Approximate Dynamic Programming

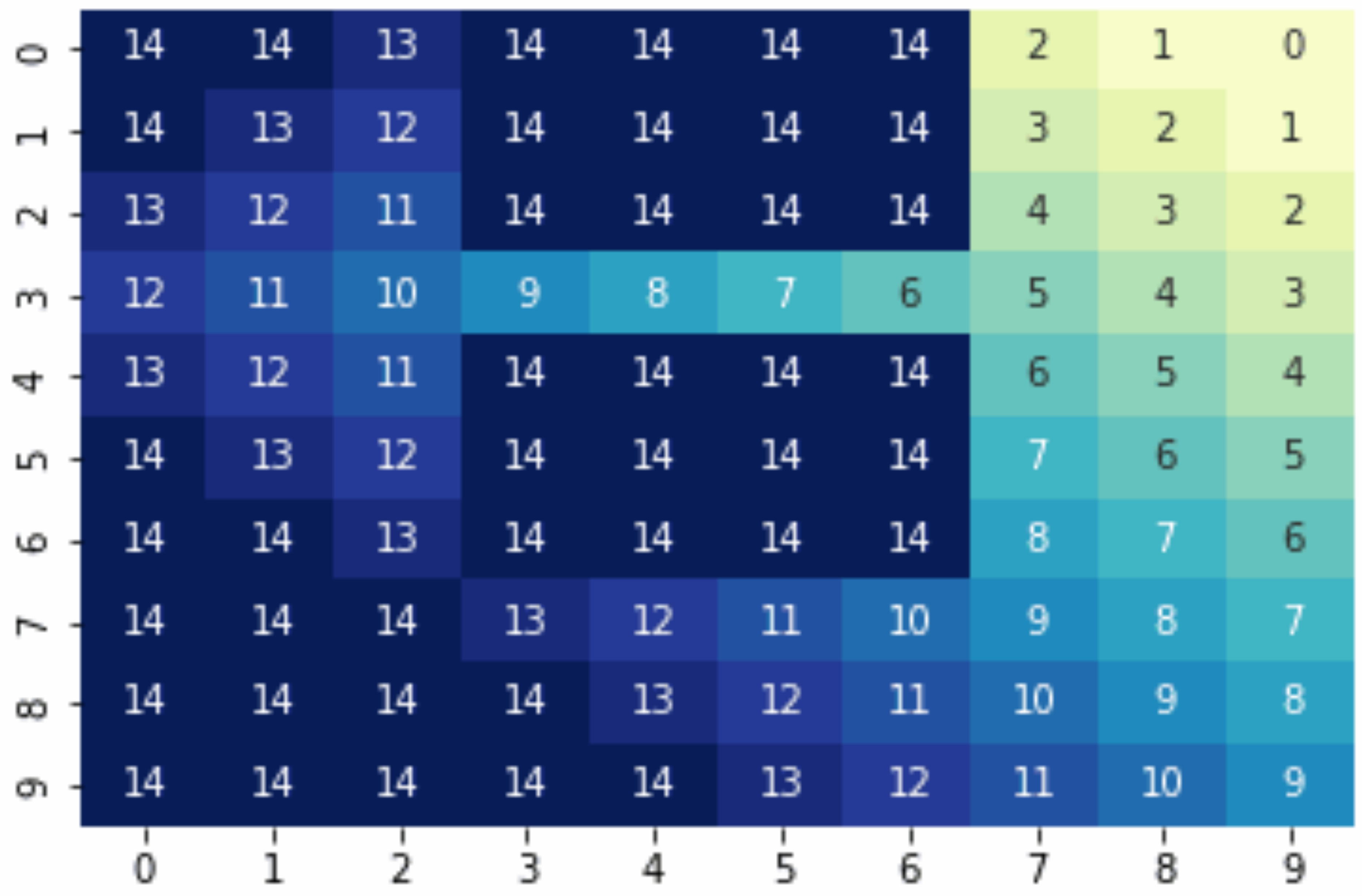
Sanjiban Choudhury

Cornell Bowers CIS
Computer Science

When the MDP is known:

Two Fundamental Ways to Solve for Optimal Policy

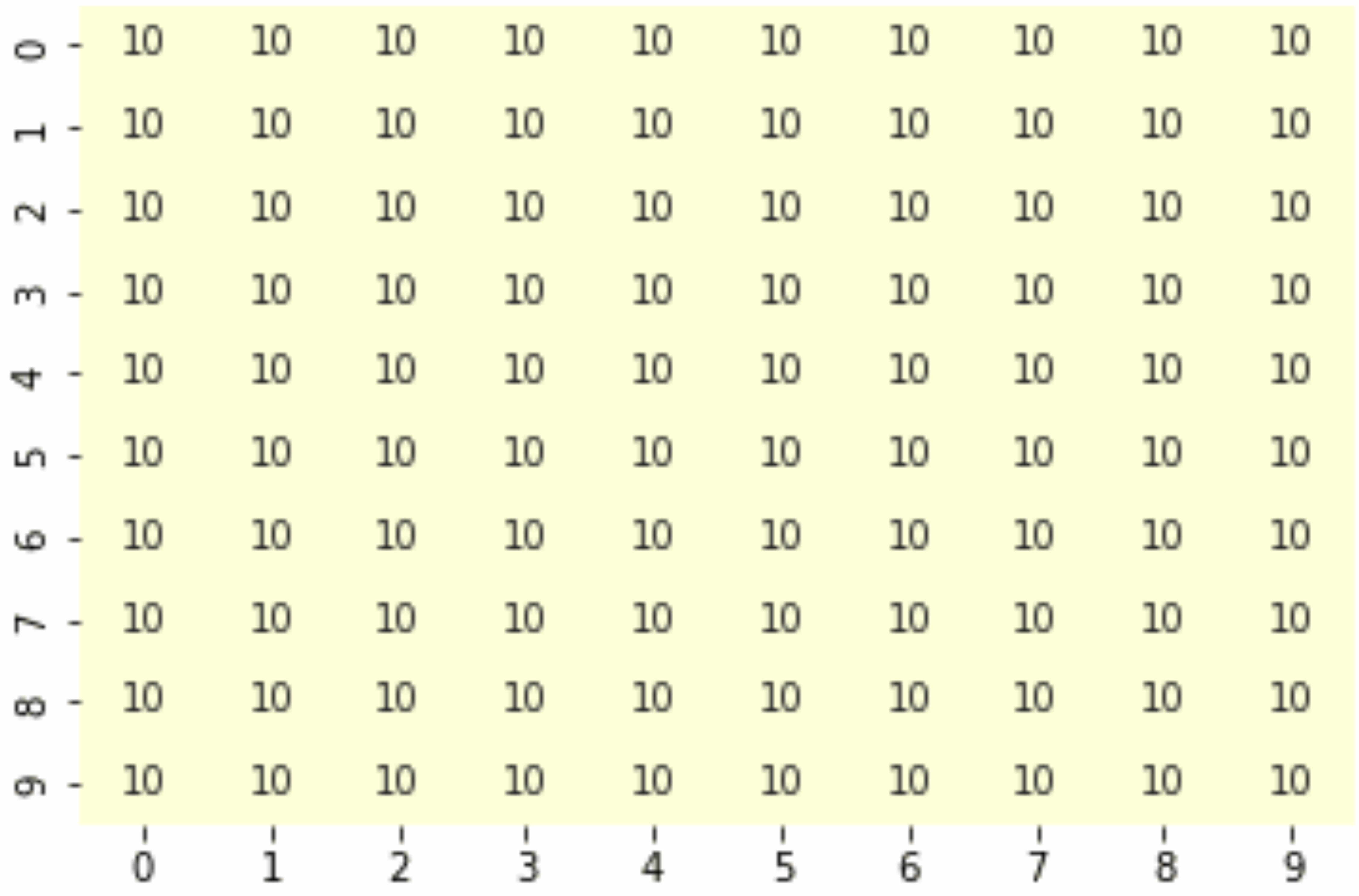
Value Iteration



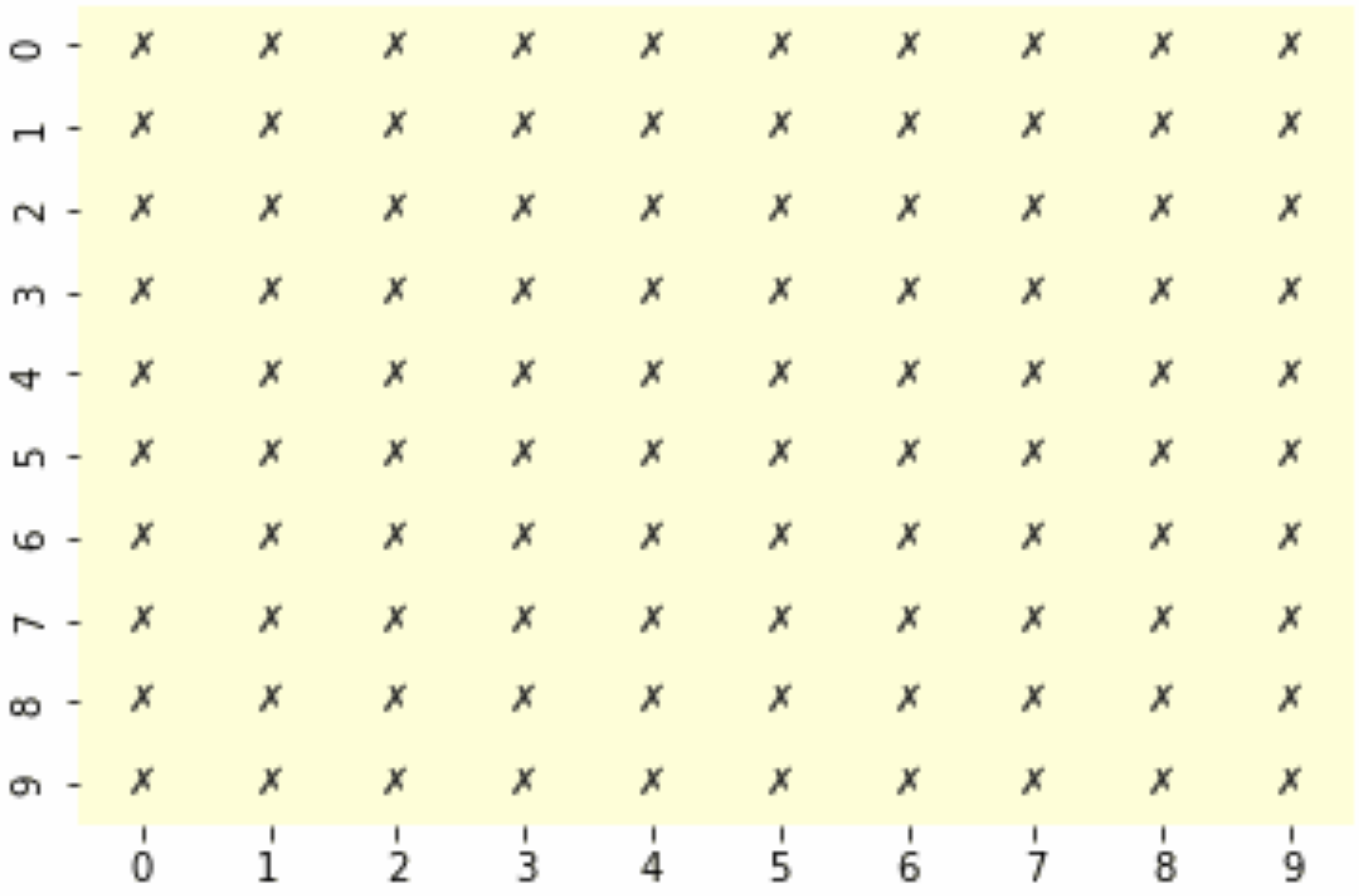
$$V^*(s) = \min_a [c(s, \pi(s)) + \gamma \mathbb{E}_{s' \sim \mathcal{T}(s,a)} V^*(s')]$$

Policy Iteration

Which one converges faster: value/policy?



Values



Policy

Policy converges faster than the value

Can we iterate over policies?

