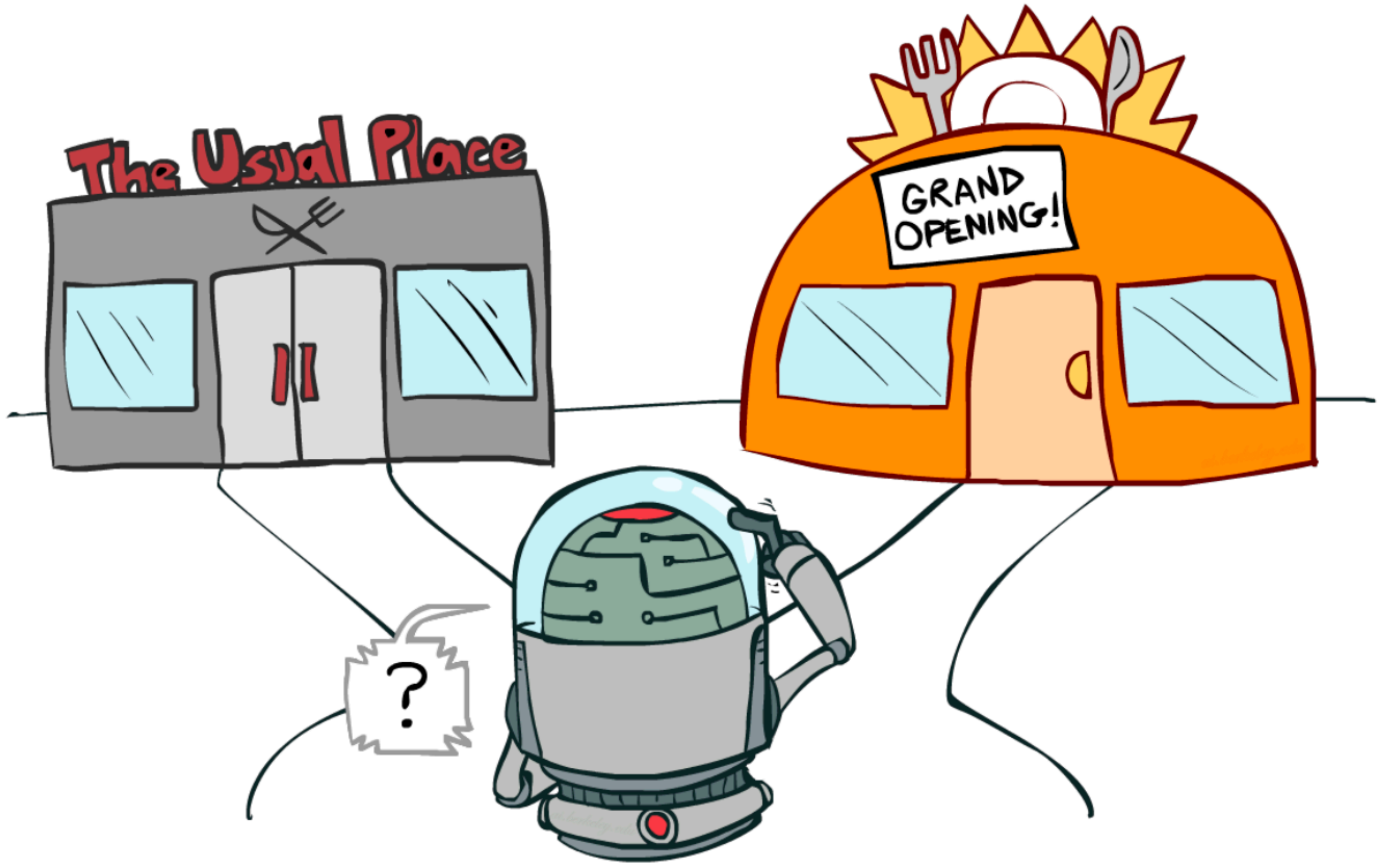
Temporal Difference Learning

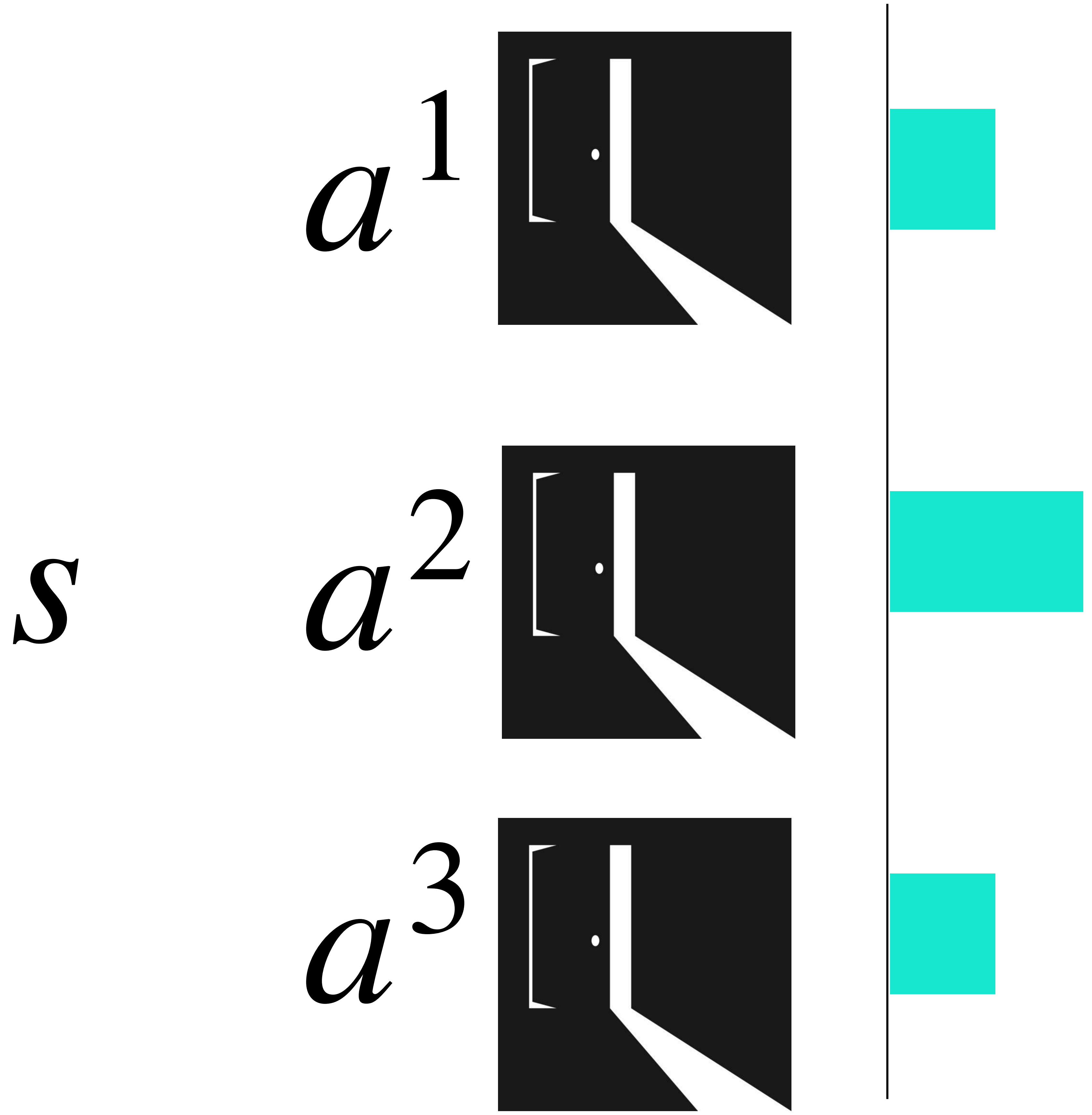
Sanjiban Choudhury

Cornell Bowers CIS
Computer Science

Two Ingredients of RL



Exploration Exploitation

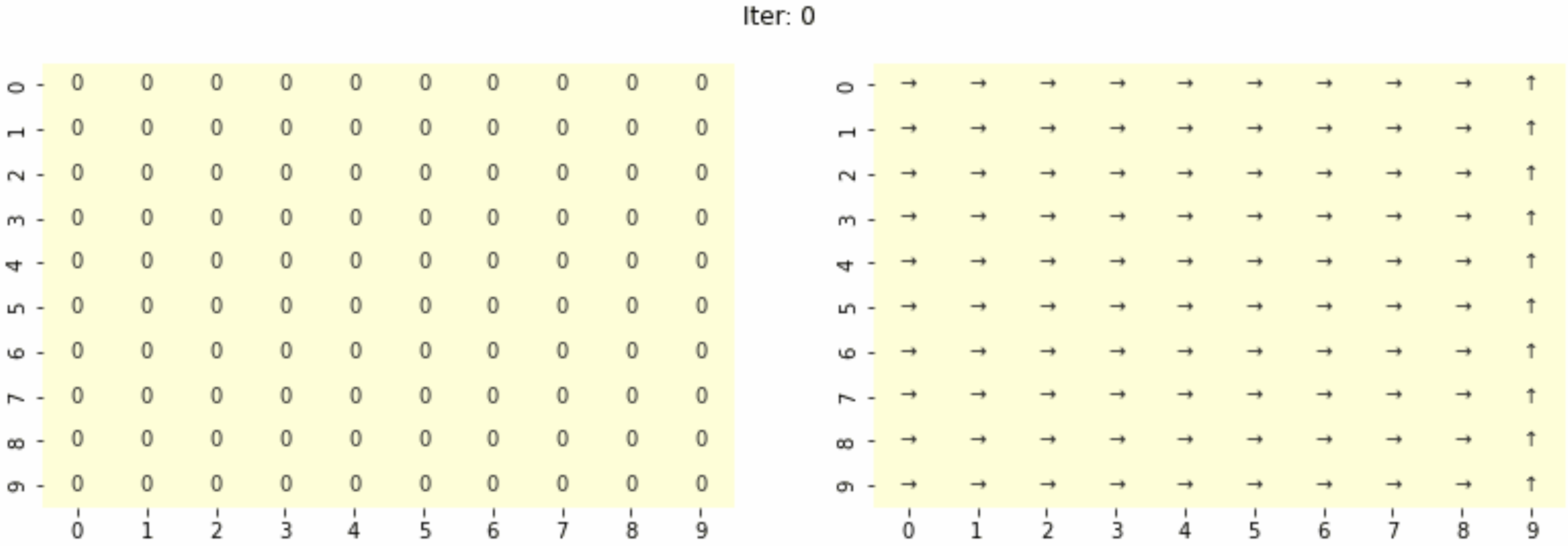


Estimate Values $Q(s, a)$

When the MDP is known!

