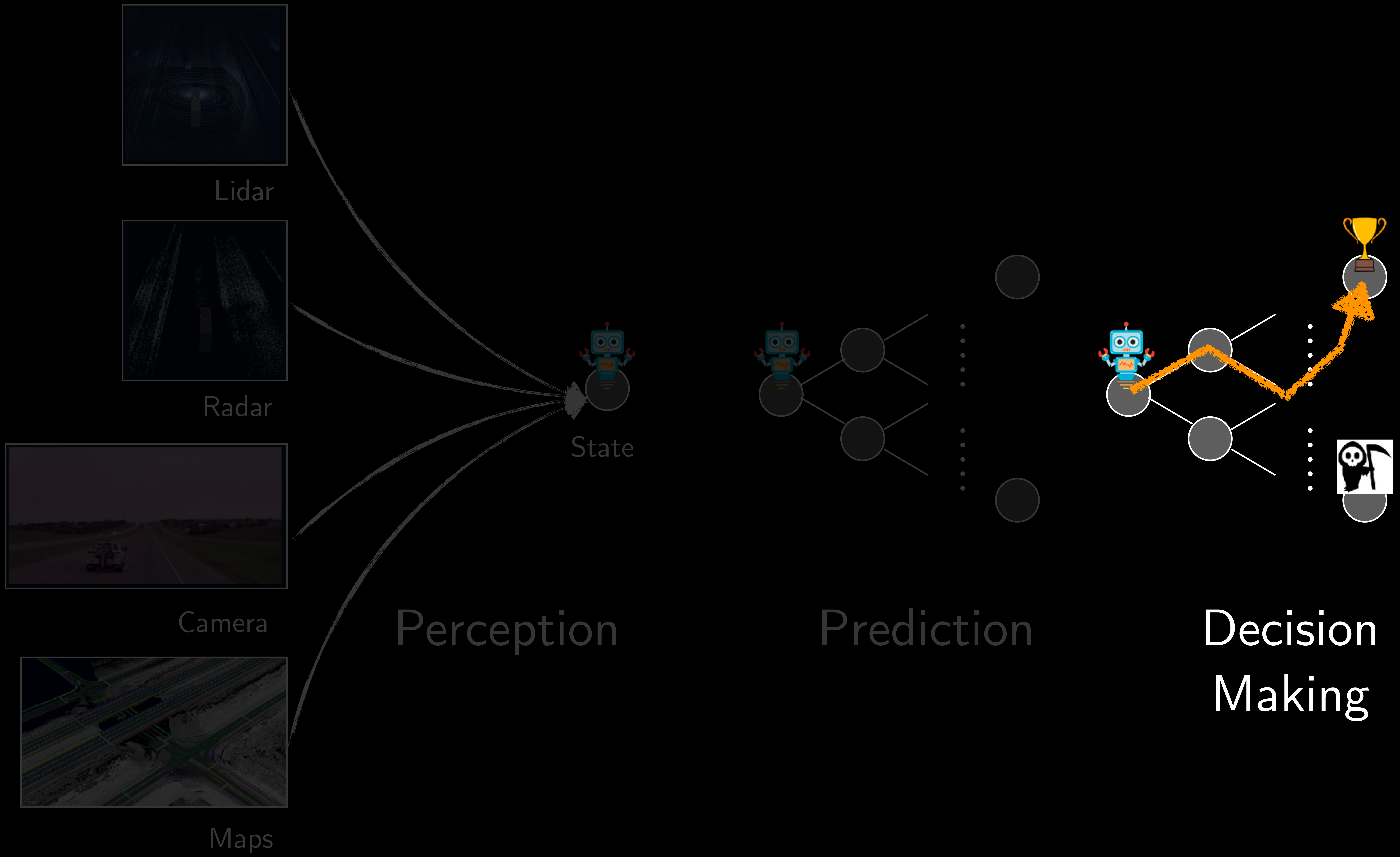
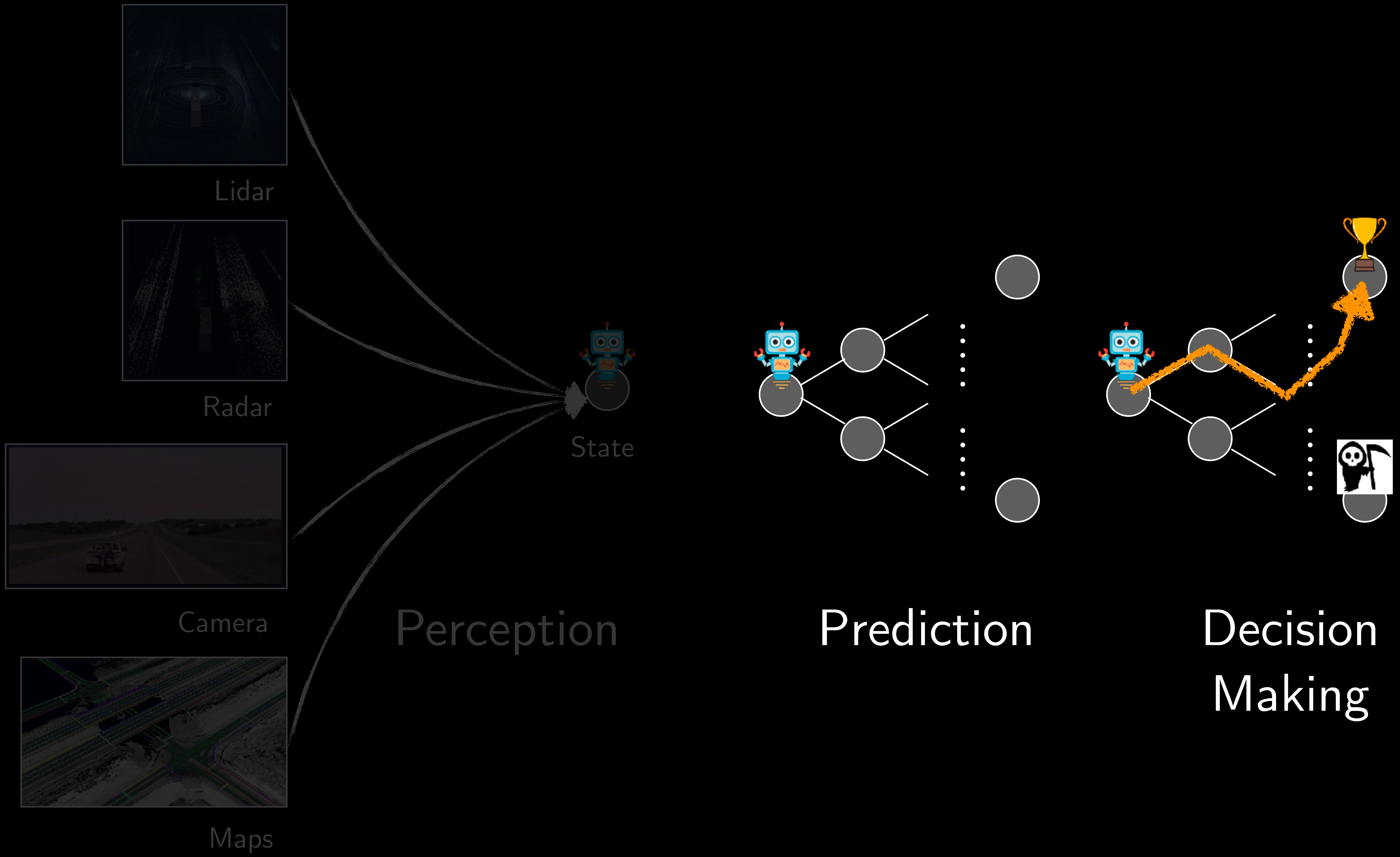
Temporal Difference & Q Learning

Sanjiban Choudhury

Cornell Bowers CIS
Computer Science

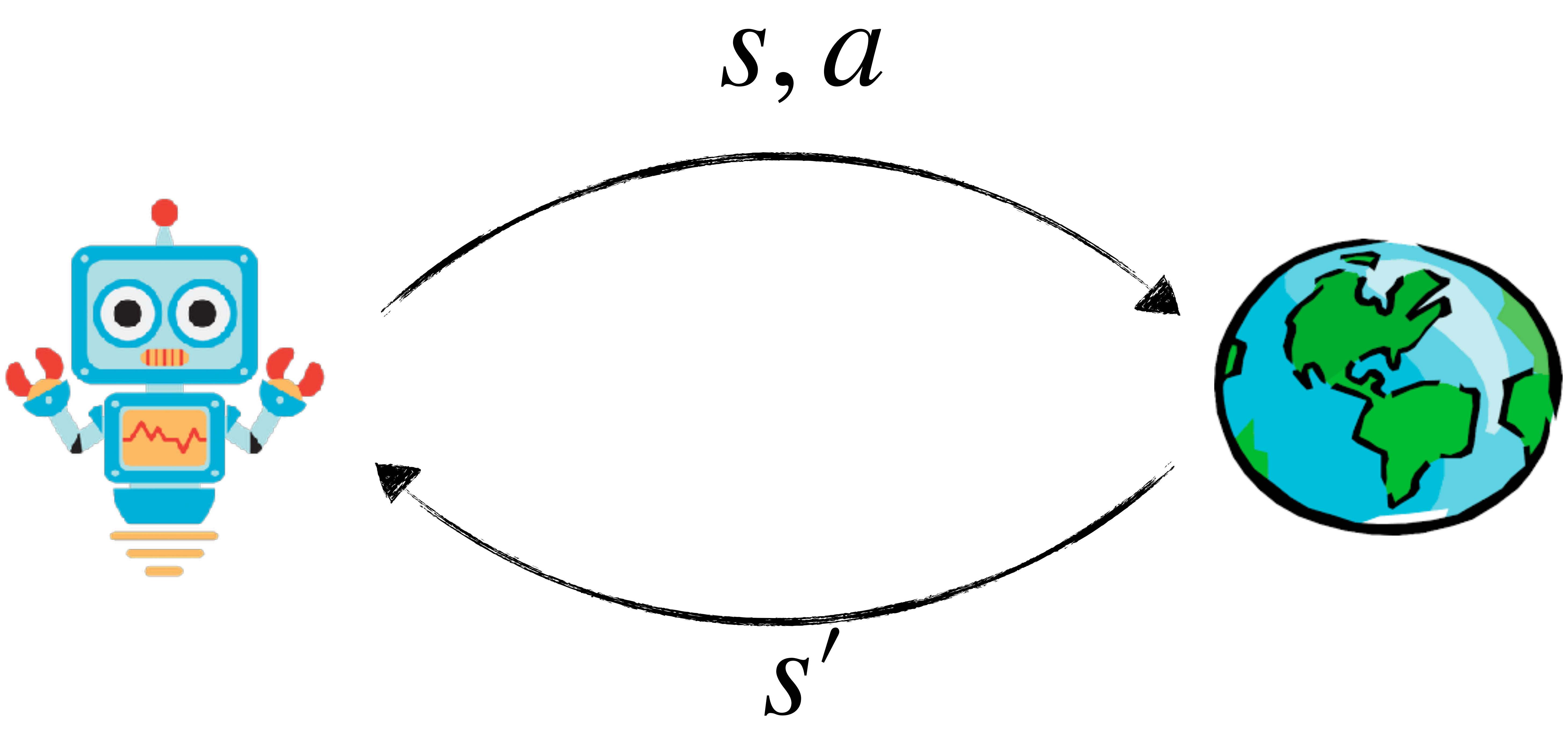
The story thus far …


Lidar
Radar
Camera
Maps
State
Perception
Prediction
Decision Making


Lidar
Radar
Camera
Maps
State
Perception
Prediction
Decision Making

What if the transitions are unknown?

$$< S , A , C , \mathcal{T} >$$



Activity!

Think-Pair-Share

Think (30 sec): What is the MDP <S, A, C, T> for this robot? Is the transition T known?

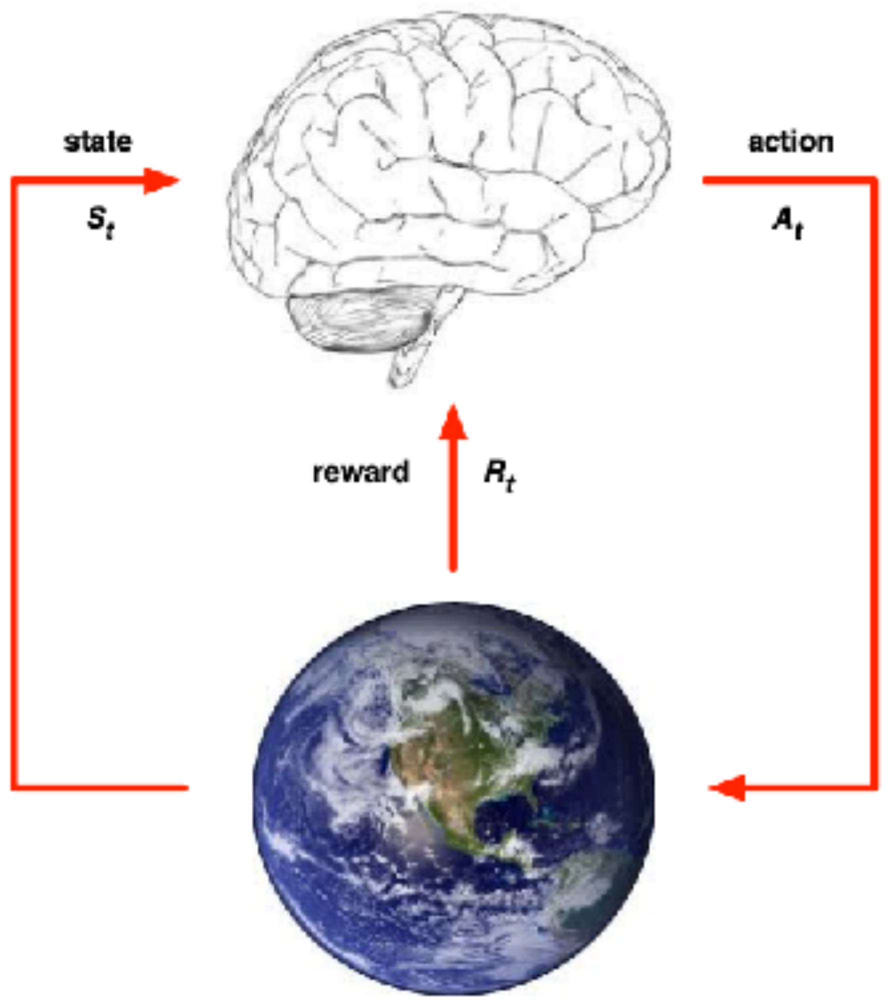
Pair: Find a partner

Share (45 sec): Partners exchange ideas

Model-Based OR Model Free?