Policy Iteration

Init with some policy π

Repeat forever

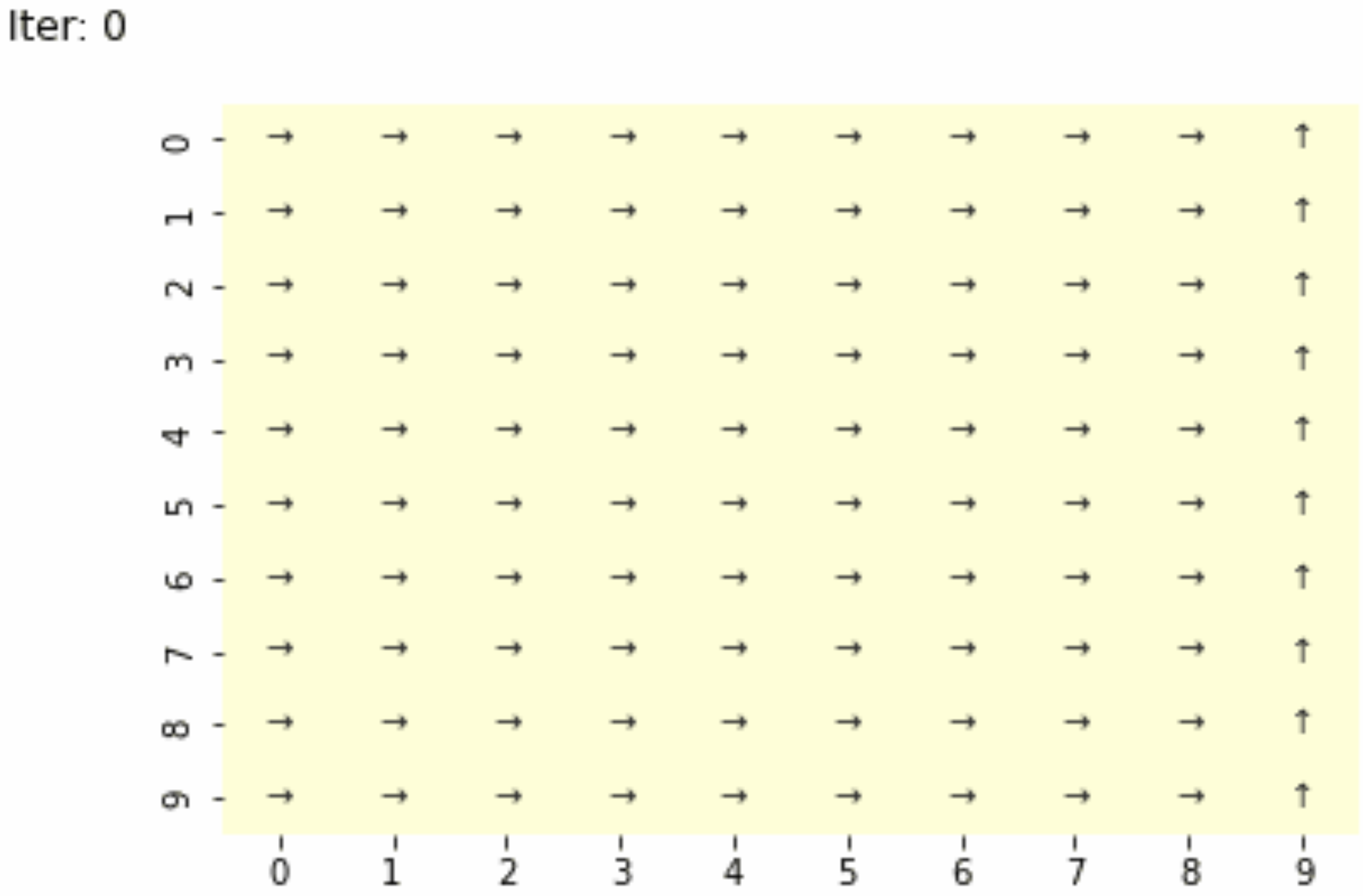
Evaluate policy

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

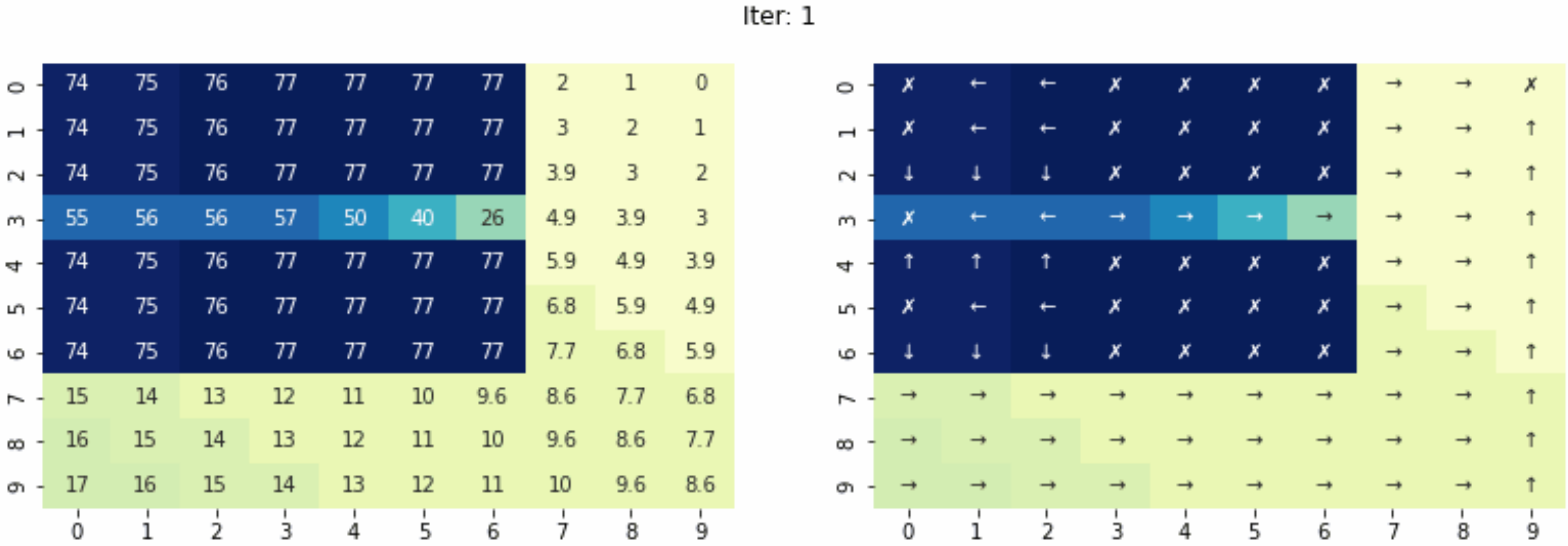
Improve policy

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

Init with some policy π



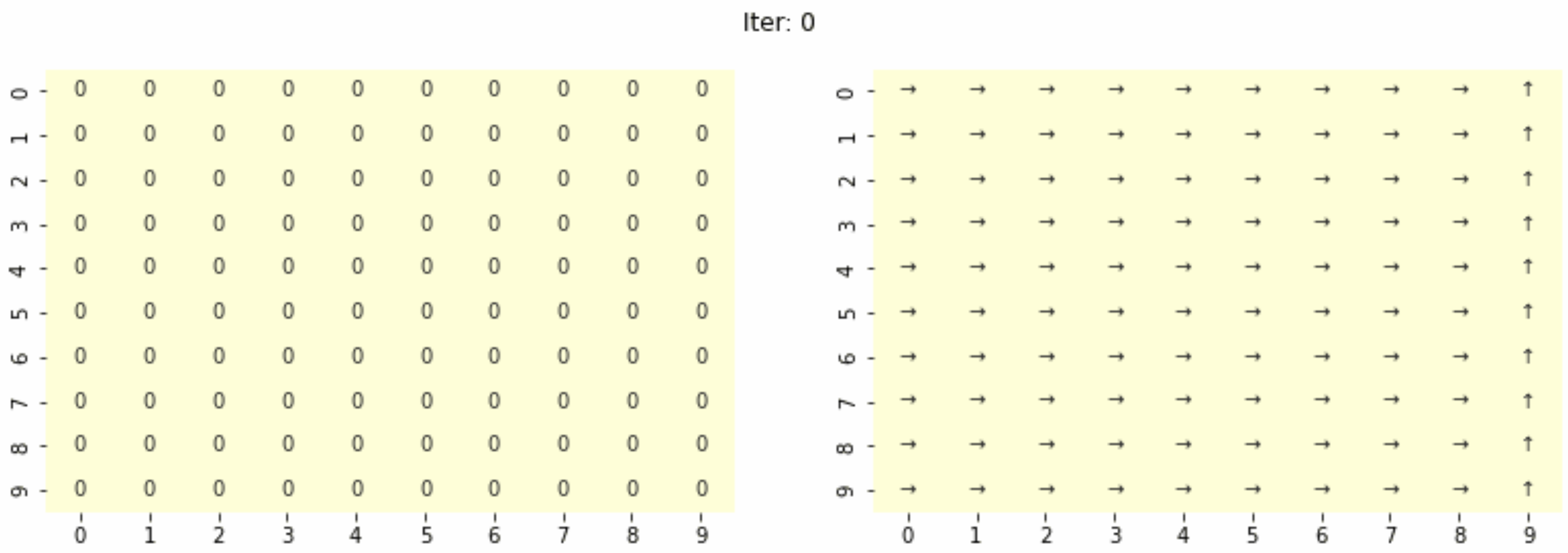
Iteration 1: Compute the value of the policy



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

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

Policy Iteration



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

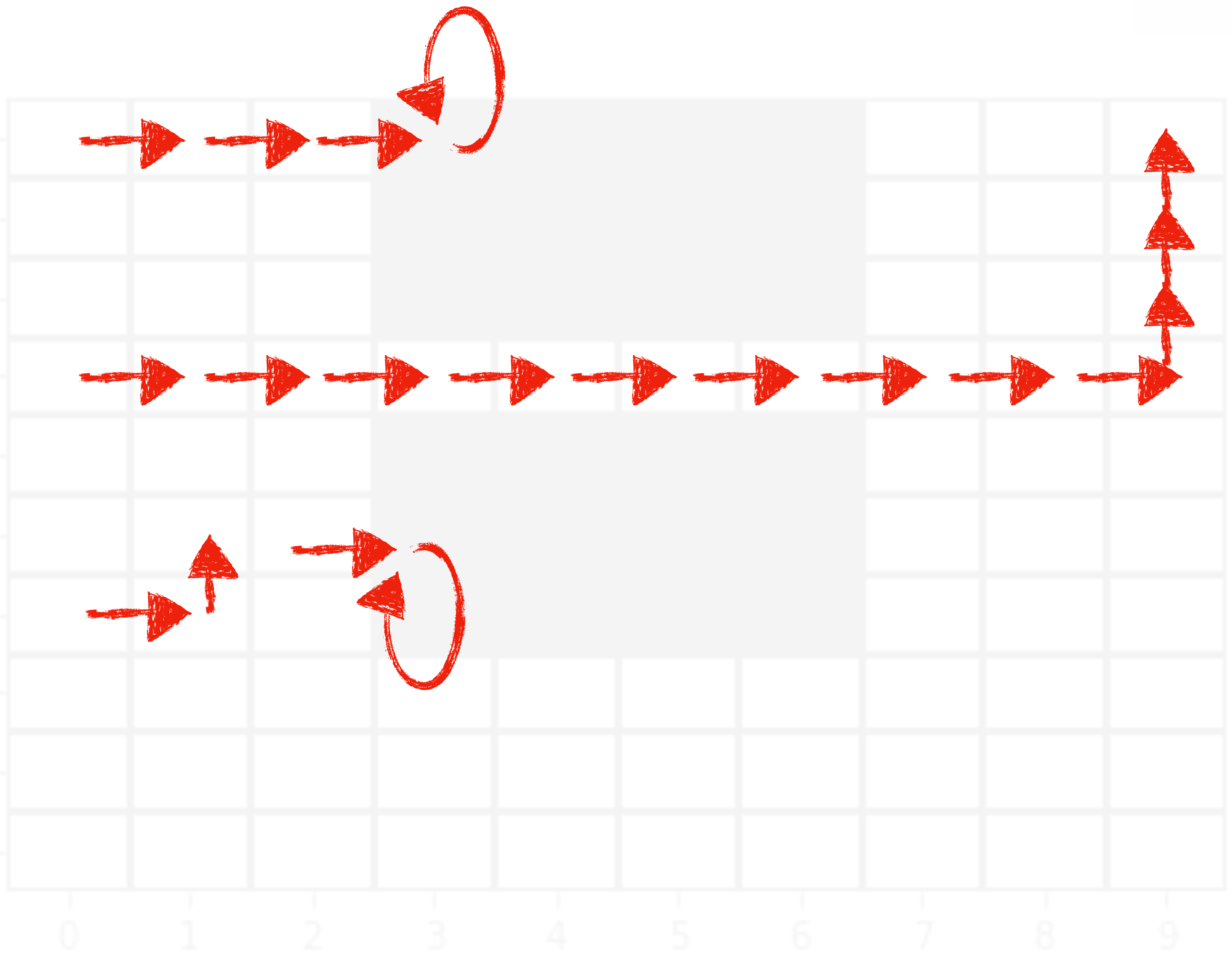
$$\pi^+(s) = \arg\min_a c(s, a) + \gamma \mathbb{E}_{s' \sim \mathcal{T}(s,a)} V^\pi(s')]$$

When the MDP is ~~known~~ unknown:

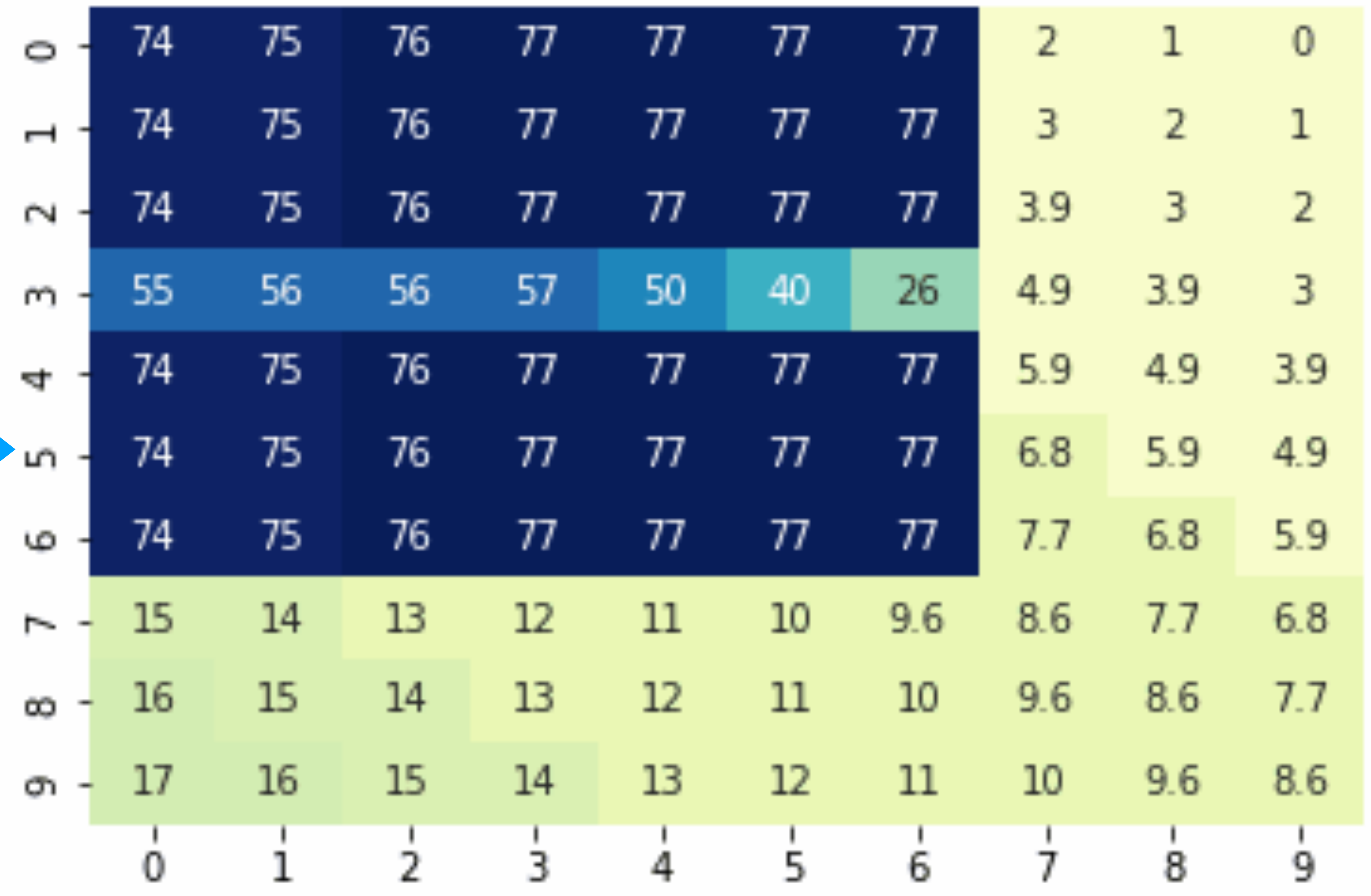
Restricted access to the transition function

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

Estimate the value of policy from sample rollouts



Roll outs



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

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

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

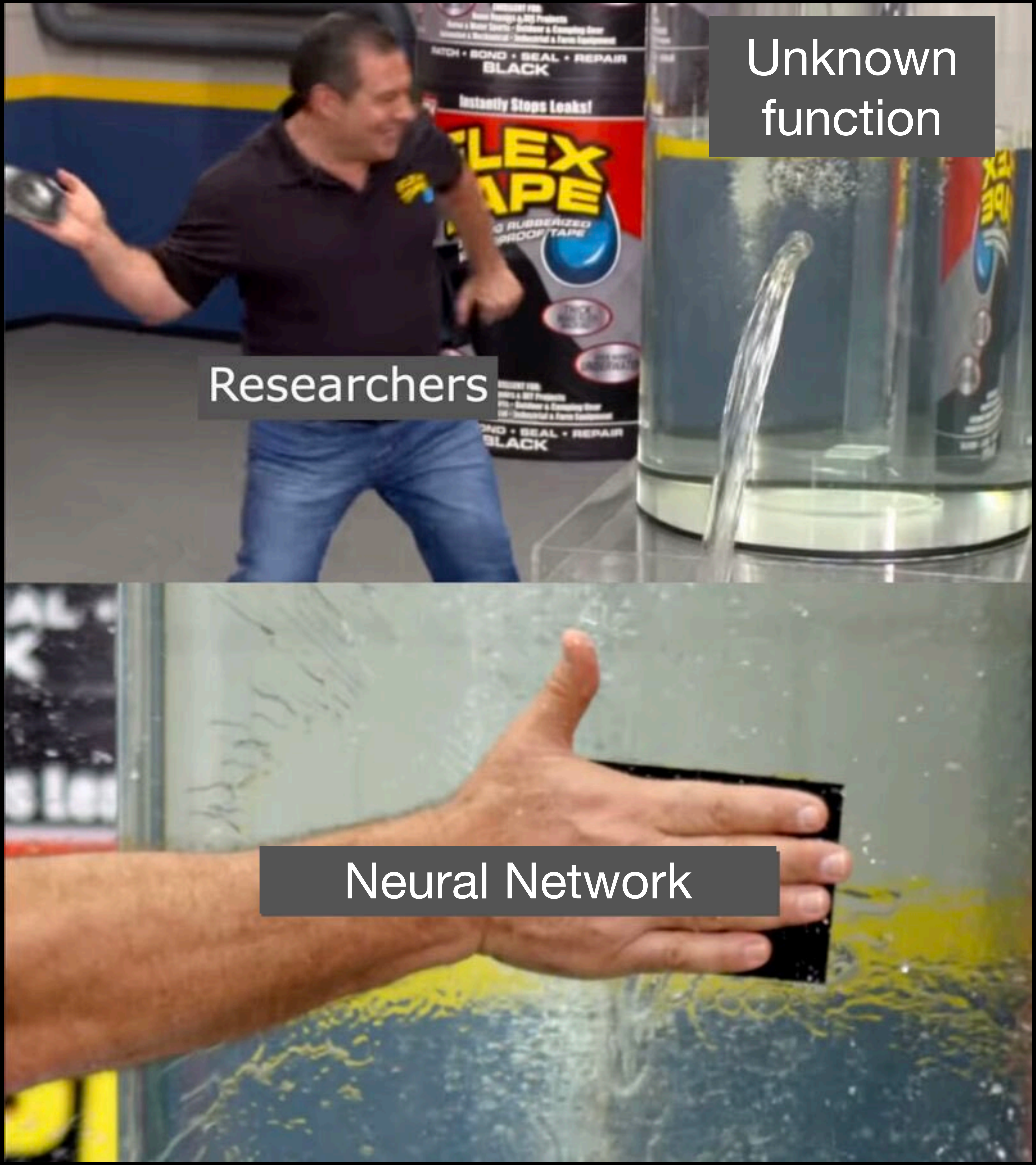
Can have bias

Low Variance

May *not* converge if using function approximation

Tabular setting is cute.