Run Value / Policy Iteration

When MDP is known: Policy Iteration



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

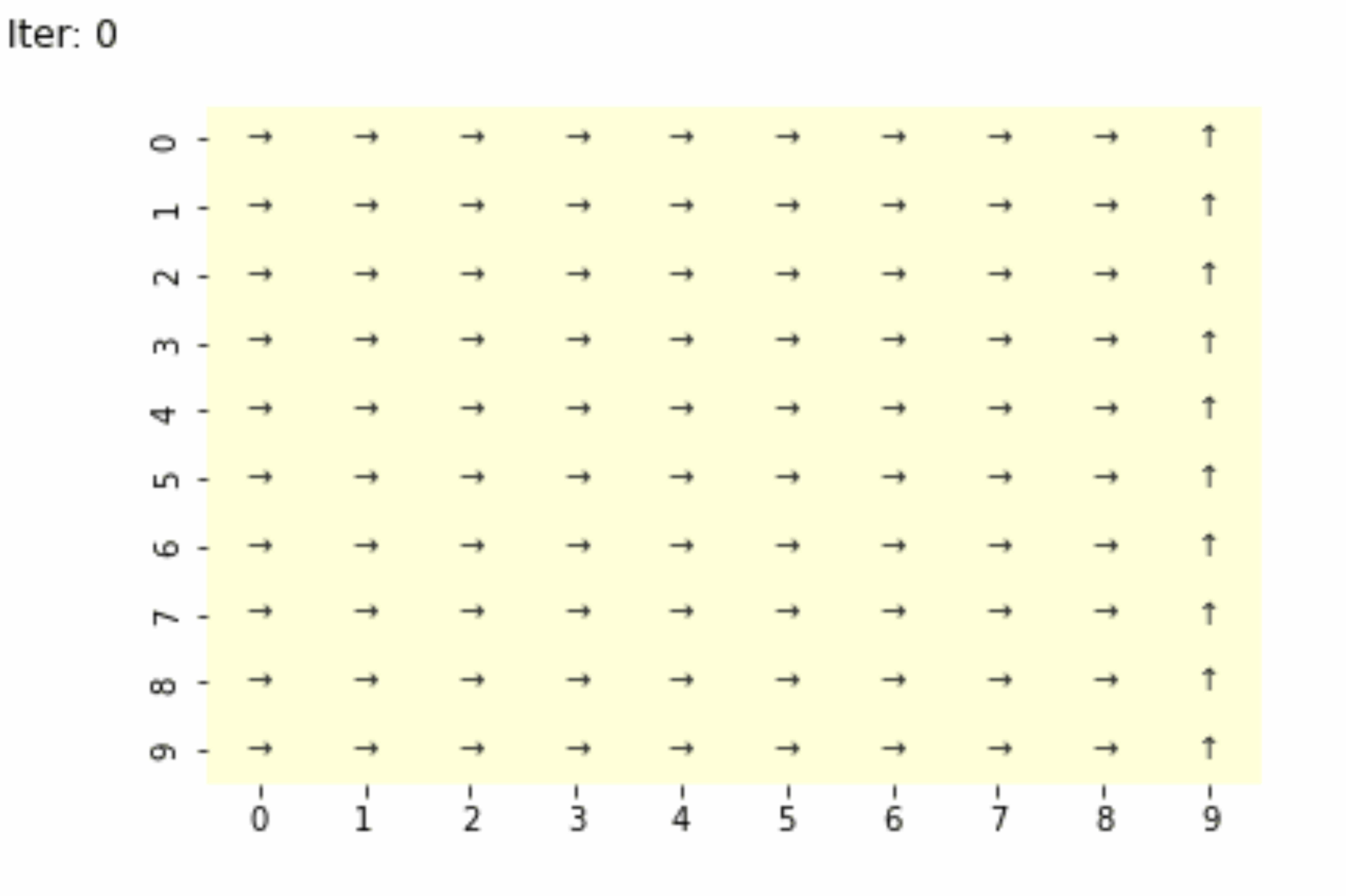
Estimate value

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

Improve policy

What happens when the MDP is *unknown?*

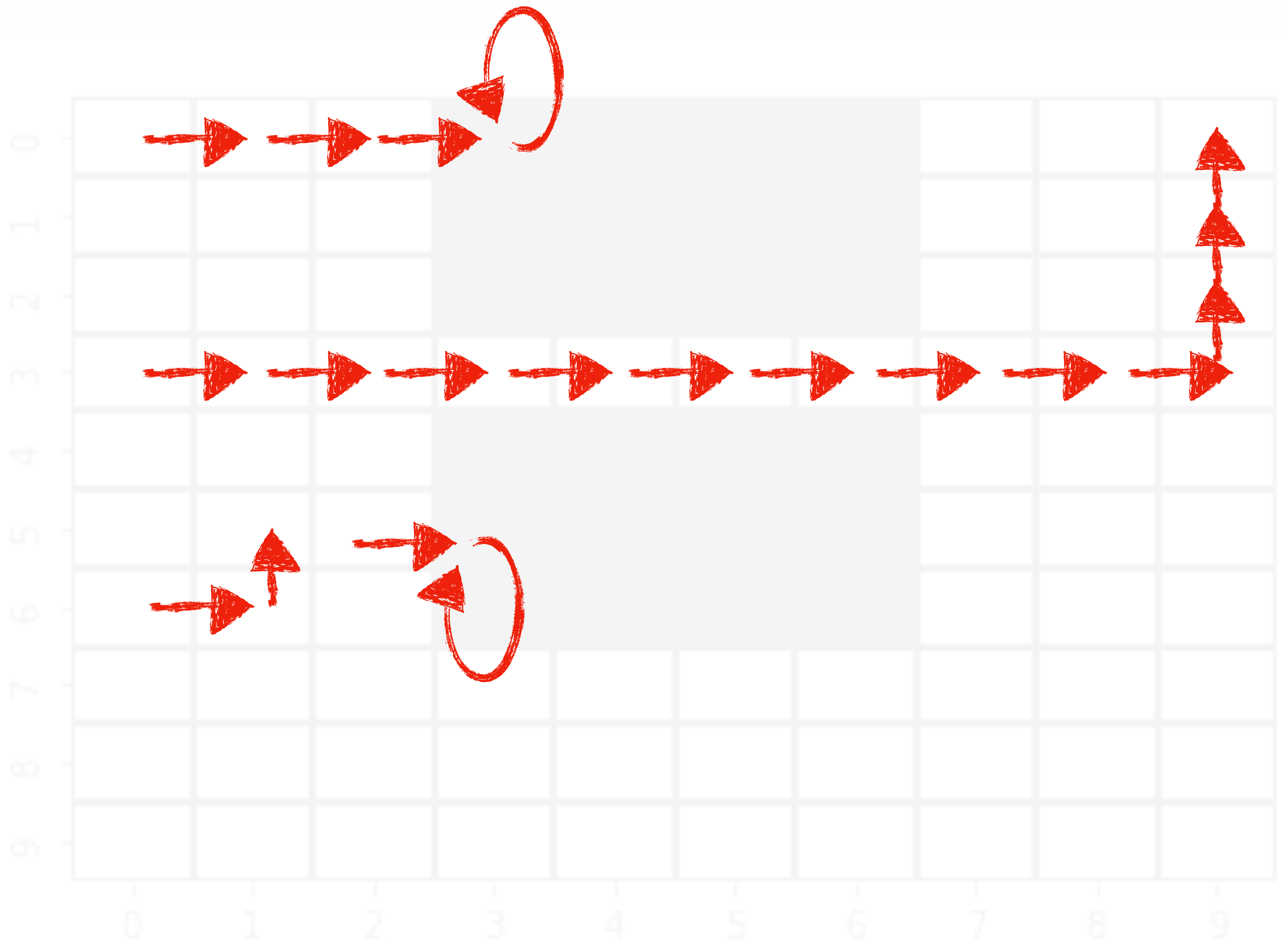
Need to *estimate the value* of policy



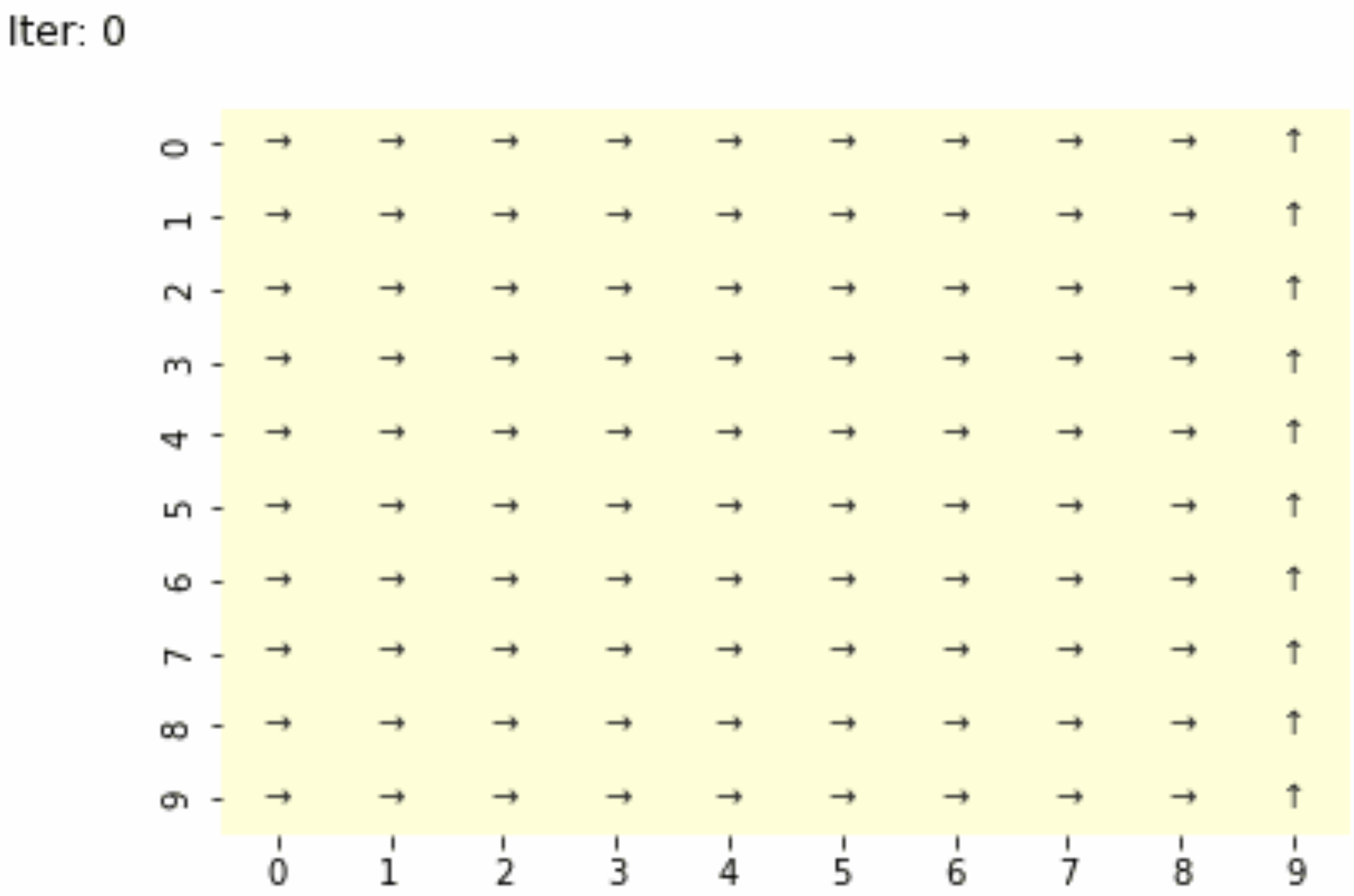
Value $V^{\pi}(s)$

Policy π

Estimate the value of policy from sample rollouts