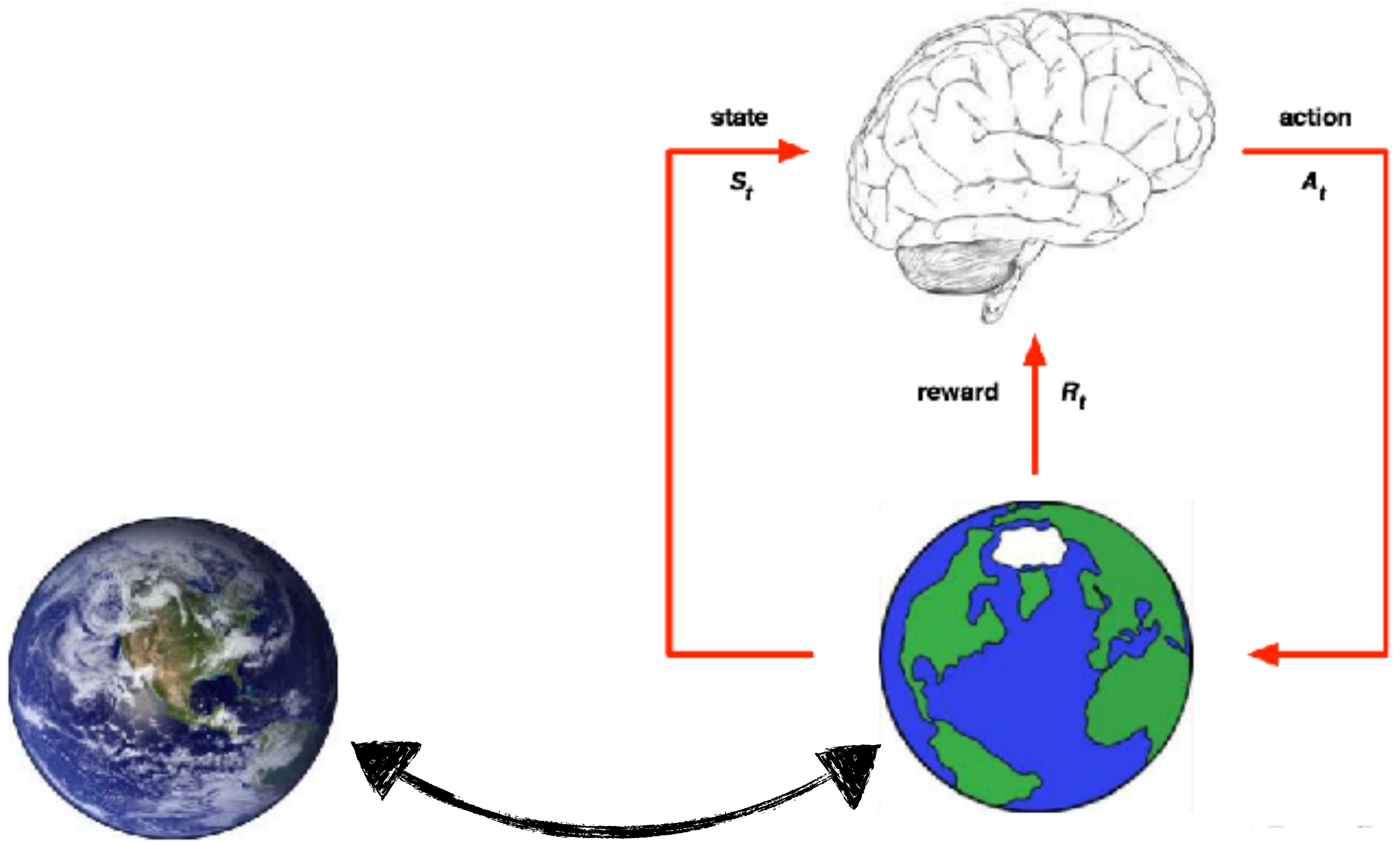
Model Free

Directly learn
π or $Q(s, a)$



Model Based

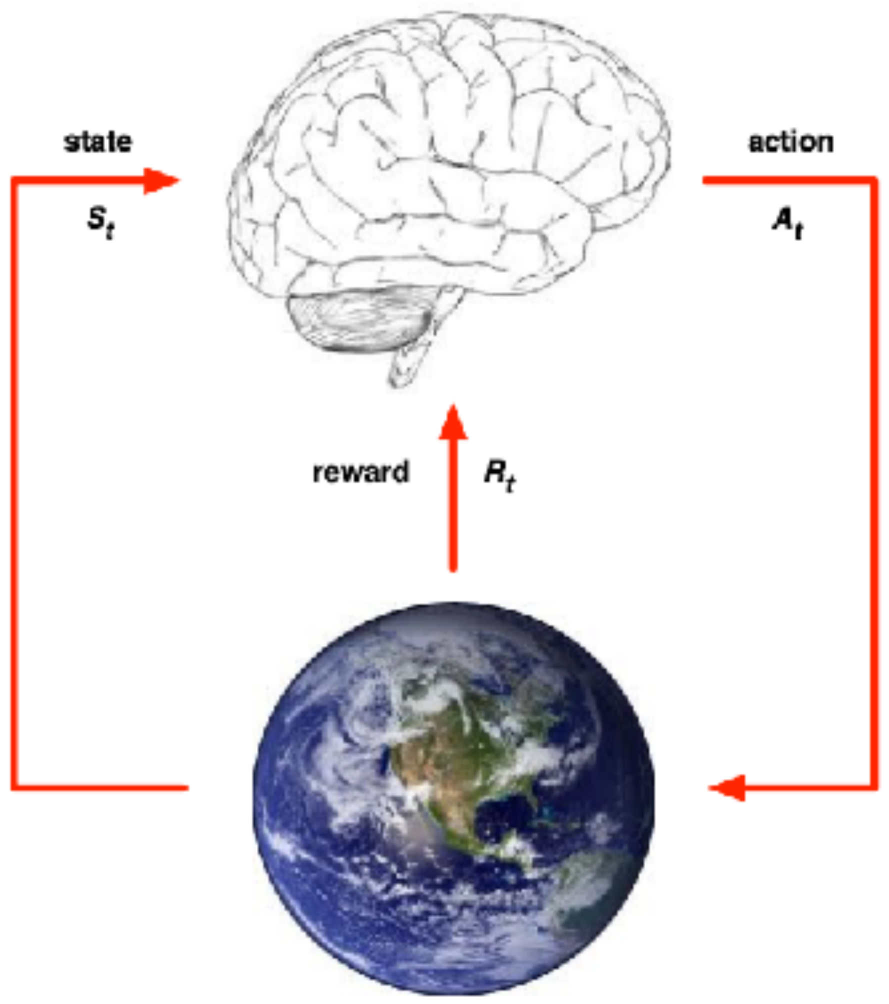
Learn a model
$T(s'|s, a)$, plan with
model to find π



Model-Based OR Model Free?

Model Free

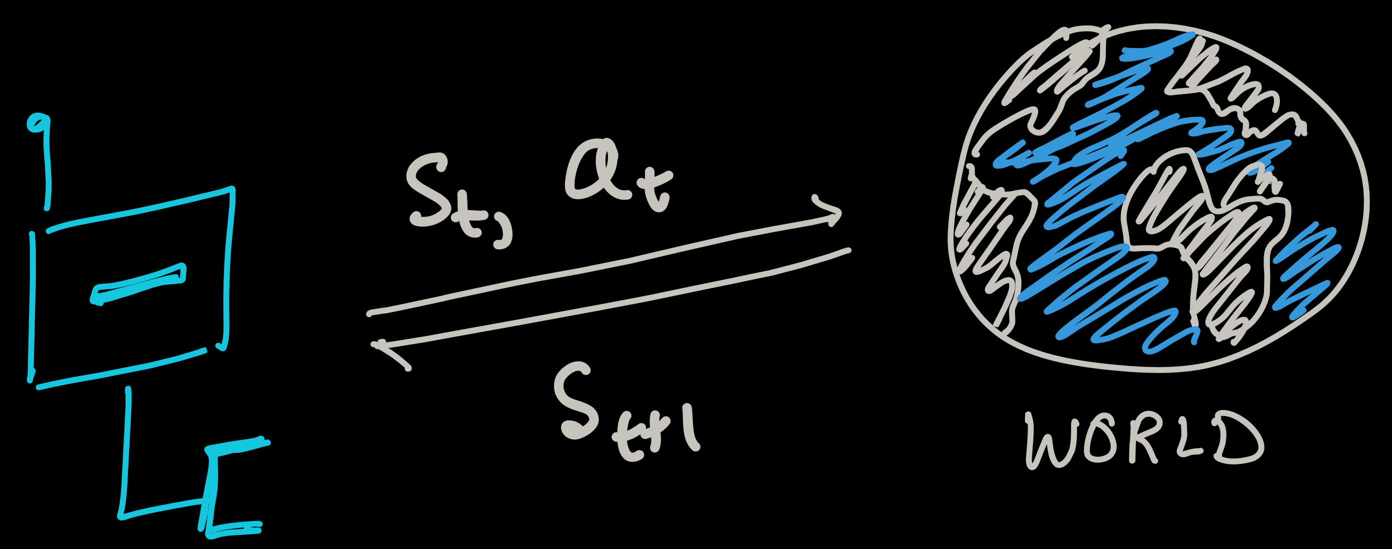
Directly learn
π or $Q(s, a)$



Model Based

Learn a model
$P(s' \mid s, a)$, plan with
model to find π

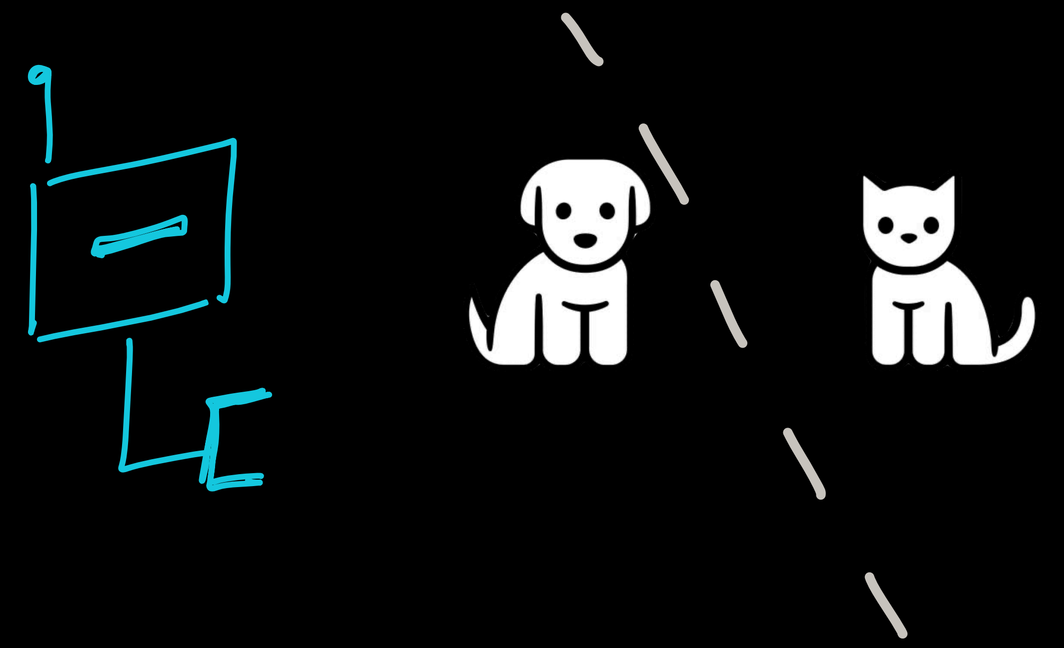
WHAT MAKES



REINFORCEMENT LEARNING

HARDER

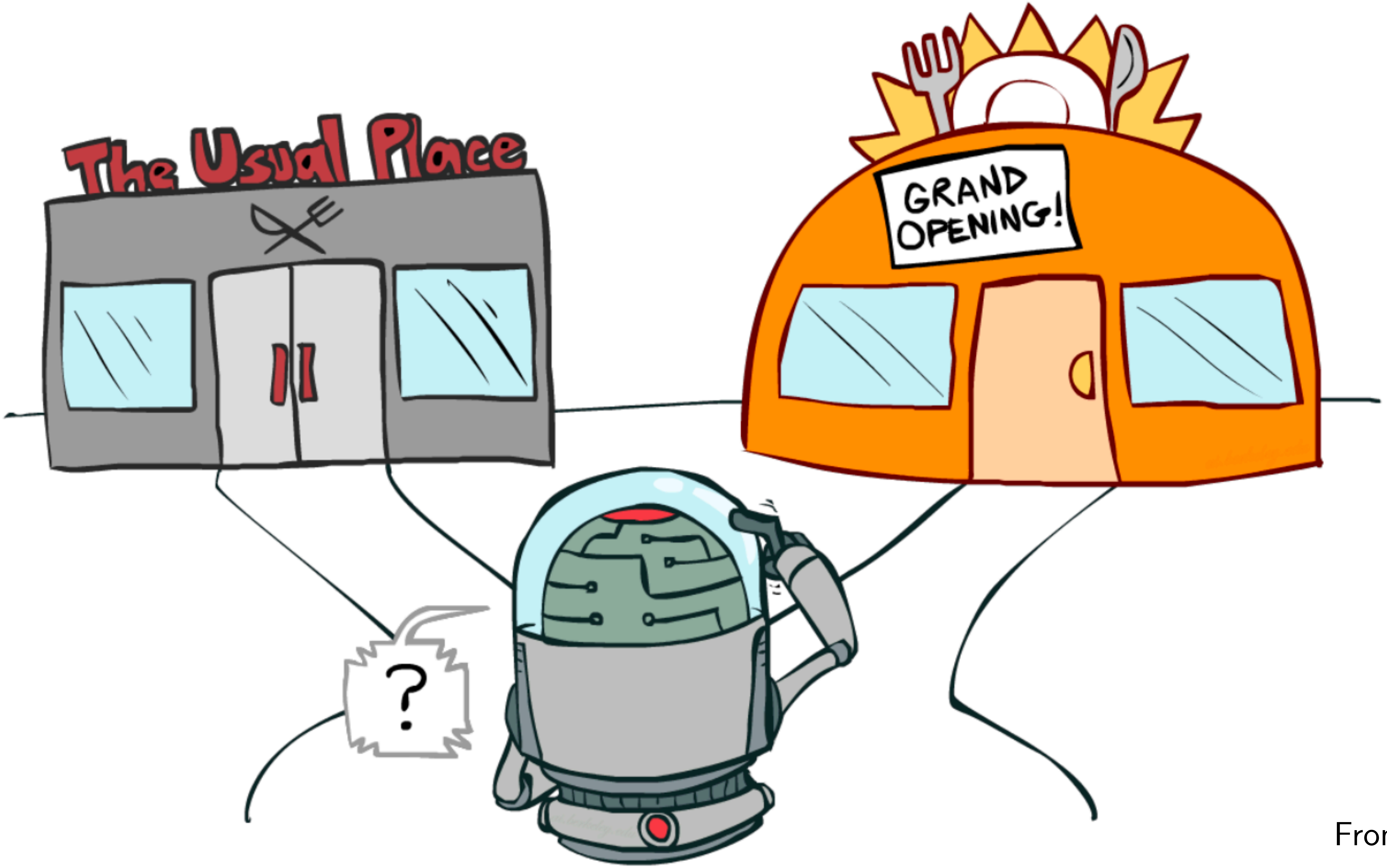
THAN



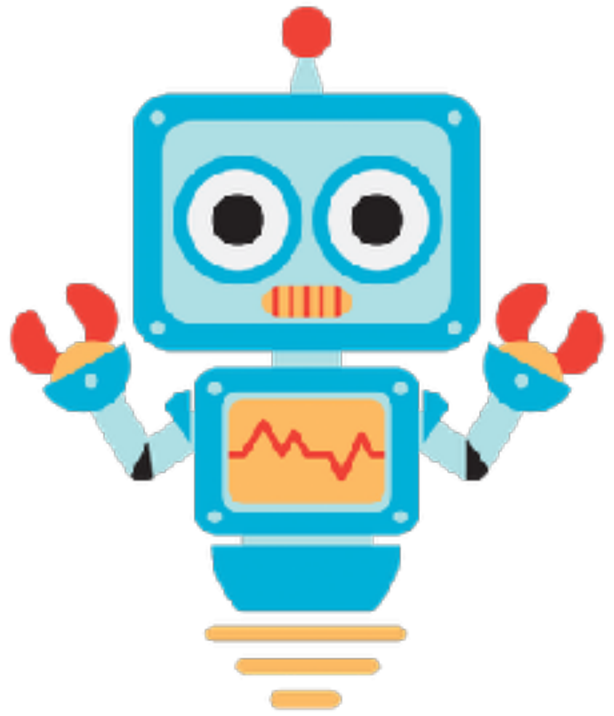
SUPERVISED LEARNING

?

Exploration vs Exploitation



From Dan Klein



Doors

a^1 ?

a^2 ?

a^3 ?