But how do we estimate V(s) in the continuous setting

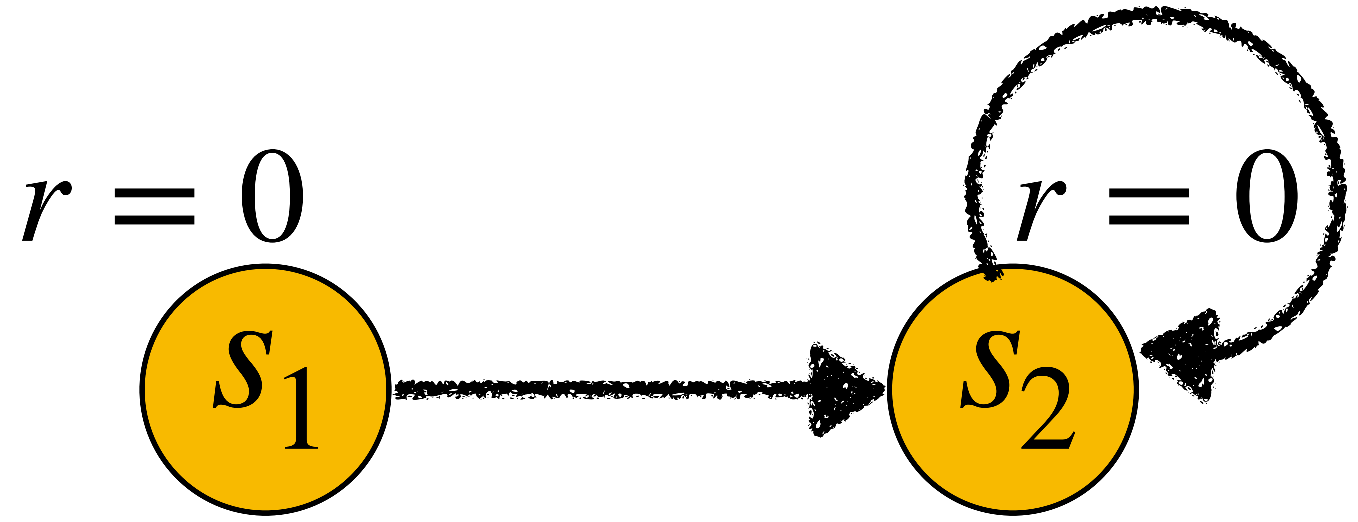

Unknown function
Researchers
Neural Network

Activity!

A *tiny* MDP

Reward for being at any state is 0.0

Discount factor $\gamma = 0.9$



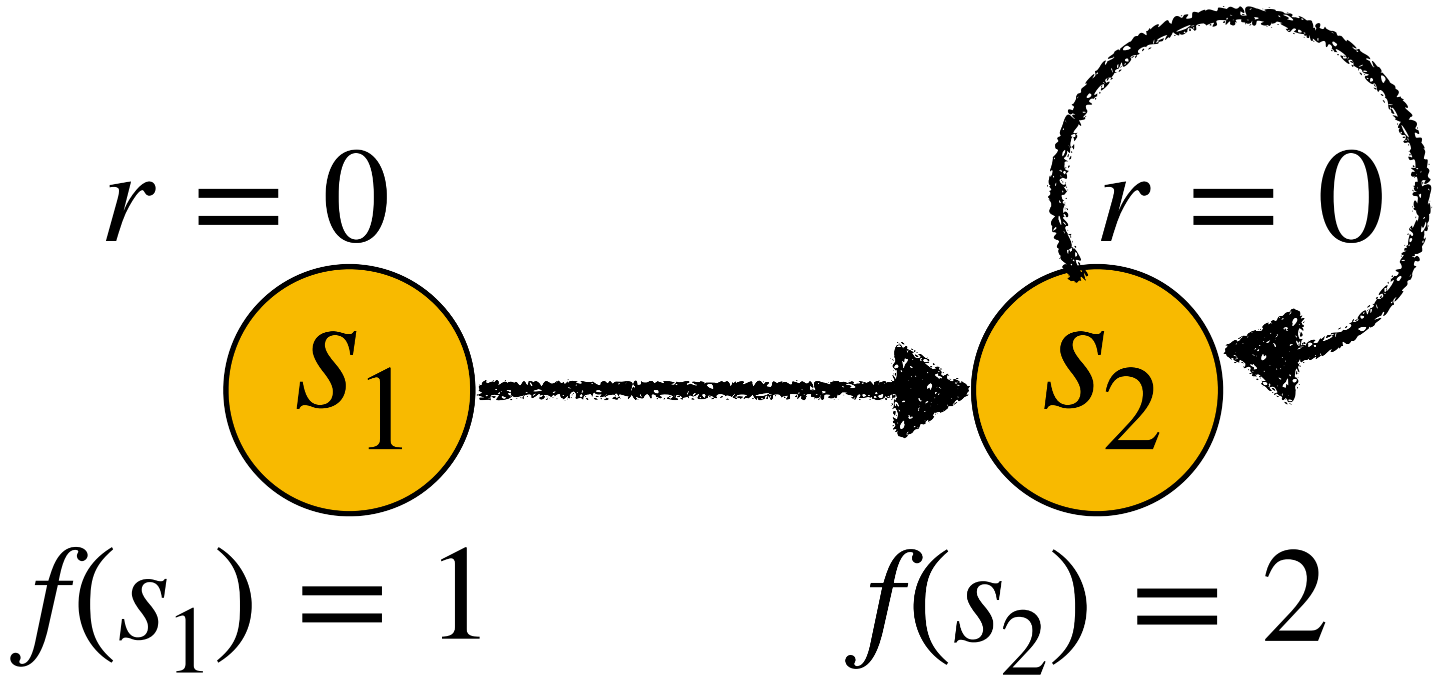
What happens when you run value iteration?

(Initialize with random values, say $V(s_1) = 1$ and 2)

A *tiny* MDP

Reward for being at any state is 0.0

Discount factor $\gamma = 0.9$



Let's say we want to use a *linear value function approximator*

$$V(s) = wf(s) = w * \begin{cases} 1 & \textit{if } s = s_1 \\ 2 & \textit{if } s = s_2 \end{cases}$$

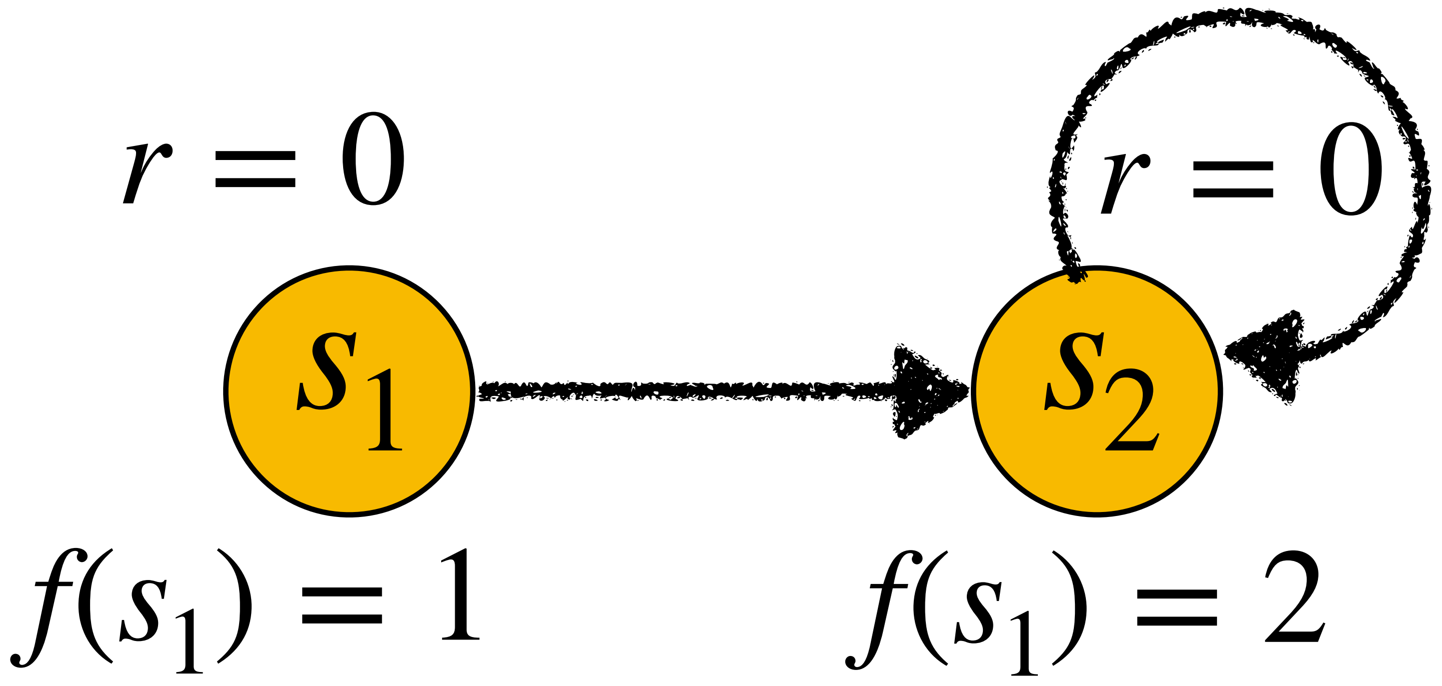
What happens if you run value iteration? (Initialize with w=1)

Think-Pair-Share

Think (30 sec): Initialize value iteration with w=1. What happens? What's the explanation?

Pair: Find a partner

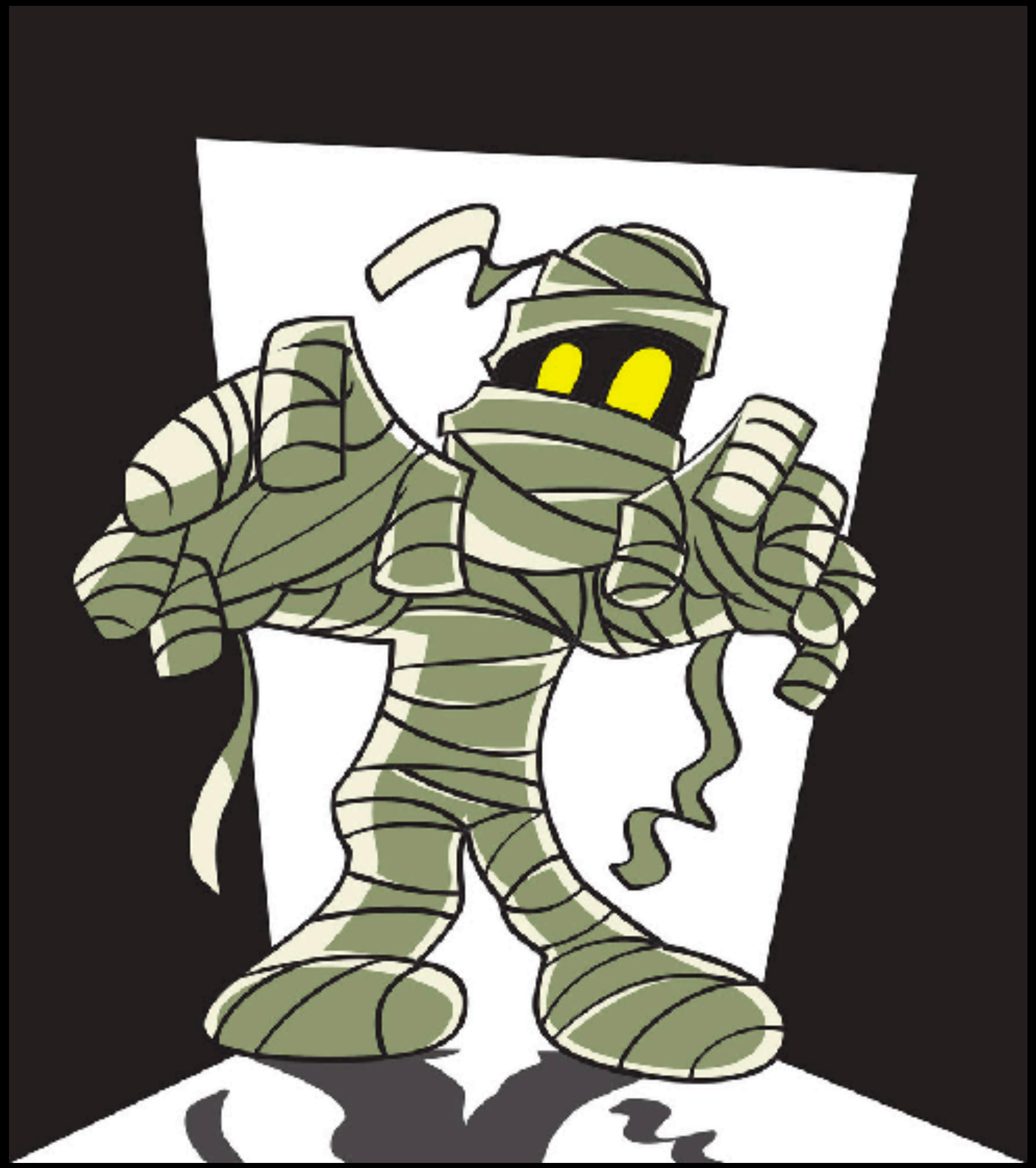
Share (45 sec): Partners exchange ideas



$$V(s) = wf(s)$$

Init with $w = 1$

CURSE OF APPROXIMATION!