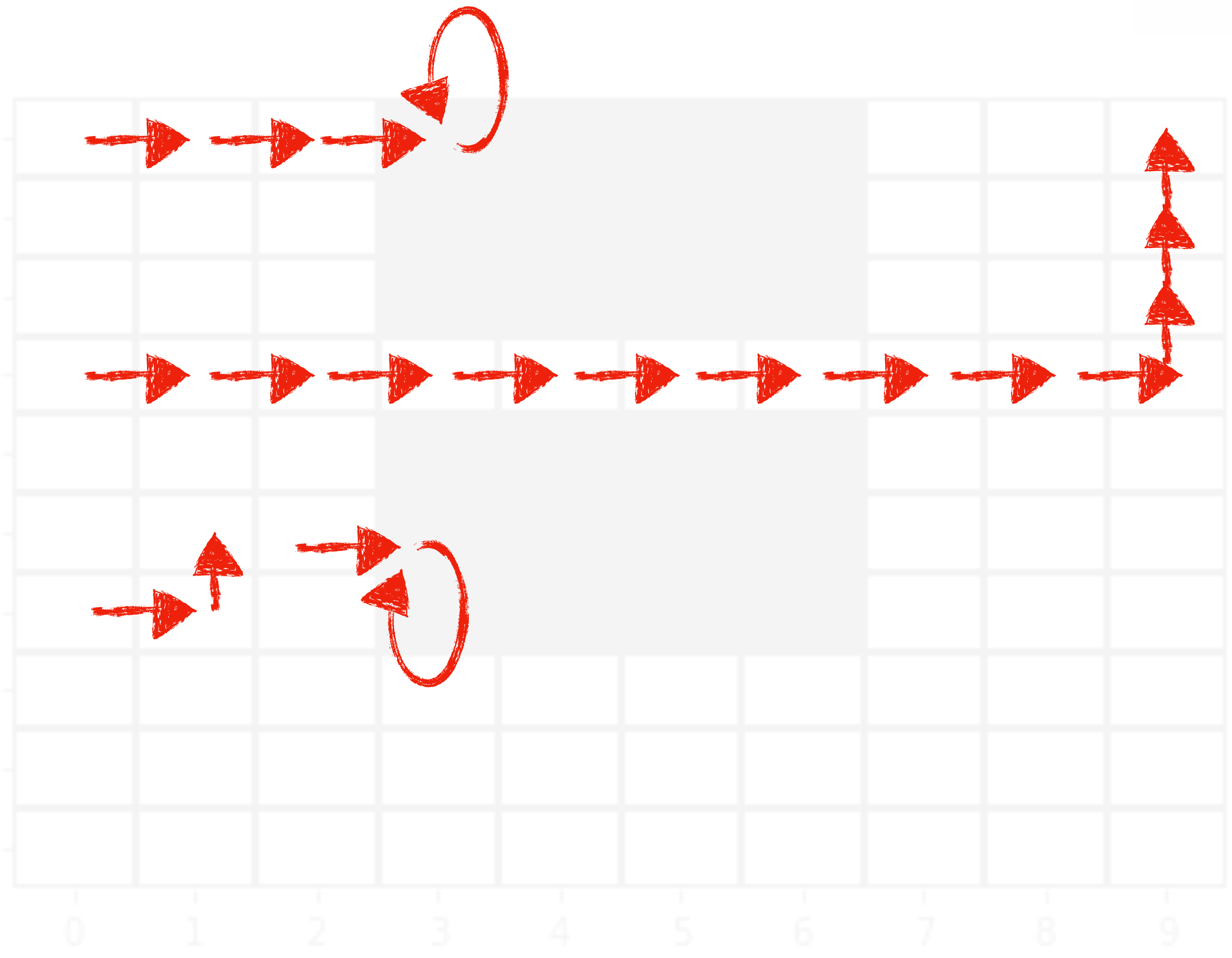


Roll outs

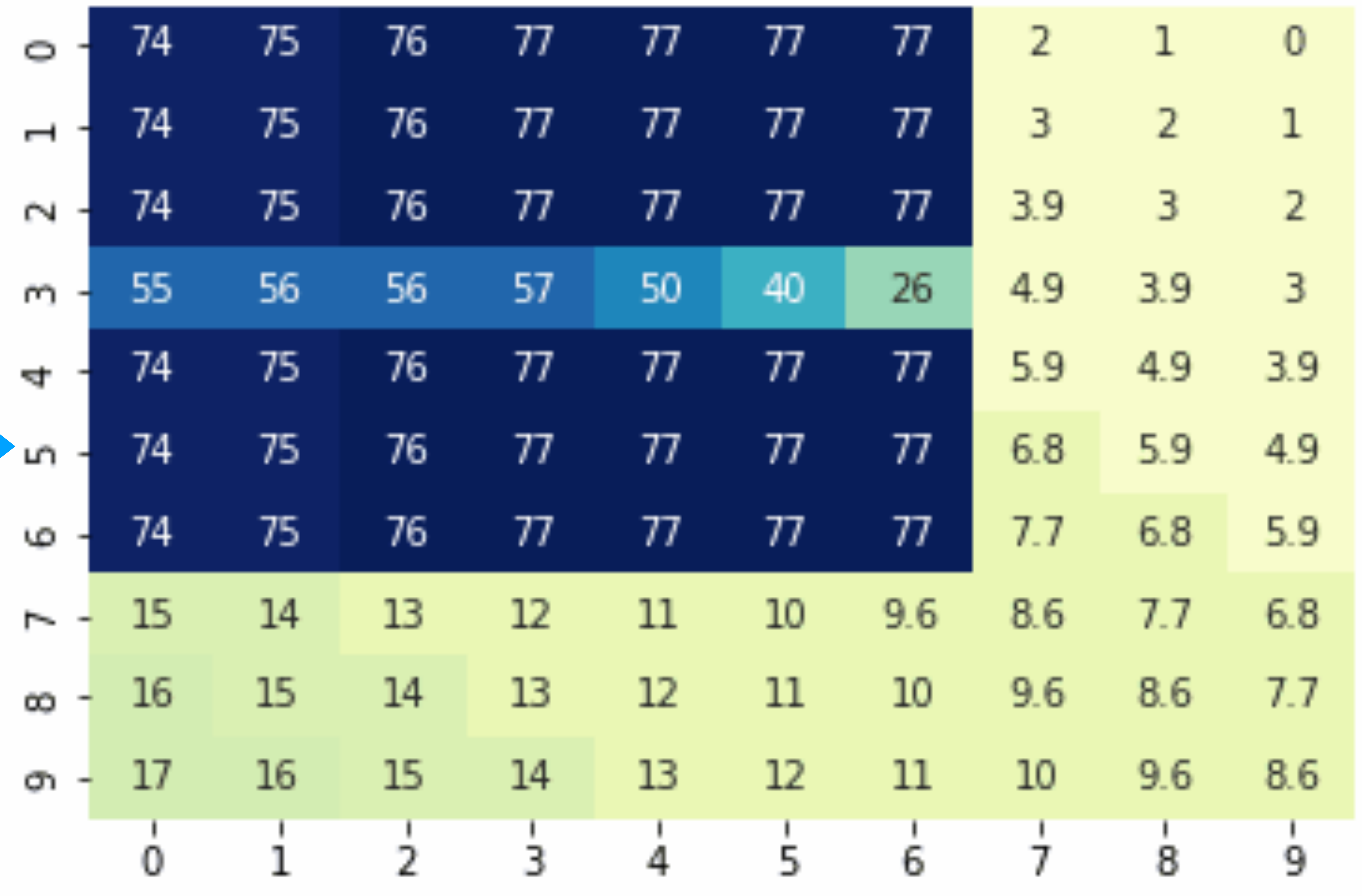


Policy π

Estimate the value of policy from sample rollouts



Roll outs



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

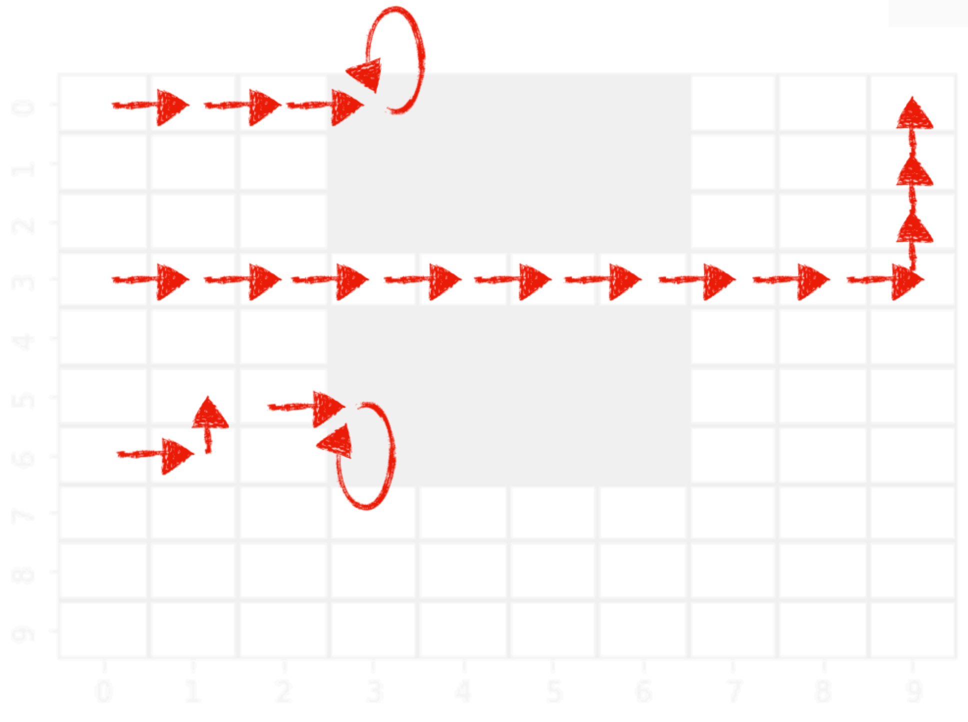
Activity!

Think-Pair-Share

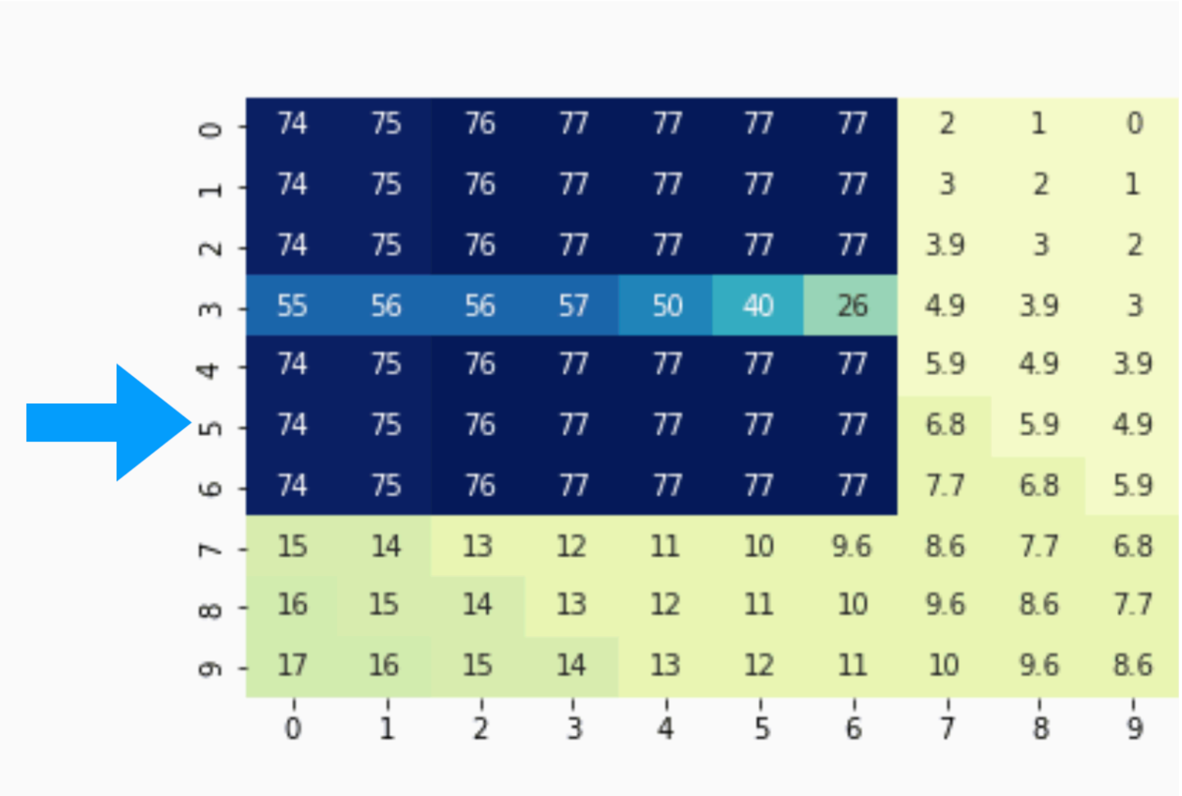
Think (30 sec): Given a bunch of roll-outs, how can you estimate value of a state? (Hint: More than one way!)

