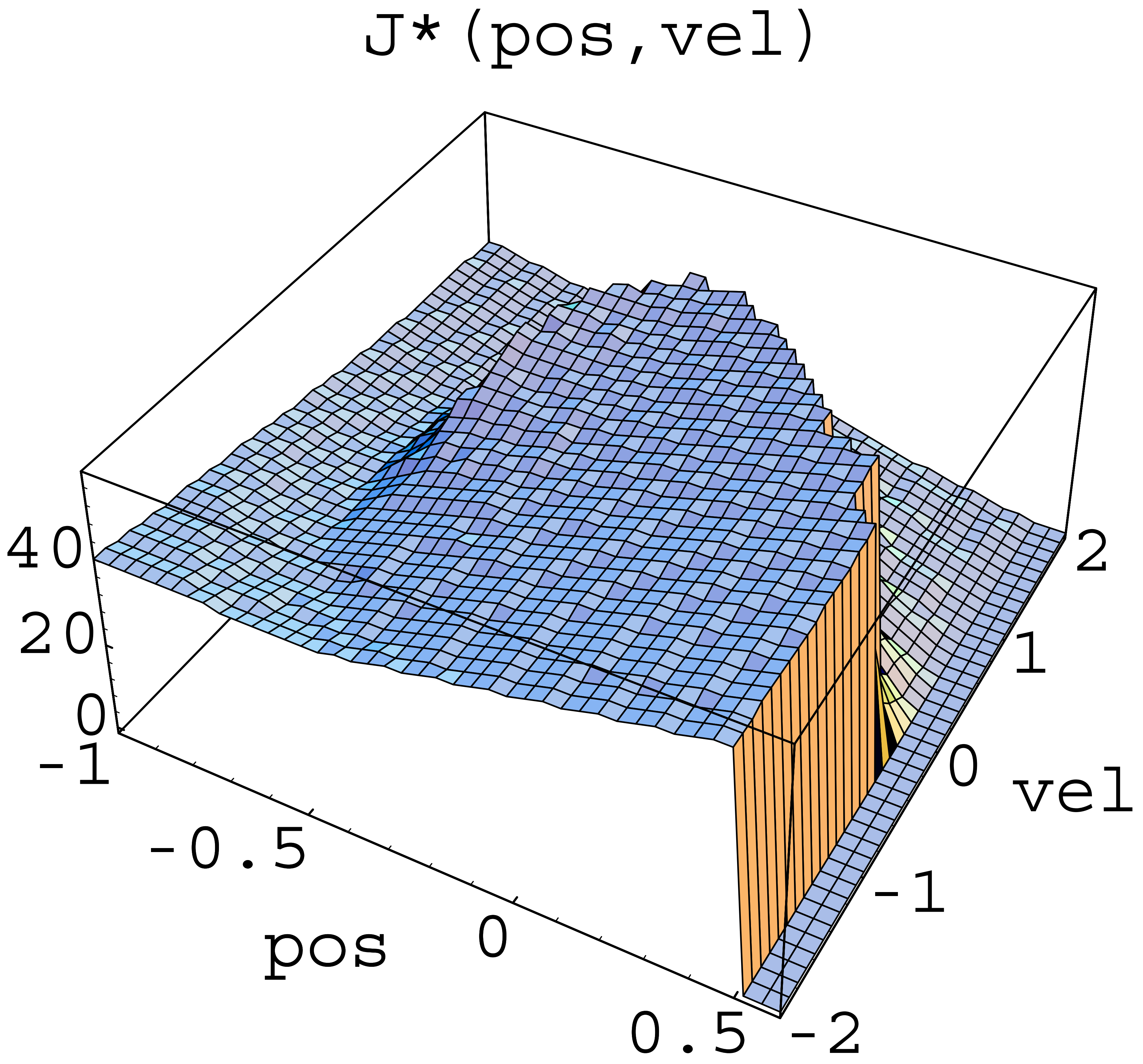
- Andrew Moore

What if we focused on finding good policies ... ?

Sometimes a policy is waaaaay simpler than the value



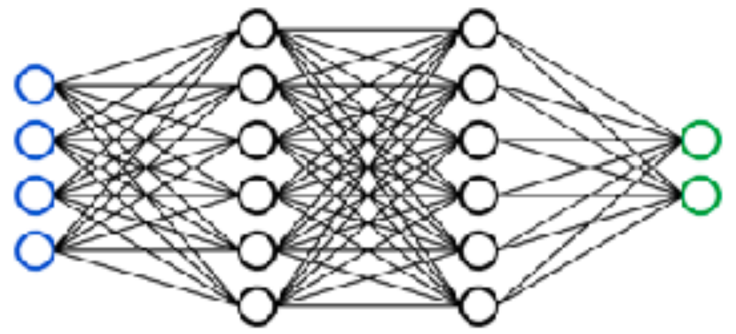
The Policy!



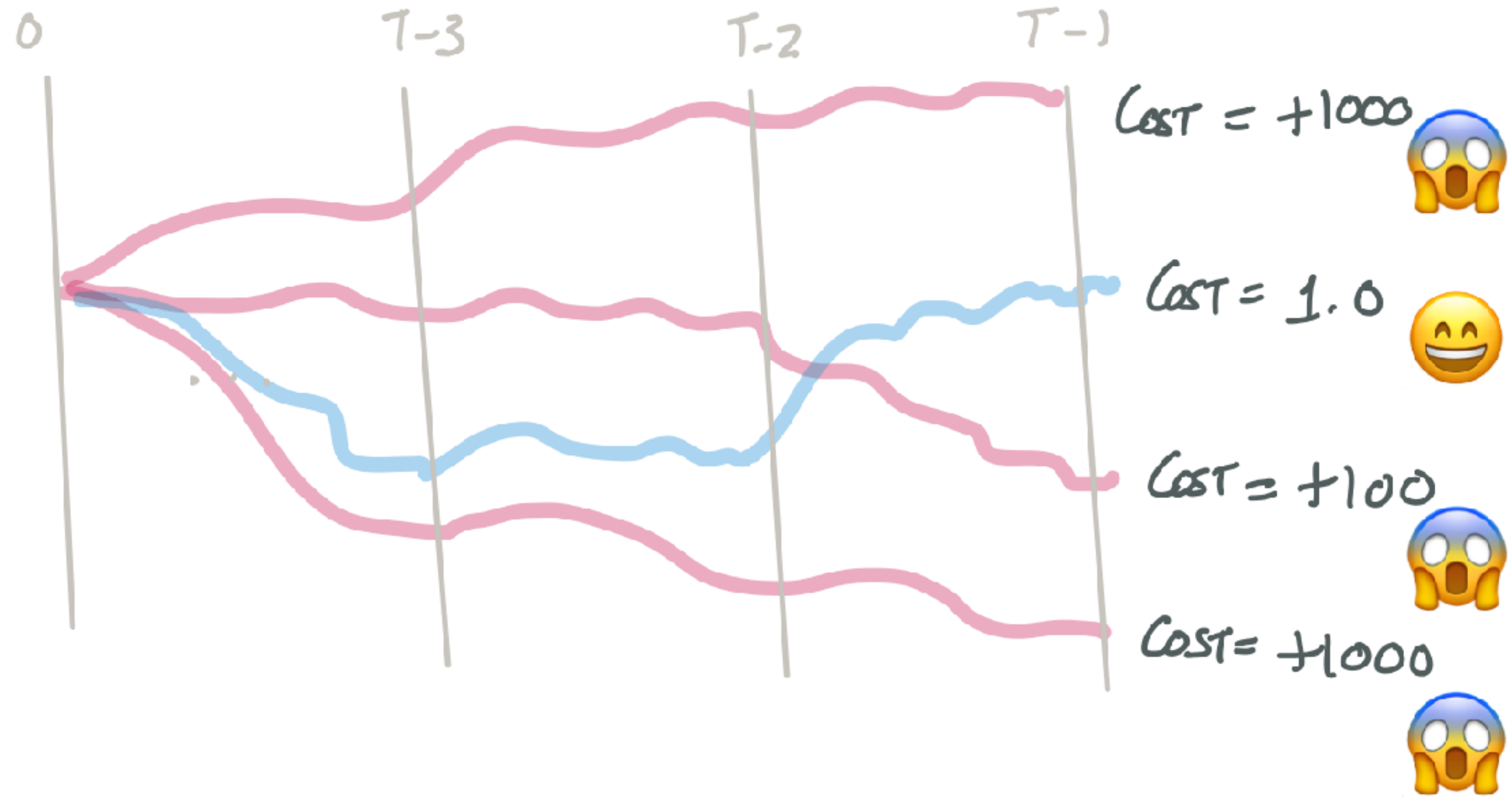
The Value!

Can we just focus on finding a good policy?