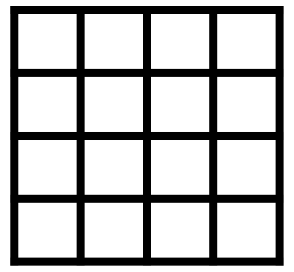
Approximation introduces an error that gets amplified by both value / policy iteration

Key separation between SL and RL

From dynamic programming to *Fitted* dynamic programming

Approximate (Fitted) Value Iteration

Q-iteration



$Q(s, a) \leftarrow 0$

while *not converged* **do**

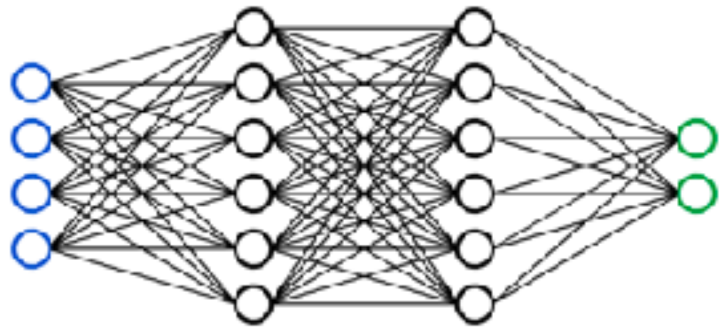
  for $s \in S, a \in A$

    $Q^{new}(s, a) = c(s, a) + \gamma \mathbb{E}_{s'} \min_{a'} Q(s', a')$

  $Q \leftarrow Q^{new}$

return Q

Fitted Q-iteration



Given $\{s_i, a_i, c_i, s'_i\}_{i=1}^{N}$

Init $Q_\theta(s, a) \leftarrow 0$

while *not converged* **do**

  $D \leftarrow \emptyset$

  for $i \in 1, \ldots, n$

    input $\leftarrow \{s_i, a_i\}$

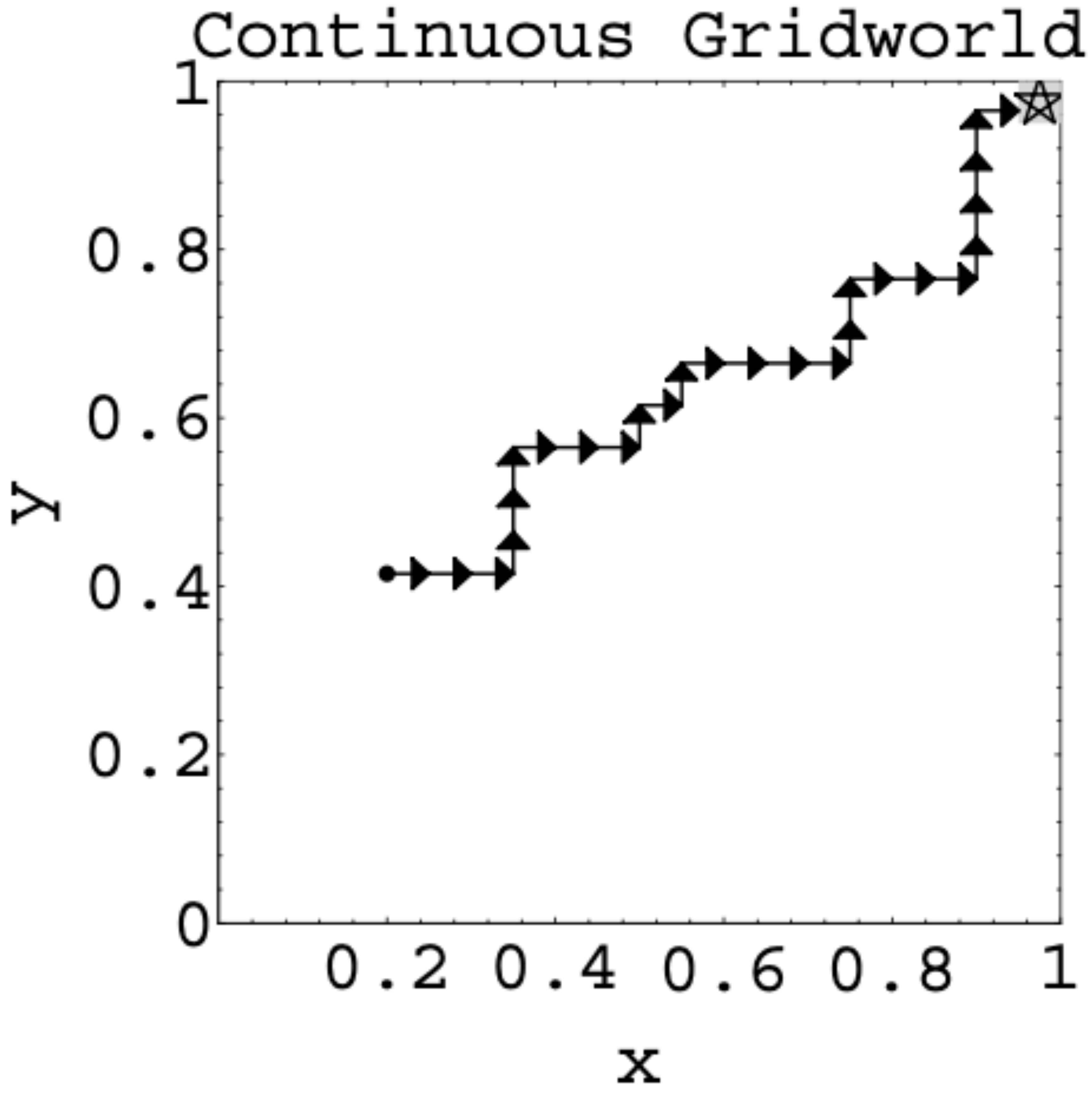
    target $\leftarrow c_i + \gamma \min_a Q_\theta(s'_i, a')$

    $D \leftarrow D \cup$ {input, output}

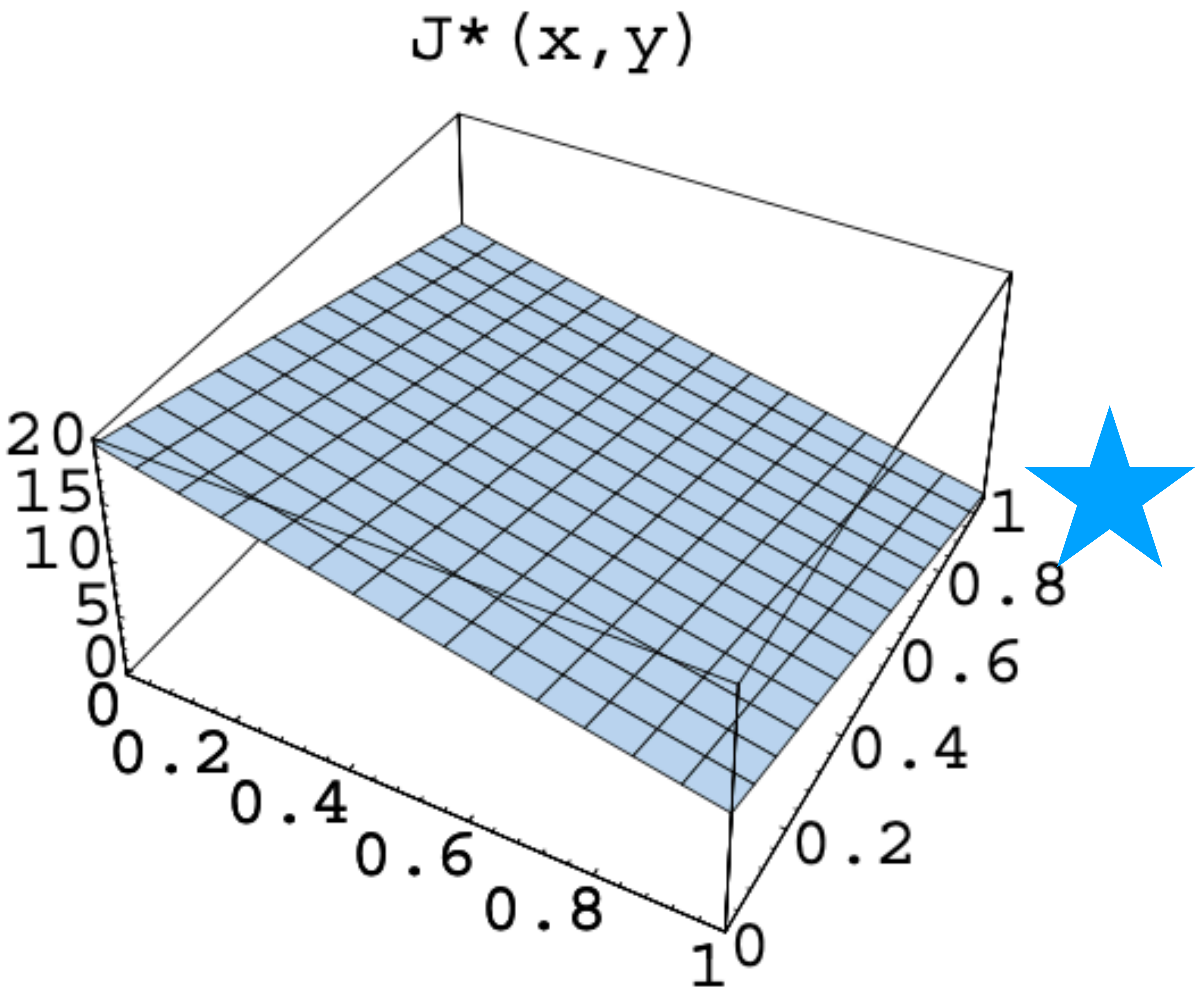
  $Q_\theta \leftarrow$ **Train**(D)