Pair: Find a partner

Share (45 sec): Partners exchange ideas



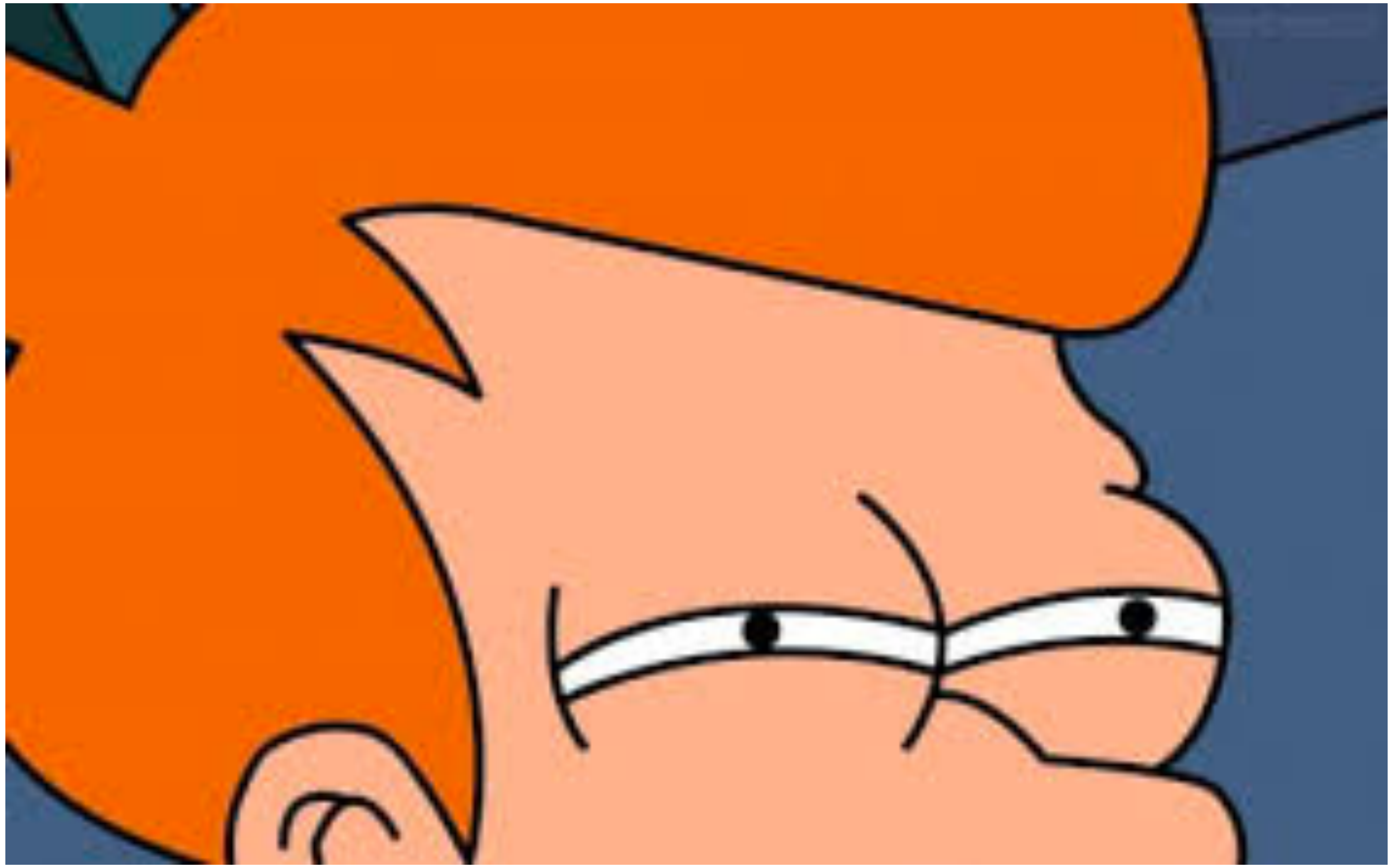
Roll outs



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

7

Option 1: Just execute the damn policy!



and look at the returns ..

Monte Carlo Evaluation

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

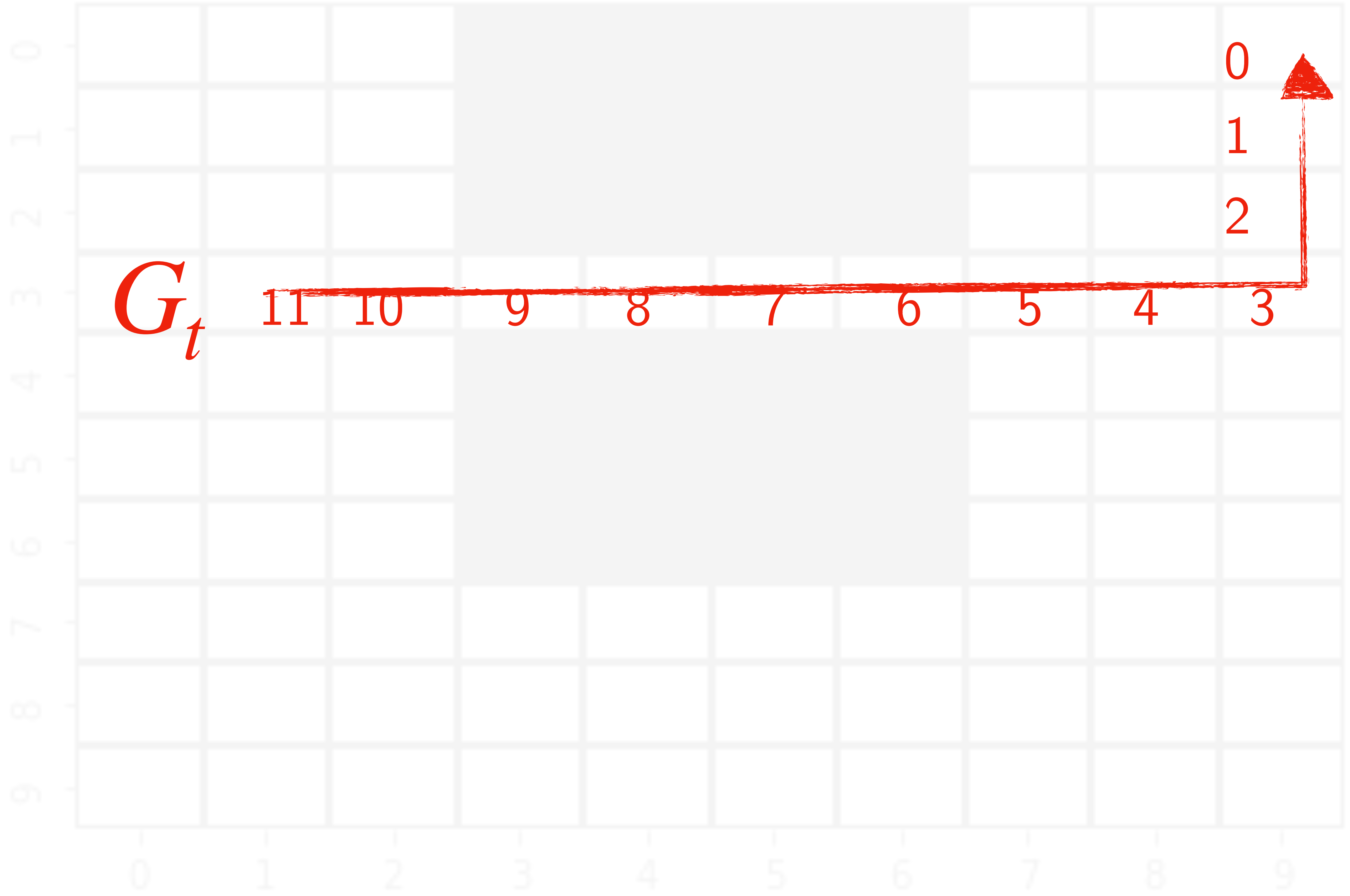
Define: *Return* is the total discounted cost

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

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

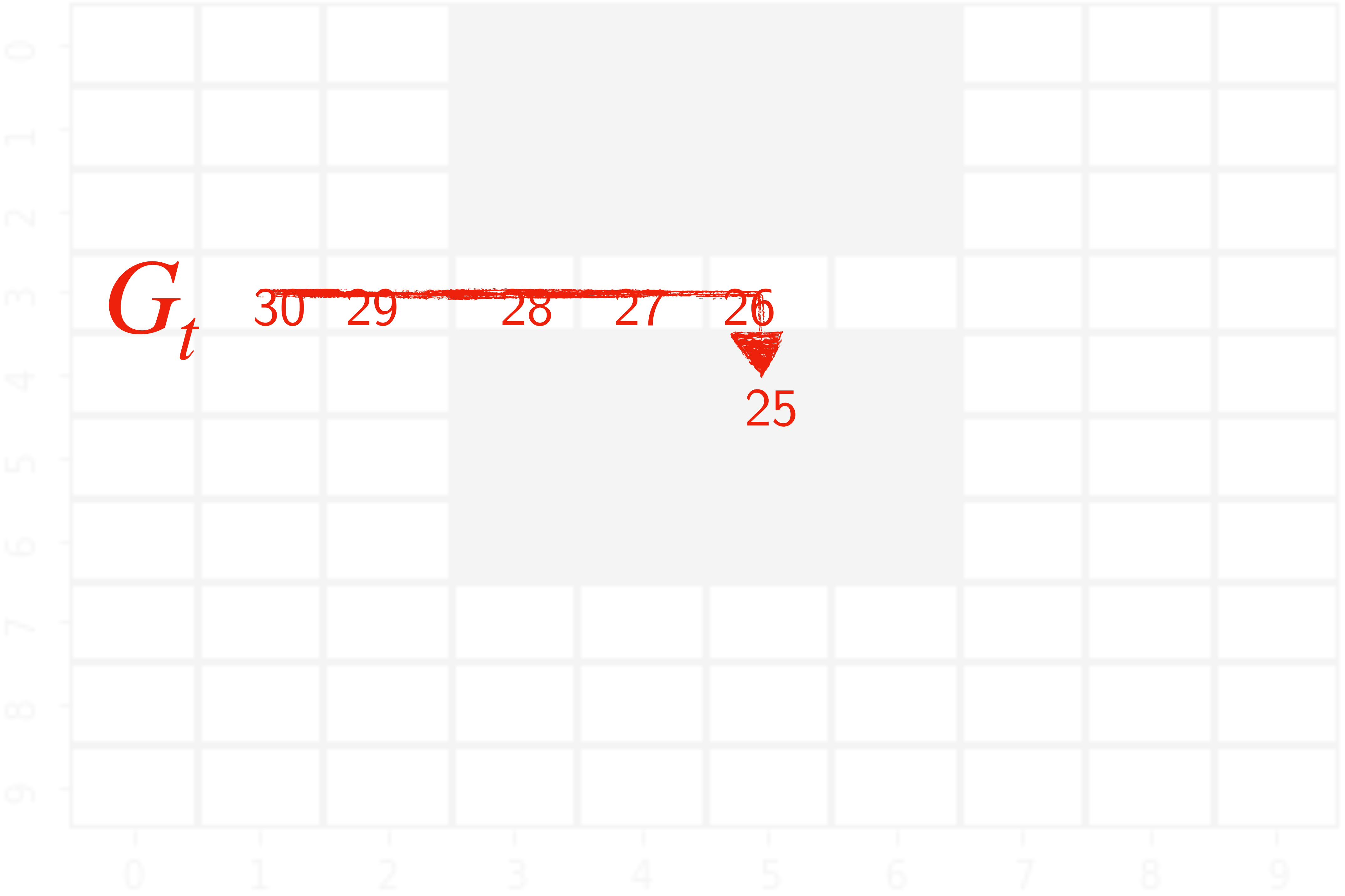
Increment counter $N(s) \leftarrow N(s) + 1$

Increment total return
$S(s) \leftarrow S(s) + G_t$

Update $V(s) = S(s)/N(s)$

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

First Visit Monte Carlo



For episode in rollouts:

If state s is visited for *first* time t

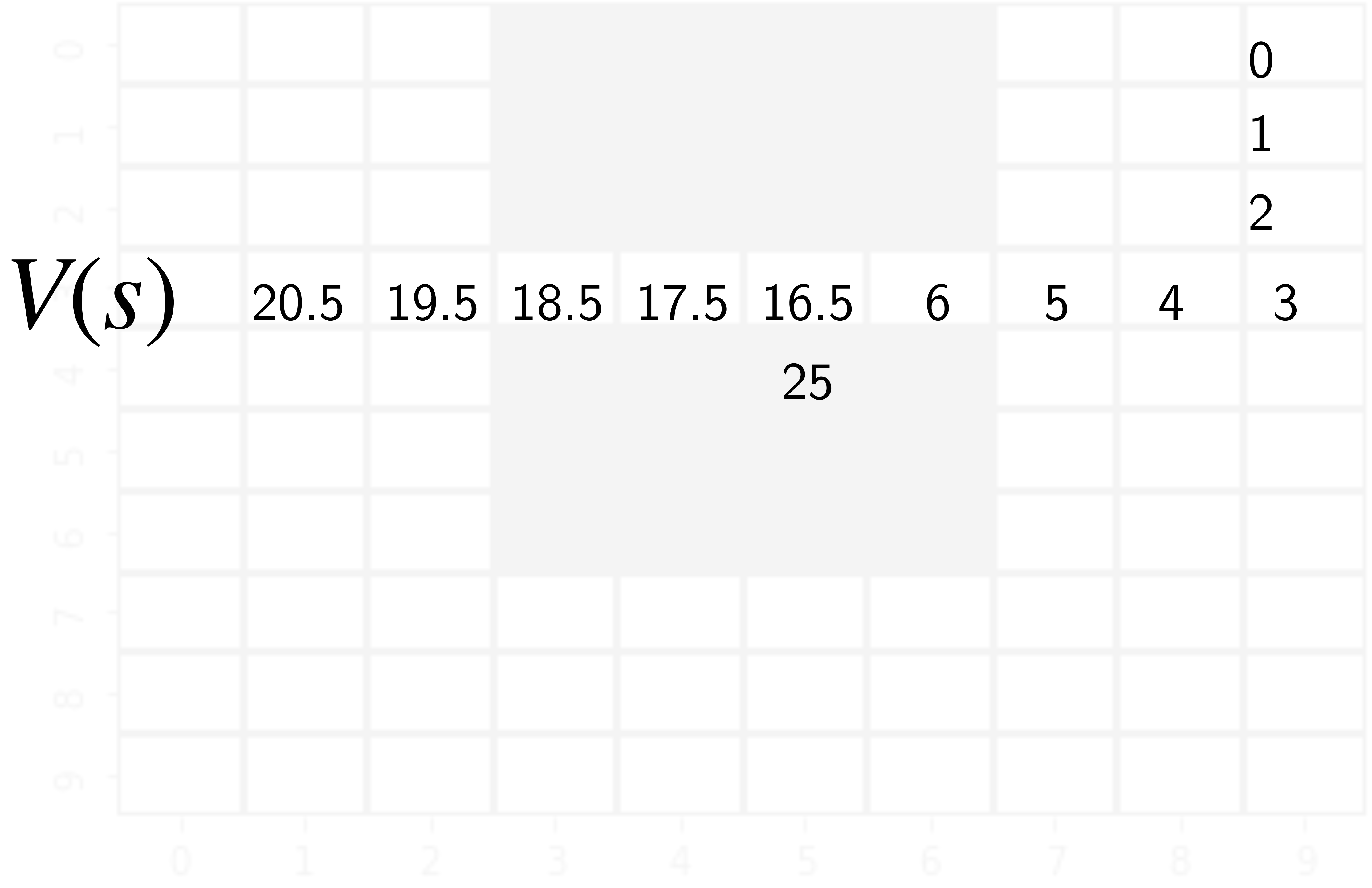
Increment counter $N(s) \leftarrow N(s) + 1$

Increment total return
$S(s) \leftarrow S(s) + G_t$

Update $V(s) = S(s)/N(s)$

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

First Visit Monte Carlo



For episode in rollouts:

If state s is visited for *first* time t

Increment counter $N(s) \leftarrow N(s) + 1$

Increment total return $S(s) \leftarrow S(s) + G_t$

Update $V(s) = S(s)/N(s)$

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

Can we incrementally update the value V(s)?

Exponential Moving Average!

For episode in rollouts:

If state s is visited for *first* time t

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

Estimation error

Facts about Monte Carlo

- MC methods learn directly from episodes of experience
- MC is *model-free*: no knowledge of MDP transitions / rewards
- MC learns from *complete* episodes: no bootstrapping
- MC uses the simplest possible idea: value = mean return
- Caveat: can only apply MC to *episodic* MDPs
    - All episodes must terminate

From David Silver

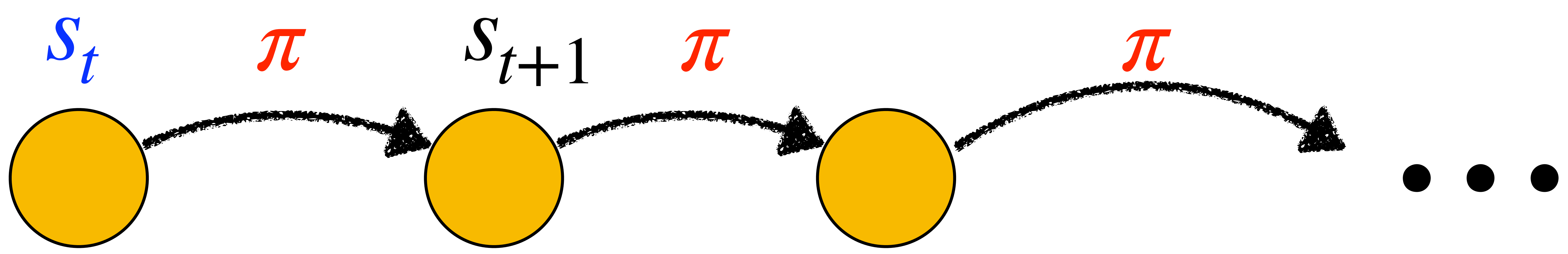
Can we do better than Monte Carlo?

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

What if we don't have complete episodes?

Option 2: Trust your value estimate

Value of a state



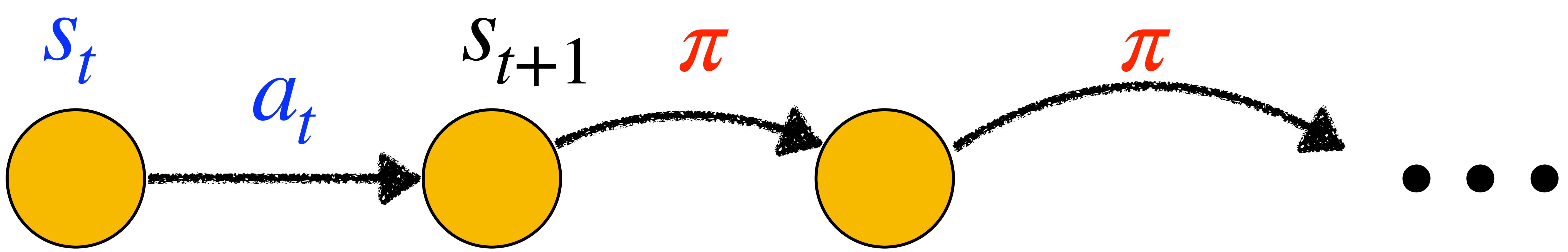
$$V^{\pi}(s_t) = c_t + \gamma c_{t+1} + \gamma^2 c_{t+2} + \dots$$

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

---

$$\pi^* = \arg\min_{\pi} \mathbb{E}_{s_0} V^{\pi}(s_0)$$

Value of a state-action



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

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

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

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

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

Temporal Difference Error

TD Learning

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

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

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

If you are interested in helping me make pretty grid world animations of MC, TD, Q-learning …

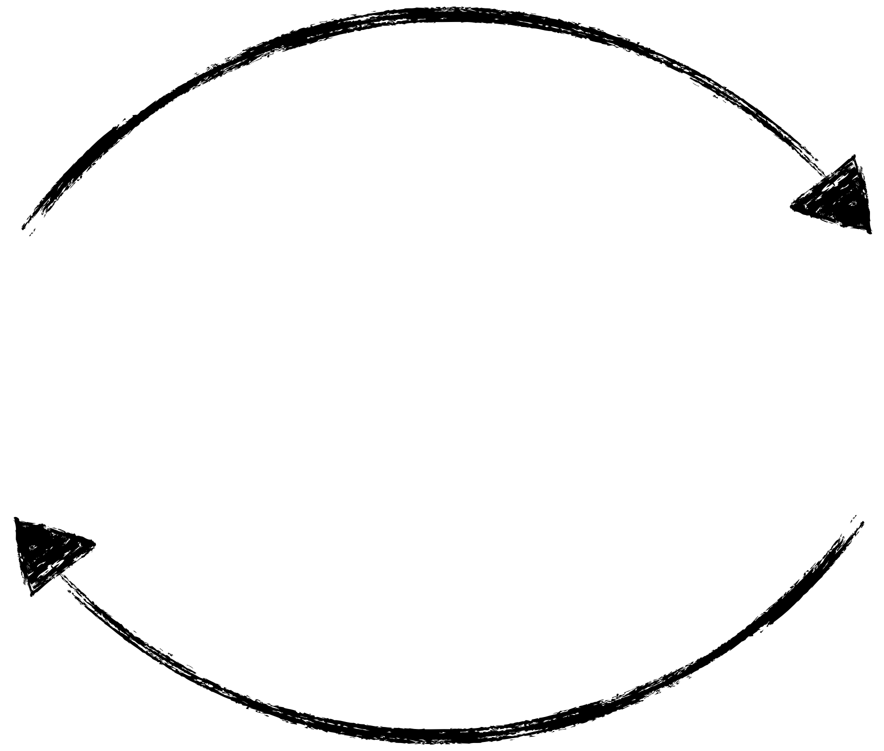
Please reach out!

So far we have been talking about estimation of $V^{\pi}(s)$.

What happens when we improve policy?

Use the same policy iteration idea ?

Estimate value $Q^{\pi}(s, a)$
using TD



Greedily improve policy
$$\pi^{+} = \arg\min_{a} Q^{\pi}(s, a)$$

Will this work?

Is greedy policy improvement the right thing to do?

"Behind one door is tenure - behind the other is flipping burgers at McDonald's."

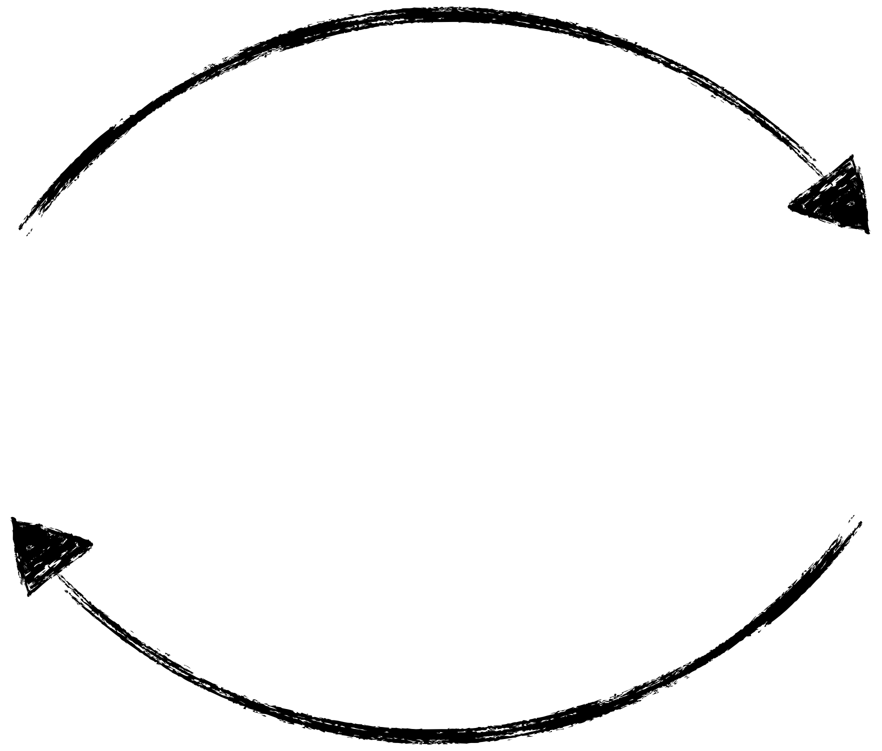
- There are two doors in front of you.
- You open the left door and get reward 0
  $V(\textit{left}) = 0$
- You open the right door and get reward $+1$
  $V(\textit{right}) = +1$
- You open the right door and get reward $+3$
  $V(\textit{right}) = +2$
- You open the right door and get reward $+2$
  $V(\textit{right}) = +2$

$$\vdots$$

- Are you sure you've chosen the best door?

SARSA

Estimate value $Q^{\pi}(s, a)$ using TD



Use epsilon-greedy to update policy

Need to explore!!

Can we learn off-policy?

Q-learning: Learning off-policy

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

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

Notice we are *not* approximating $Q^\pi(s_t, a_t)$

We don't even care about π

We can learn from any data!

Is this ... magic?