$$\pi_\theta : s_t \rightarrow a_t$$

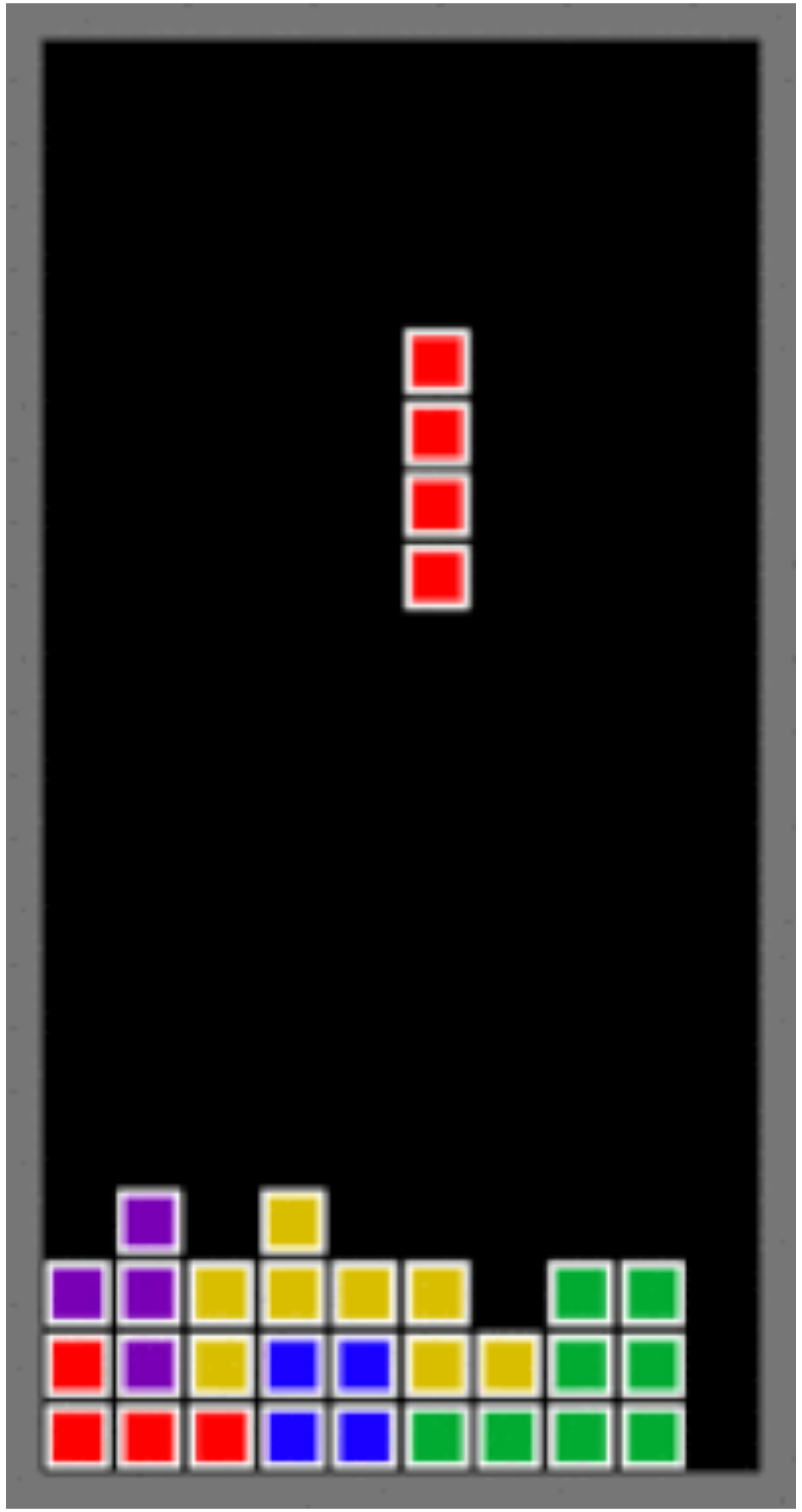


Learn a mapping from states to actions



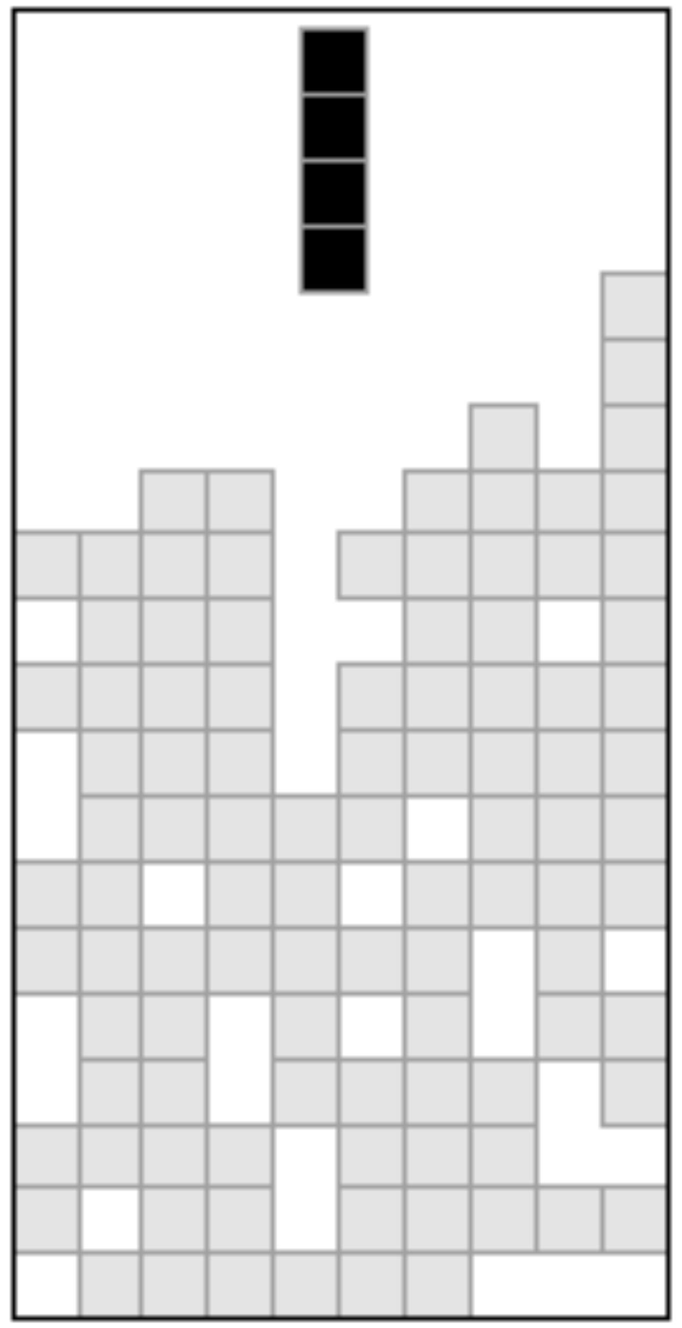
Roll-out policies in the real-world to estimate value

The Game of Tetris

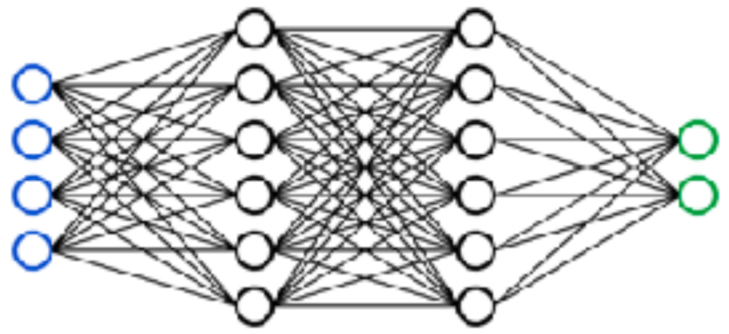


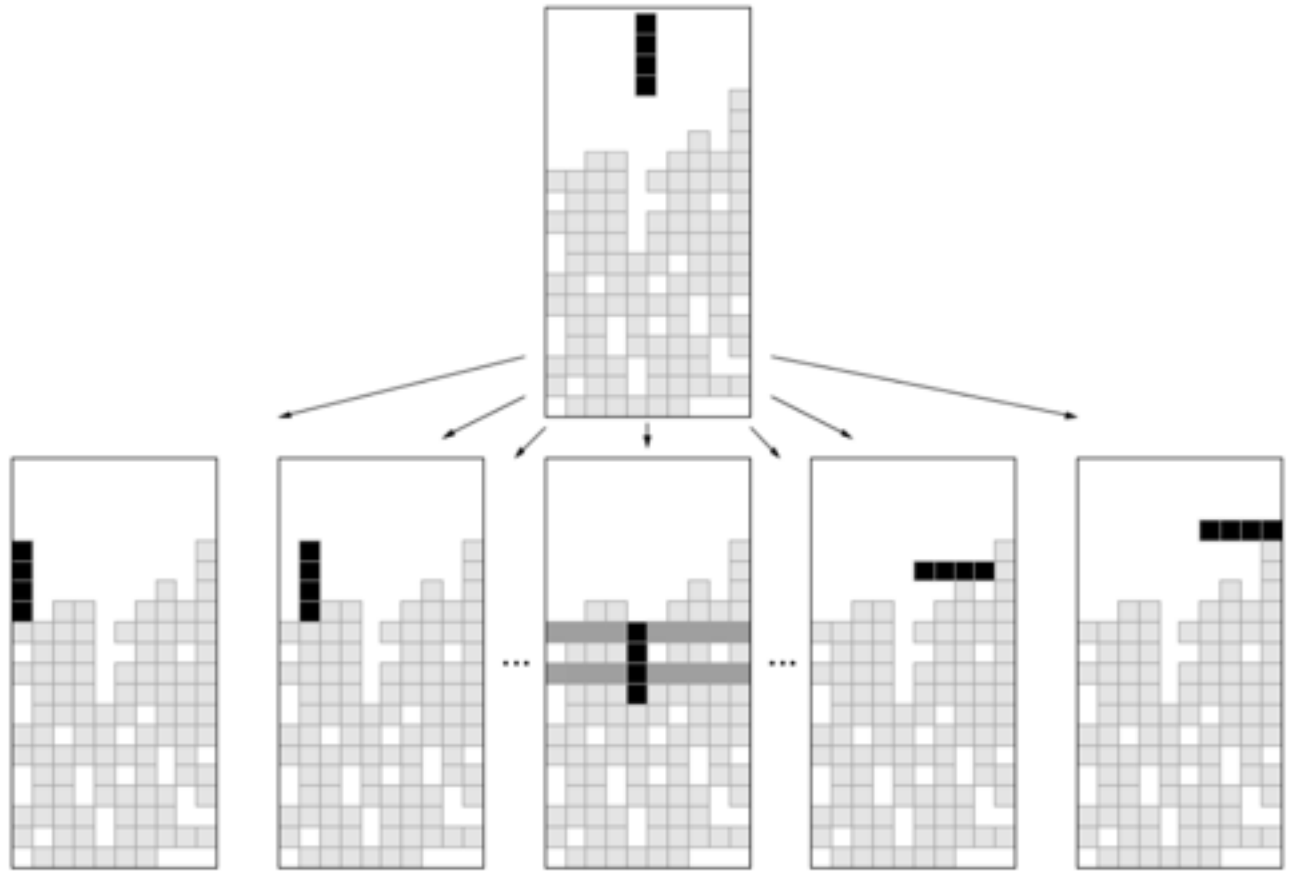
What's a good policy representation for Tetris?