⋮

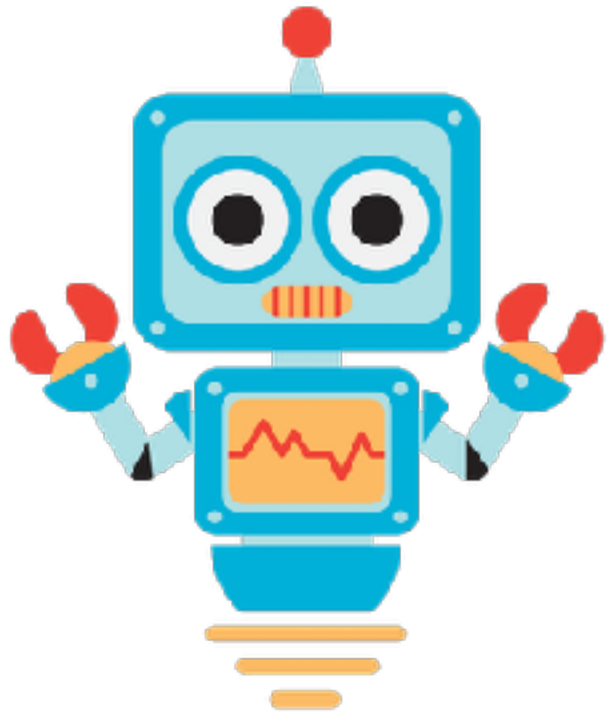
+100

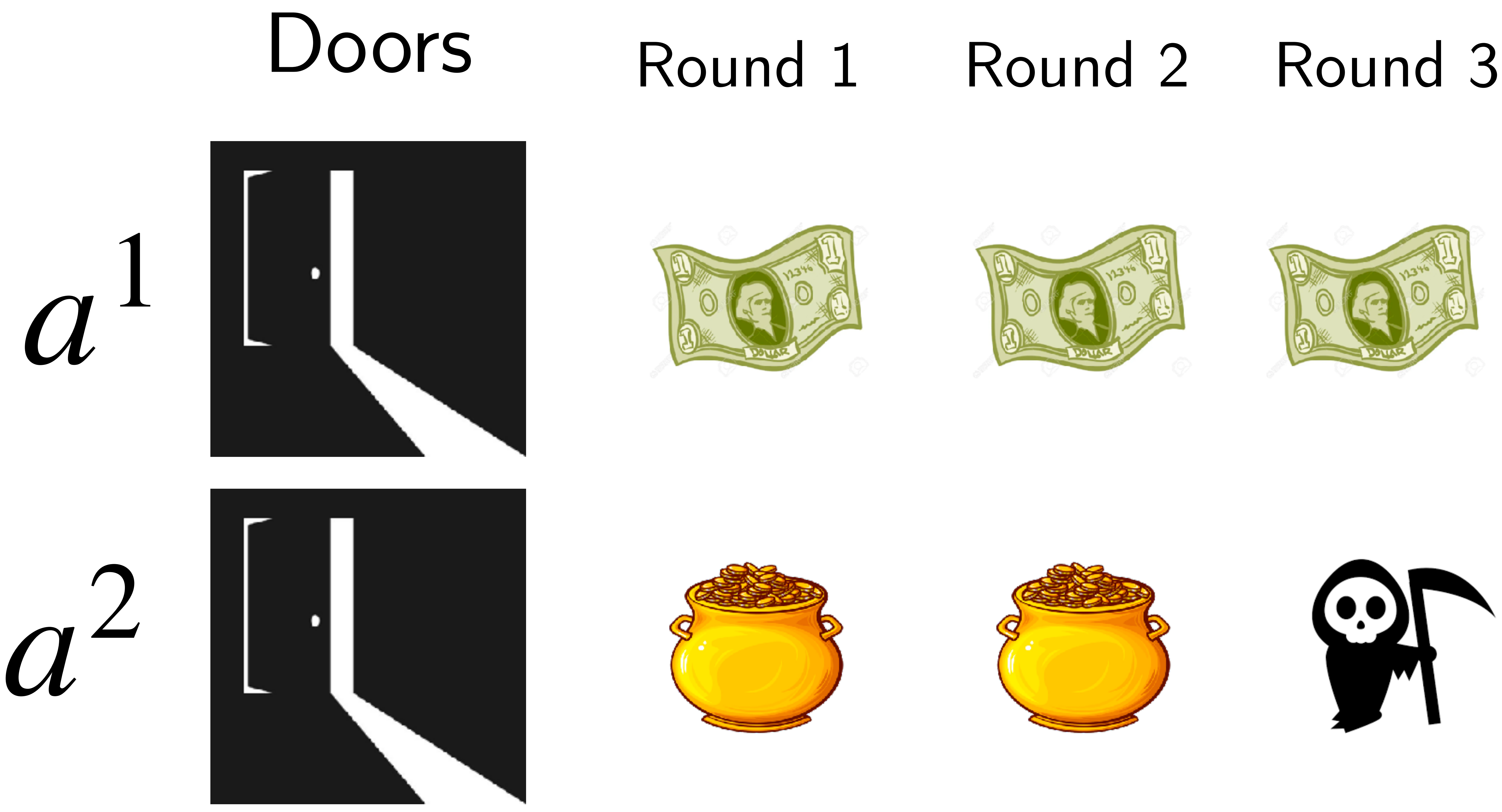
+1

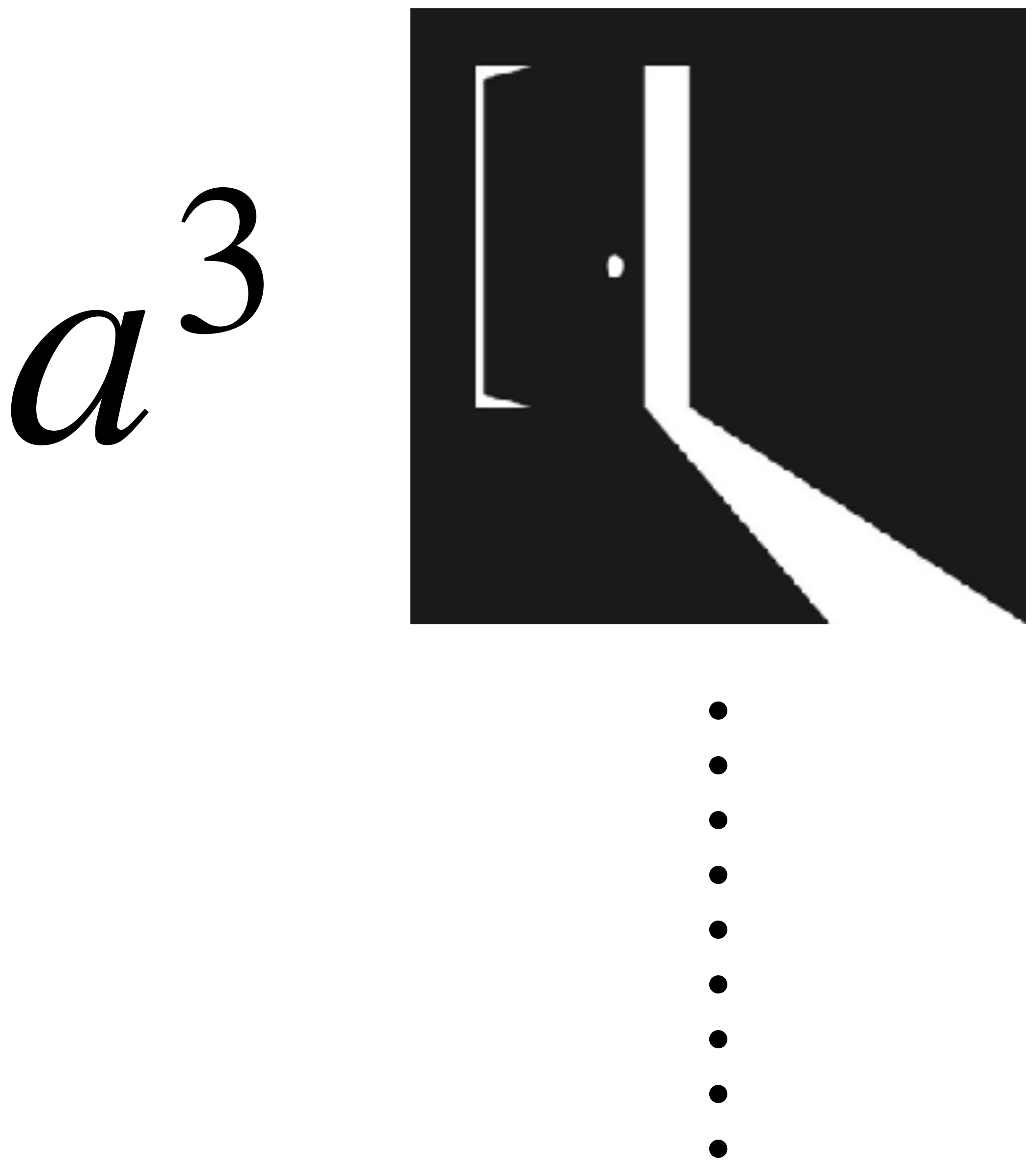
-1000


Doors
Round 1
Round 2
Round 3
a^1
a^2
a^3
+100
+1
-1000


Doors
Round 1
Round 2
Round 3
a^1
a^2
a^3
+100
+1
-1000



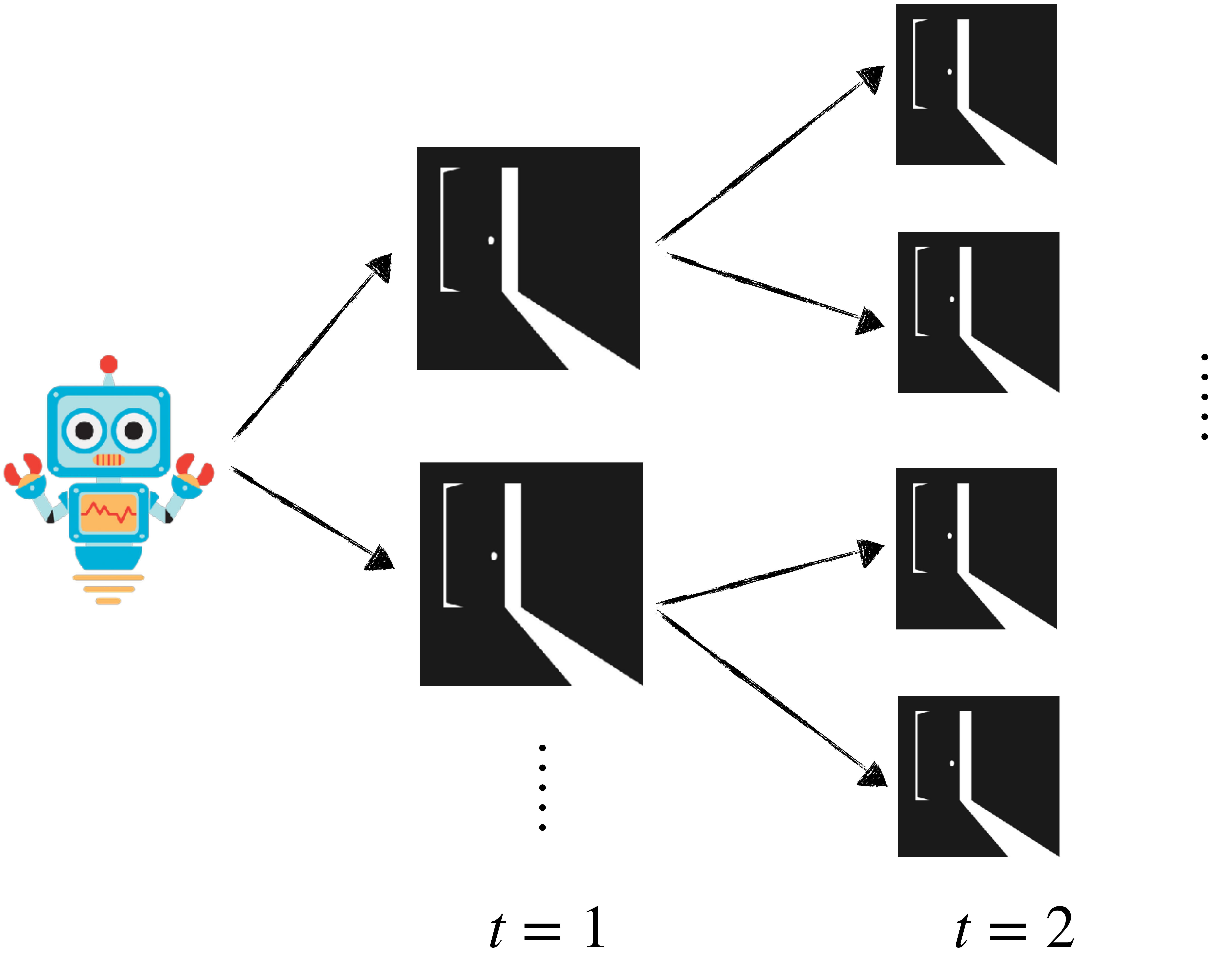




How do we explore/ exploit when picking doors?

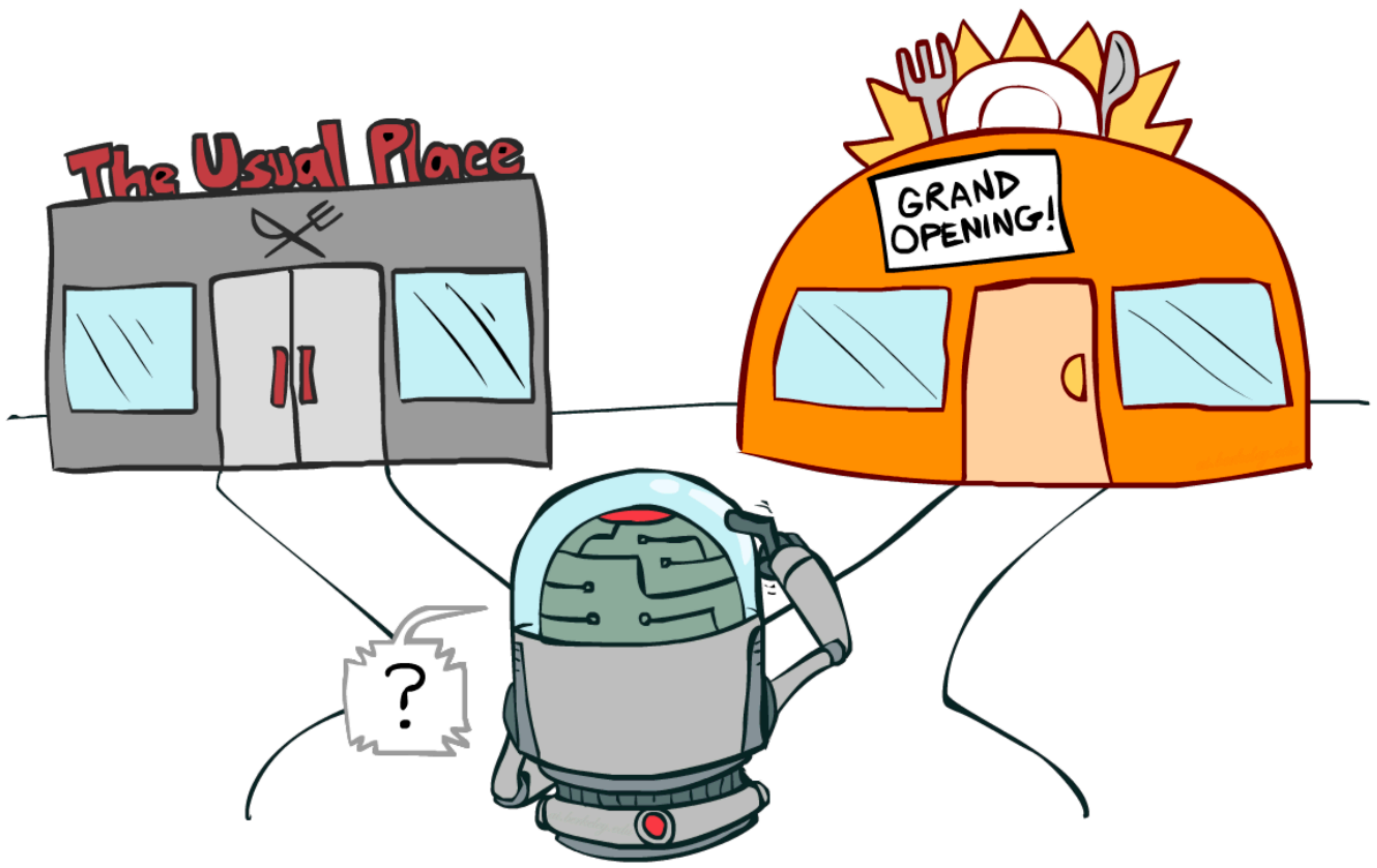
What if we played the game over multiple time steps?


+100
+1
$t = 1$
$t = 2$

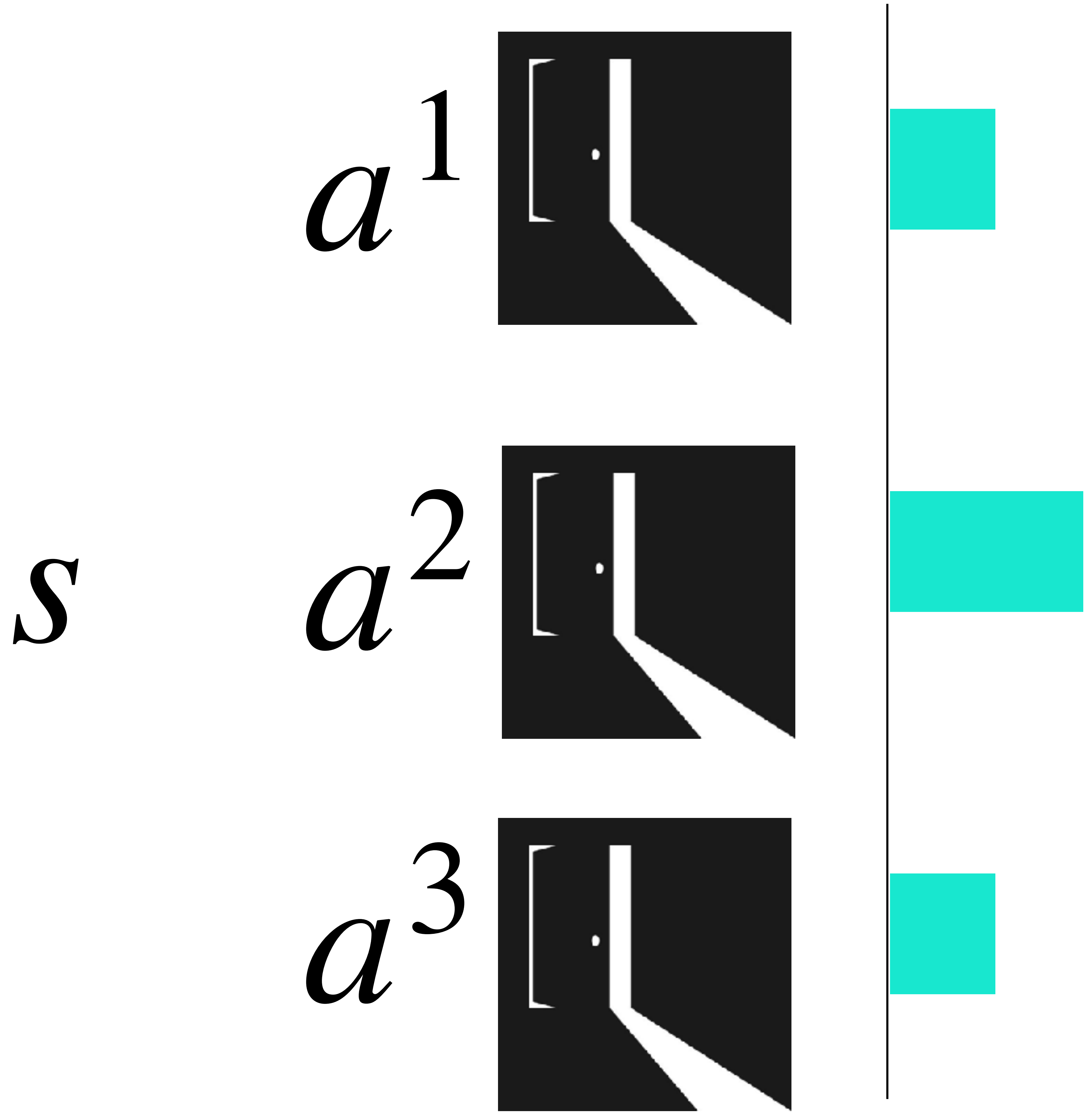


How do we estimate values of each door?