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

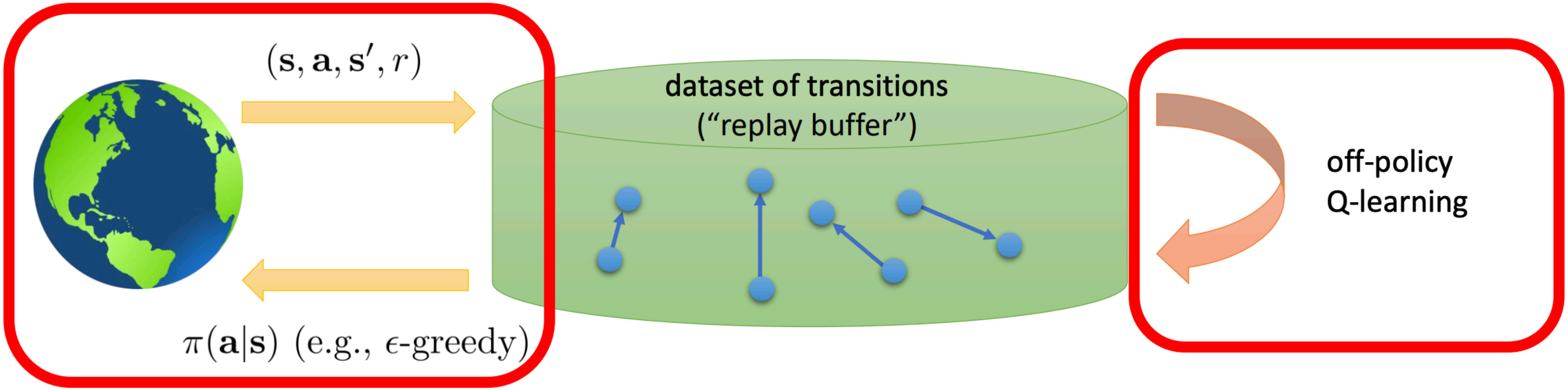
It's not magic. Q-learning relies on a set of assumptions:

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

How can we use a (s,a,c,s') more than once?

What happens if samples are highly correlated?

Solution: Use a replay buffer!



Credit S. Levine

Human-level control through deep reinforcement learning

Volodymyr Mnih[1]*, Koray Kavukcuoglu[1]*, David Silver[1]*, Andrei A. Rusu[1], Joel Veness[1], Marc G. Bellemare[1], Alex Graves[1], Martin Riedmiller[1], Andreas K. Fidjeland[1], Georg Ostrovski[1], Stig Petersen[1], Charles Beattie[1], Amir Sadik[1], Ioannis Antonoglou[1], Helen King[1], Dharshan Kumaran[1], Daan Wierstra[1], Shane Legg[1] & Demis Hassabis[1]

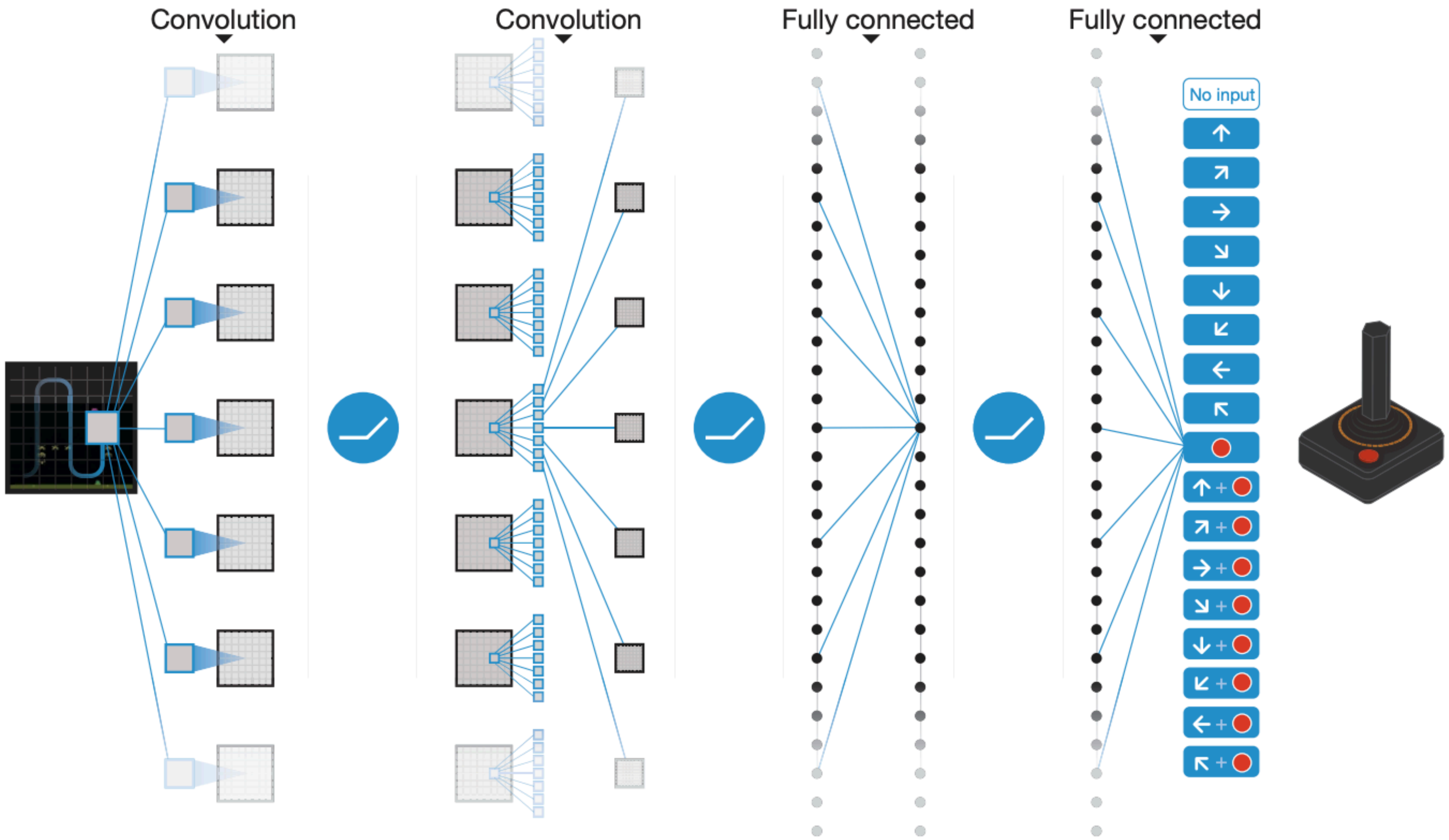
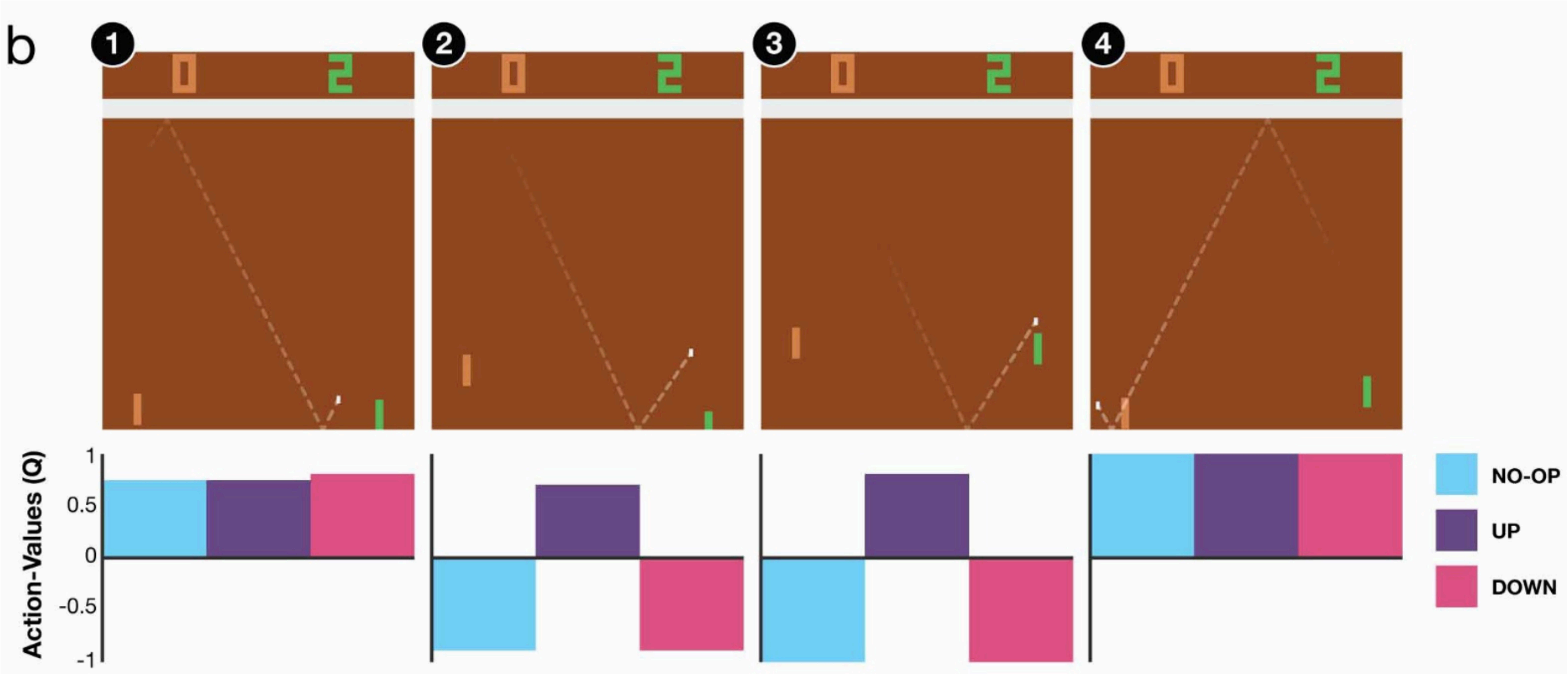
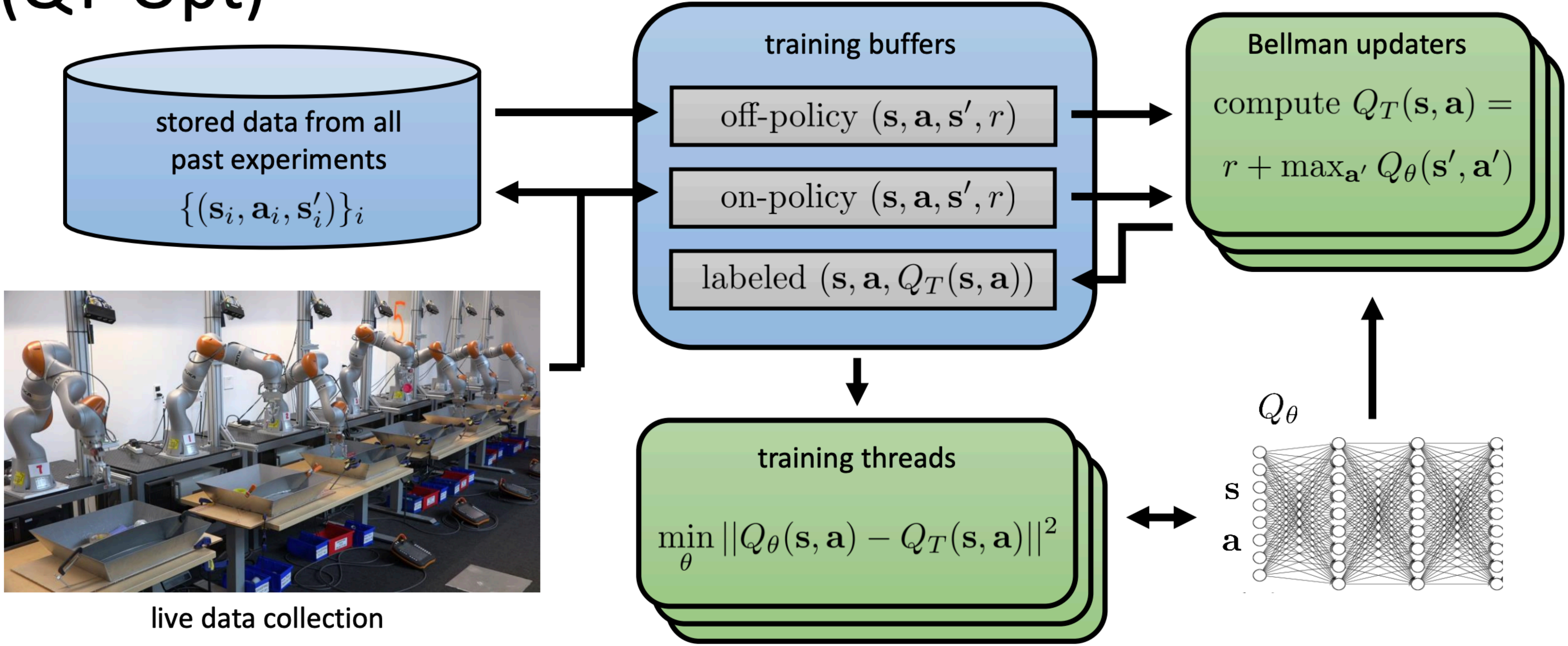


Figure 1 | Schematic illustration of the convolutional neural network. The details of the architecture are explained in the Methods. The input to the neural network consists of an $84 \times 84 \times 4$ image produced by the preprocessing map ϕ, followed by three convolutional layers (note: snaking blue line symbolizes sliding of each filter across input image) and two fully connected layers with a single output for each valid action. Each hidden layer is followed by a rectifier nonlinearity (that is, $\max(0,x)$).


b
1
2
3
4
0
2
Action-Values (Q)
1
0.5
0
-0.5
-1
NO-OP
UP
DOWN

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**


The system is trained on about
1000 visually and physically diverse objects

Making Q-learning better!