(4 rotations)*(10 slots) - (6 impossible poses) = 34



$$\pi_\theta : s_t \rightarrow a_t$$





State (s_t)

Action (a_t)

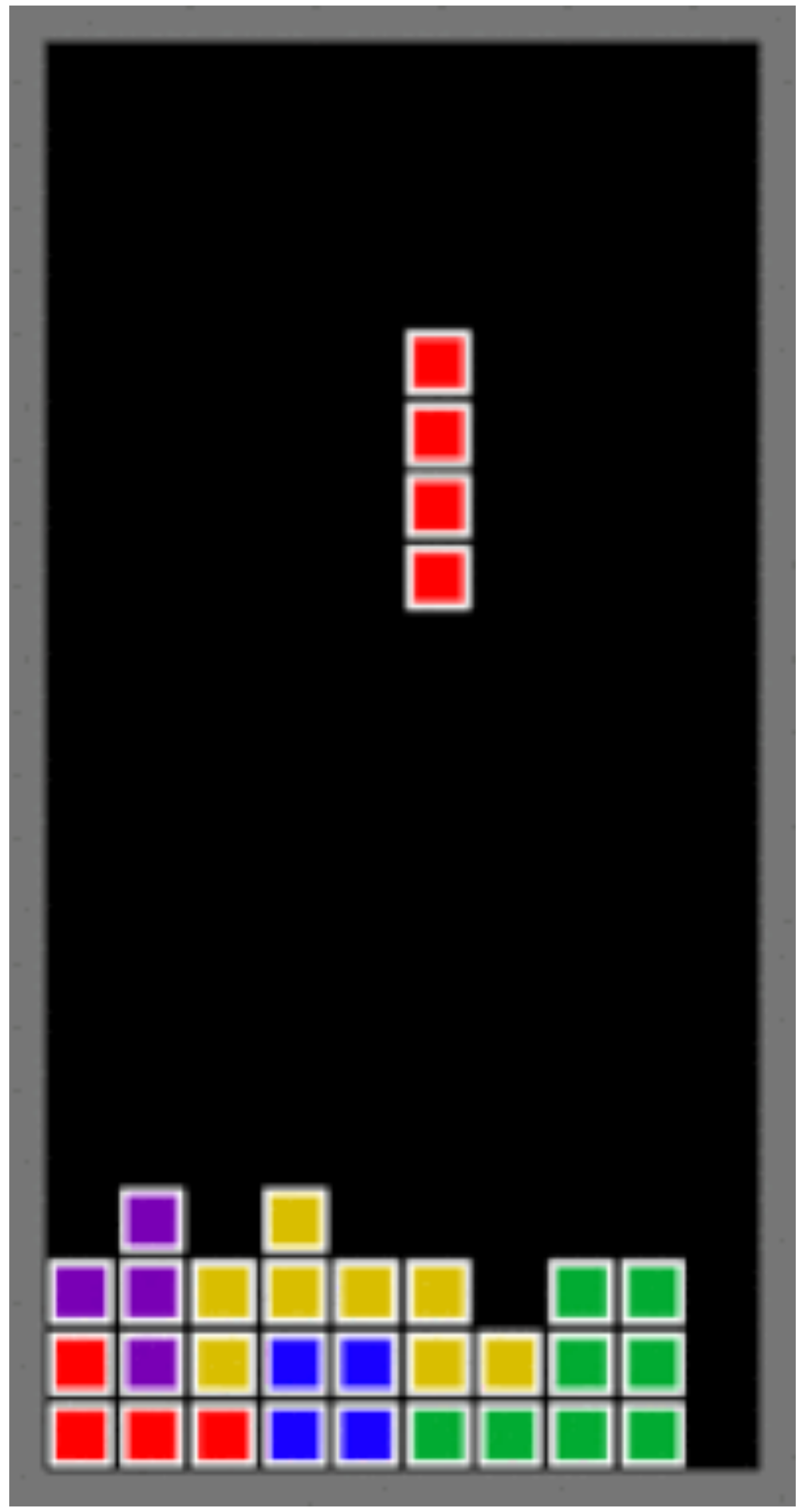
Activity!

Think-Pair-Share

Think (30 sec): Ideas for how to represent policy for tetris?

Pair: Find a partner

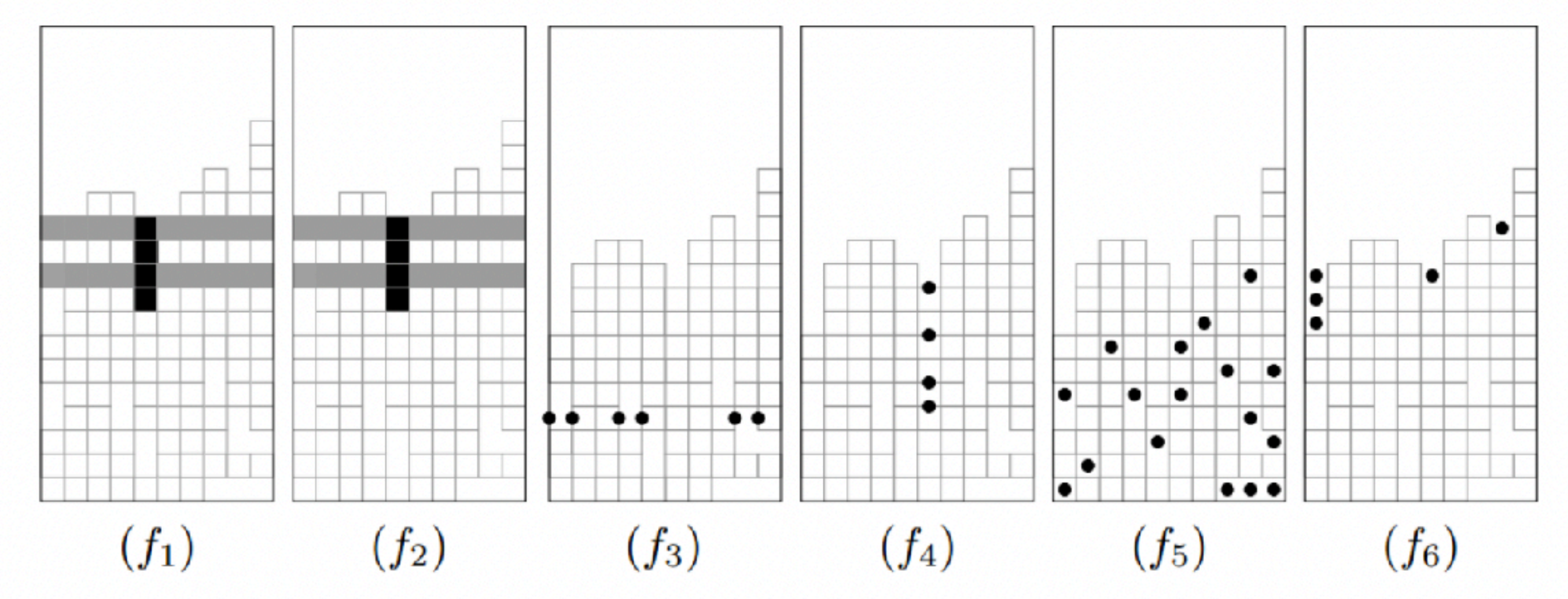
Share (45 sec): Partners exchange ideas



Some inspiration for Tetris policy

Until 2008, the best artificial Tetris player was handcrafted, as reported by Fahey (2003). Pierre Dellacherie, a self declared average Tetris player, identified six simple features and tuned the weights by trial and error.

Dellacherie Features



Landing Heights

Eroded Cells

Row Transitions

Column Transitions

Holes

Cumulative Wells

The contribution of the last piece to the cleared lines time the number of cleared lines.

The number of filled cells adjacent to the empty cells summed over all rows

A well is a succession of empty cells and the cells to the left and right are occupied

A *magic* formula ?!?

$$
\begin{aligned}
&-4 \times holes - cumulative\,wells \\
&- row\,transitions - column\,transitions \\
&- landing\,height + eroded\,cells
\end{aligned}
$$

A *magic* formula ?!?

$$-4 \times holes - cumulative\ wells$$
$$-\ row\ transitions - column\ transitions$$
$$-\ landing\ height + eroded\ cells$$

This linear evaluation function cleared an average of 660,000 lines on the full grid ...
... In the simplified implementation used by the approaches discussed earlier, the games would have continued further, until every placement would overflow the grid. Therefore, this report underrates this simple linear rule compared to other algorithms.

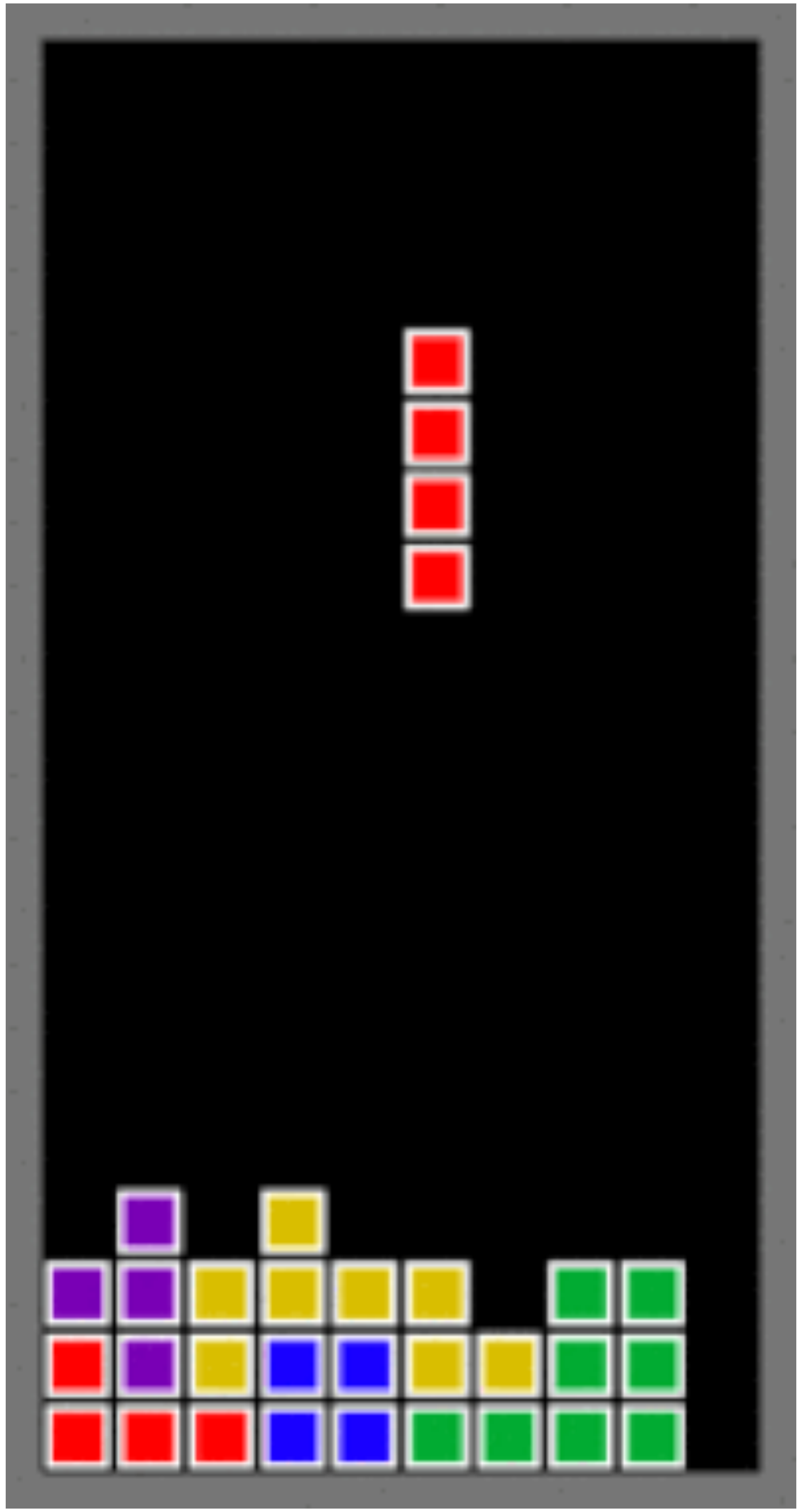
Tetris Policy

$$\pi_\theta(a|s) = \frac{\exp\left(\theta^\top f(s,a)\right)}{\sum_{a'} \exp\left(\theta^\top f(s,a')\right)}$$

$f_1(s,a) = \#$ number of holes

$f_2(s,a) = \#$ max height

....