Two Ingredients of RL

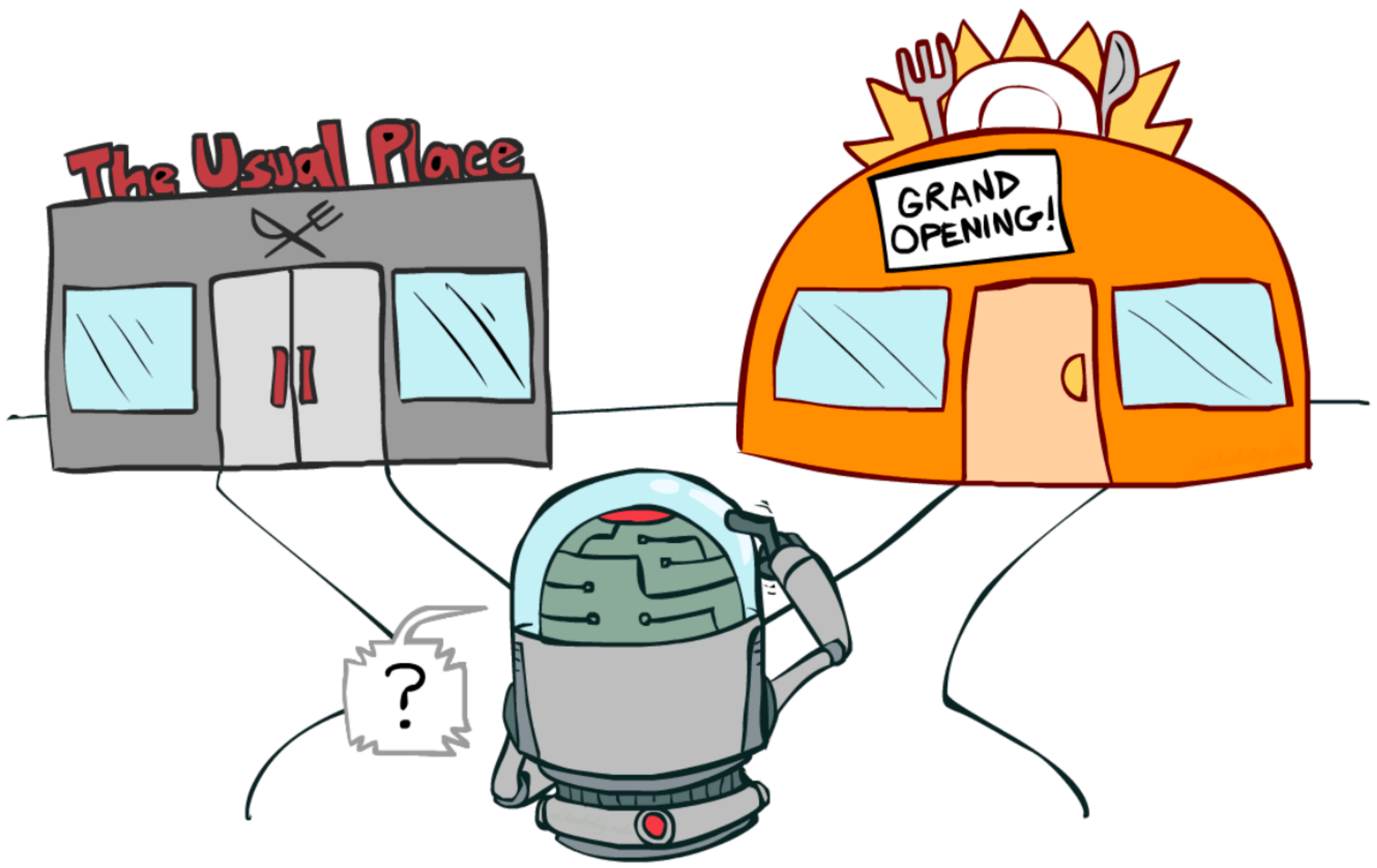


Exploration Exploitation

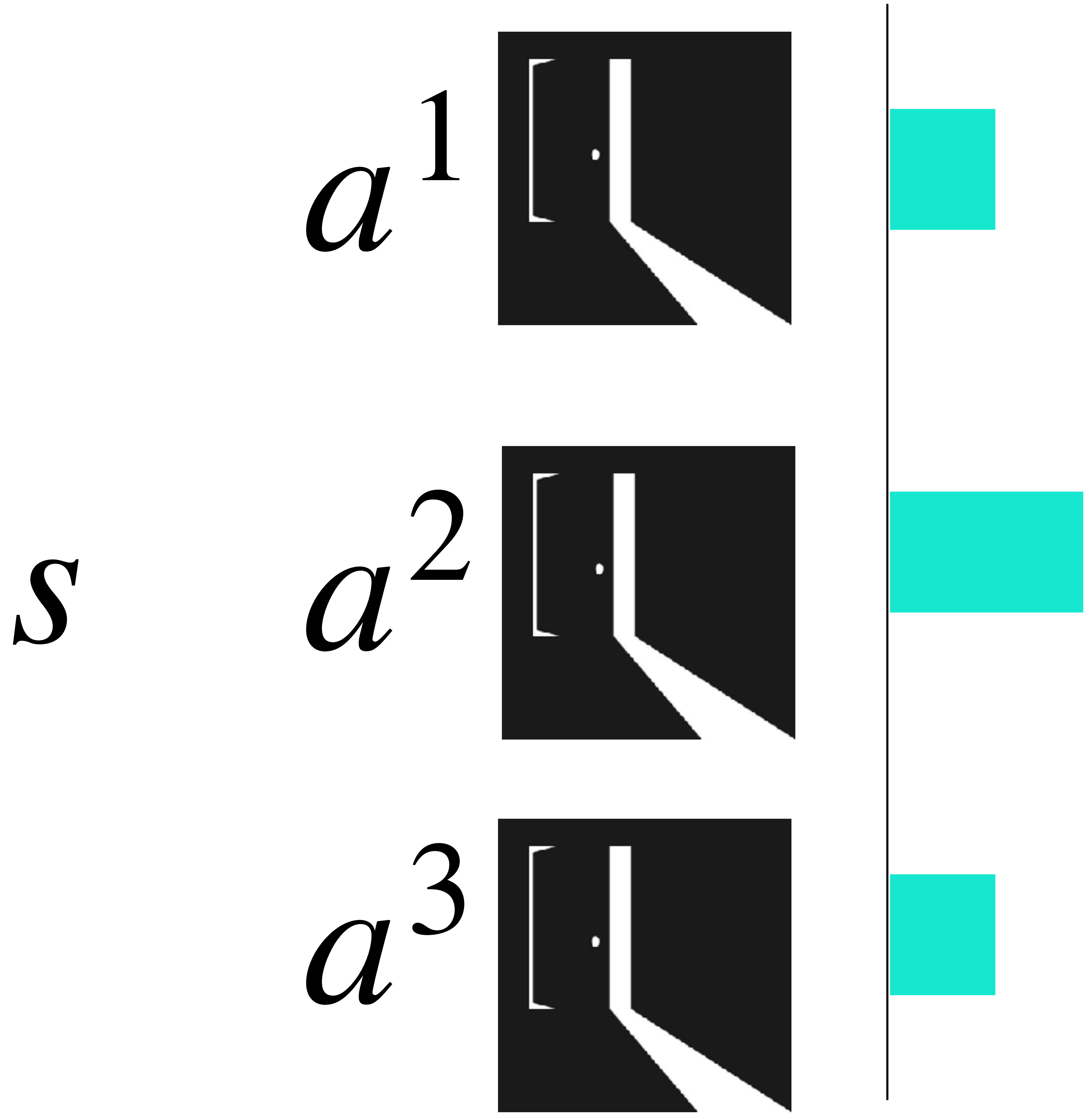


Estimate Values $Q(s, a)$

Two Ingredients of RL

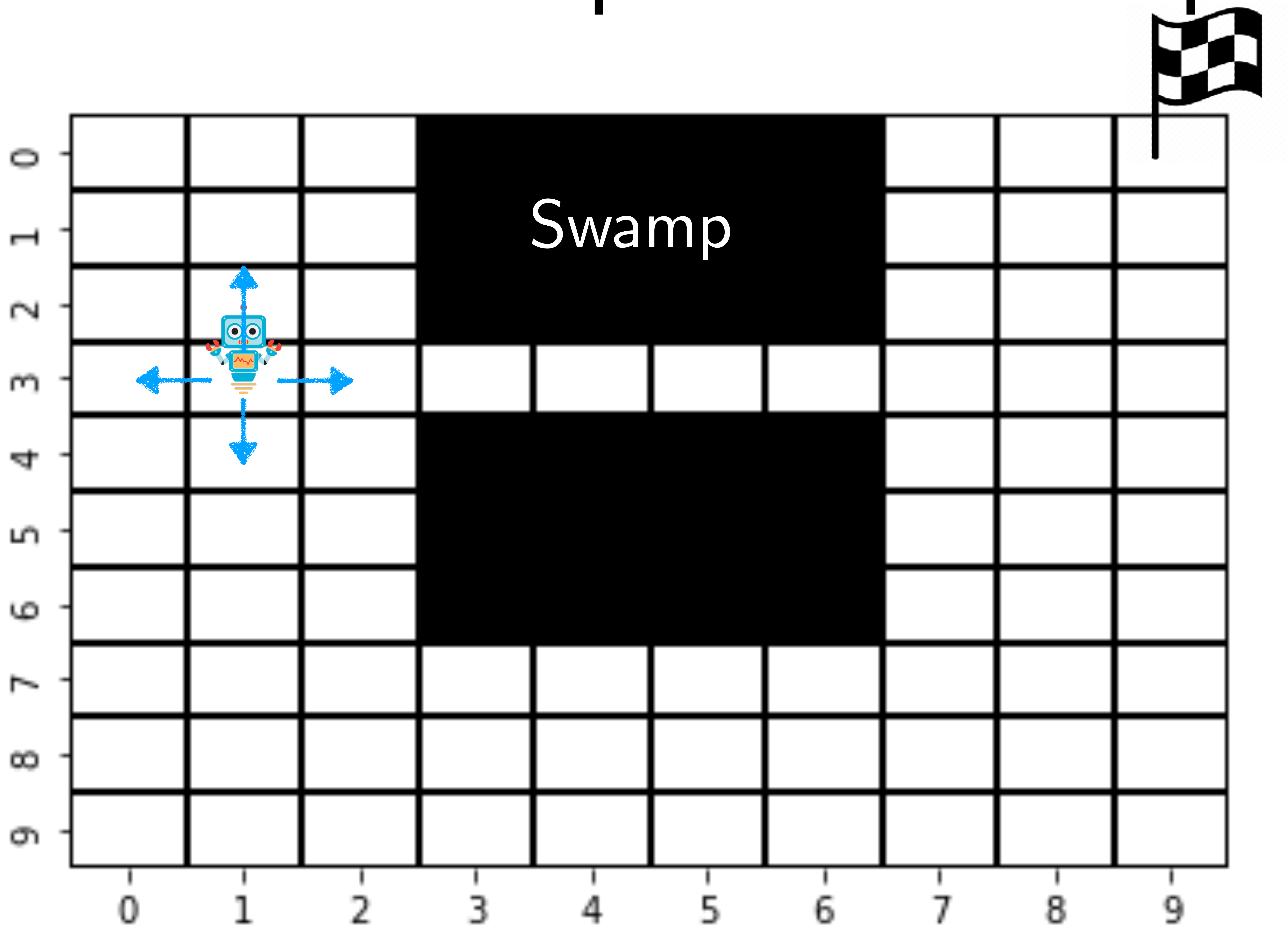


Exploration Exploitation



Estimate Values $Q(s, a)$

Recap: The Swamp MDP



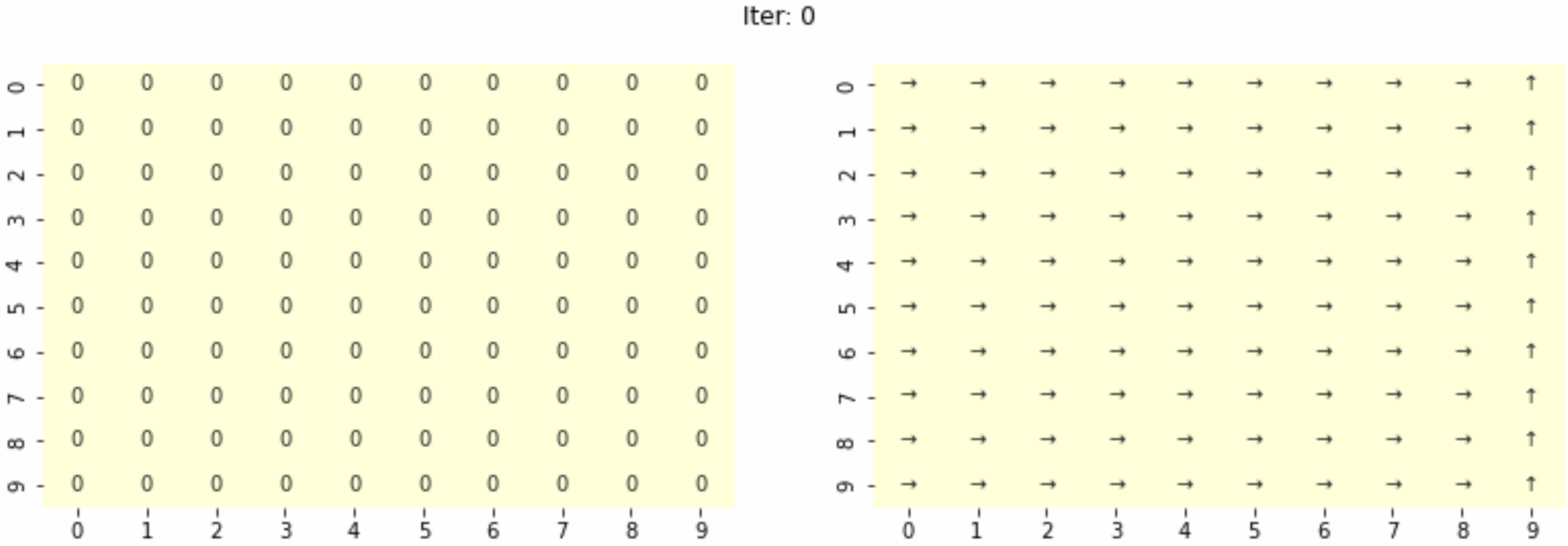
$$< S\,, A\,, C\,, \mathcal{T} >$$

- Two absorbing states: Goal and Swamp
- Cost of each state is 1 till you reach the goal
- Let's set T = 30

When the
MDP is known!

Run Value
/ Policy Iteration

When MDP is known: Policy Iteration



$$V^\pi(s) = c(s, \pi(s)) + \gamma \mathbb{E}_{s' \sim \mathcal{T}(s,a)} V^\pi(s')]$$

$$\pi^+(s) = \arg\min_a c(s,a) + \gamma \mathbb{E}_{s' \sim \mathcal{T}(s,a)} V^\pi(s')]$$