Problem: Q-learning suffers from an estimation bias $\min_{a'} Q^*(s_{t+1}, a')$

Solution: Double Q-learning $Q^*(s_{t+1}, \arg\min_{a'} \tilde{Q}(s_{t+1}, a'))$

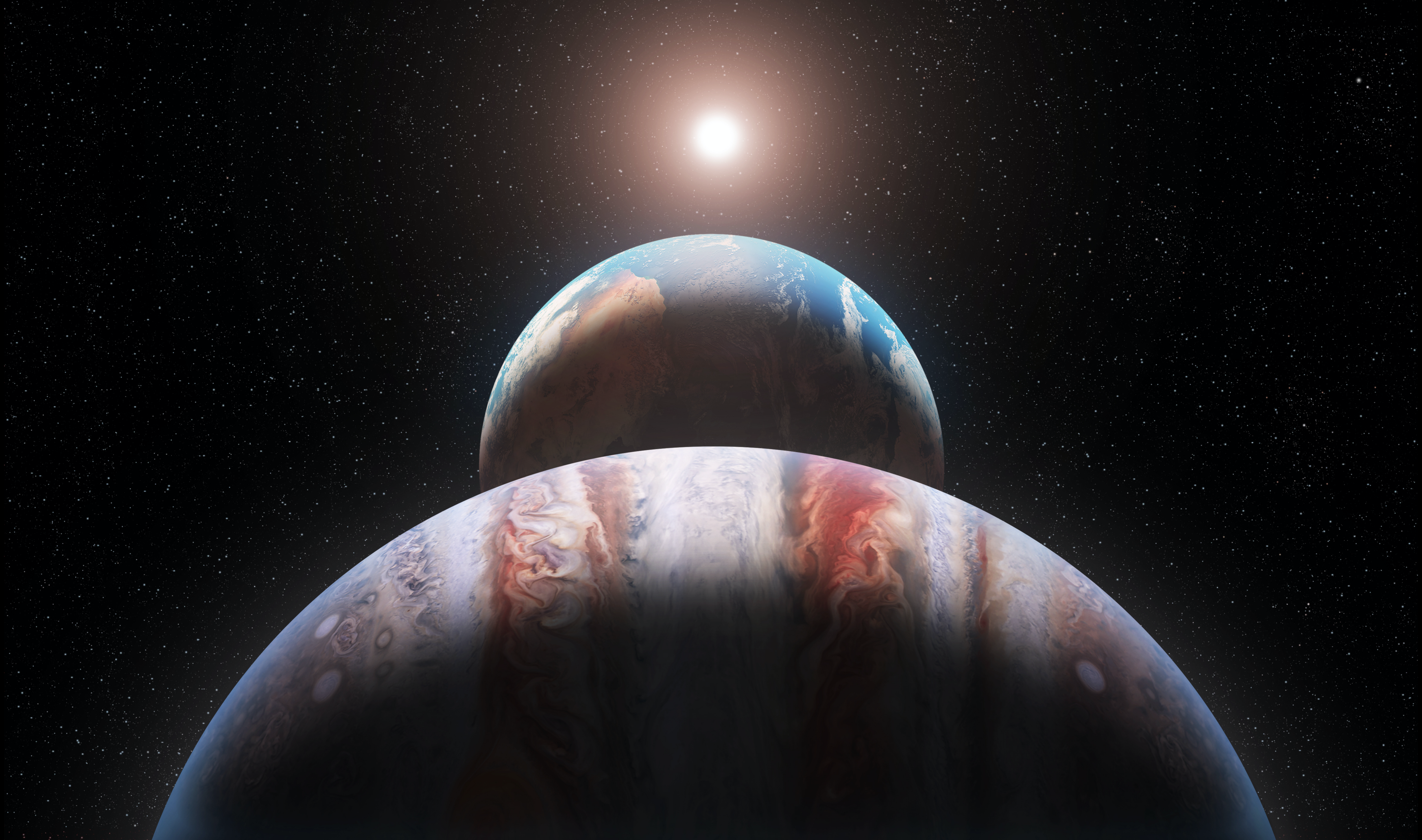
Problem: Q-learning samples uniformly from replay buffer

Solution: Prioritized DQN - samples states with higher bellman error

Problem: Q-learning doesn't seem to learn ....

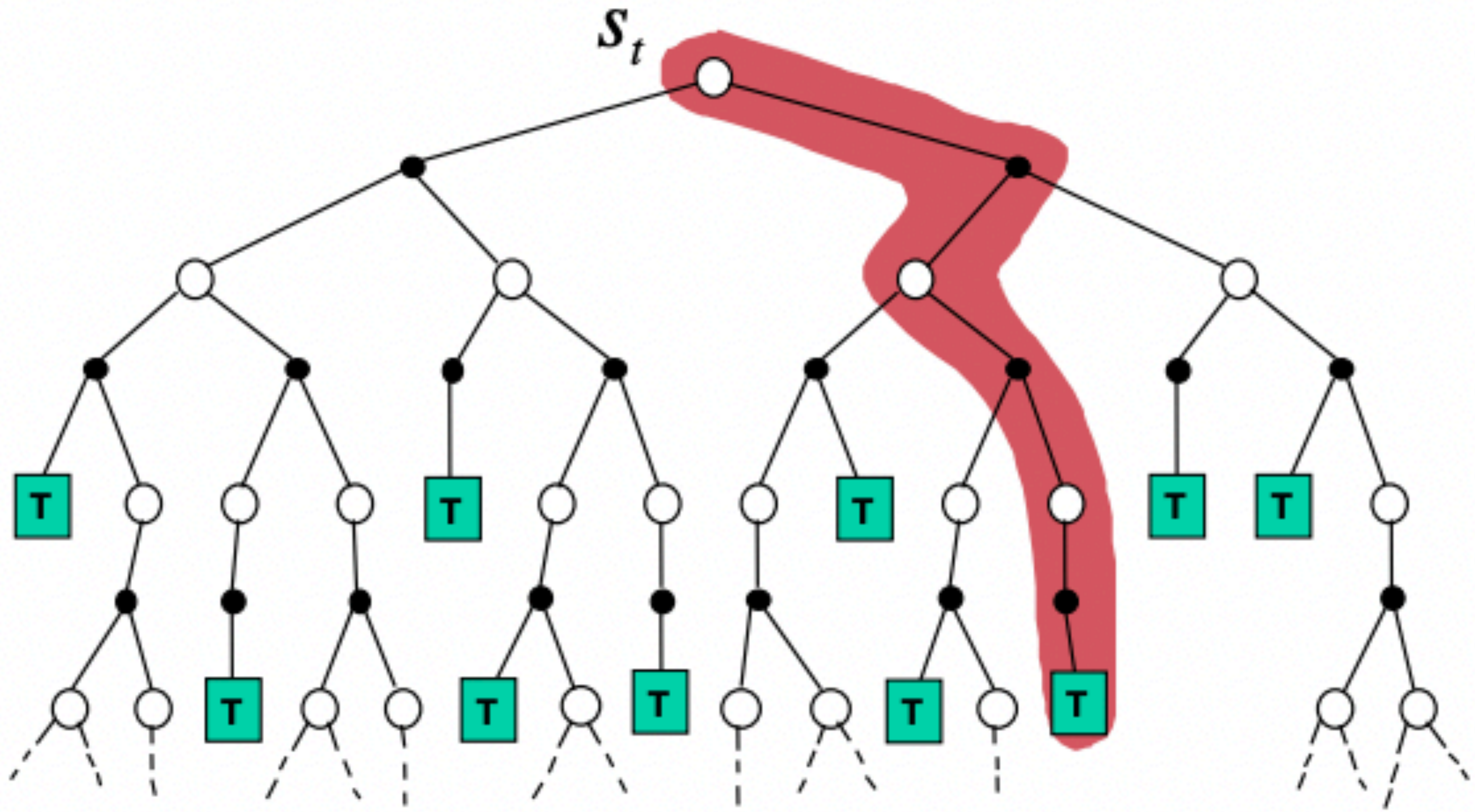
Solution: Start with high exploration + learning rate, anneal!