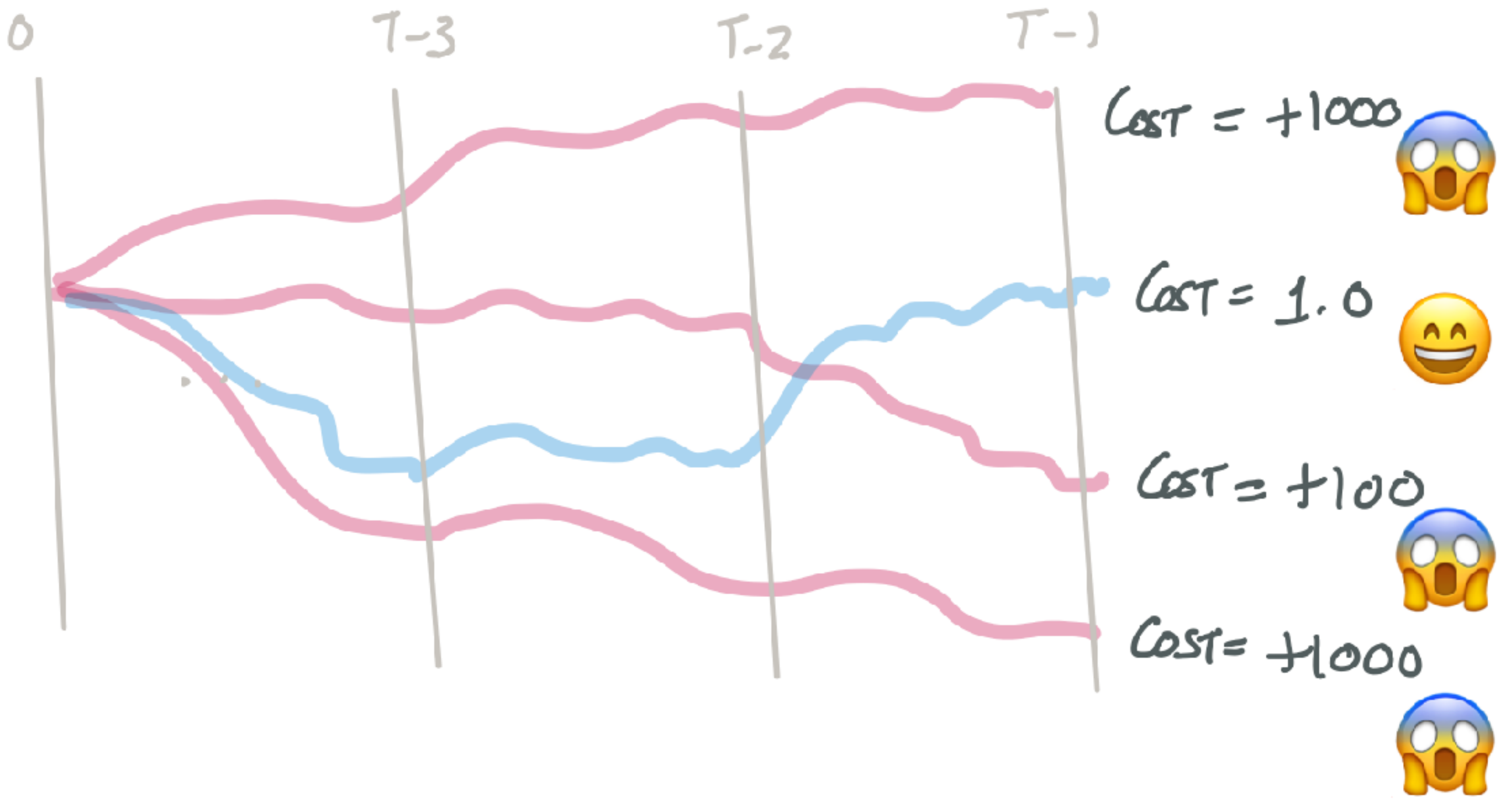


The Goal of Policy Optimization

$$\pi_\theta(a|s) = \frac{\exp\left(\theta^\top f(s,a)\right)}{\sum_{a'} \exp\left(\theta^\top f(s,a')\right)}$$

#Think of f(s,a) being dellacherie features

$$\min_\theta J(\theta) = \sum_{t=0}^{T-1} \mathbb{E}_{\pi_\theta} c(s_t, a_t)$$



#Think of c(s,a) as
-num_rows_cleared

Can we do gradient descent if we don't know the dynamics??

The Likelihood Ratio Trick!

REINFORCE

Algorithm 20: The REINFORCE algorithm.

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

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

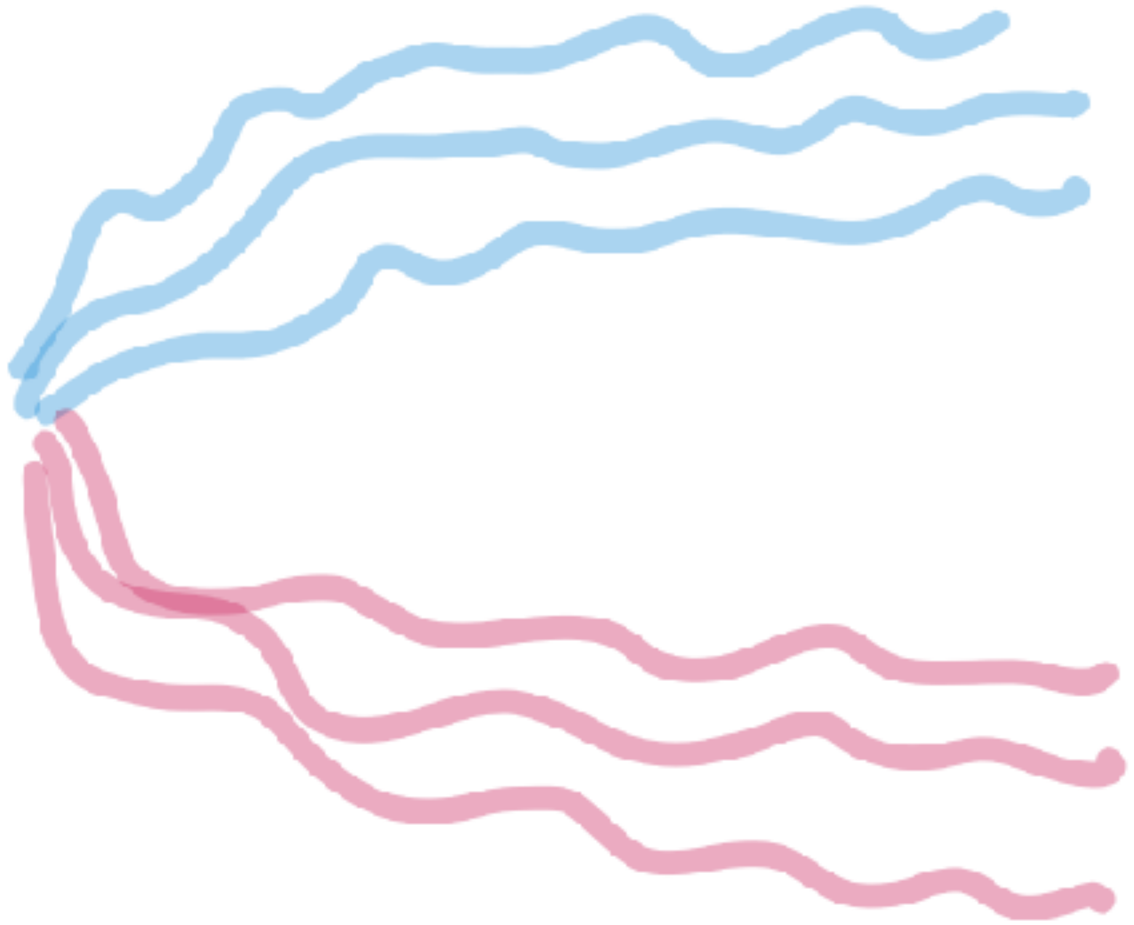
  Update parameters $\theta \leftarrow \theta + \alpha\, \widetilde{\nabla}_\theta J$
return π_θ

Chugging through the gradient ..

$$
\begin{aligned}
\nabla_\theta \log \pi_\theta(a|s) &= \nabla_\theta \left[\theta^\top f(s,a) - \log \sum_{a'} \exp\left(\theta^\top f(s,a')\right) \right] \\
&= f(s,a) - \frac{\sum_{a'} f(s,a') \exp\left(\theta^\top f(s,a')\right)}{\sum_{a'} \exp\left(\theta^\top f(s,a')\right)} \\
&= f(s,a) - \sum_{a'} f(s,a')\, \pi_\theta\left(a'|s\right) \\
&= f(s,a) - E_{\pi_\theta(a'|s)}\left[f(s,a')\right]
\end{aligned}
$$

Understanding the REINFORCE update

LET $f_1(s,a) = \#\text{ holes}.$



R = +1
R = +1
R = +1

$$\Rightarrow \sum_{(s,a)\in\tau} \underbrace{f_1(s,a)}_{\text{Feature}} - \underbrace{\mathop{E}_{a'\sim\pi_\theta} f_1(s,a')}_{\text{Average feature}} = -5$$