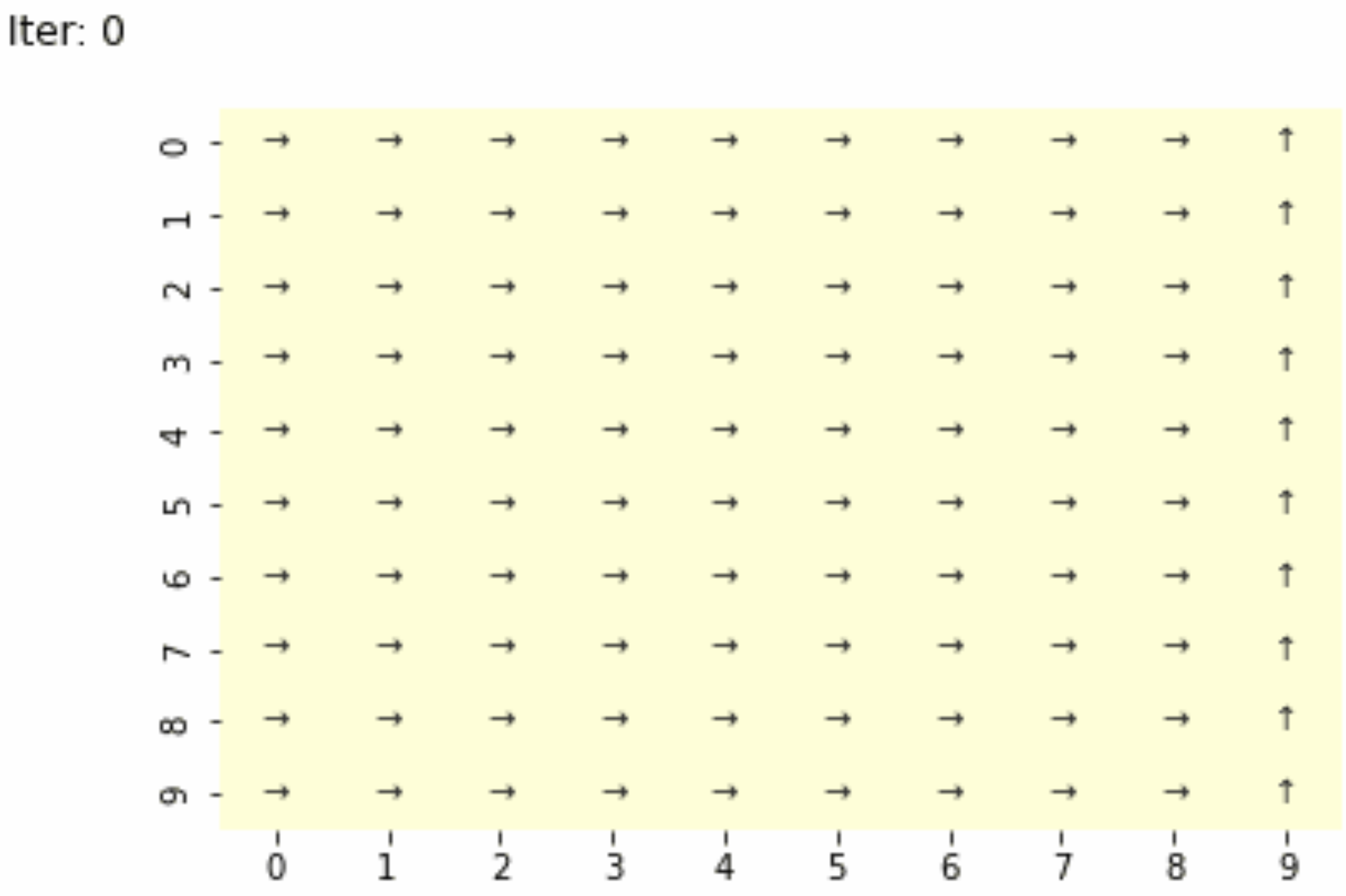
Estimate value

Improve policy

What happens when the MDP is *unknown?*

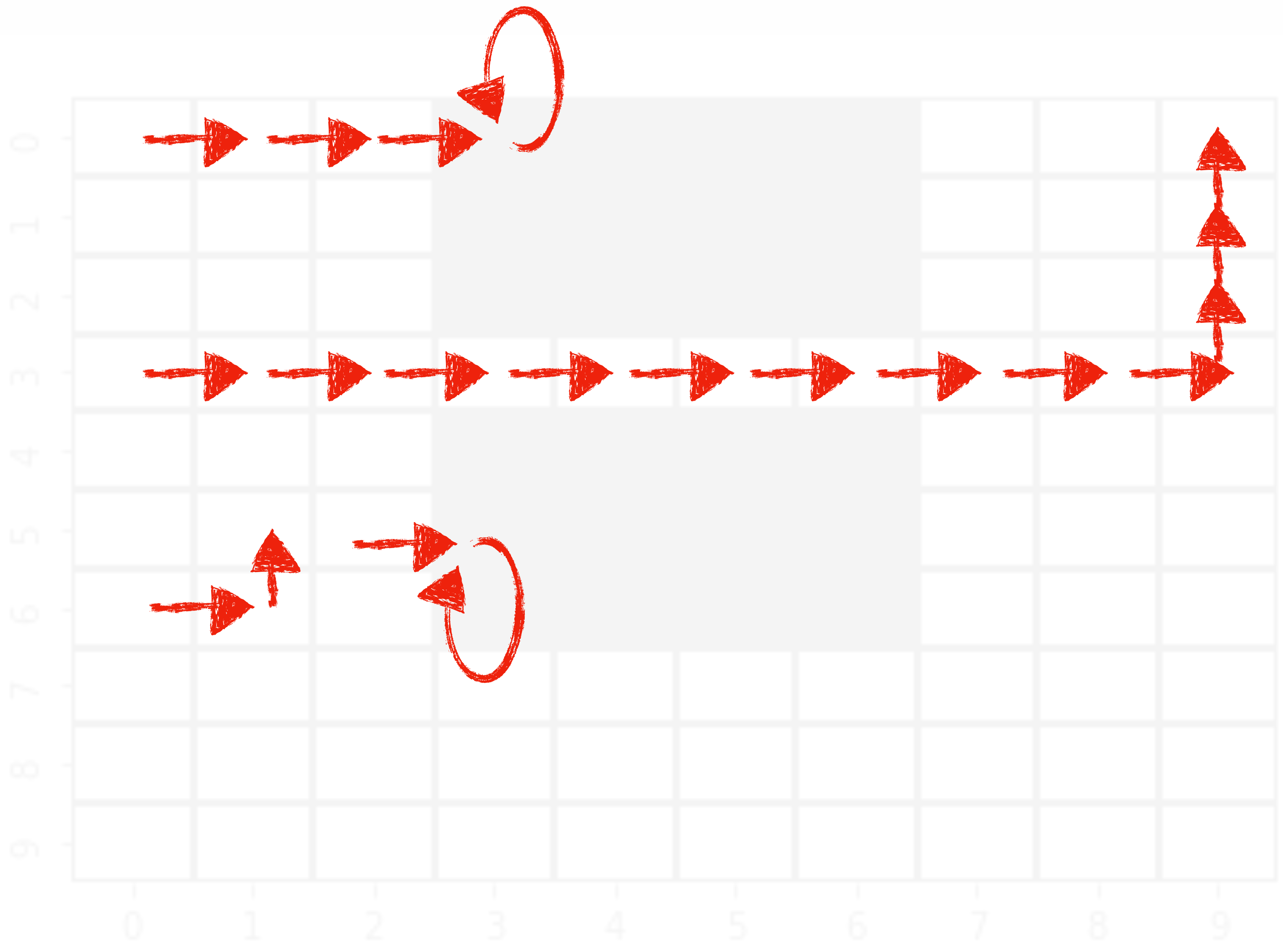
Need to *estimate the value* of policy

Value $V^{\pi}(s)$

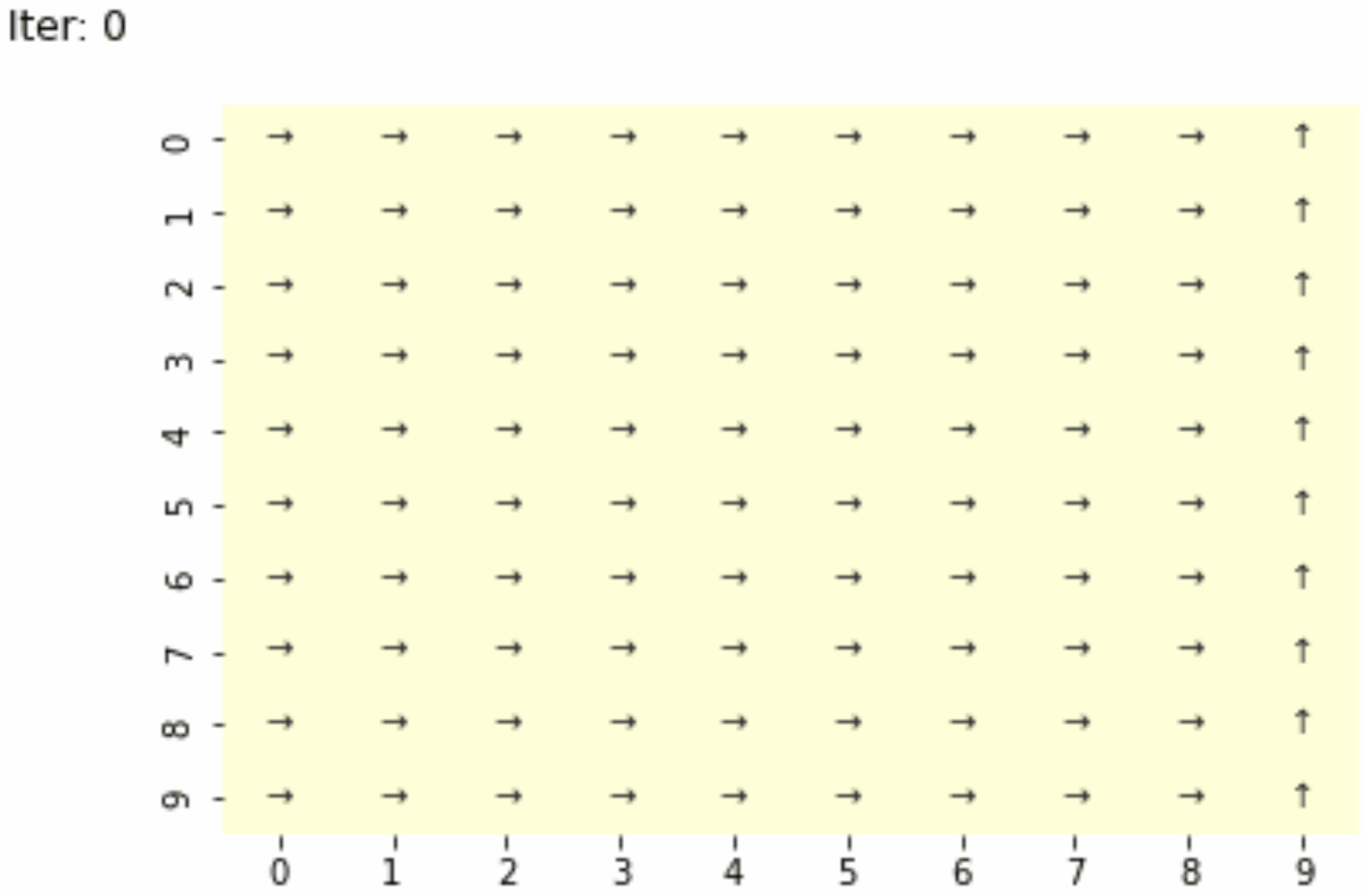


Policy π

Estimate the value of policy from sample rollouts

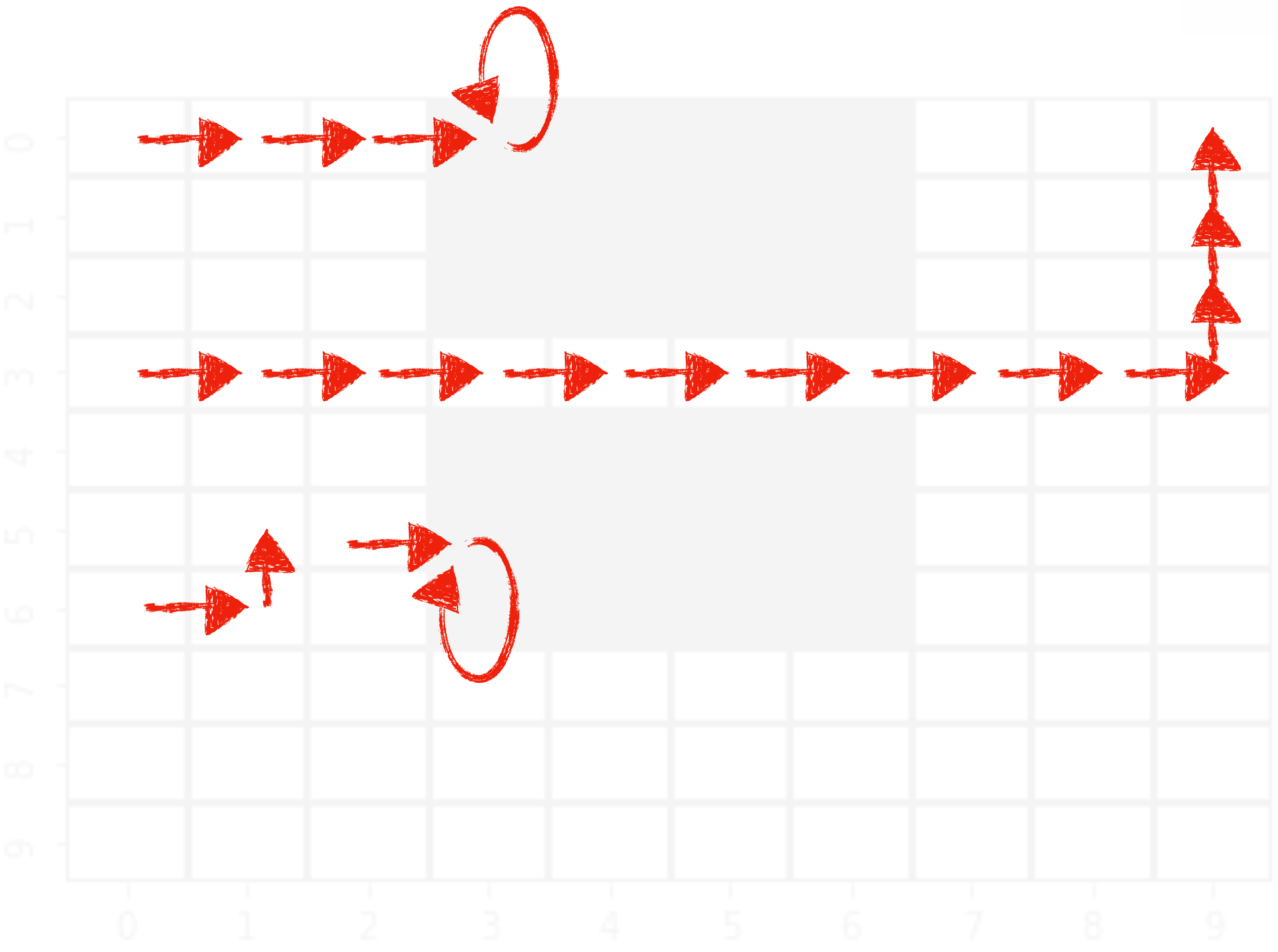


Roll outs

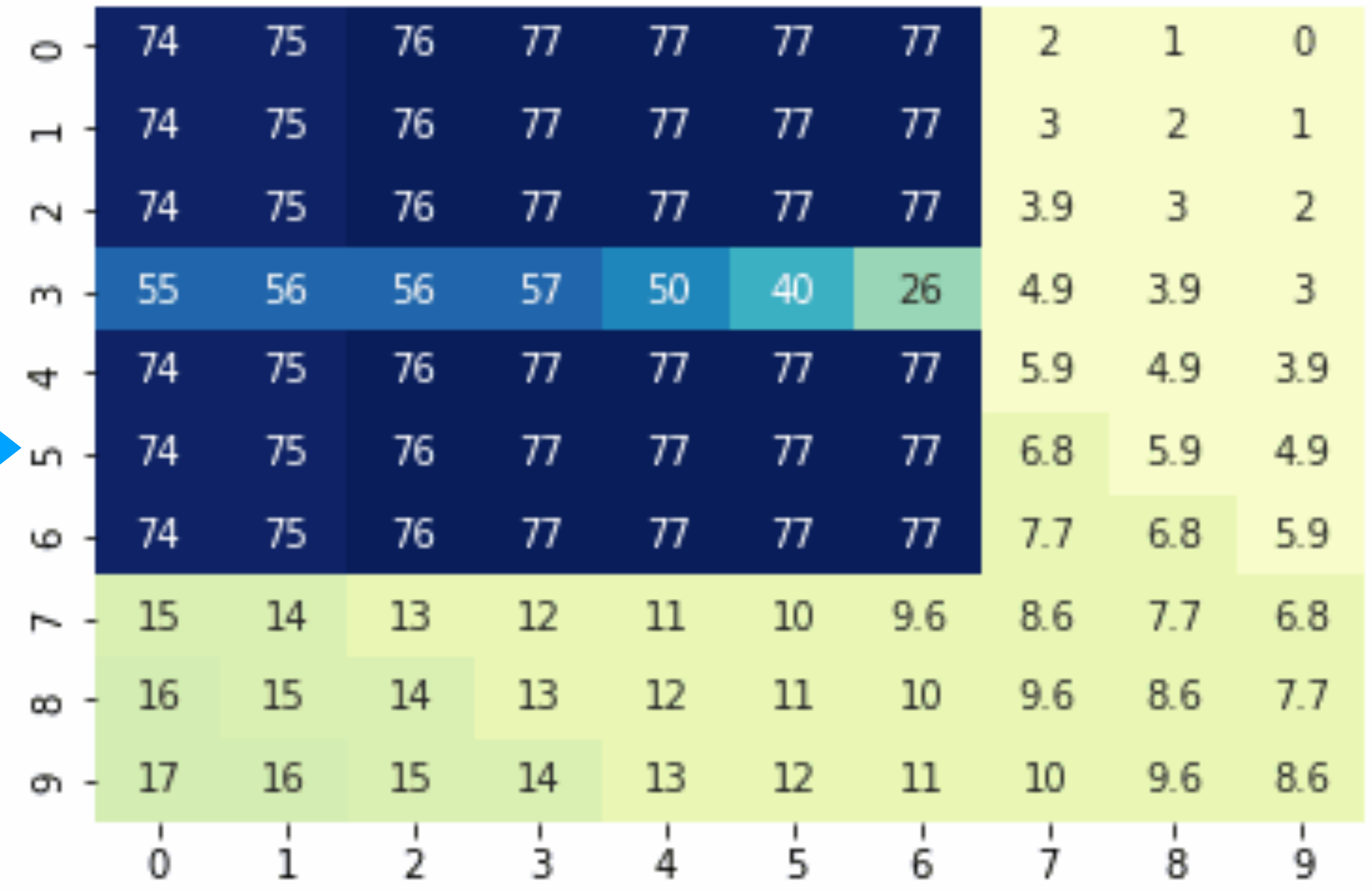


Policy π

Estimate the value of policy from sample rollouts



Roll outs



Value $V^{\pi}(s)$

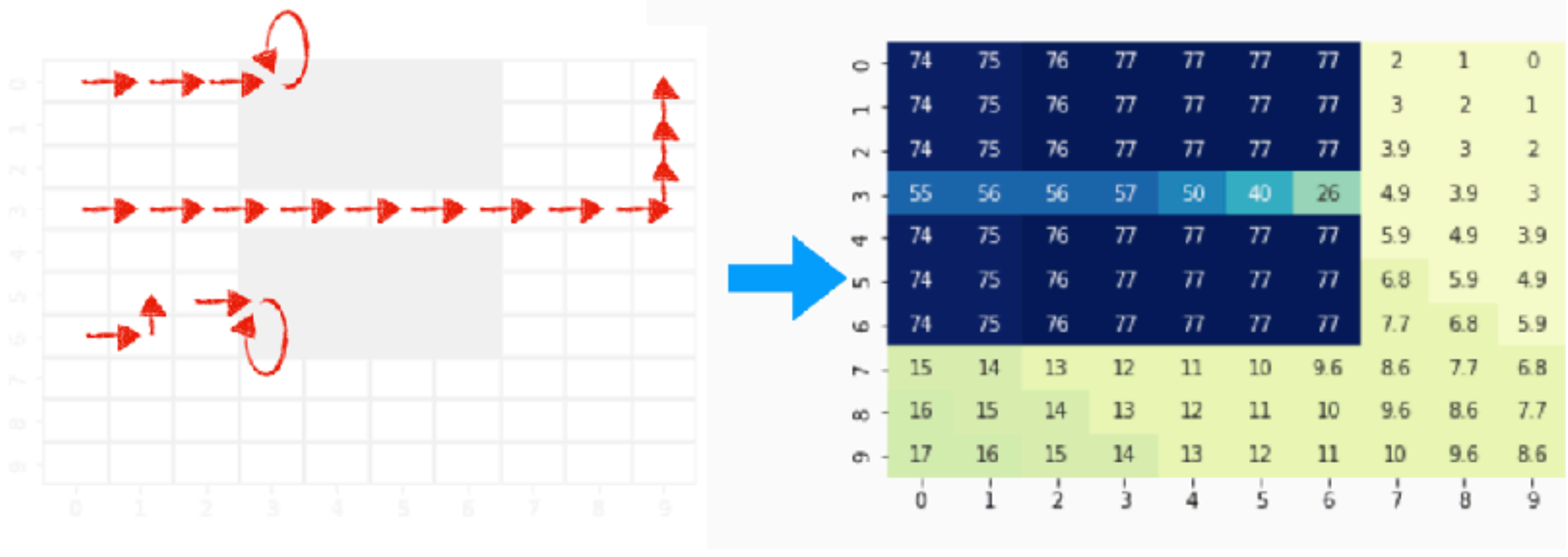
Activity!

Think-Pair-Share

Think (30 sec): Given a bunch of roll-outs, how can you estimate value of a state? (Hint: More than one way!)

Pair: Find a partner

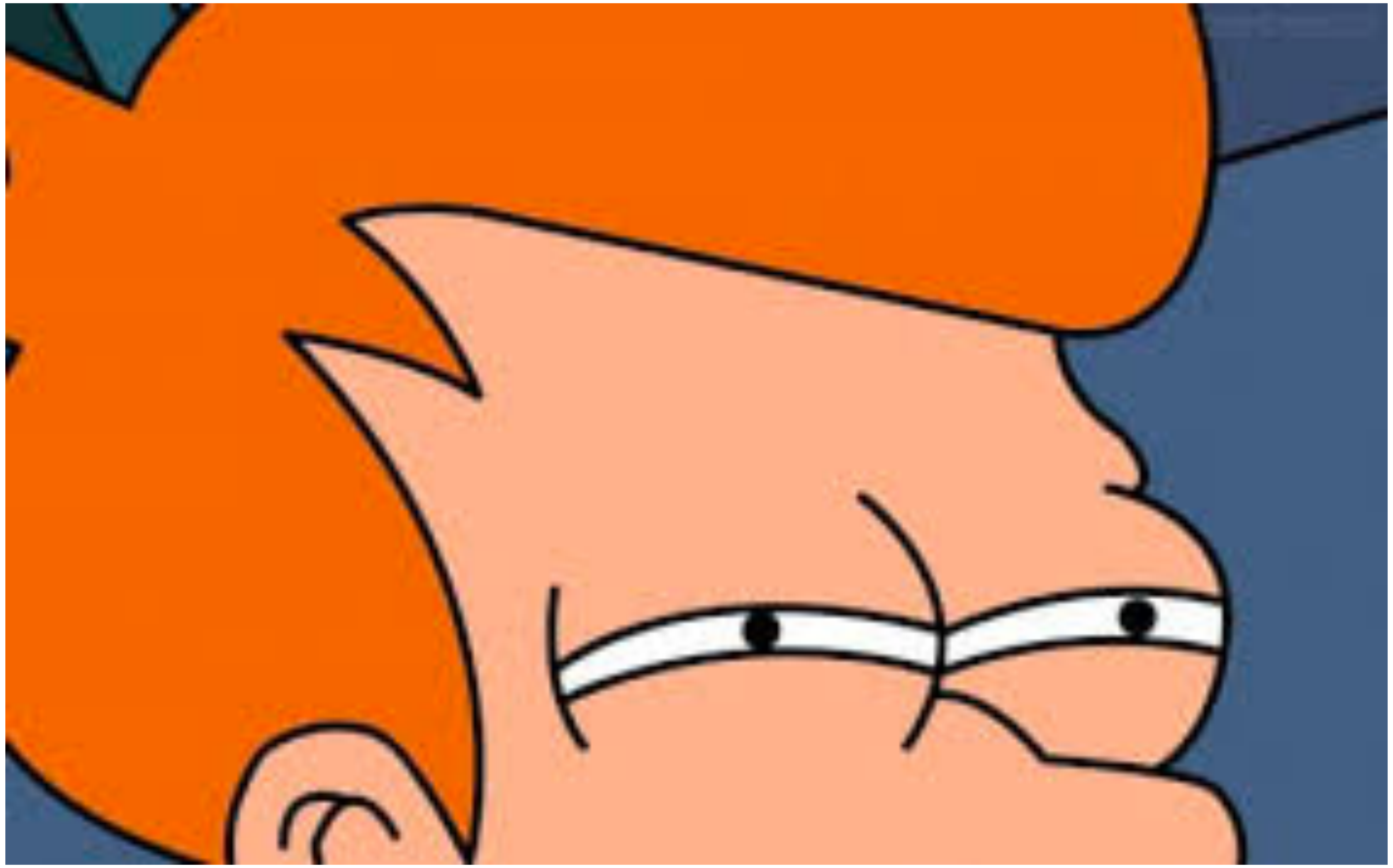
Share (45 sec): Partners exchange ideas



Roll outs

Value $V^{\pi}(s)$

Option 1:Trust only the actual returns!