return Q_θ

A simple example: Gridworld



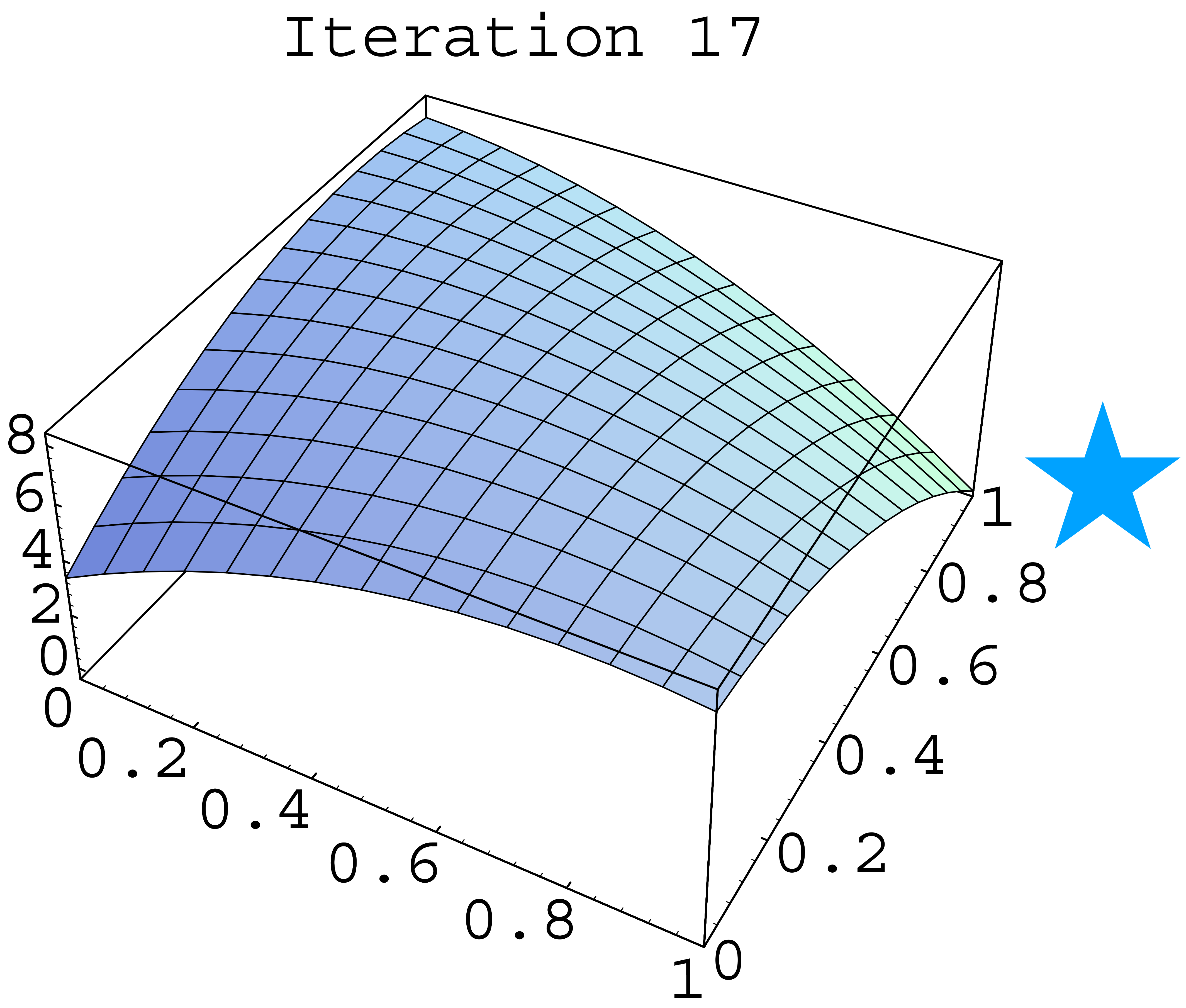
Optimal path



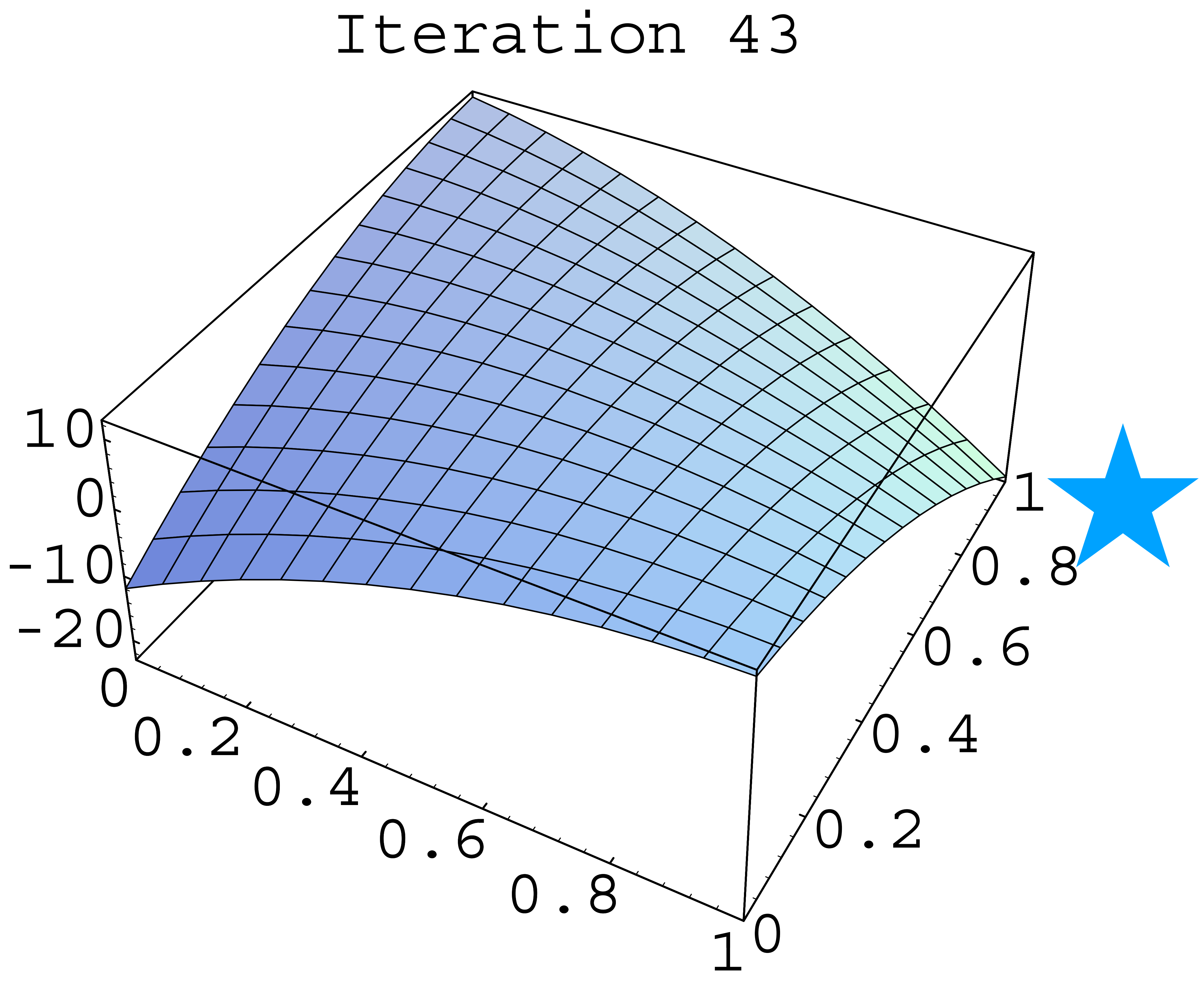
True value function

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

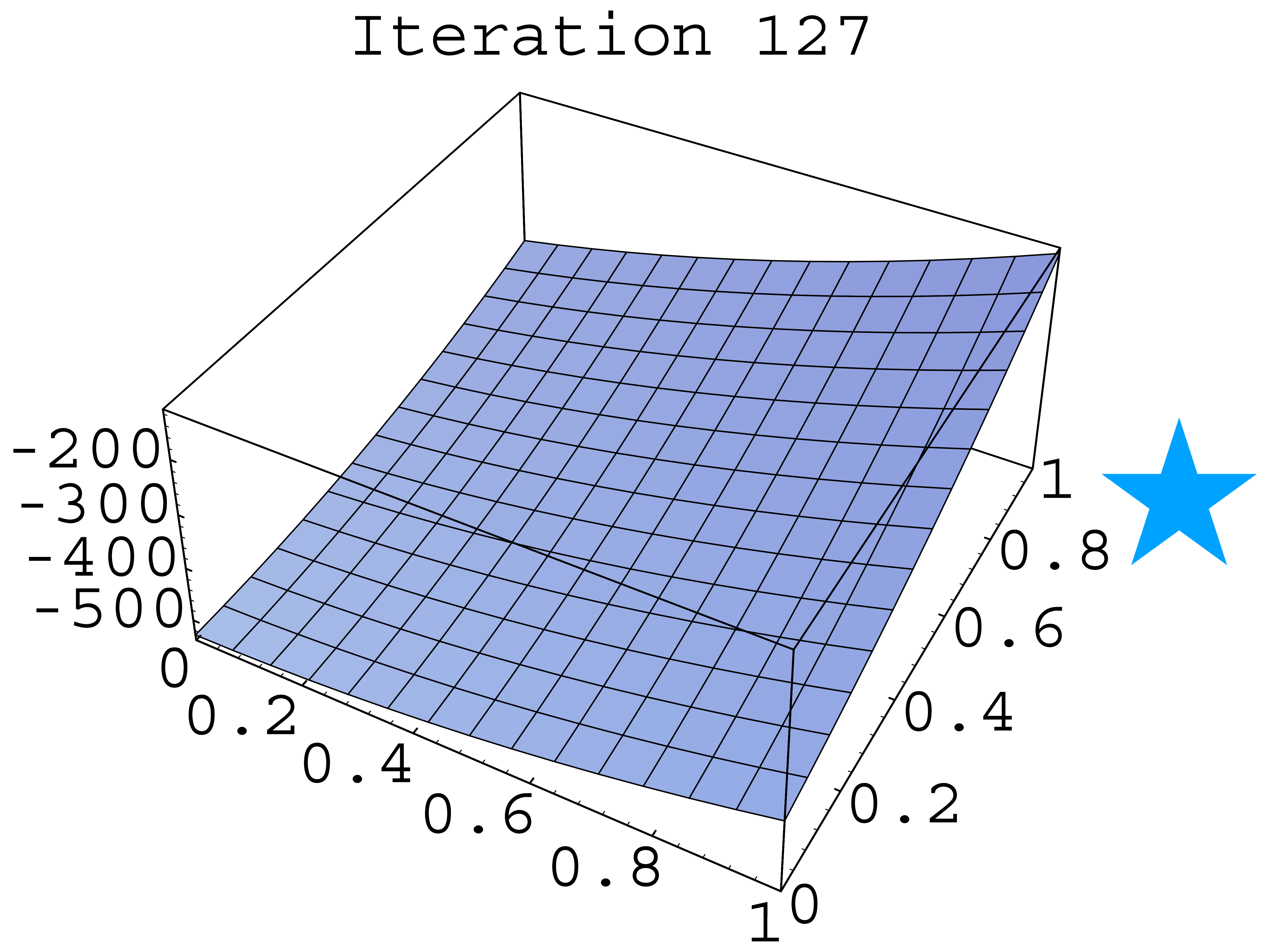
What happens when we run value iteration with a *quadratic*?



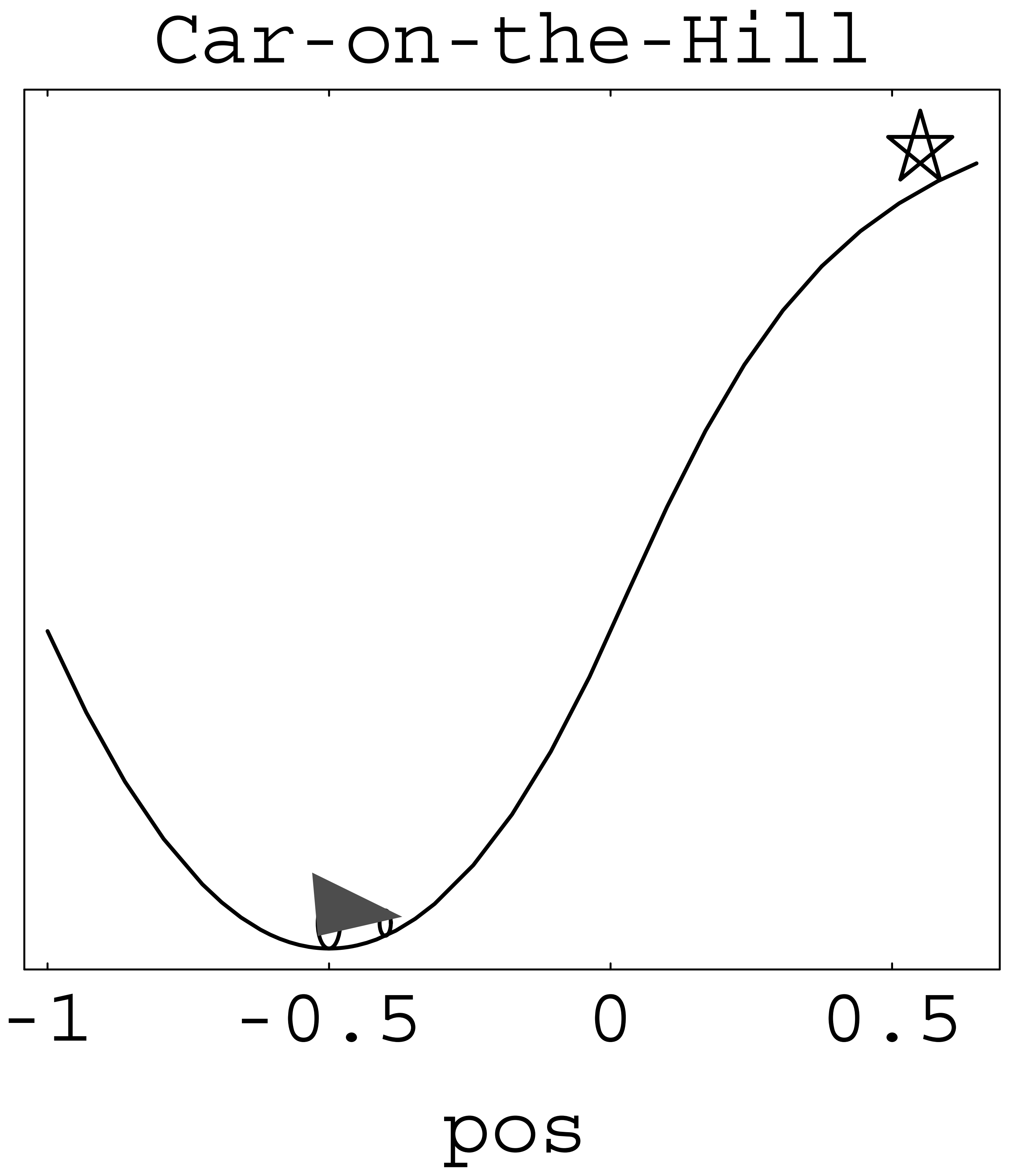
What happens when we run value iteration with a *quadratic*?

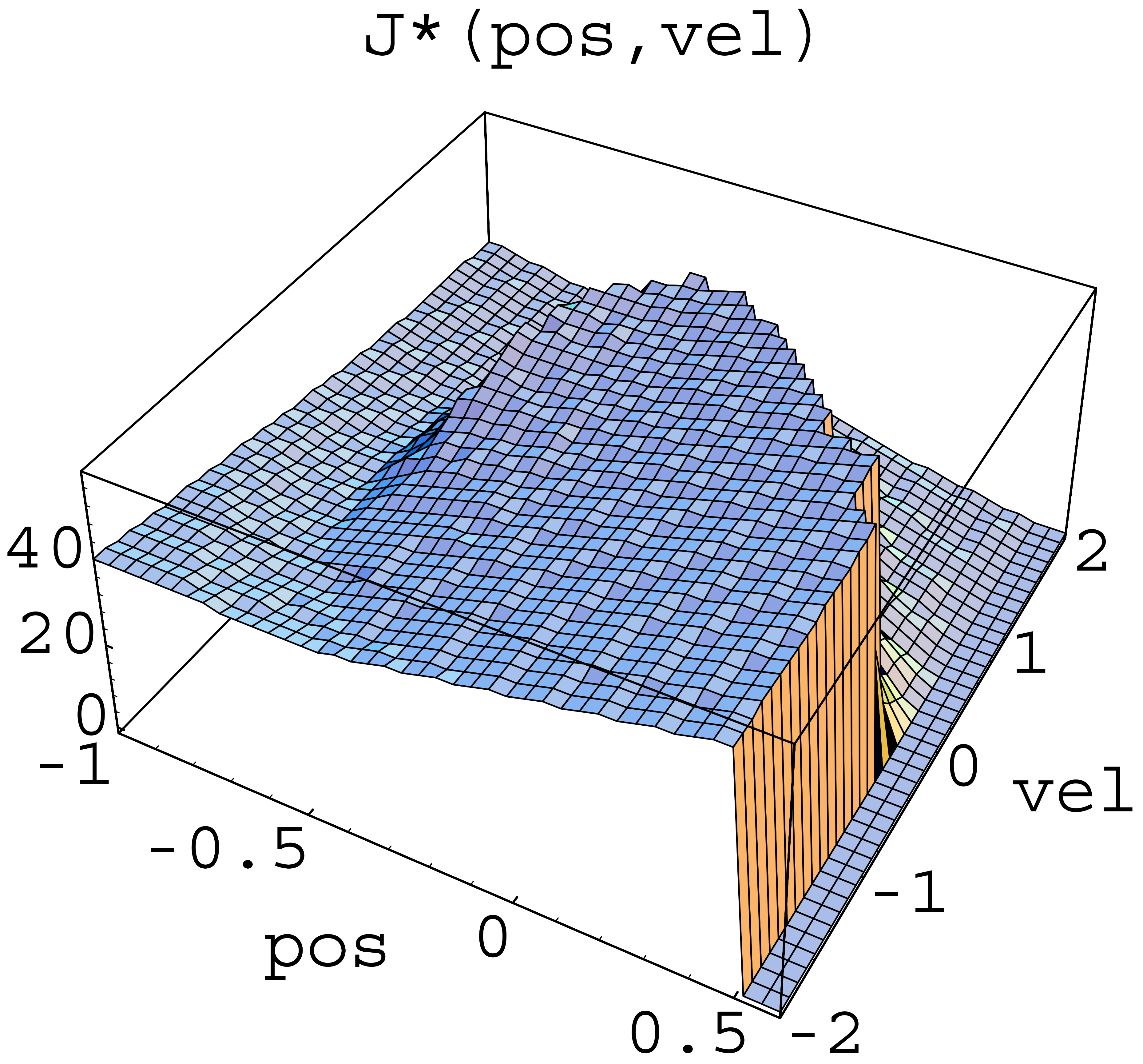


What happens when we run value iteration with a *quadratic*?

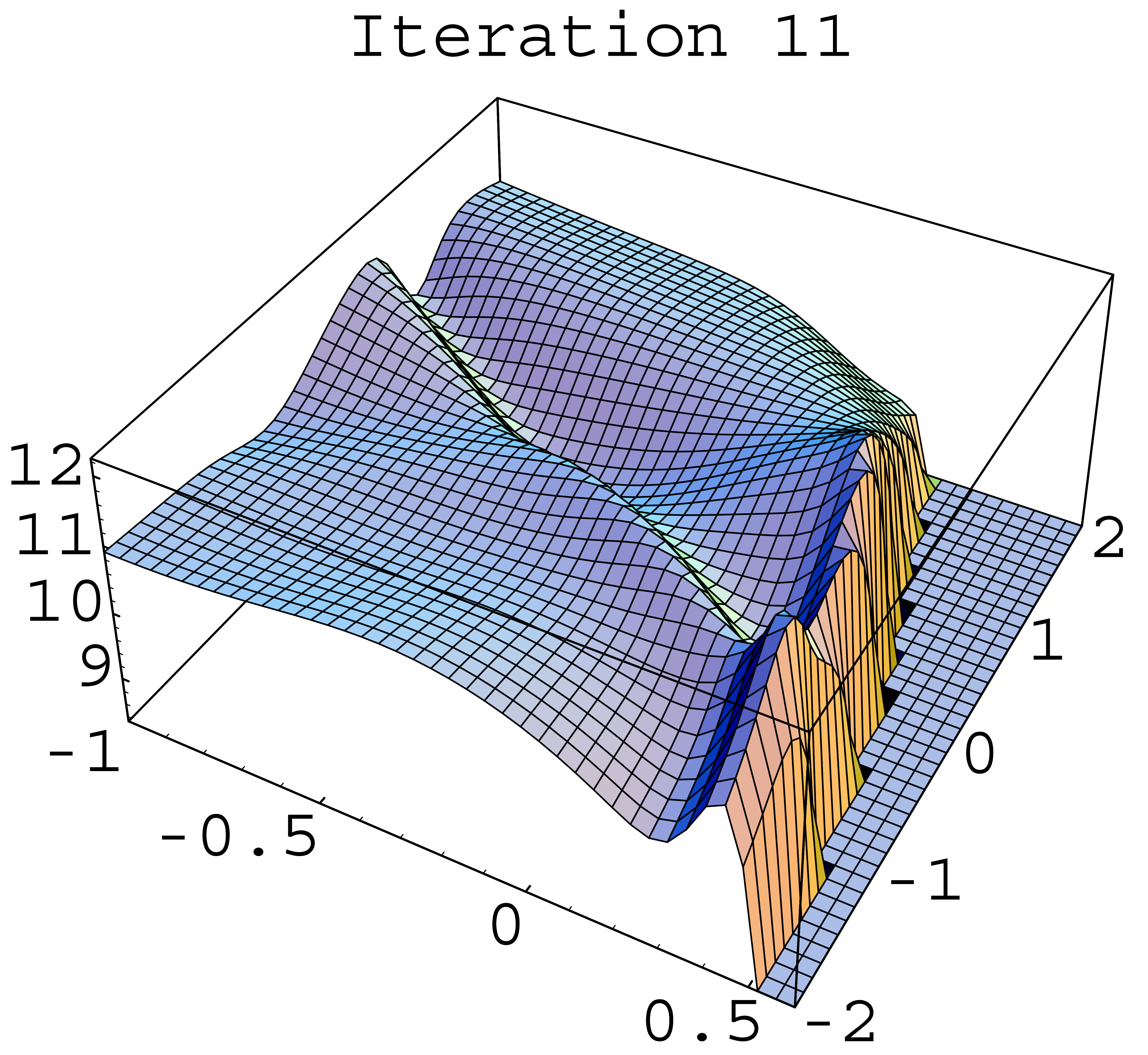


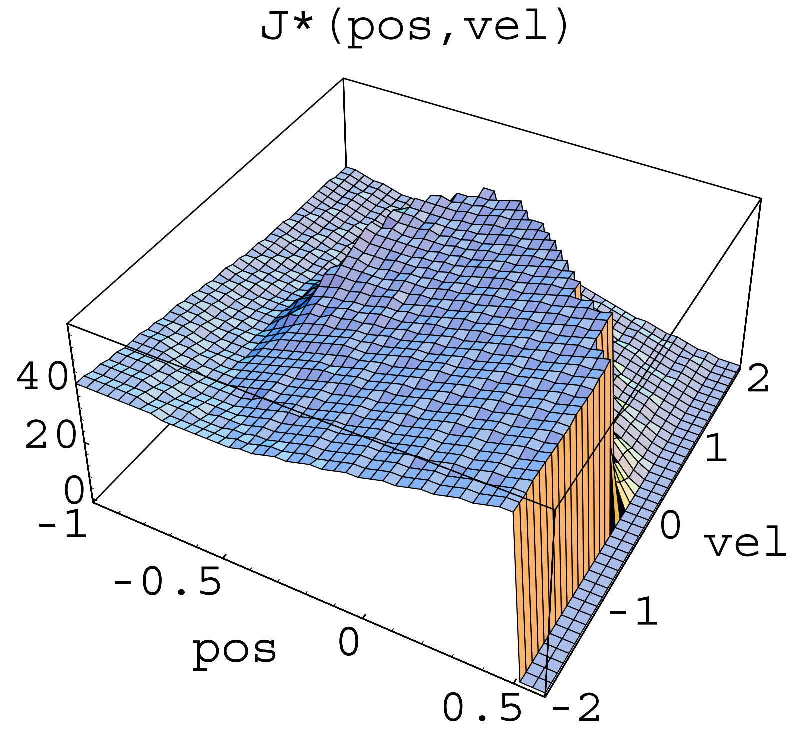
Another Example: Mountain Car!