A Unified View of Reinforcement Learning



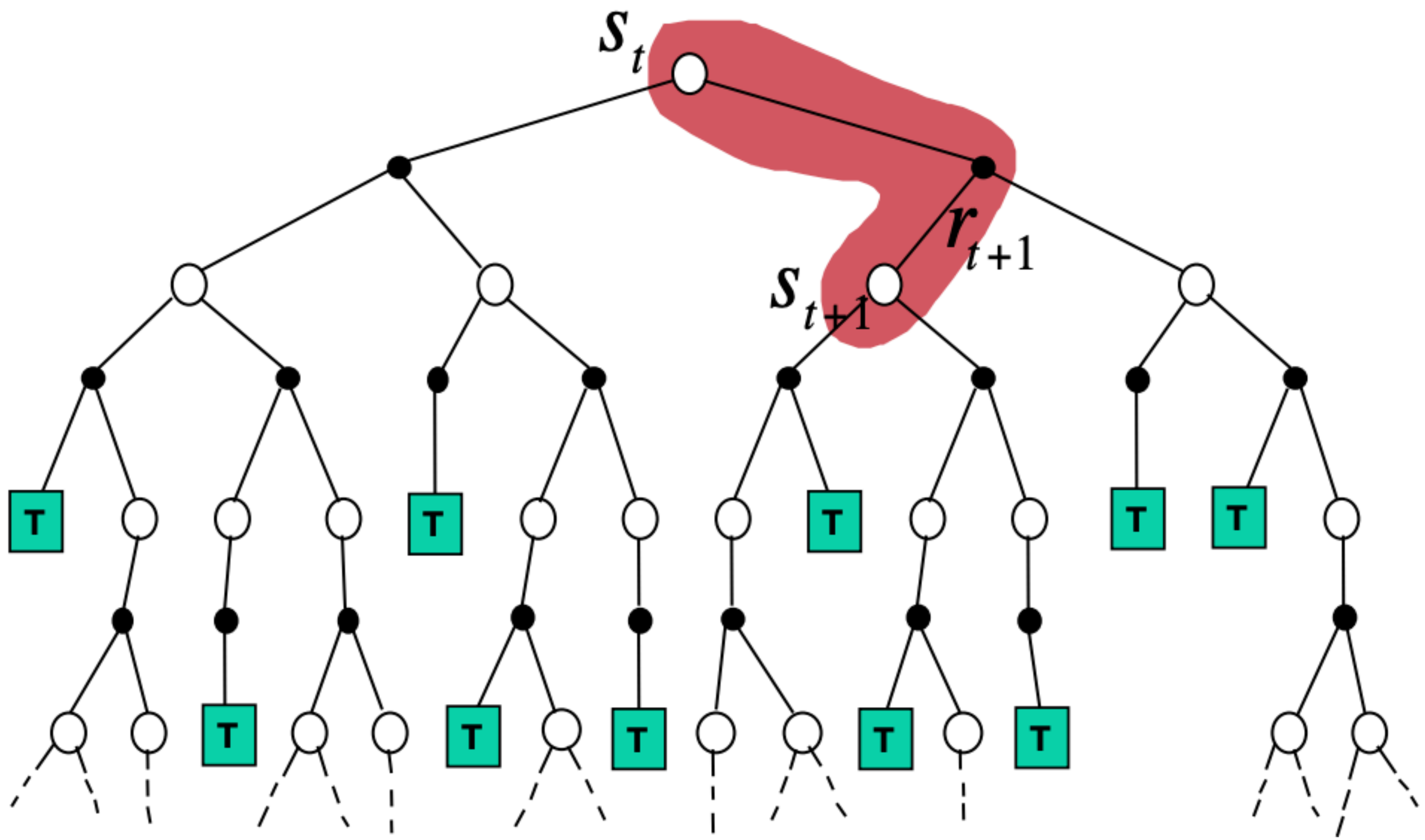
Monte-Carlo

$$V(S_t) \leftarrow V(S_t) + \alpha \left(G_t - V(S_t) \right)$$



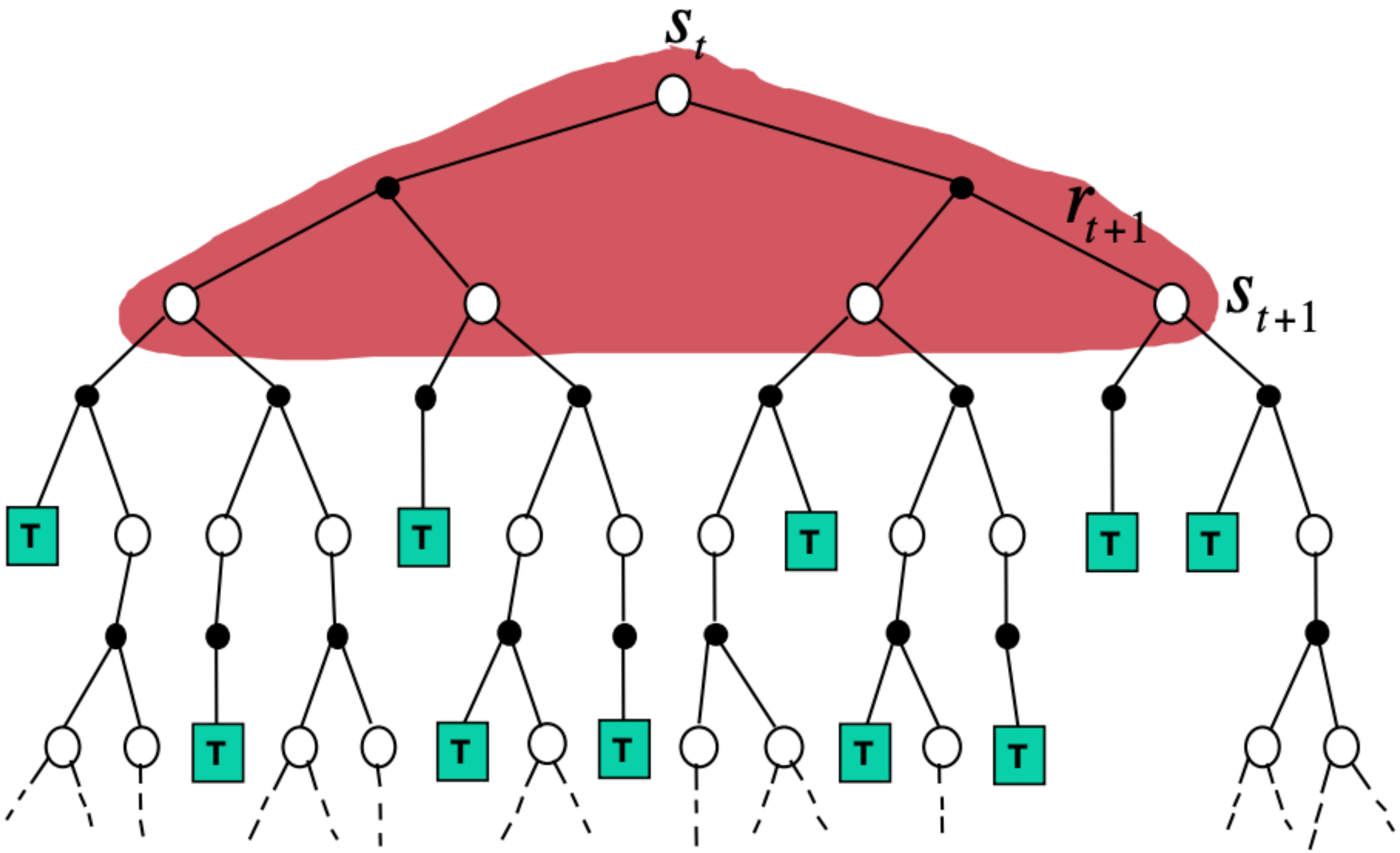
Temporal Difference Learning

$$V(S_t) \leftarrow V(S_t) + \alpha \left(R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \right)$$

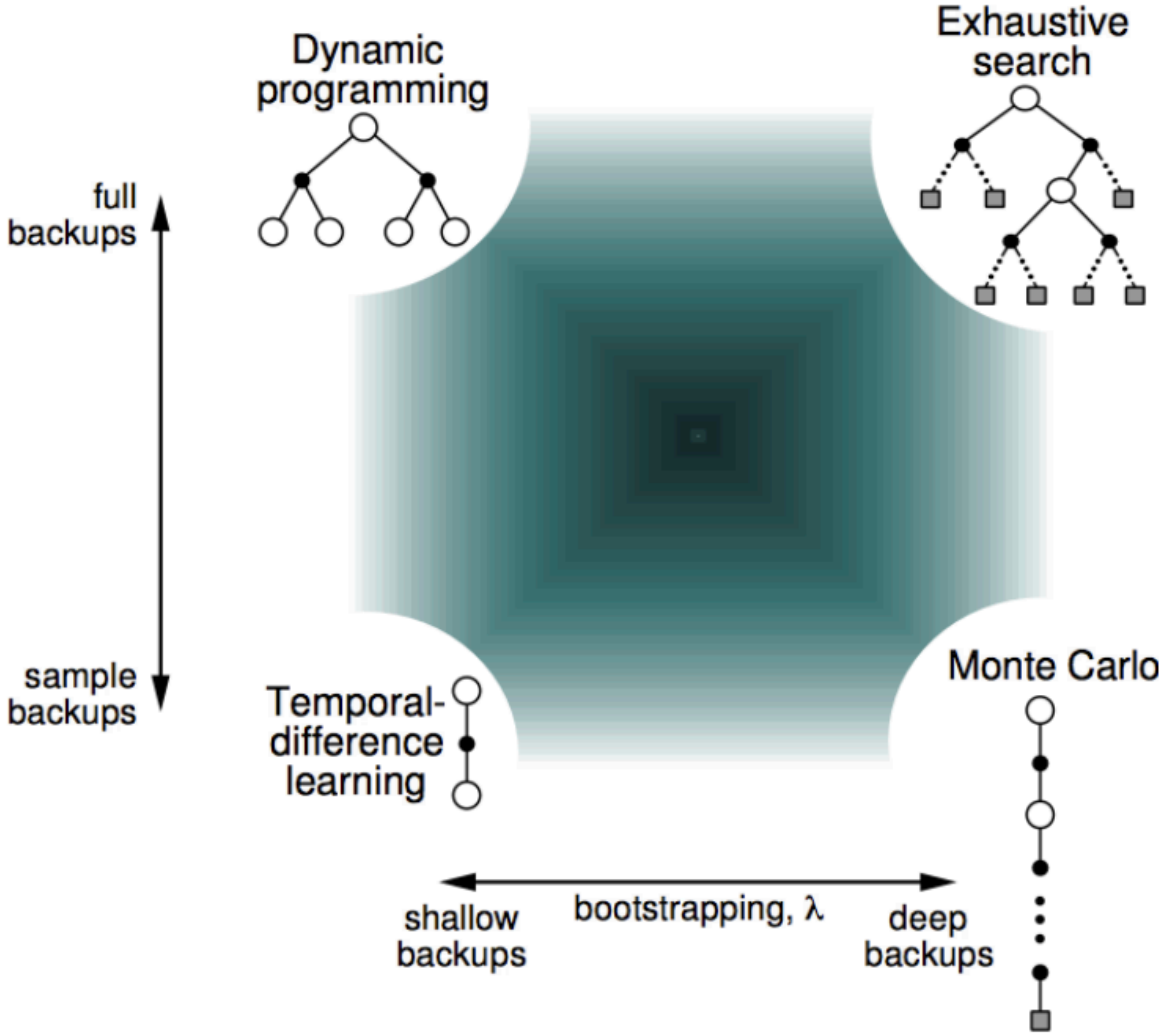


Dynamic Programming

$$V(S_t) \leftarrow \mathbb{E}_\pi \left[R_{t+1} + \gamma V(S_{t+1})\right]$$

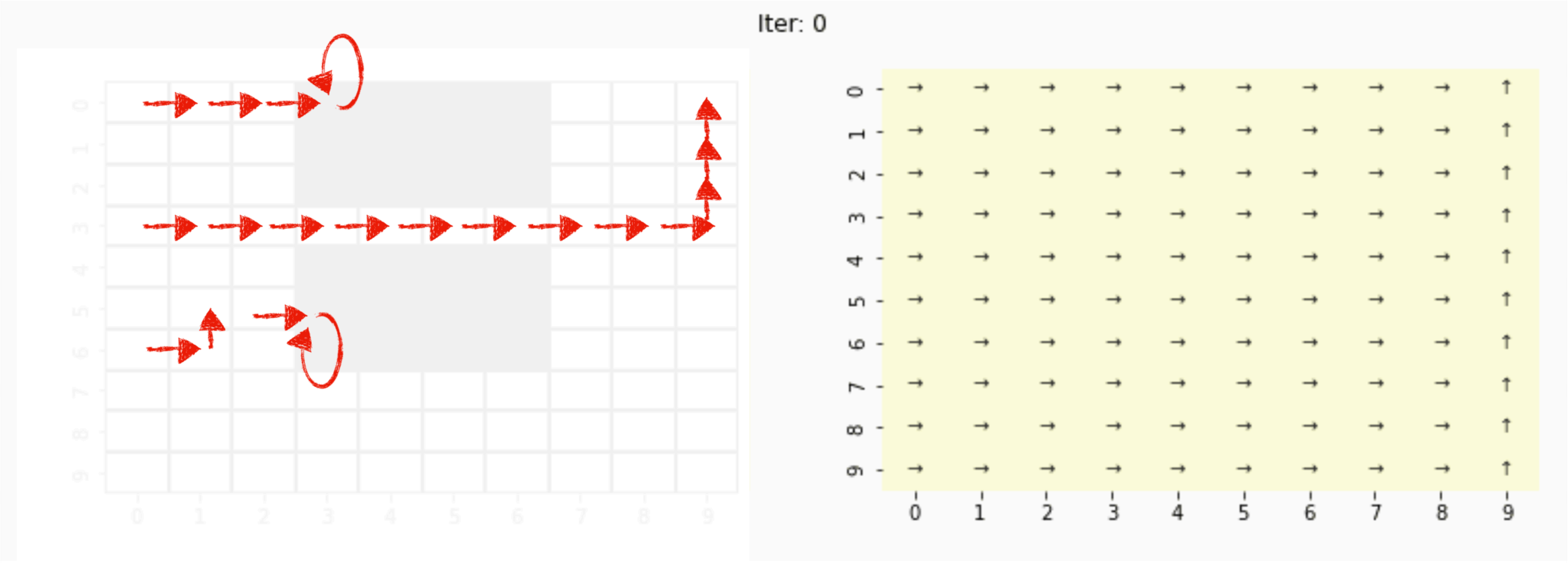


The Unified View



tl;dr

Estimate the value of policy from sample rollouts



Roll outs

Policy π

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

Q-learning: Learning off-policy

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

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

Notice we are *not* approximating $Q^\pi(s_t, a_t)$

We don't even care about π

We can learn from any data!