Monte Carlo Evaluation

Goal: Learn $V^\pi(s)$ from complete rollout $s_1, a_1, c_1, s_2, a_2, c_2, \ldots \sim \pi$

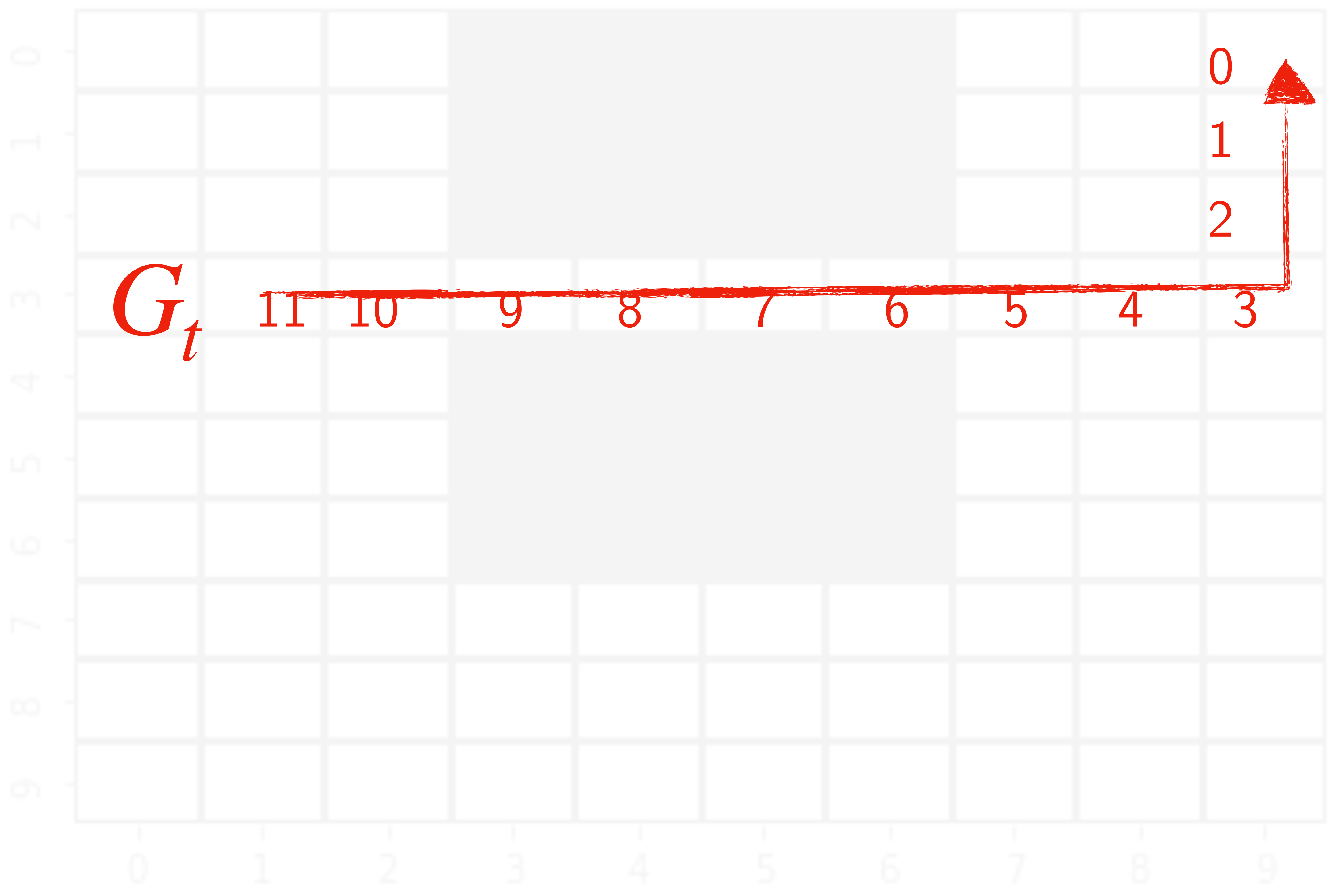
Define: *Return* is the total discounted cost

$$G_t = c_{t+1} + \gamma c_{t+2} + \gamma^2 c_{t+3} + \ldots$$

Value function is the expected return

$$V^\pi(s) = \mathbb{E}_\pi[G_t \,|\, s_t = s]$$

First Visit Monte Carlo



For episode in rollouts:

If state s is visited for *first* time t

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

Law of large numbers: $V(s) \rightarrow V^{\pi}(s)$

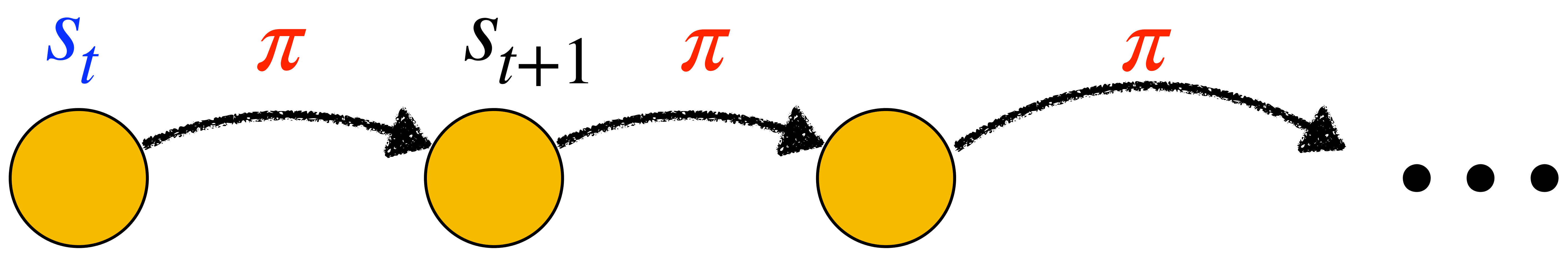
Can we do better than Monte Carlo?

What if we want quick updates?
(No patience to wait till end)

What if we don't have complete episodes?

Option 2: Believe in Bellman

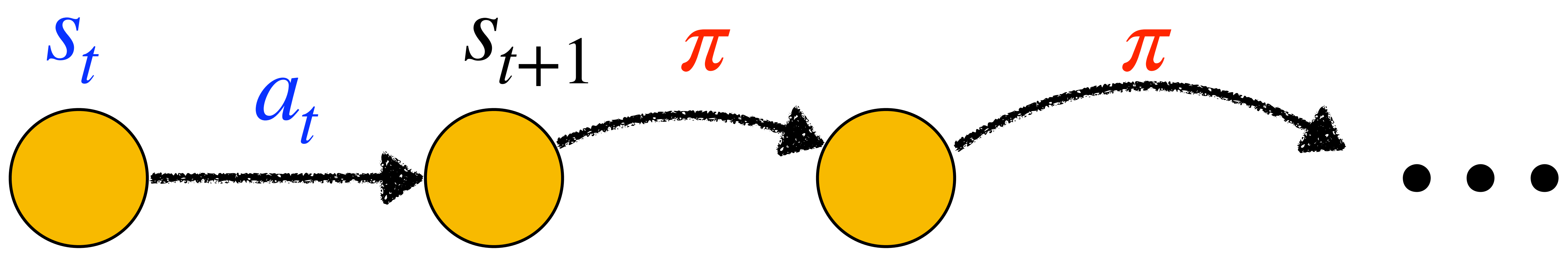
Recap: Value of a state



$$V^{\pi}(s_t) \quad = c_t + \quad \gamma c_{t+1} + \quad \gamma^2 c_{t+2} +$$

Expected discounted sum of cost from
starting at a state
and following a policy from then on

Recap: Value of a state-action



$$Q^{\pi}(s_t, a_t) = c_t + \gamma c_{t+1} + \gamma^2 c_{t+2} + \dots$$

Expected discounted sum of cost from starting at a state, executing action and following a policy from then on

---

$$Q^{\pi}(s_t, a_t) = c(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim \mathcal{T}(s_t, a_t)} V^{\pi}(s_{t+1})$$

Temporal Difference (TD) learning

Goal: Learn $V^{\pi}(s)$ from traces

$$\left(s_t, a_t, c_t, s_{t+1}\right) \qquad \left(s_t, a_t, c_t, s_{t+1}\right) \qquad \left(s_t, a_t, c_t, s_{t+1}\right) \qquad \left(s_t, a_t, c_t, s_{t+1}\right)$$

Recall value function $V^{\pi}(s)$ satisfies

$$V^{\pi}(s) = c(s, \pi(s)) + \gamma \mathbb{E}_{s'} V^{\pi}(s')$$

TD Idea: Update value using estimate of next state value

$$V(s_t) \leftarrow V(s_t) + \alpha \left(\underbrace{c_t + \gamma V(s_{t+1}) - V(s_t)}_{\text{Temporal Difference Error}} \right)$$

TD Learning

For every $\left(s_t, a_t, c_t, s_{t+1}\right)$

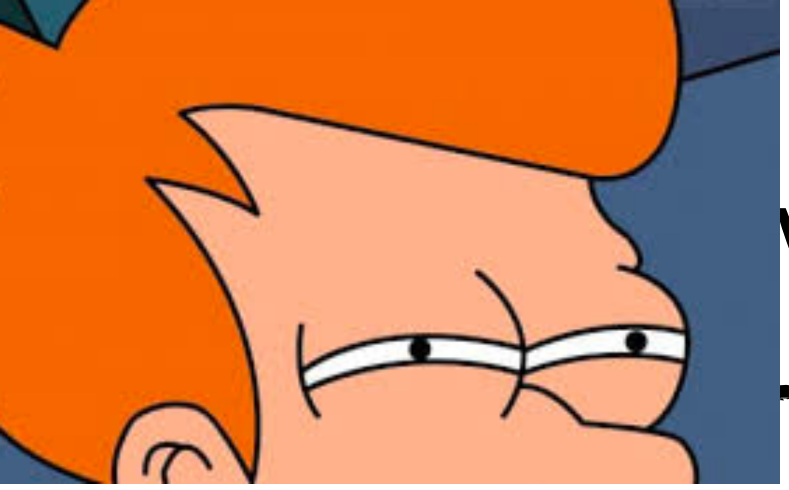
$$V(s_t) \leftarrow V(s_t) + \alpha(c_t + \gamma V(s_{t+1}) - V(s_t))$$

Did you spot the trick?

$$V^{\pi}(s) = c(s, \pi(s)) + \gamma \mathbb{E}_{s'} V^{\pi}(s')$$

$$\Updownarrow$$

$$V(s_t) \leftarrow V(s_t) + \alpha(c_t + \gamma V(s_{t+1}) - V(s_t))$$



Monte-Carlo

Temporal Difference

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

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

Monte-Carlo	Temporal Difference
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

We have been talking about trying to learn the value of a given policy π

$V^{\pi}(s)$ / $Q^{\pi}(s,a)$

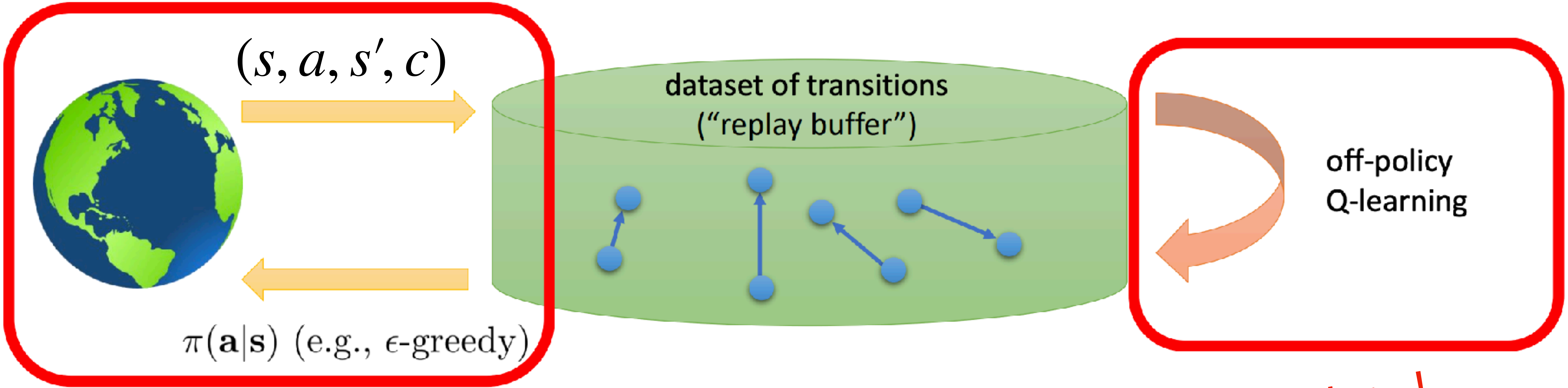
What if we wanted to learn the optimal value function

$V^{*}(s)$ / $Q^{*}(s,a)$

QT-Opt: Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation


Training time
Distributed RL
-
=
Reward: Grasp success determined by subtracting pre and post-drop images
State, Action, Reward
Learned weights
Inference time
State: 472x472 Image and gripper aperture
Camera
Robot
Critic Function
Q(State, Action)
Action proposals
Q-Values
Action: Gripper displacement and aperture
Cross-Entropy Method
arg max Q(State, Action)
Action

Q-learning: Learning off-policy

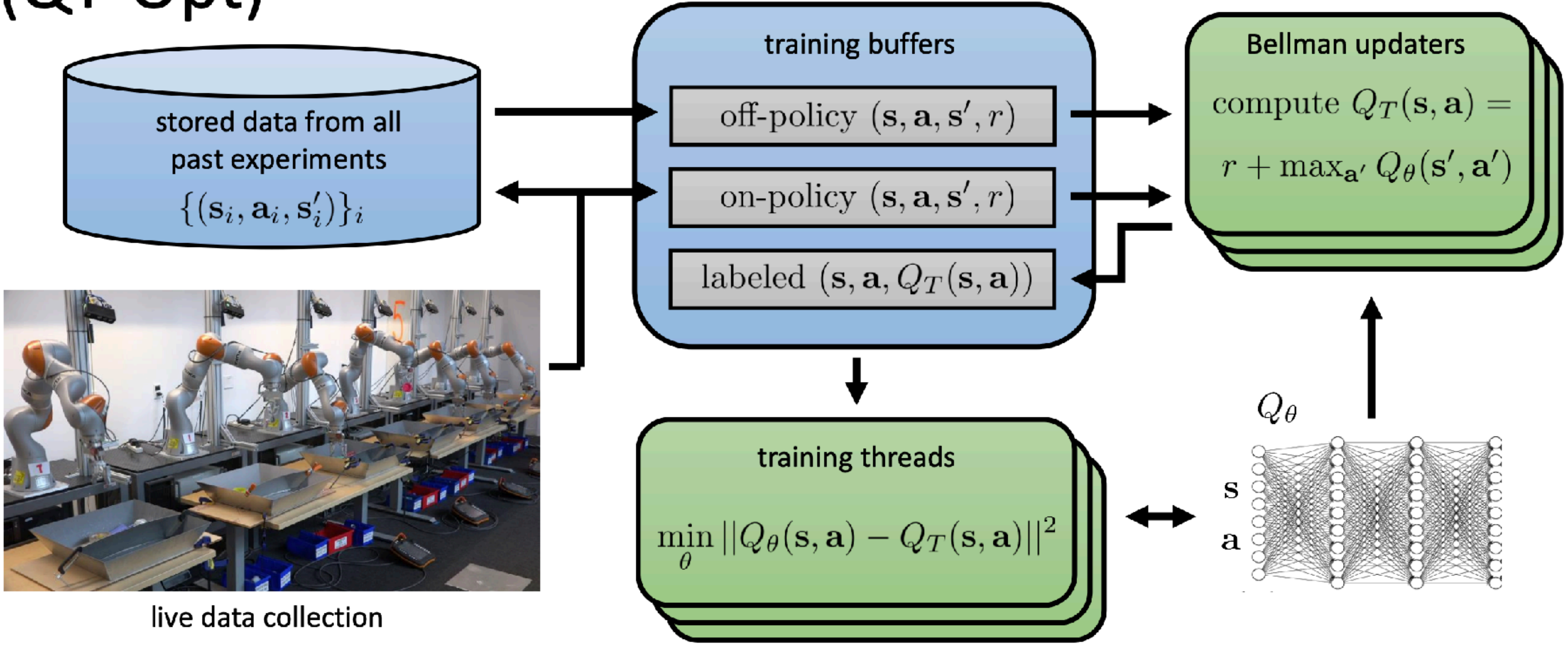


Can learn from any data!