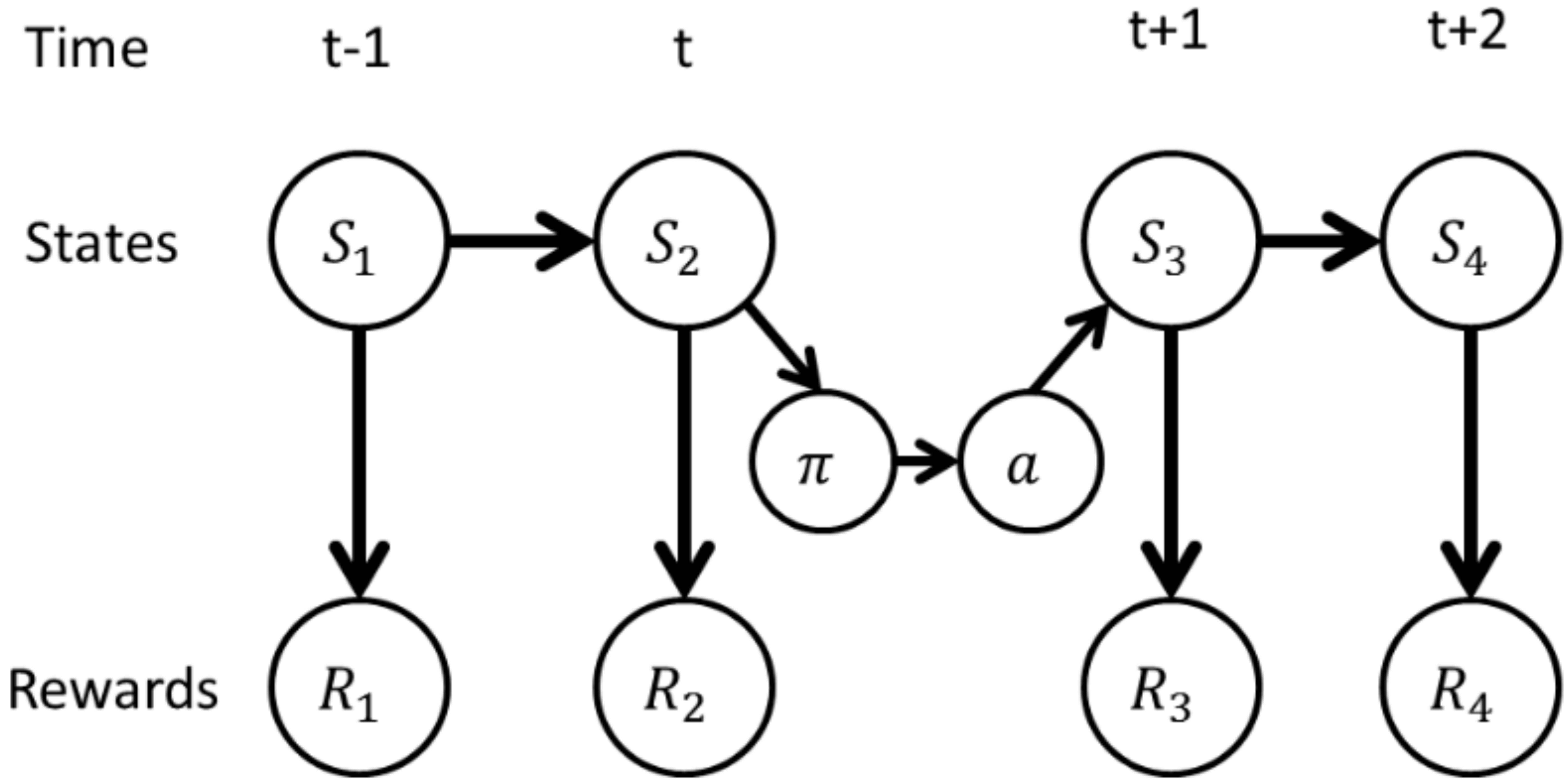
R = −1
R = −1
R = −1

$$\Rightarrow \sum_{(s,a)\in\tau} \underbrace{f_1(s,a)}_{\text{Feature}} - \underbrace{\mathop{E}_{a'\sim\pi_\theta} f_1(s,a')}_{\text{Average feature}} = +3$$

$$\theta_1 = \theta_1 + \sum_{(s,a)\in\tau}\left(\nabla_\theta \log \pi_\theta(a|s)\right)R(\tau) = \theta_1 + \alpha\left(-5\times(+1) + 3\,(-1)\right)$$

$$= \theta_1 - \alpha\, 8 \quad \left(\text{Bump down this feature}\right)$$

Causality: Can actions affect the past?



The Policy Gradient Theorem

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

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

Life is good!

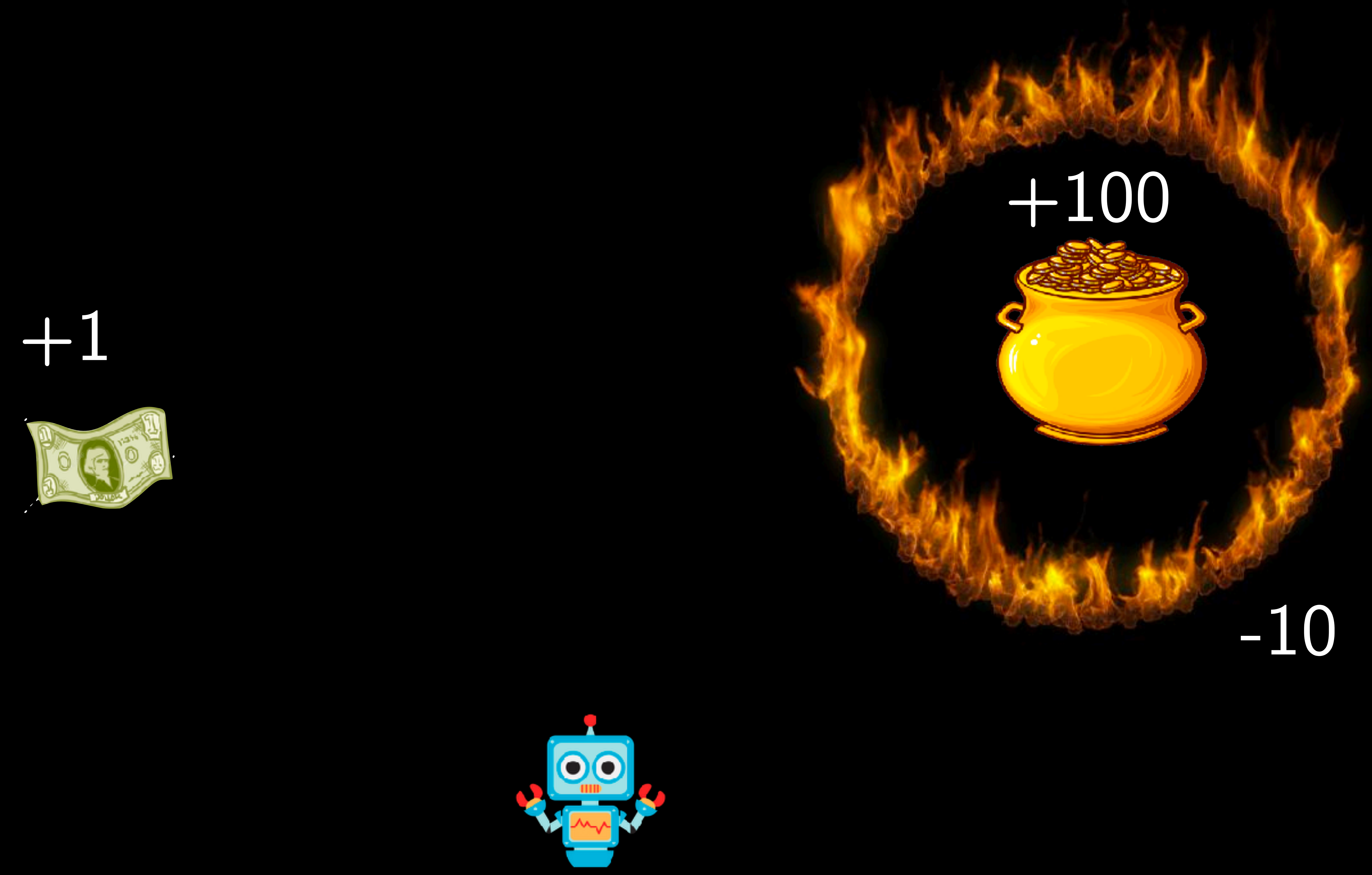
This solves everything ...

The Three Nightmares of Policy Optimization

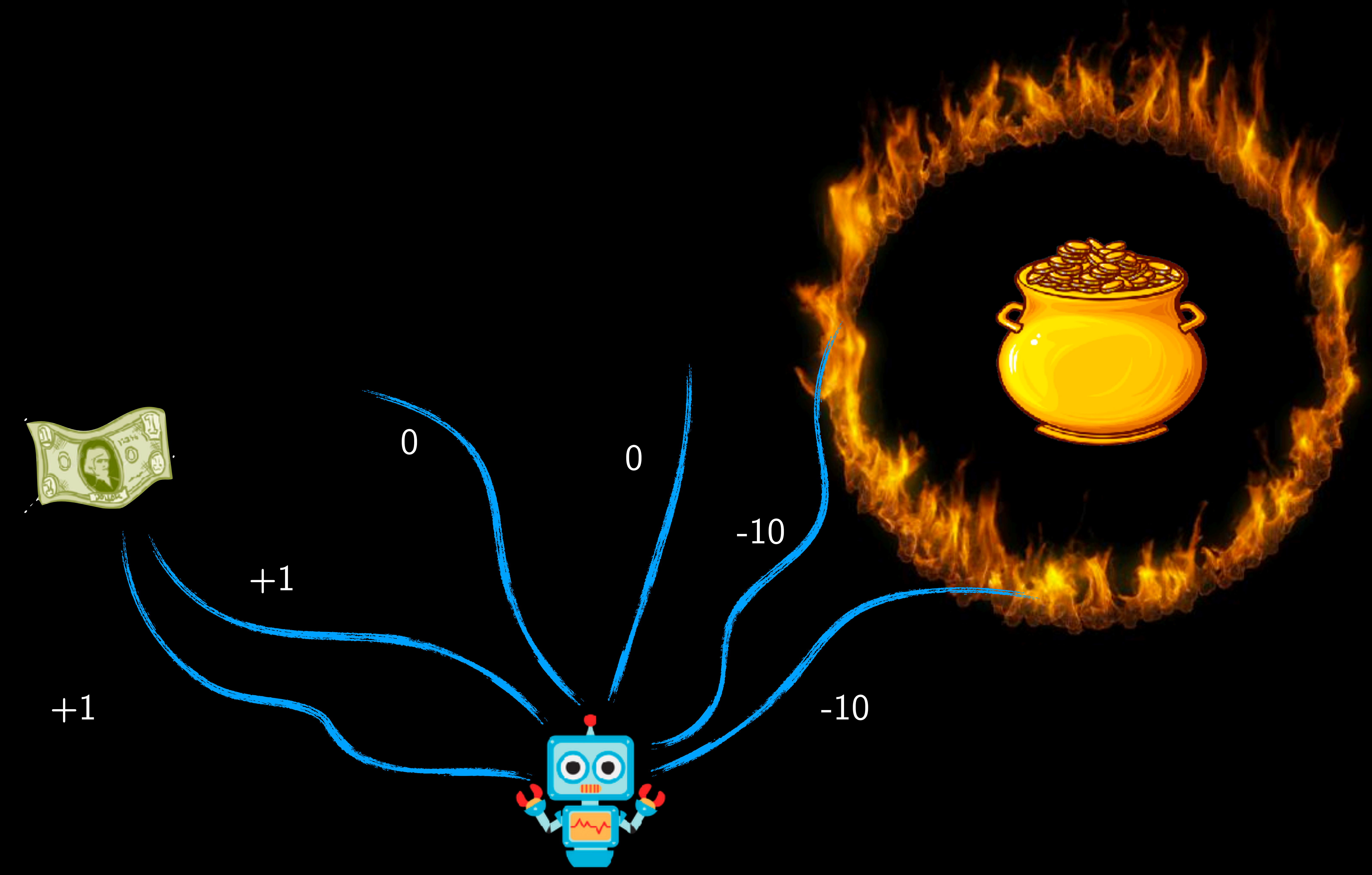
Nightmare 1:

Local Optima

The Ring of Fire

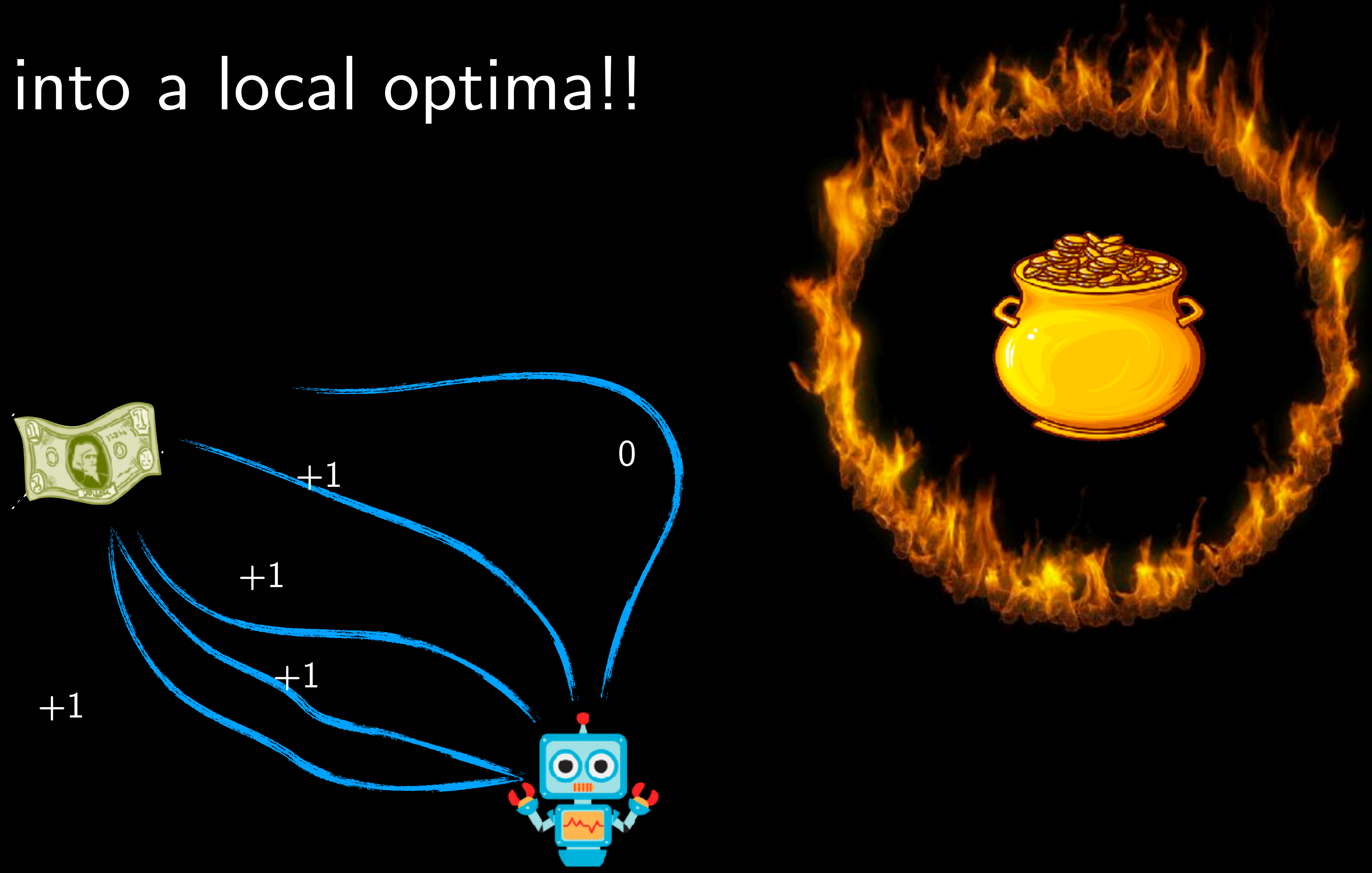


The Ring of Fire



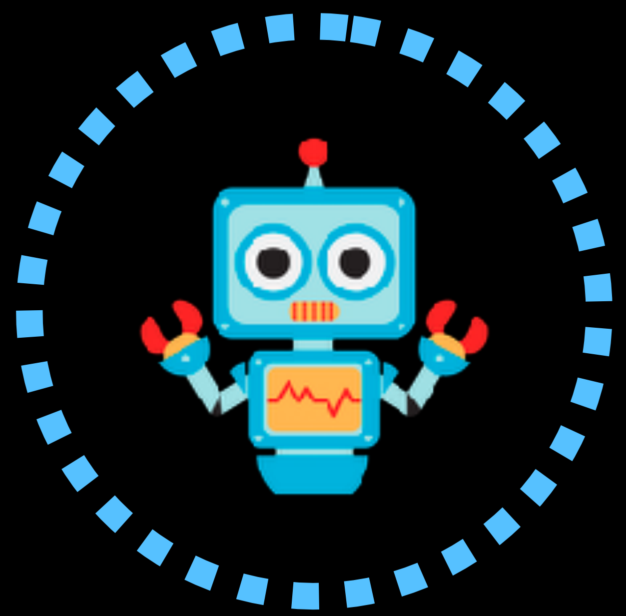
The Ring of Fire

Get's sucked into a local optima!!

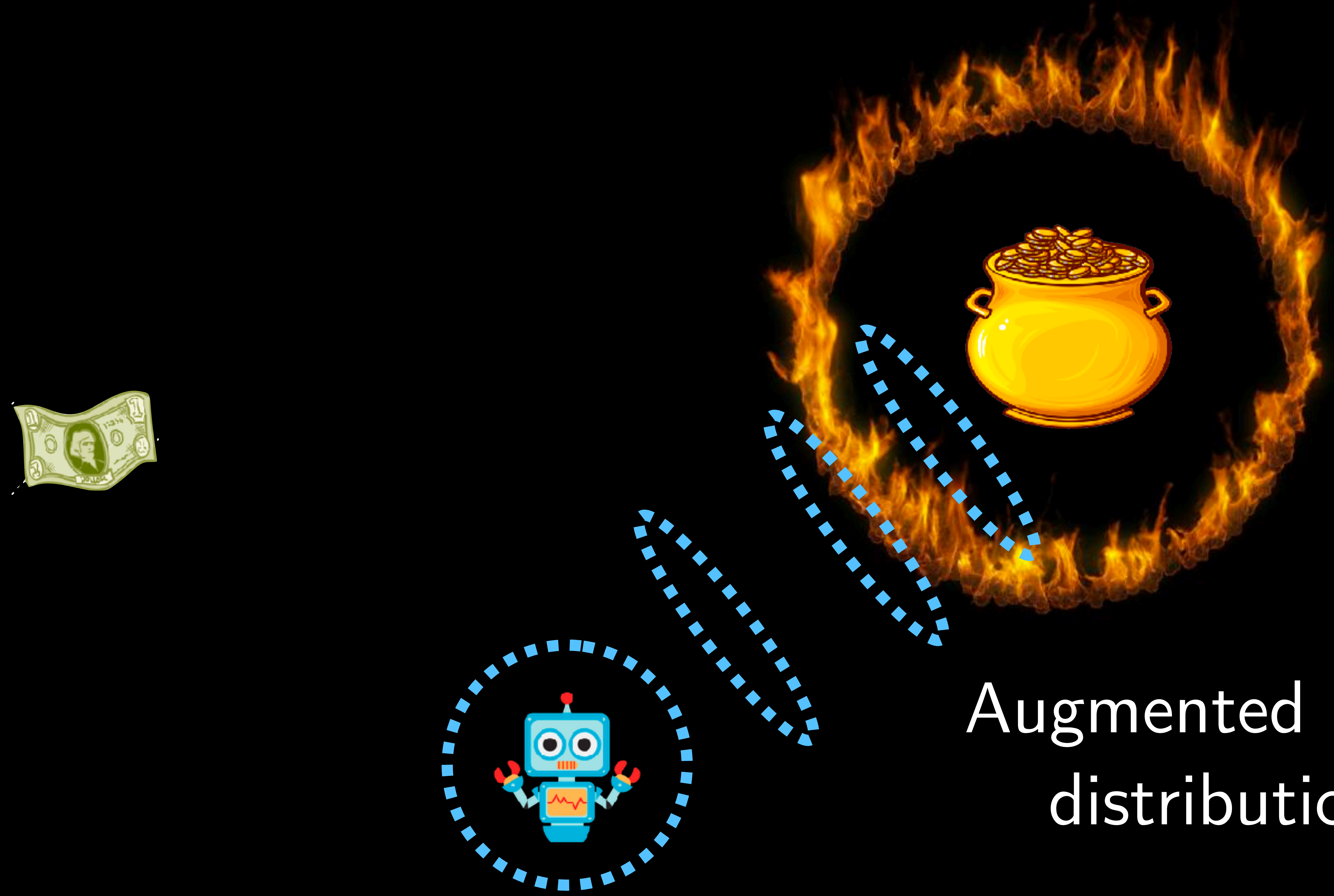


Idea: What if we had a "good reset distribution?"