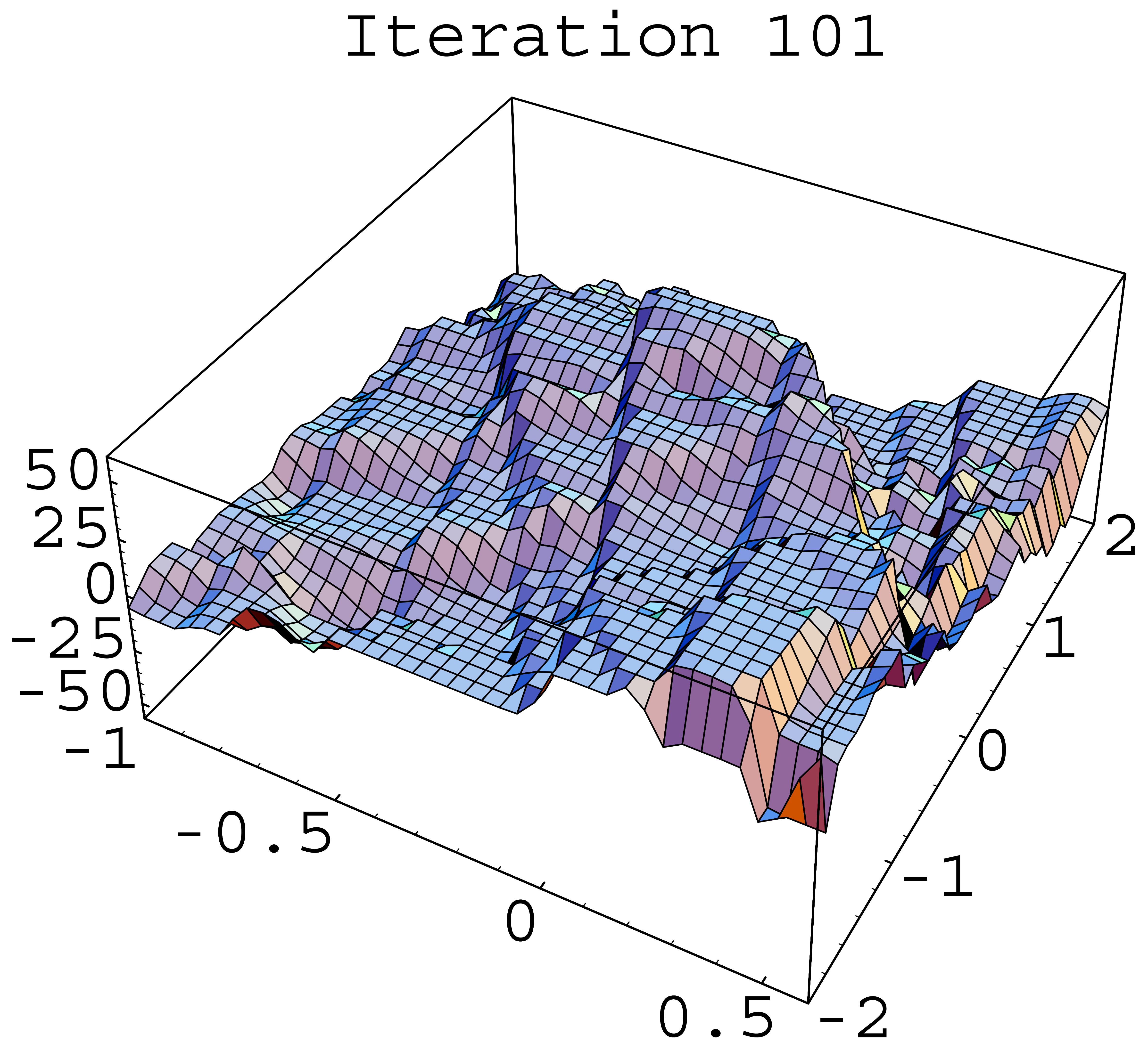


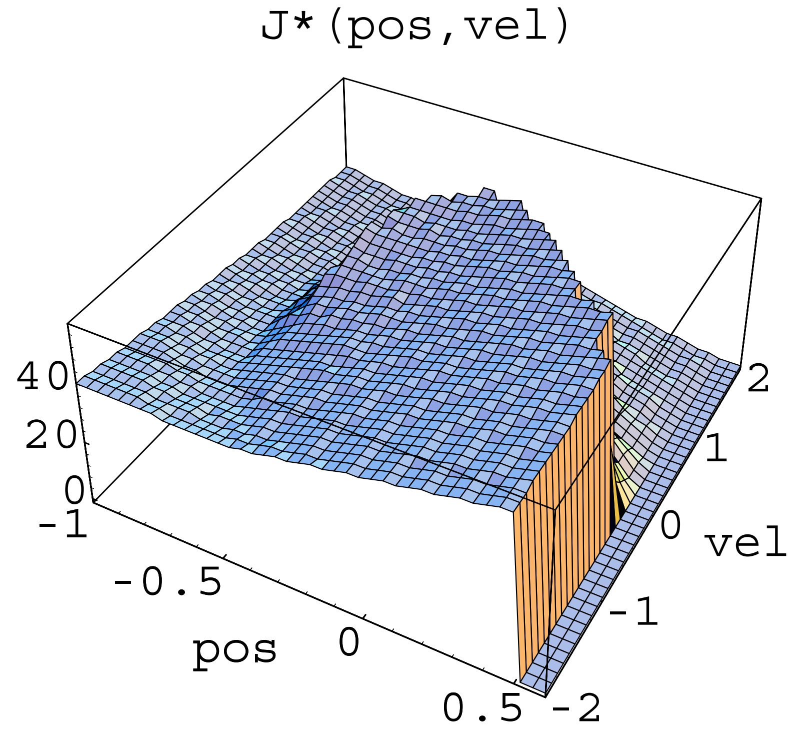
What happens when we run value iteration with a *2 Layer MLP?*



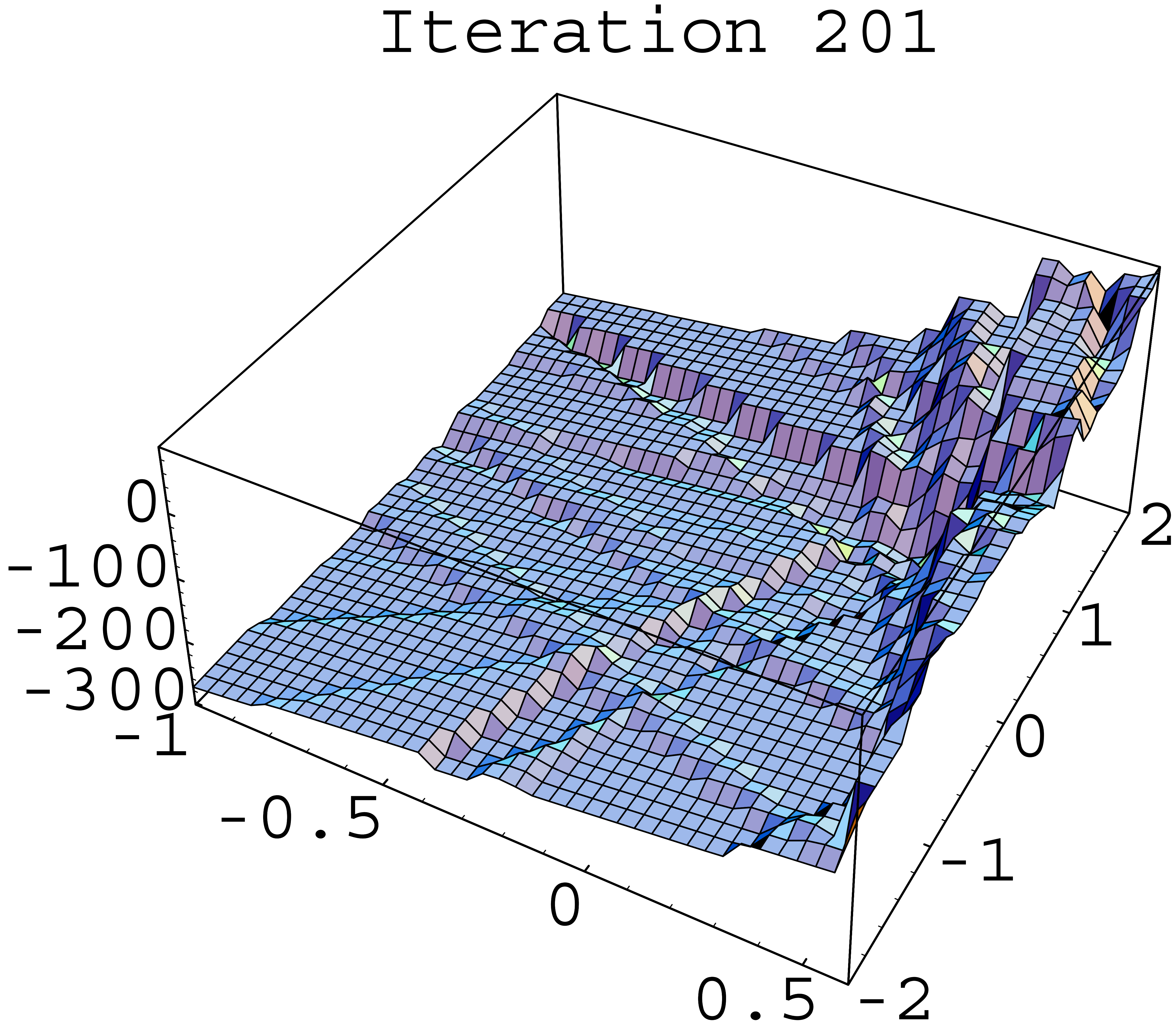


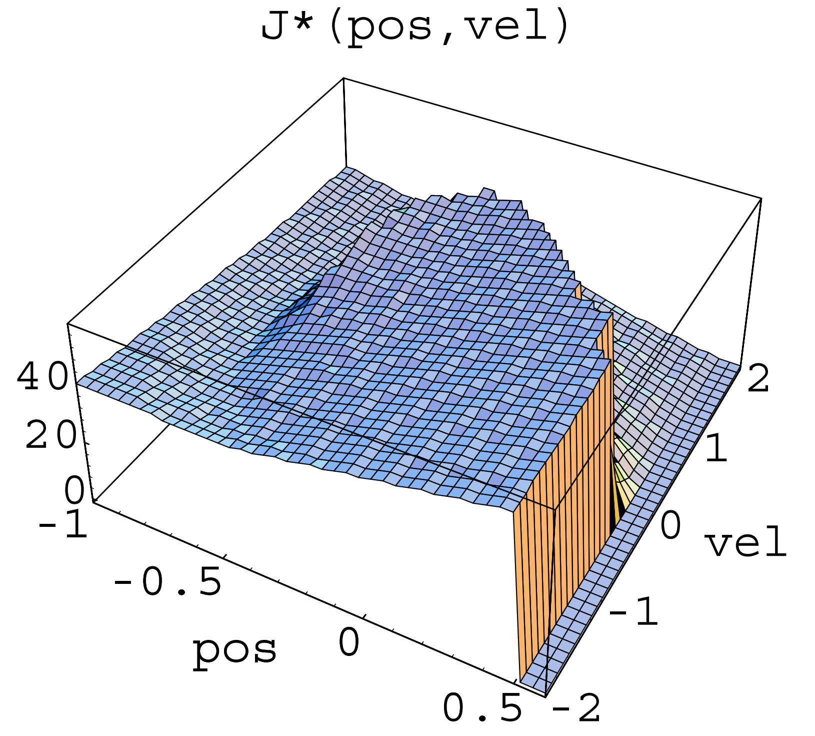
What happens when we run value iteration with a *2 Layer MLP?*





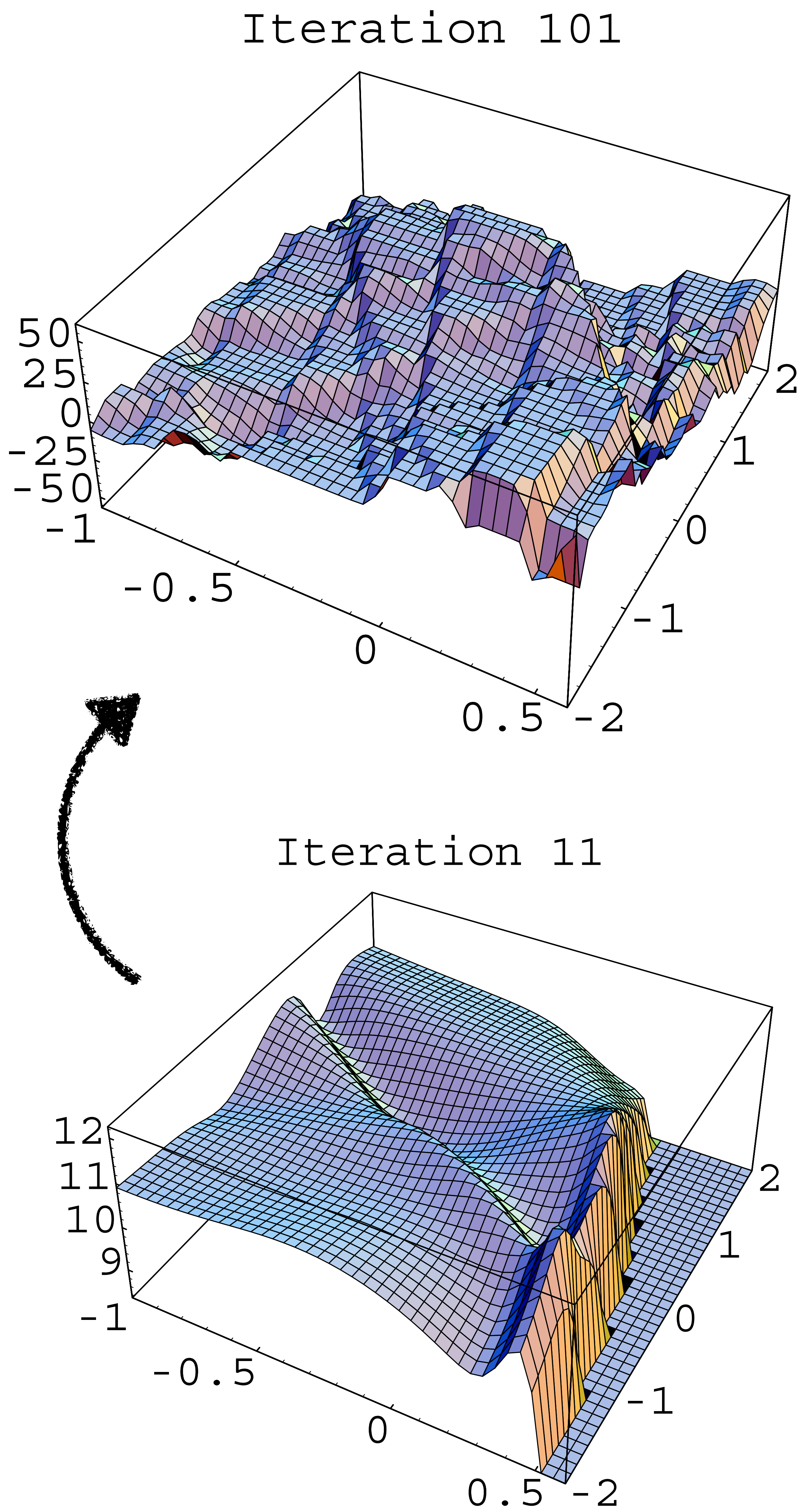
What happens when we run value iteration with a *2 Layer MLP?*





The problem of Bootstrapping!

max()



The problem of Bootstrapping!