For every (s_t, a_t, c_t, s_{t+1})

$$Q^*(s_t, a_t) = Q^*(s_t, a_t) + \alpha(c(s_t, a_t) + \gamma \min_{a'} Q^*(s_{t+1}, a') - Q^*(s_t, a_t))$$

Large-scale Q-learning with continuous actions (QT-Opt)



Kalashnikov, Irpan, Pastor, Ibarz, Herzong, Jang, Quillen, Holly, Kalakrishnan, Vanhoucke, Levine. **QT-Opt: Scalable Deep Reinforcement Learning of Vision-Based Robotic Manipulation Skills**

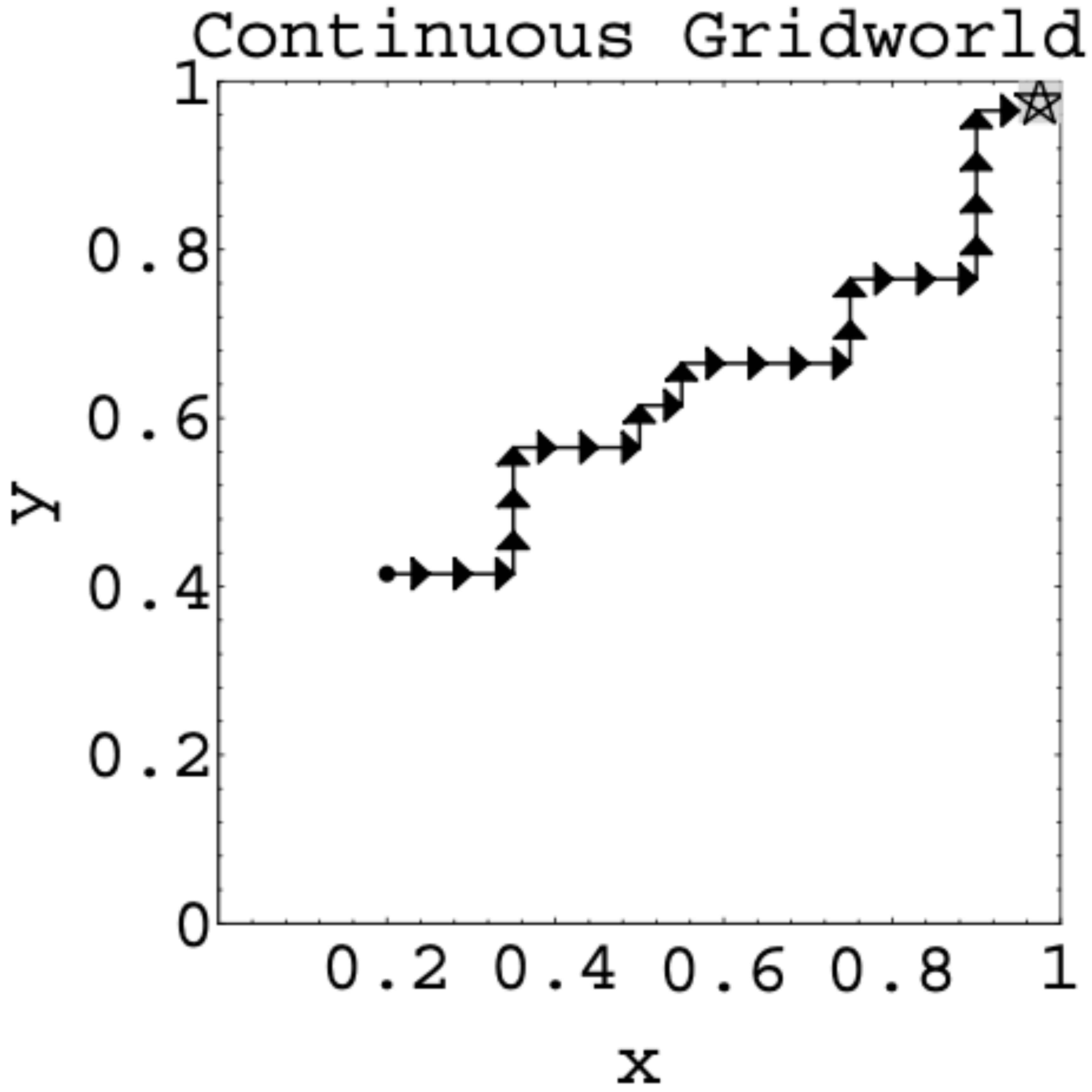
Is this ... magic?

We just learned in IL how distribution shift is a big deal ...

It's not magic. Q-learning relies on a set of assumptions:

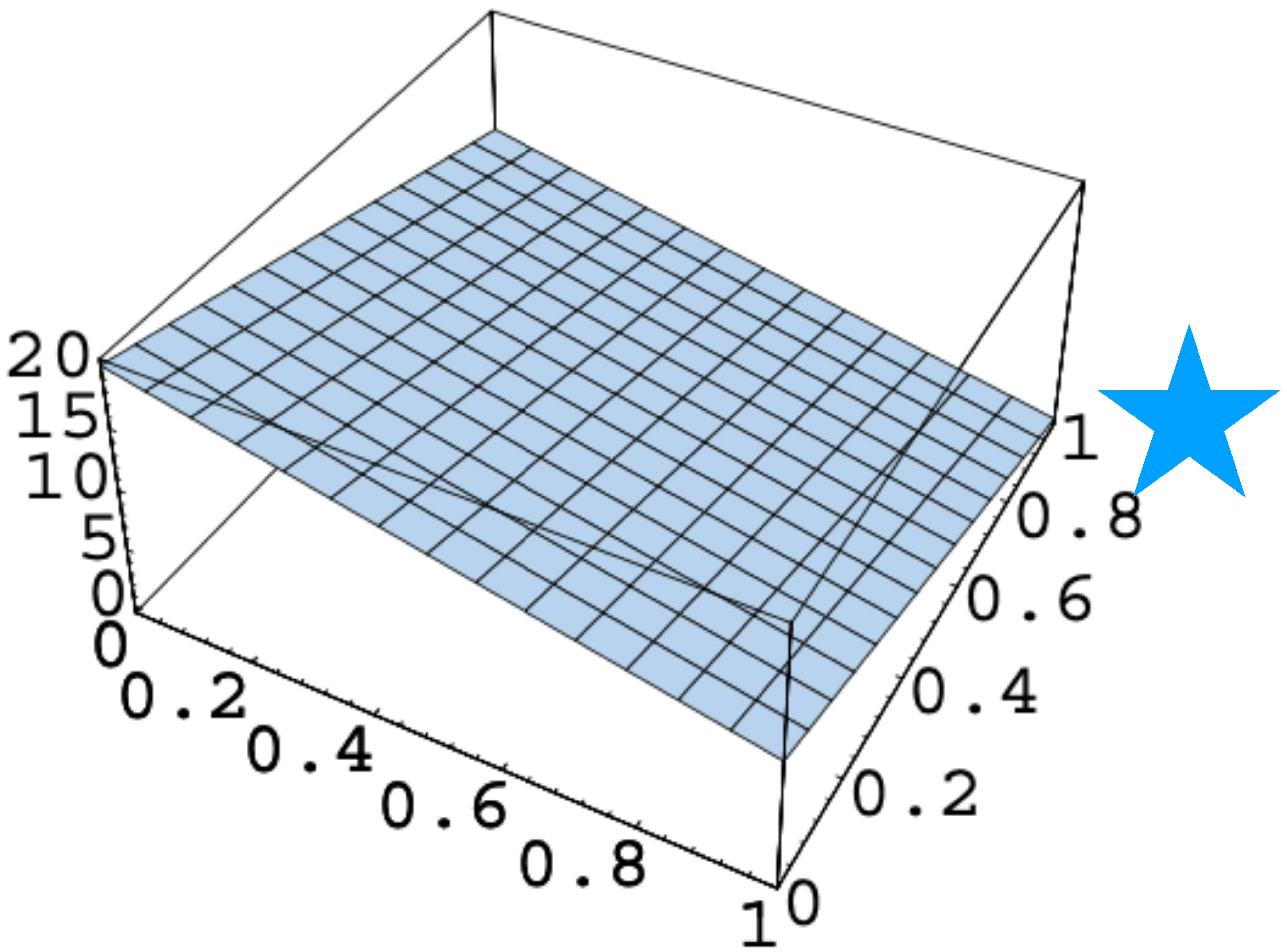
1. Each state-action is visited *infinite* times
2. Learning rate α must be annealed over time

When things fail!



Optimal path

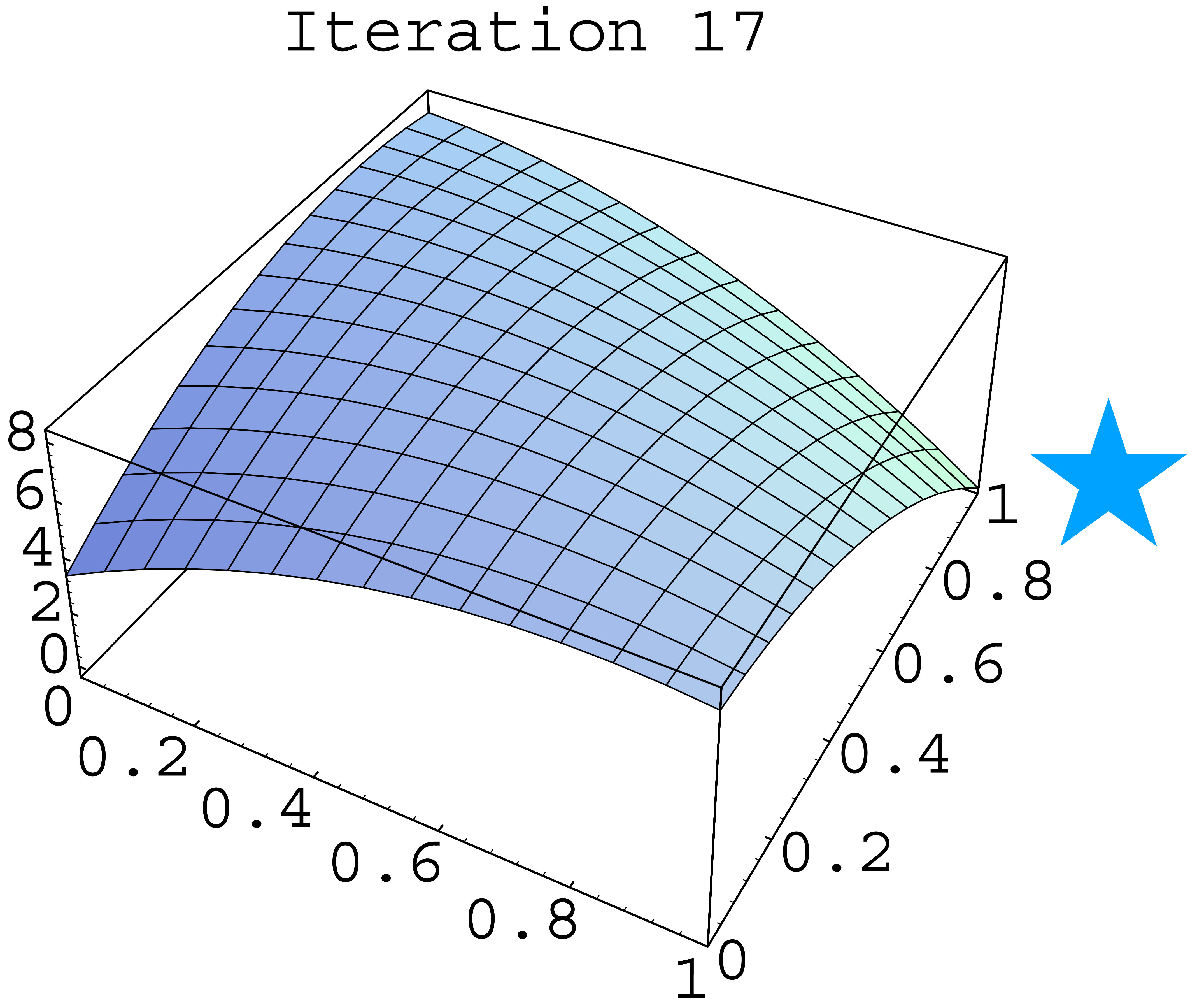
$V^*(x, y)$

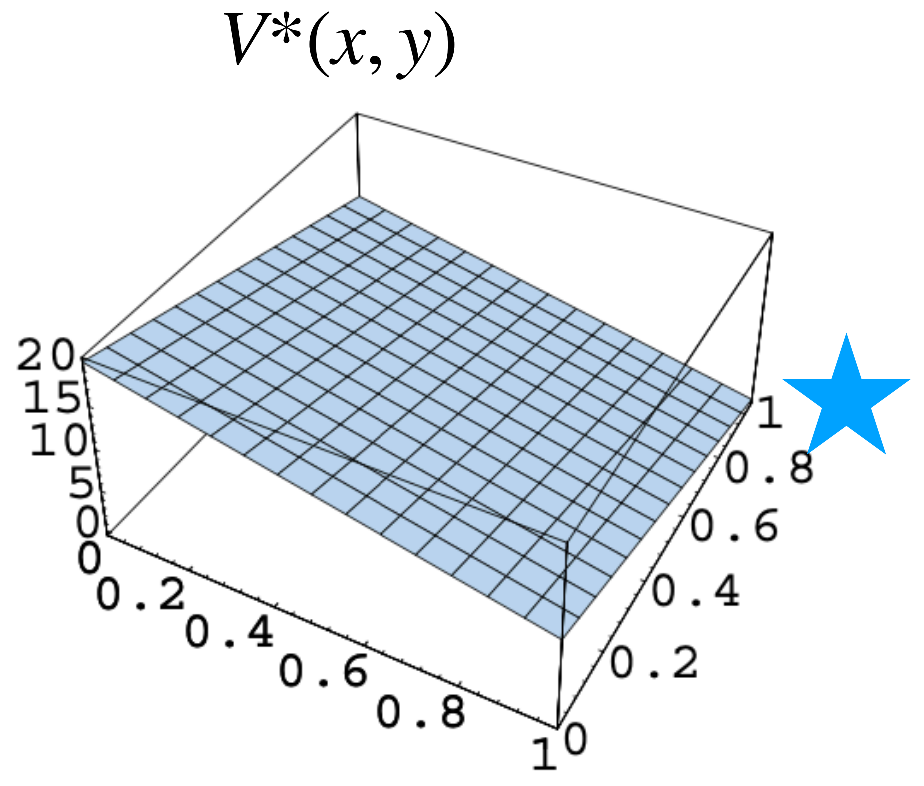


True value function

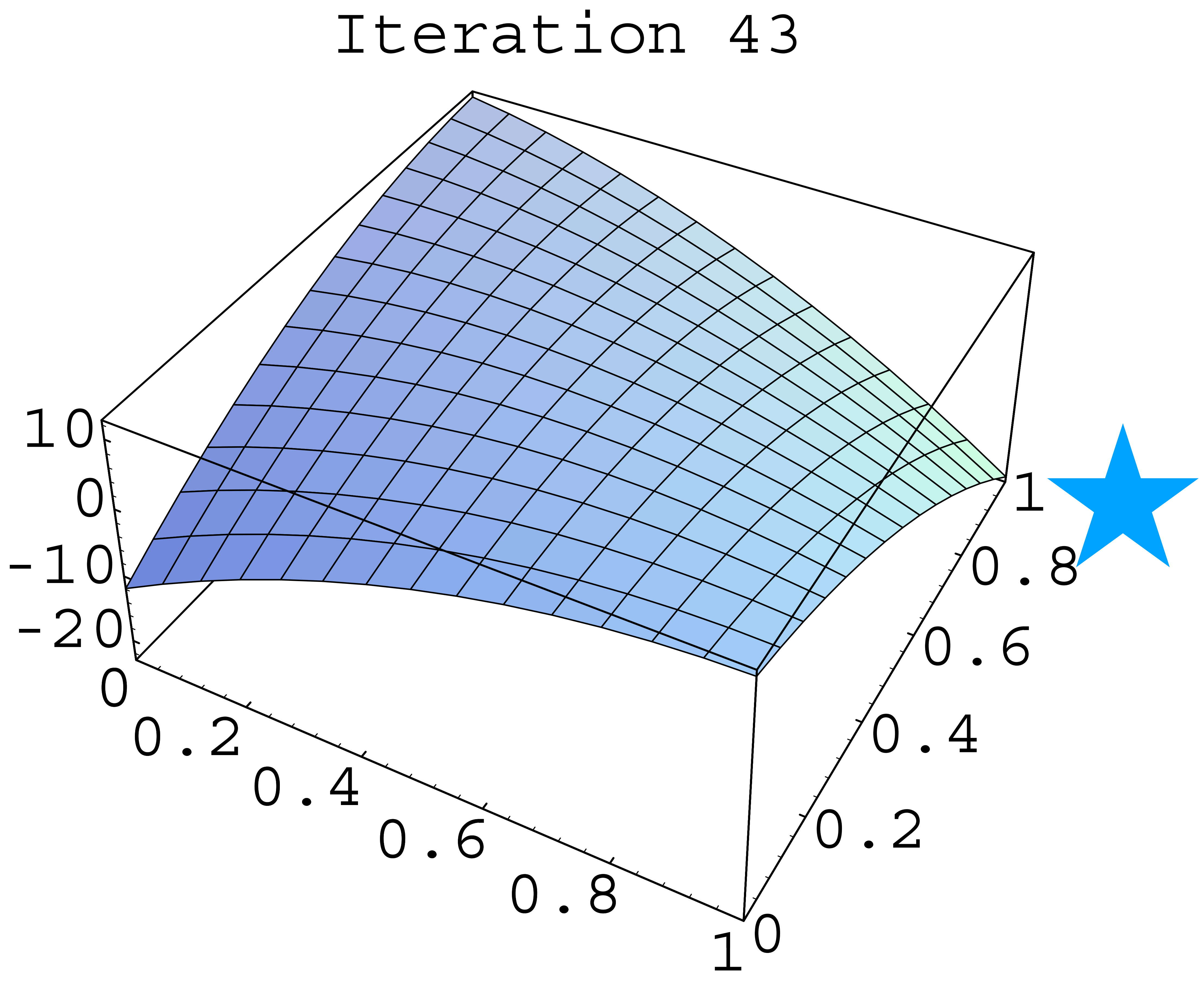
Boyan,Justin A and Moore, Andrew W, Generalization in Reinforcement Learning: Safely Approximating the Value Function. NeurIPS 1994

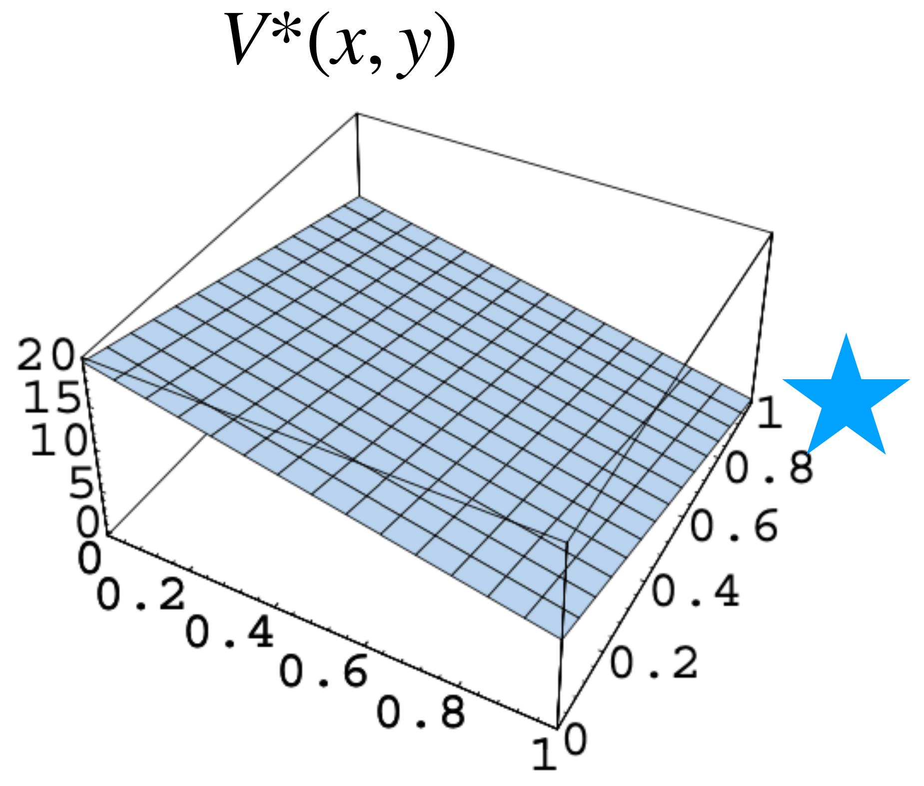
What happens when we run value iteration with a *quadratic*?