Nominal reset distribution

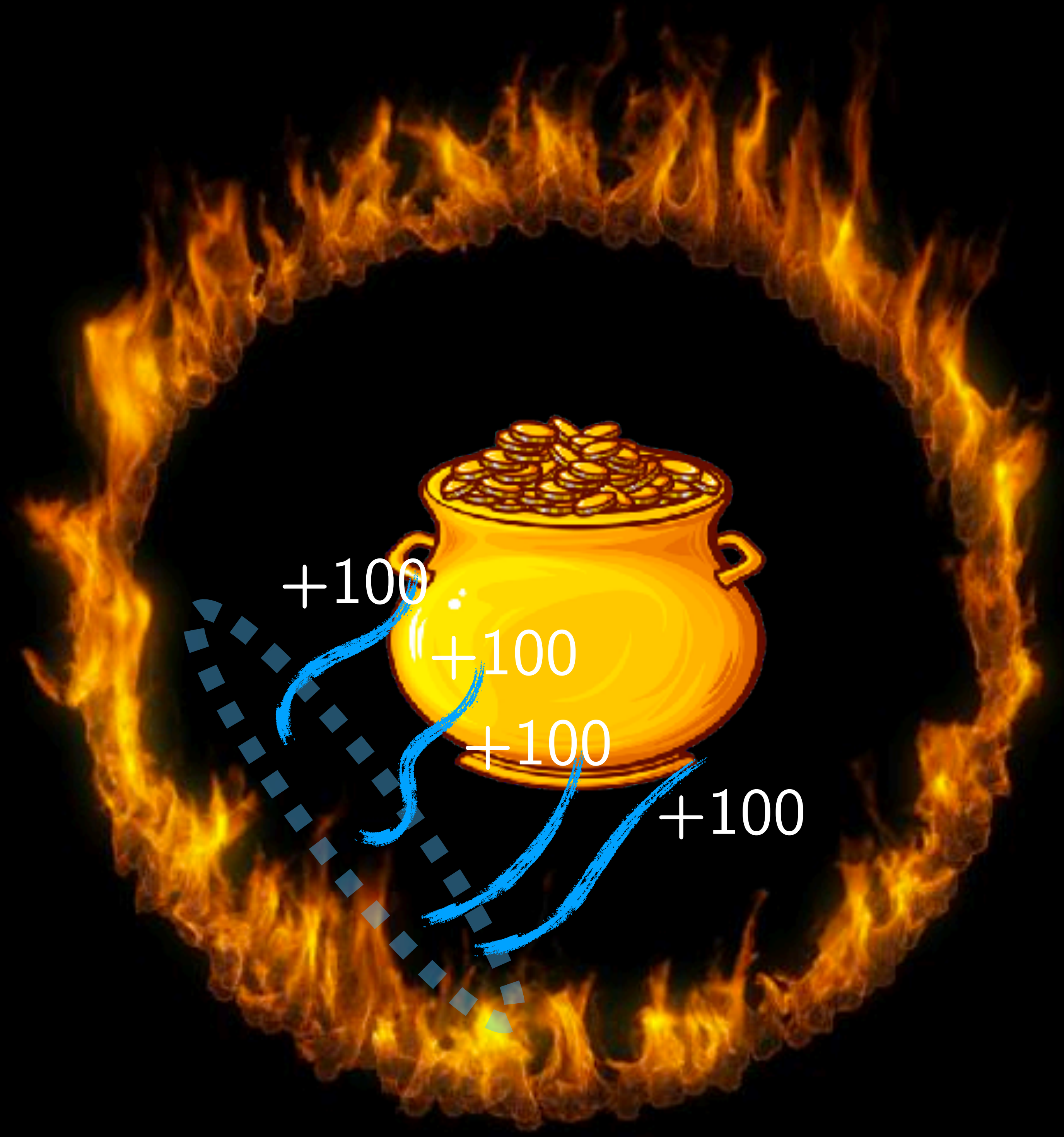


Idea: What if we had a “good reset distribution?”



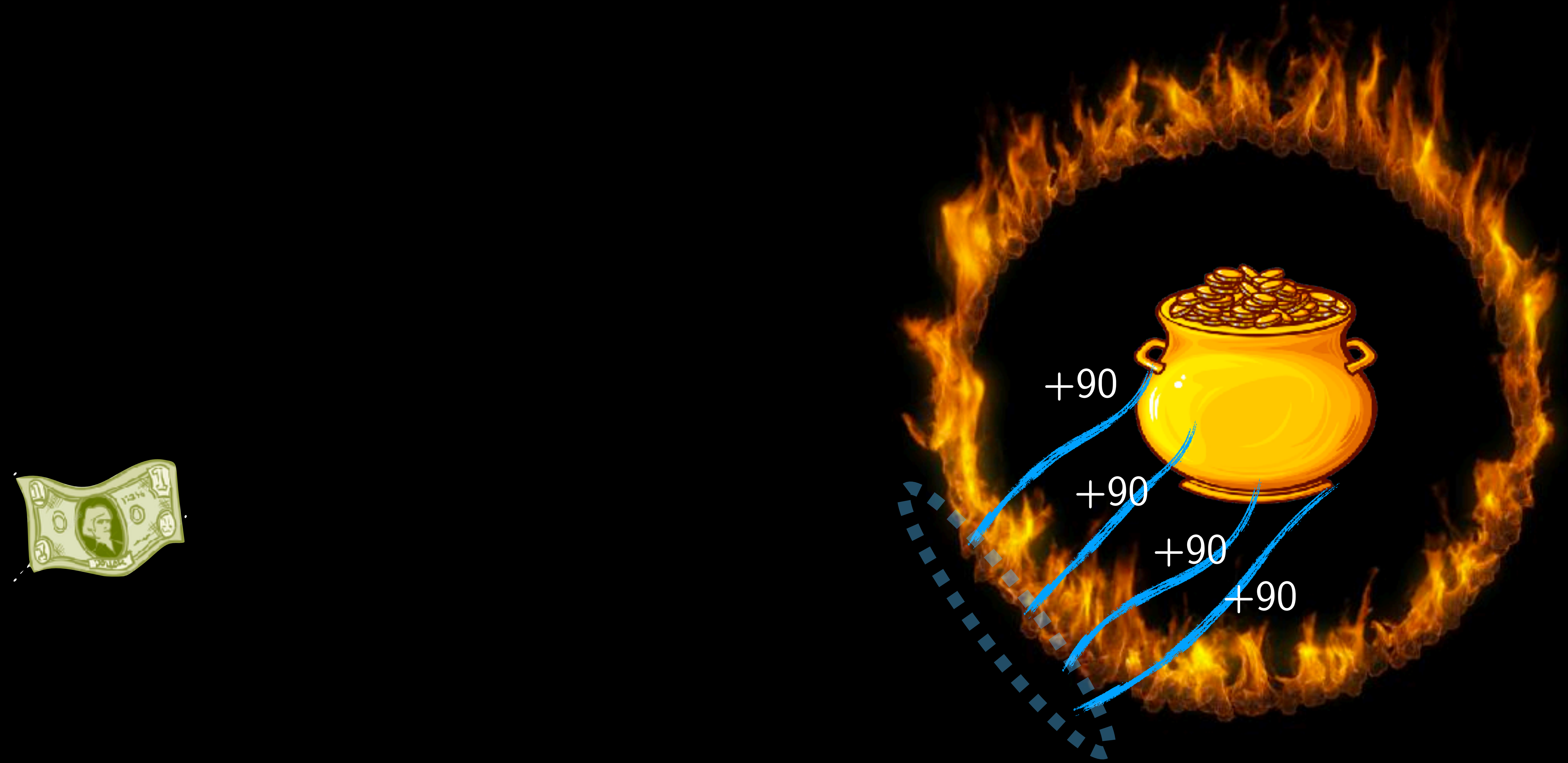
Augmented reset distribution

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



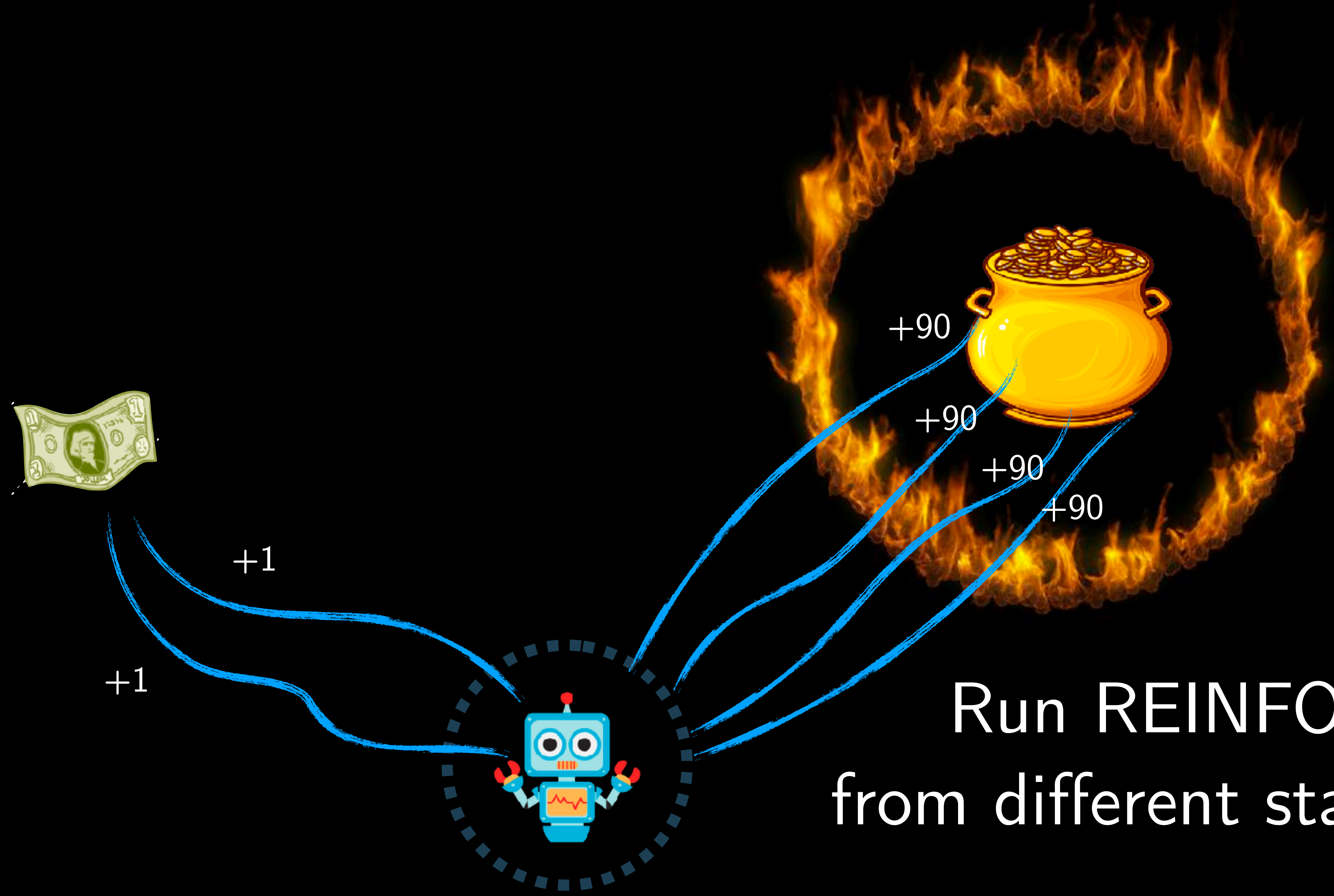
Run REINFORCE
from different start states

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 “restart” distribution

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

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

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

Nightmare 2:

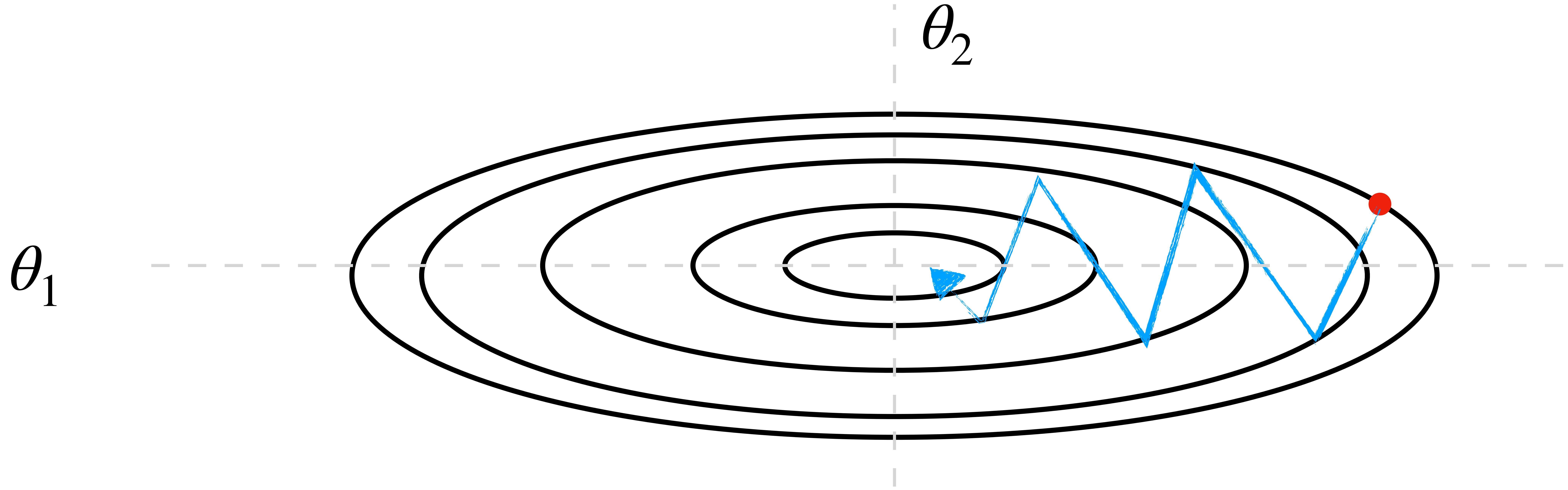
Distribution Shift

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?



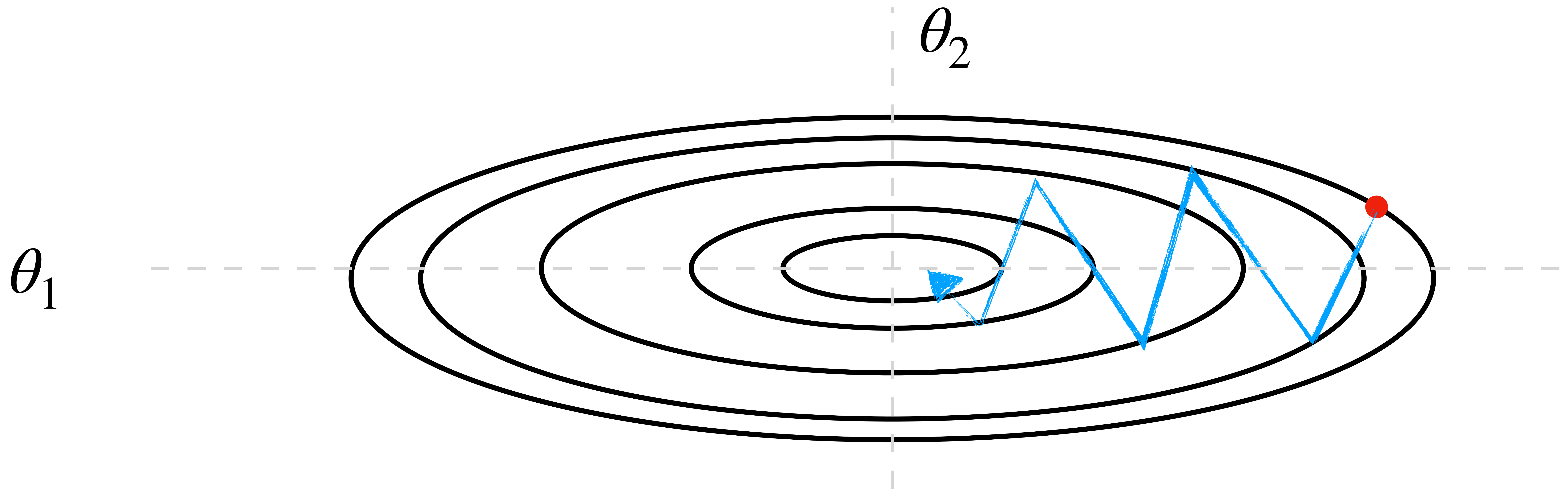
What assumption does it make that is breaking?

How can we make it choose a better direction?

Gradient Descent as Steepest Descent

Gradient Descent is simply Steepest Descent with L2 norm

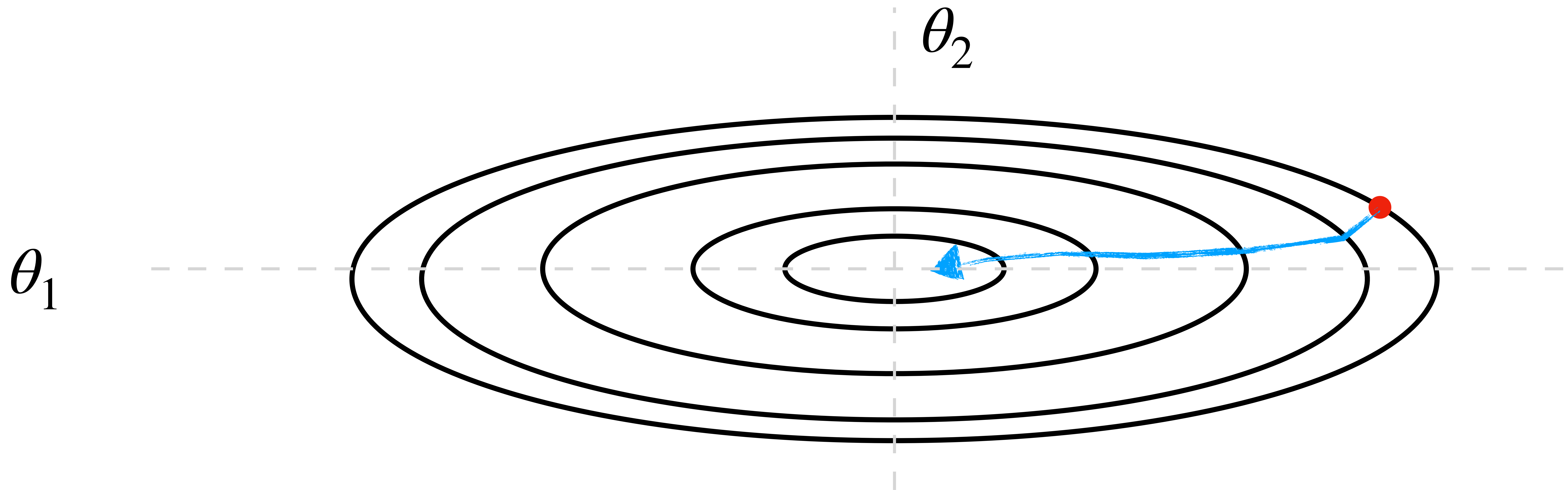
$$\min_{\Delta\theta} J(\theta + \Delta\theta) \ \text{ s.t } \ ||\Delta\theta|| \leq \epsilon \longrightarrow \Delta\theta = -\nabla_\theta J(\theta)$$



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 $\pi_\theta(a \,|\, s)$"

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

"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

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

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

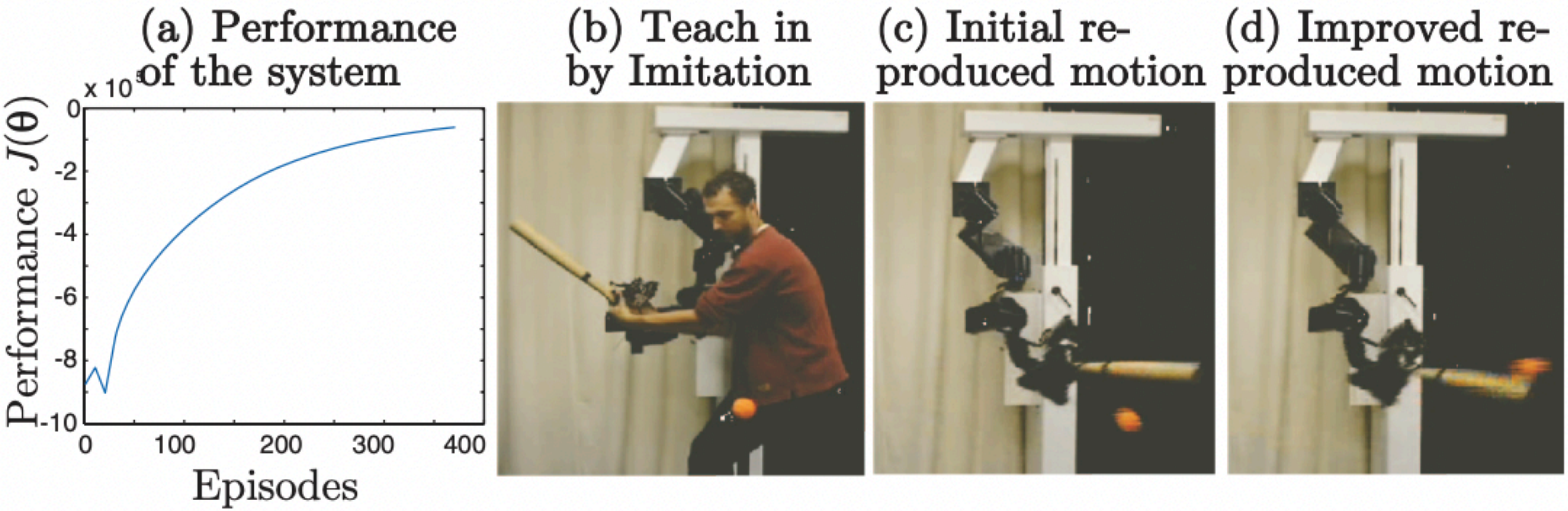
Update parameters $\theta \leftarrow \theta + \alpha \boxed{\tilde{G}^{-1}(\theta)} \tilde{\nabla}_\theta J$.
return π_θ

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.

Nightmare 3:

High Variance

tl;dr

The Policy Gradient Theorem

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

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

15

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