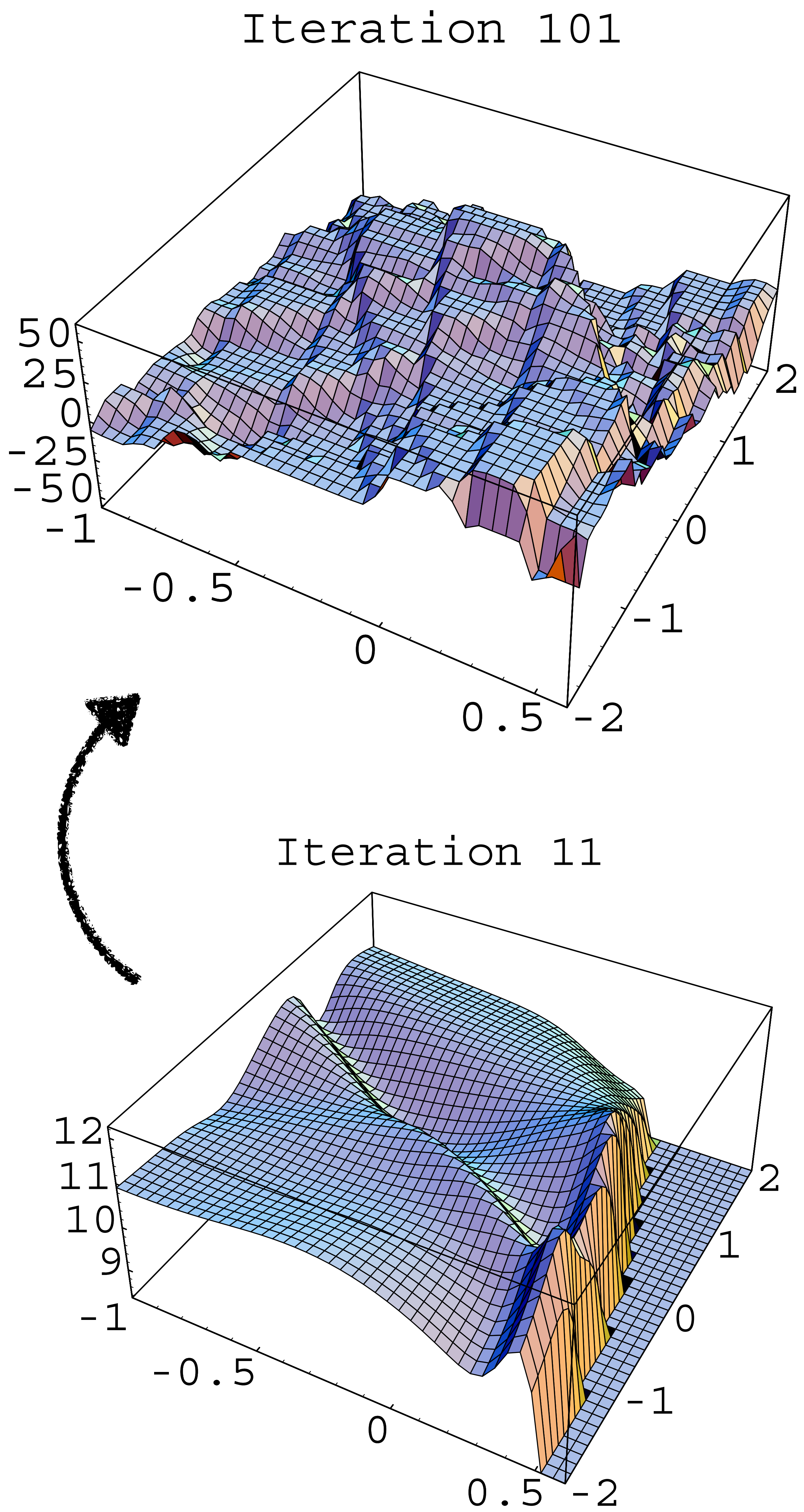
Errors in approximation are amplified

Key reason is the minimization

Minimization operation visit states where approximate values is less than the true value of that state – that is to say, states that look more attractive than they should.

Typically states where you have very few samples

max()

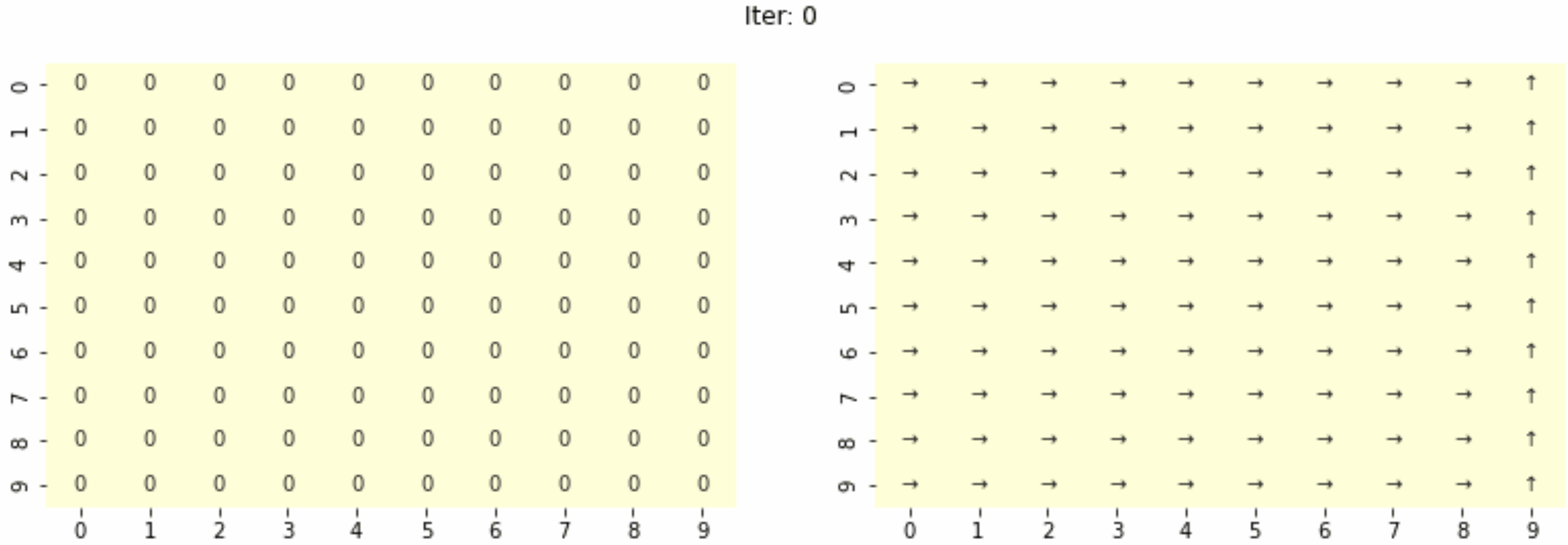


What about policy iteration?

Policy Iteration

Policy Evaluation

Policy Improvement



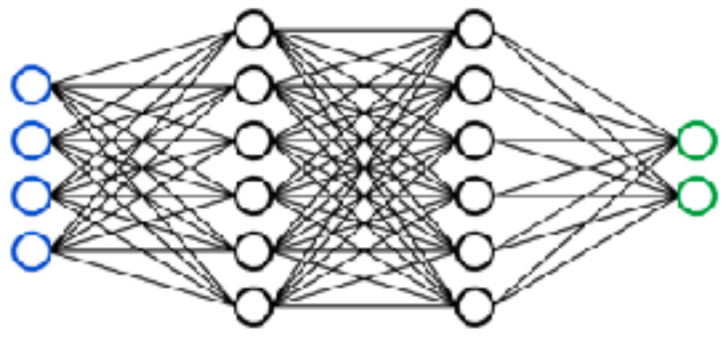
$$Q^{\pi}(s,a) = c(s,a)) + \gamma \mathbb{E}_{s' \sim \mathcal{T}(s,a)} Q^{\pi}(s', \pi(s'))]$$

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

Approximate (Fitted) Policy Iteration

Fitted policy evaluation

Given $\{s_i, a_i, c_i, s'_i\}_{i=1}^N$



Init $Q_\theta(s, a) \leftarrow 0$

while *not converged* **do**

$D \leftarrow \varnothing$

for $i \in 1, \ldots, n$

input $\leftarrow \{s_i, a_i\}$

target $\leftarrow c_i + \gamma Q_\theta(s'_i, \pi(s'_i))$

$D \leftarrow D \cup$ {input, output}

$Q_\theta \leftarrow$ **Train**(D)

return Q_θ

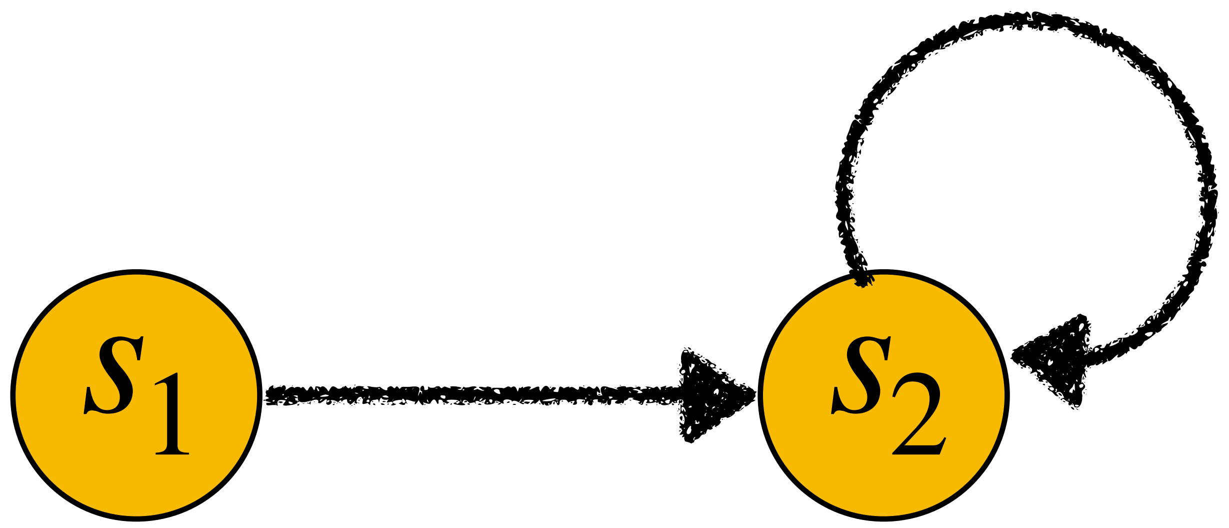
Policy Improvement

This remains the same!

$$\pi^+(s) = \arg\min_a Q^\pi(s, a)$$

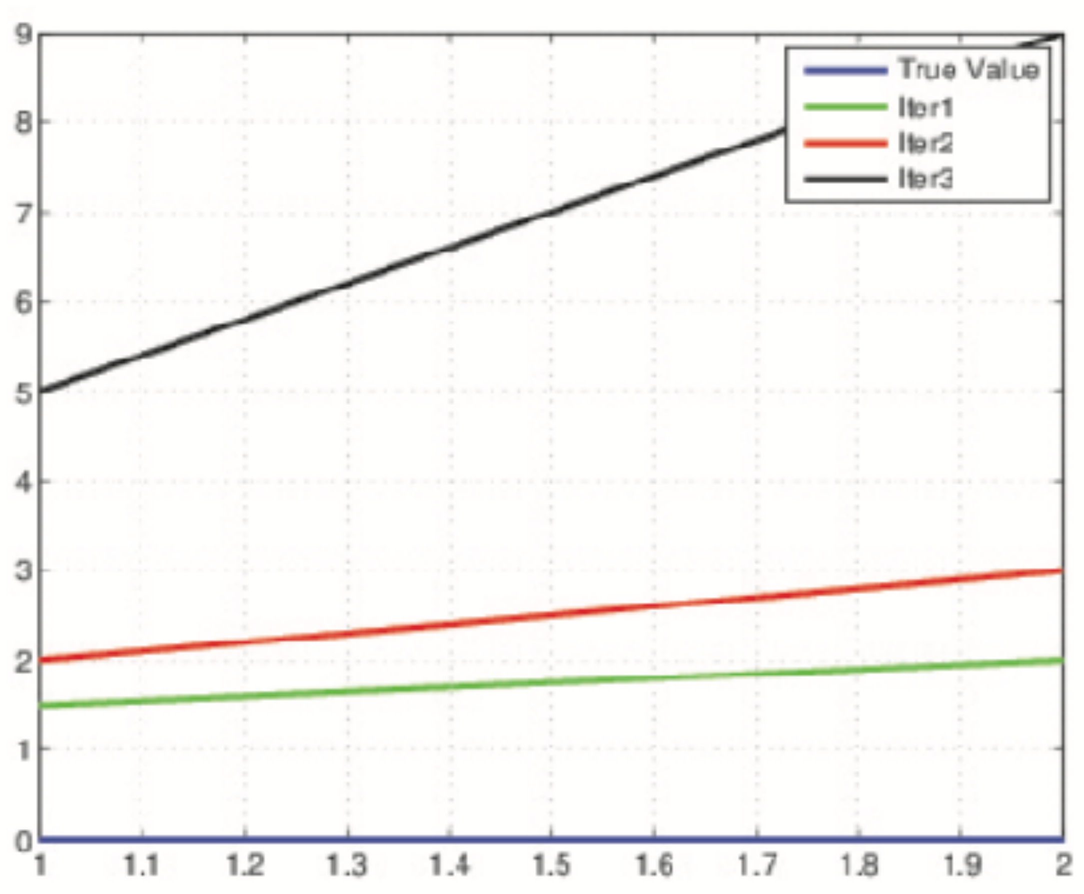
Surely approximate value evaluation is more stable than approximate value iteration? (There is no min()!)

Well … not quite

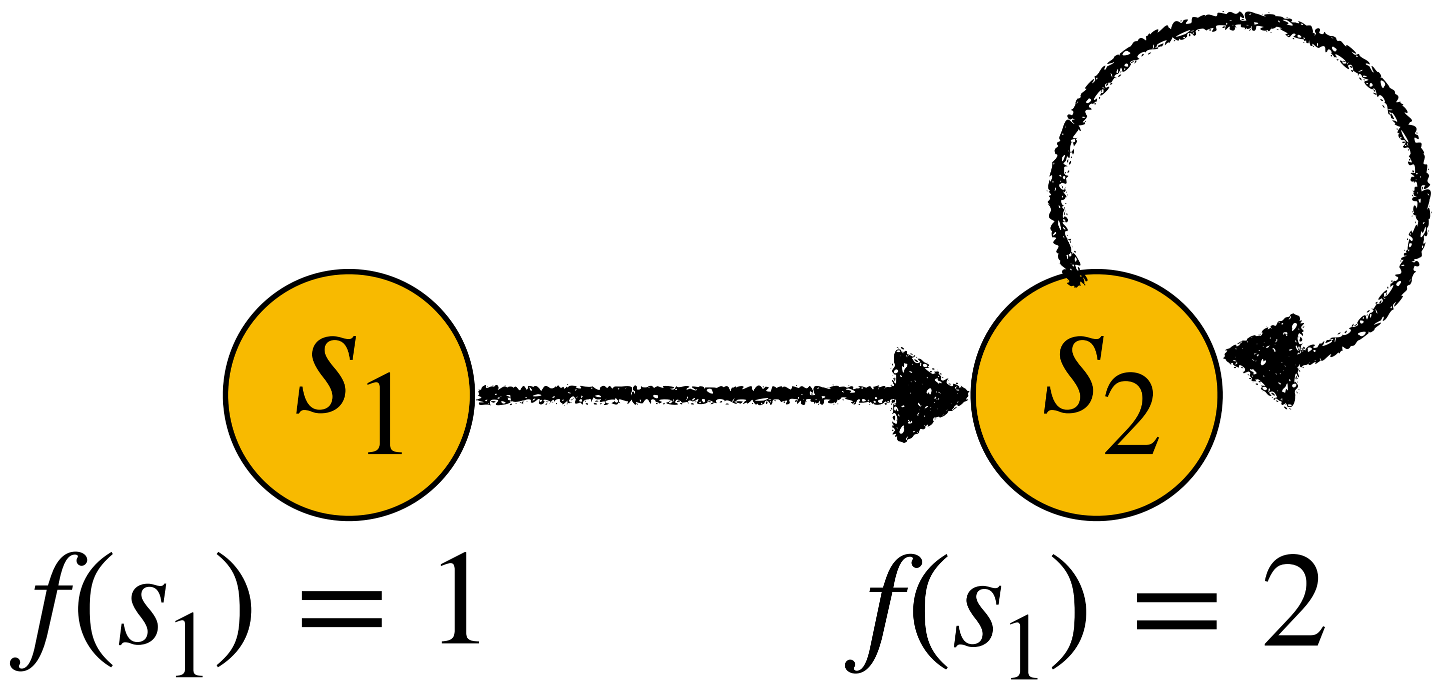


$f(s_1) = 1$ $f(s_1) = 2$

w blows up!