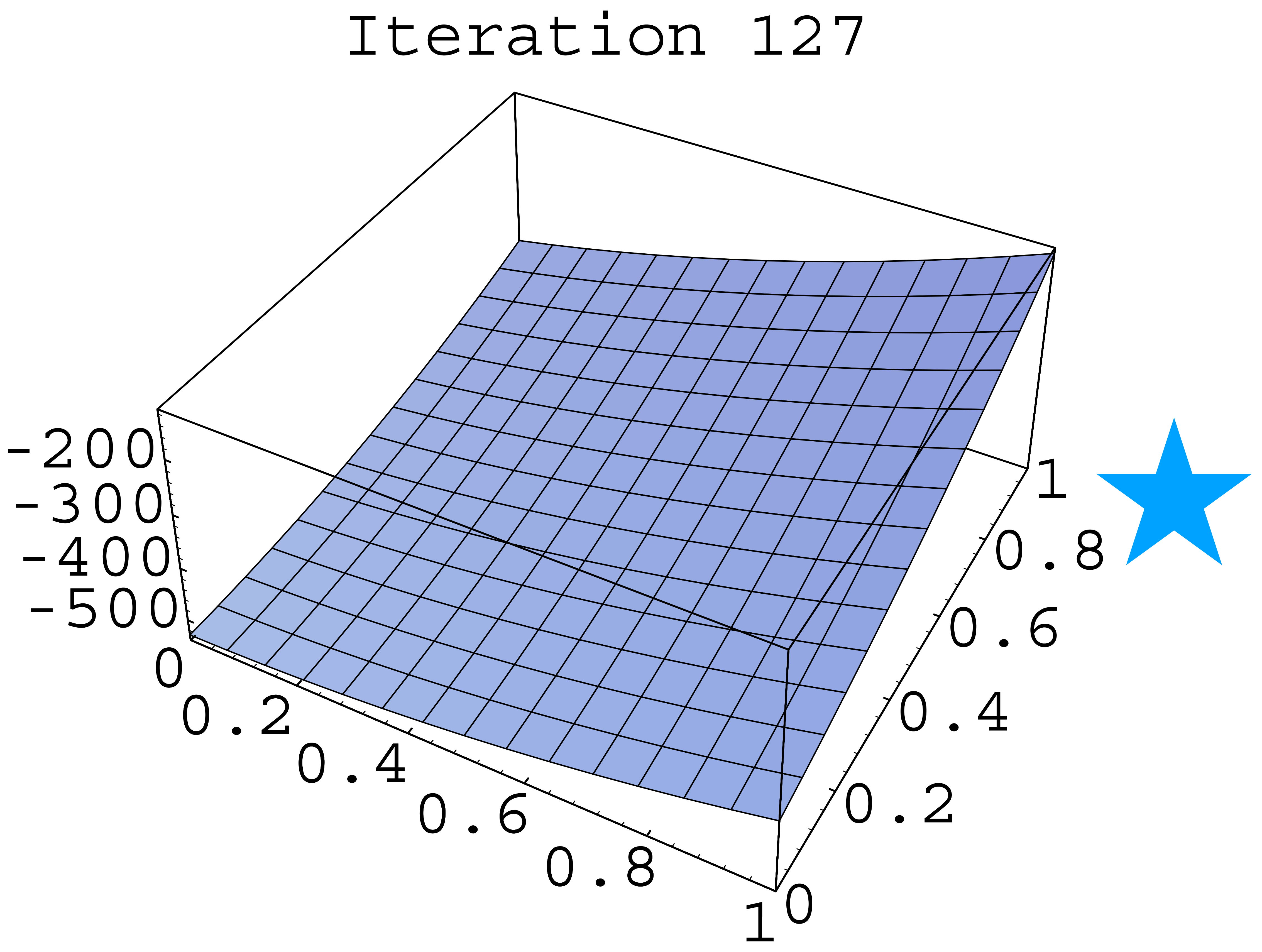


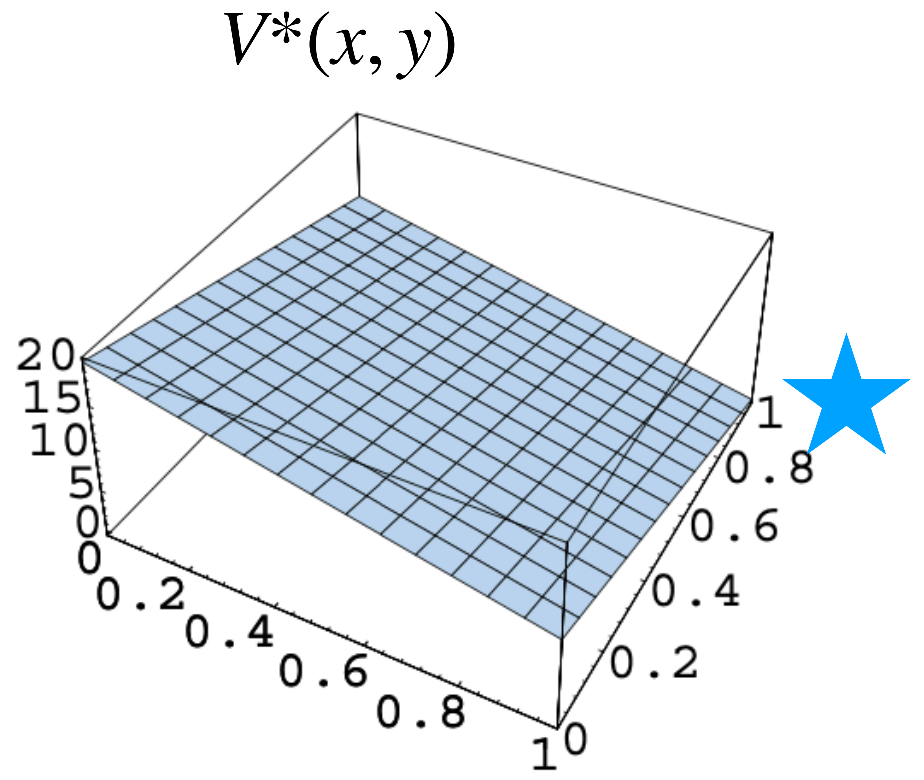
What happens when we run value iteration with a *quadratic*?





What happens when we run value iteration with a *quadratic*?

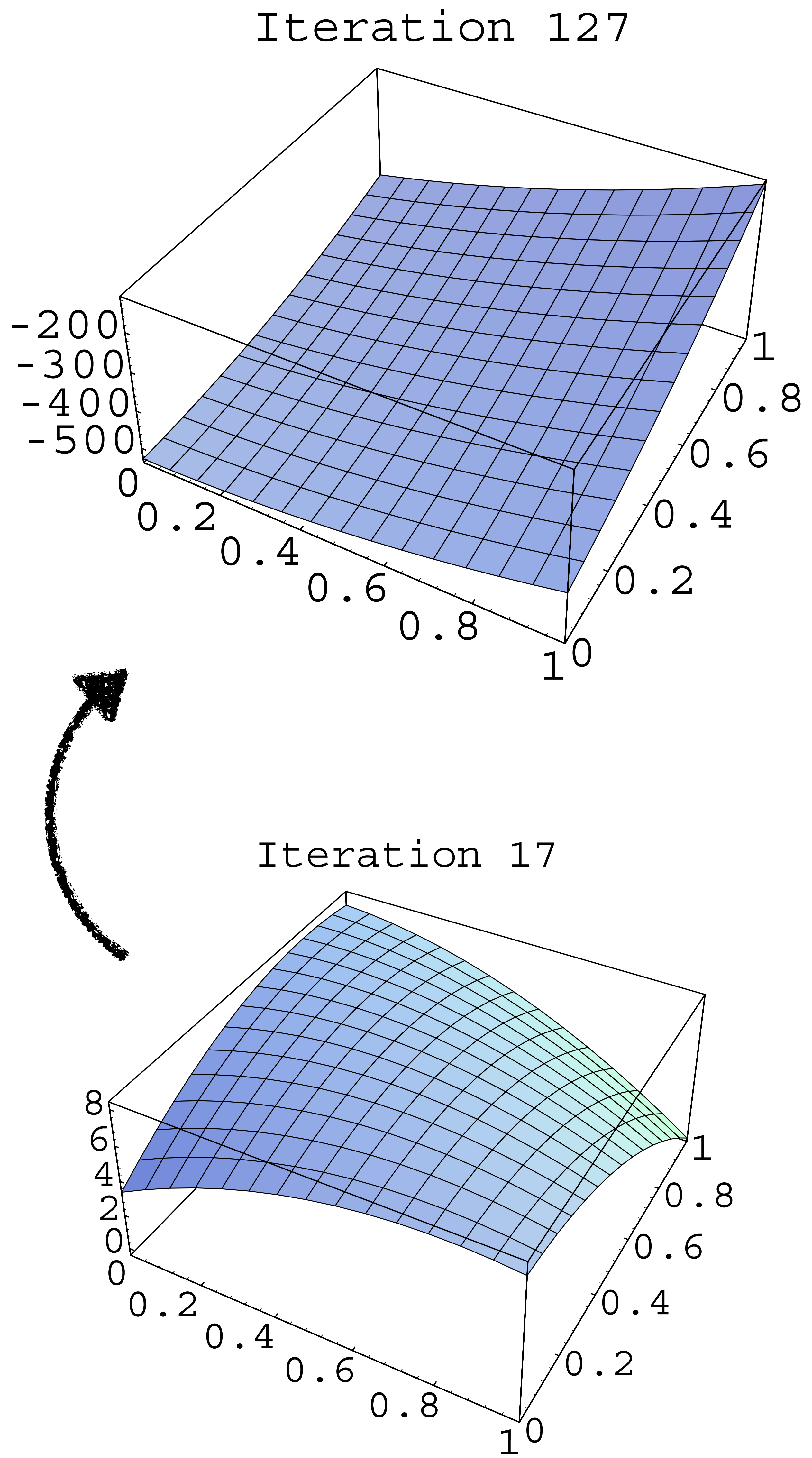




The problem of Bootstrapping!

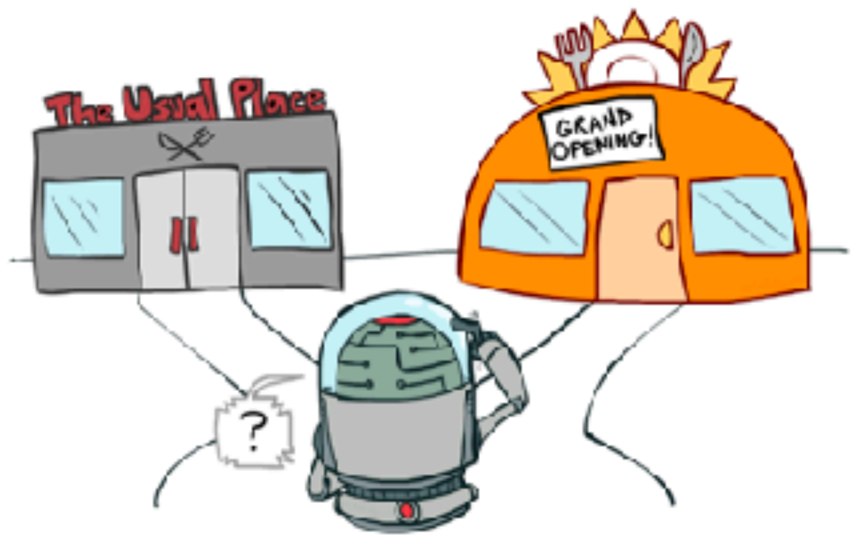
$$Q^*(s_t, a_t) = Q^*(s_t, a_t) + \alpha(c(s_t, a_t) + \gamma \min_{a'} Q^*(s_{t+1}, a') - Q^*(s_t, a_t))$$

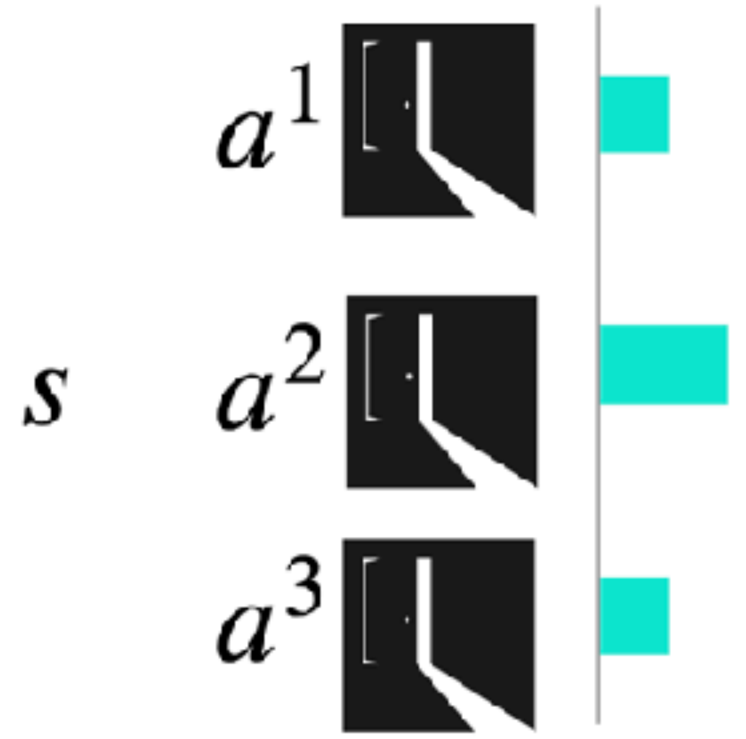
min()



tl;dr

Two Ingredients of RL

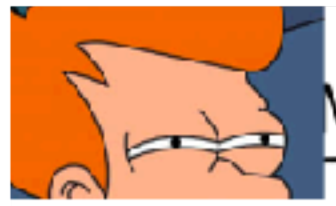




Exploration Exploitation

Estimate Values $Q(s, a)$

2



Monte-Carlo

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

Zero Bias

High Variance

Always convergence
(Just have to wait till heat death of the universe)

Temporal Difference

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

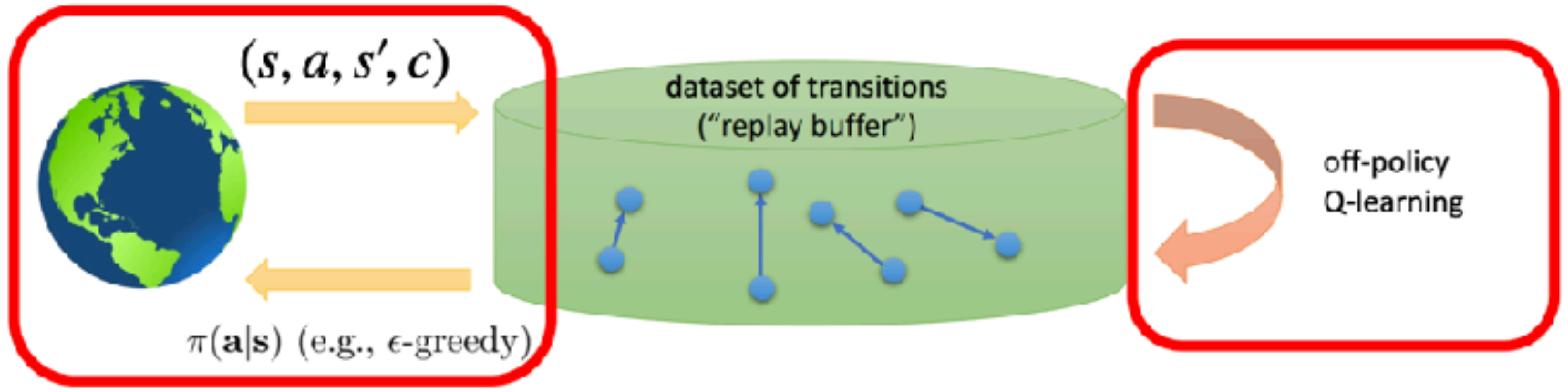
Can have bias

Low Variance

May *not* converge if using function approximation

38

Q-learning: Learning off-policy



For every (s_t, a_t, c_t, s_{t+1})

$$Q^*(s_t, a_t) = Q^*(s_t, a_t) + \alpha(c(s_t, a_t) + \gamma \min_{a'} Q^*(s_{t+1}, a') - Q^*(s_t, a_t))$$

41