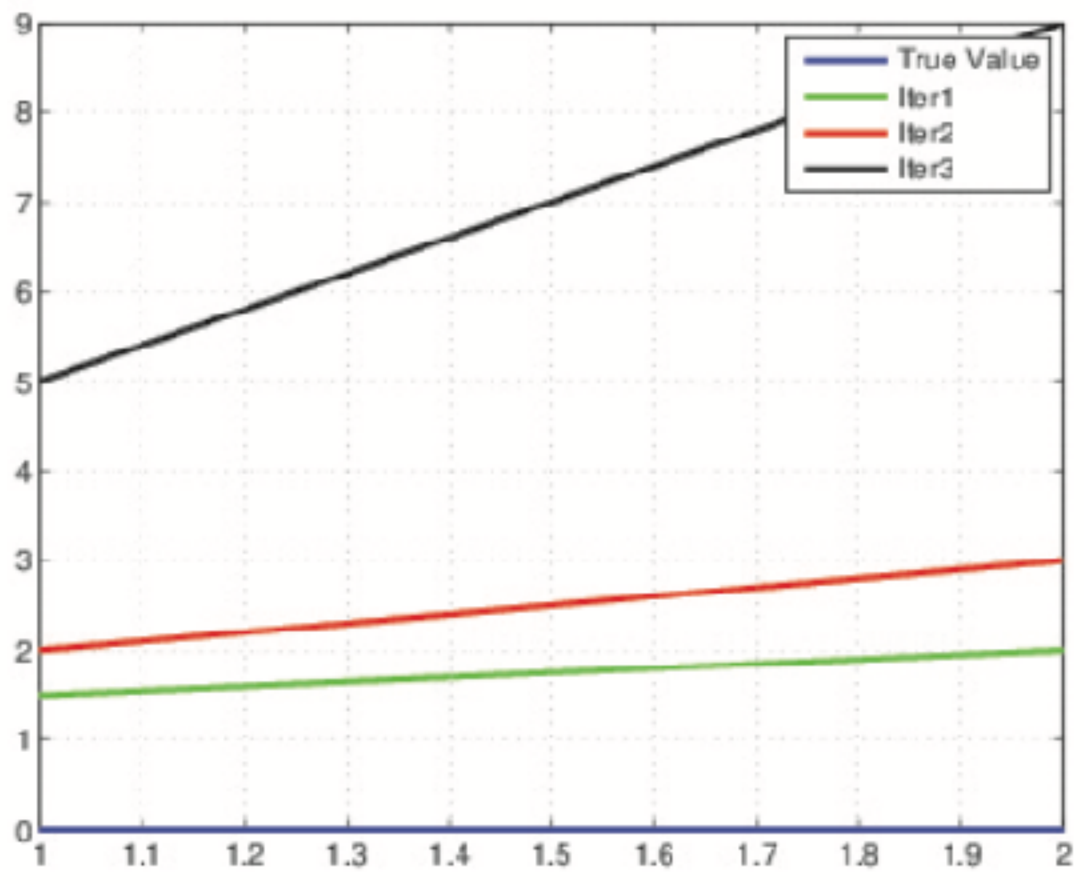


Well … not quite



But we can fix this by *on-policy weighting*

w blows up!



Weight each datapoint by how often the policy visits it.

But what about policy improvement?

Fitted policy evaluation

Given $\{s_i, a_i, c_i, s'_i\}_{i=1}^N$

Init $Q_\theta(s, a) \leftarrow 0$

while *not converged* **do**

$D \leftarrow \emptyset$

for $i \in 1, \ldots, n$

input $\leftarrow \{s_i, a_i\}$

target $\leftarrow c_i + \gamma Q_\theta(s'_i, \pi(s'_i))$

$D \leftarrow D \cup$ {input, output}

$Q_\theta \leftarrow$ **Train**(D)

return Q_θ

This is fine..

Policy Improvement

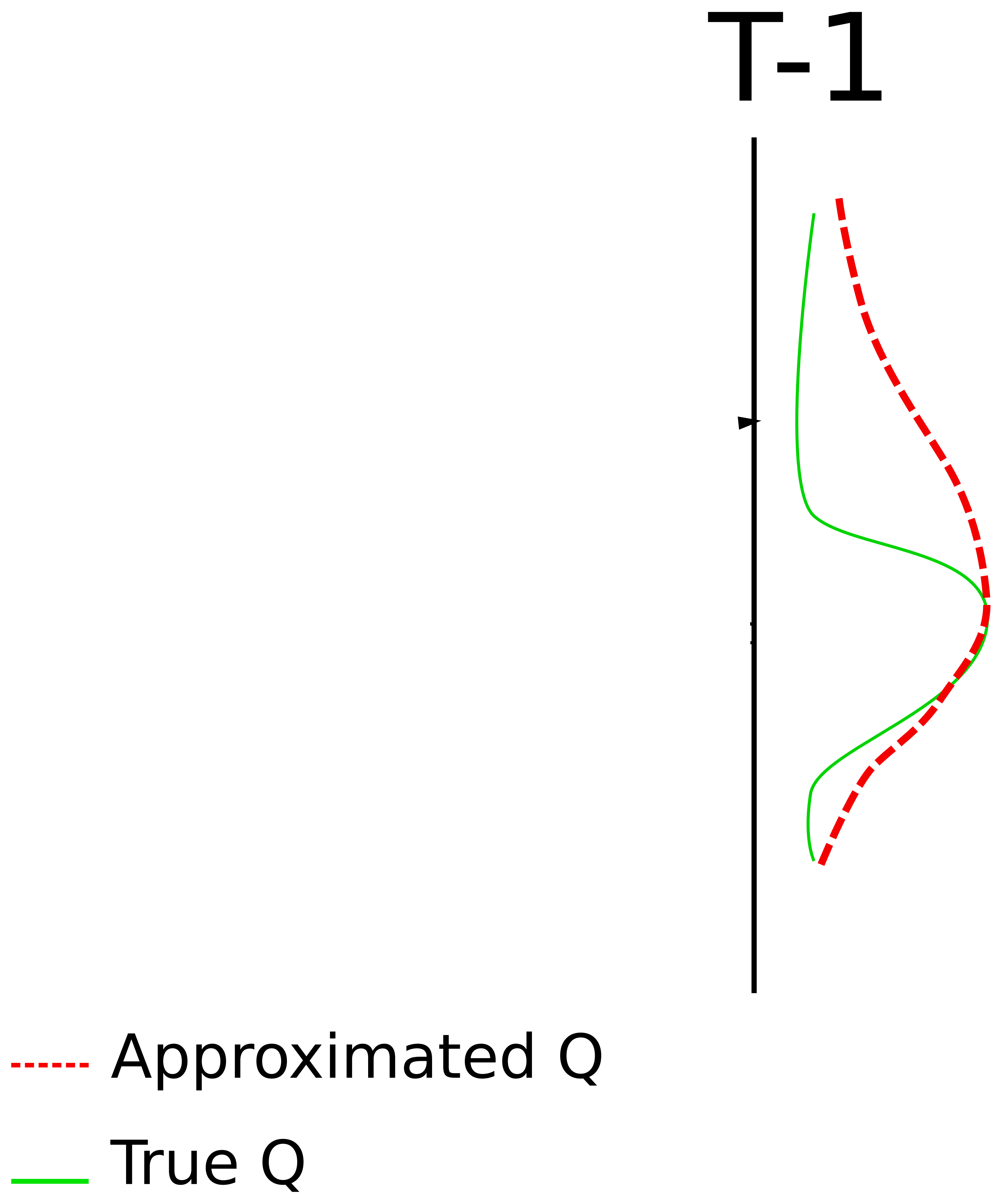
But this has the min() step!

$$\pi^+(s) = \arg\min_a Q^\pi(s, a)$$

The problem of distribution shift

Upper half of state
is BAD

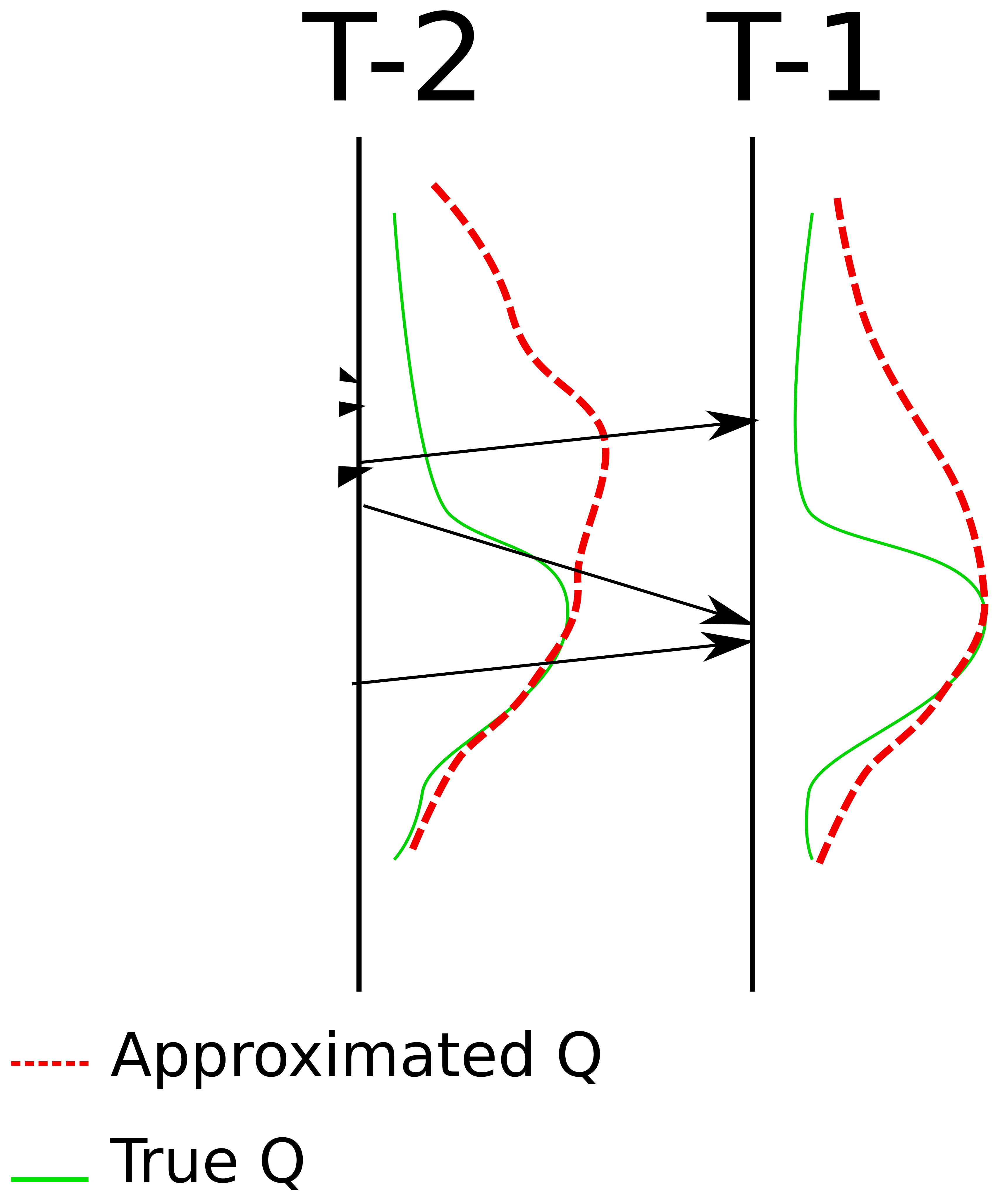
Lower half of state
is GOOD



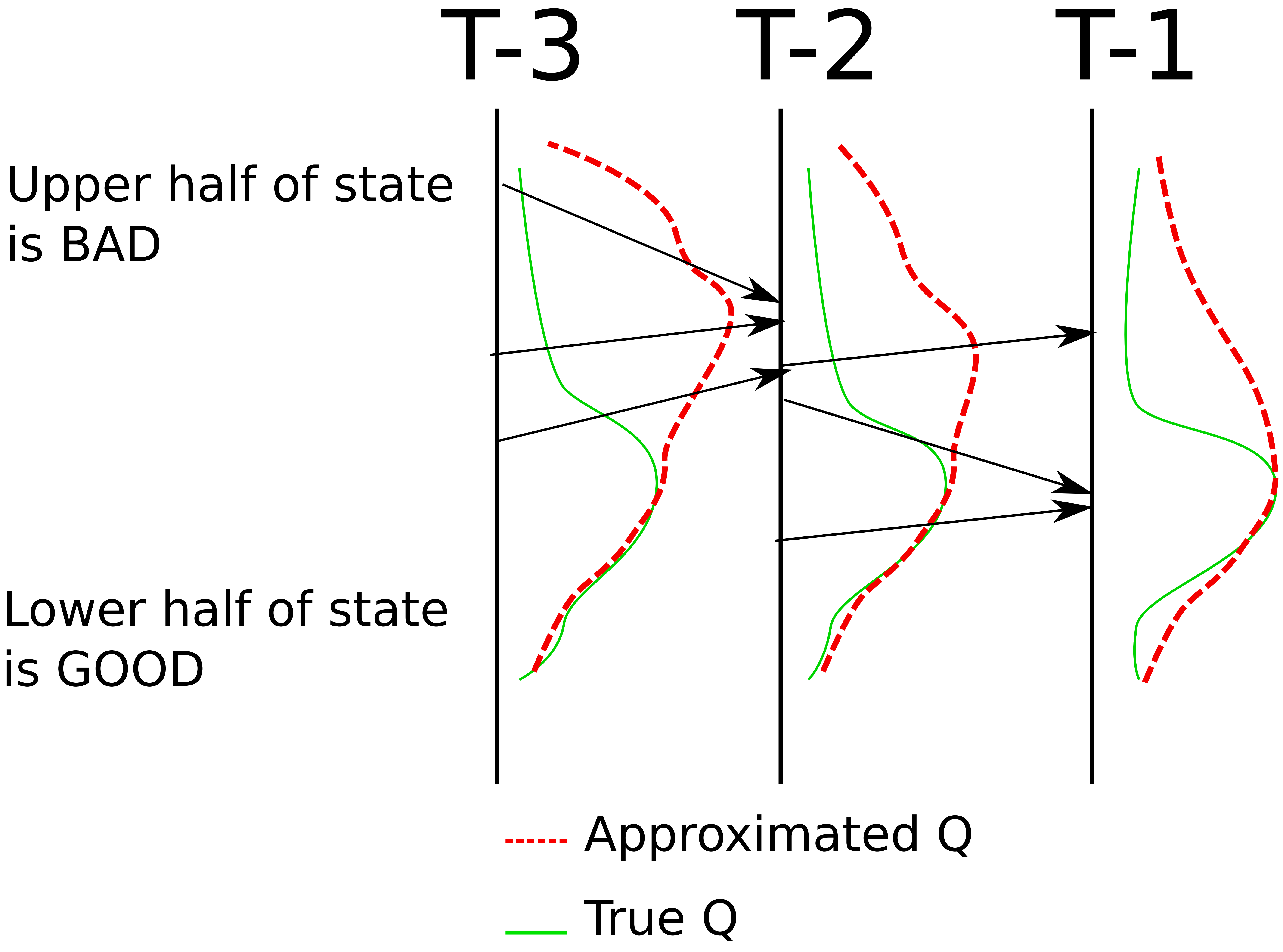
The problem of distribution shift

Upper half of state
is BAD

Lower half of state
is GOOD



The problem of distribution shift

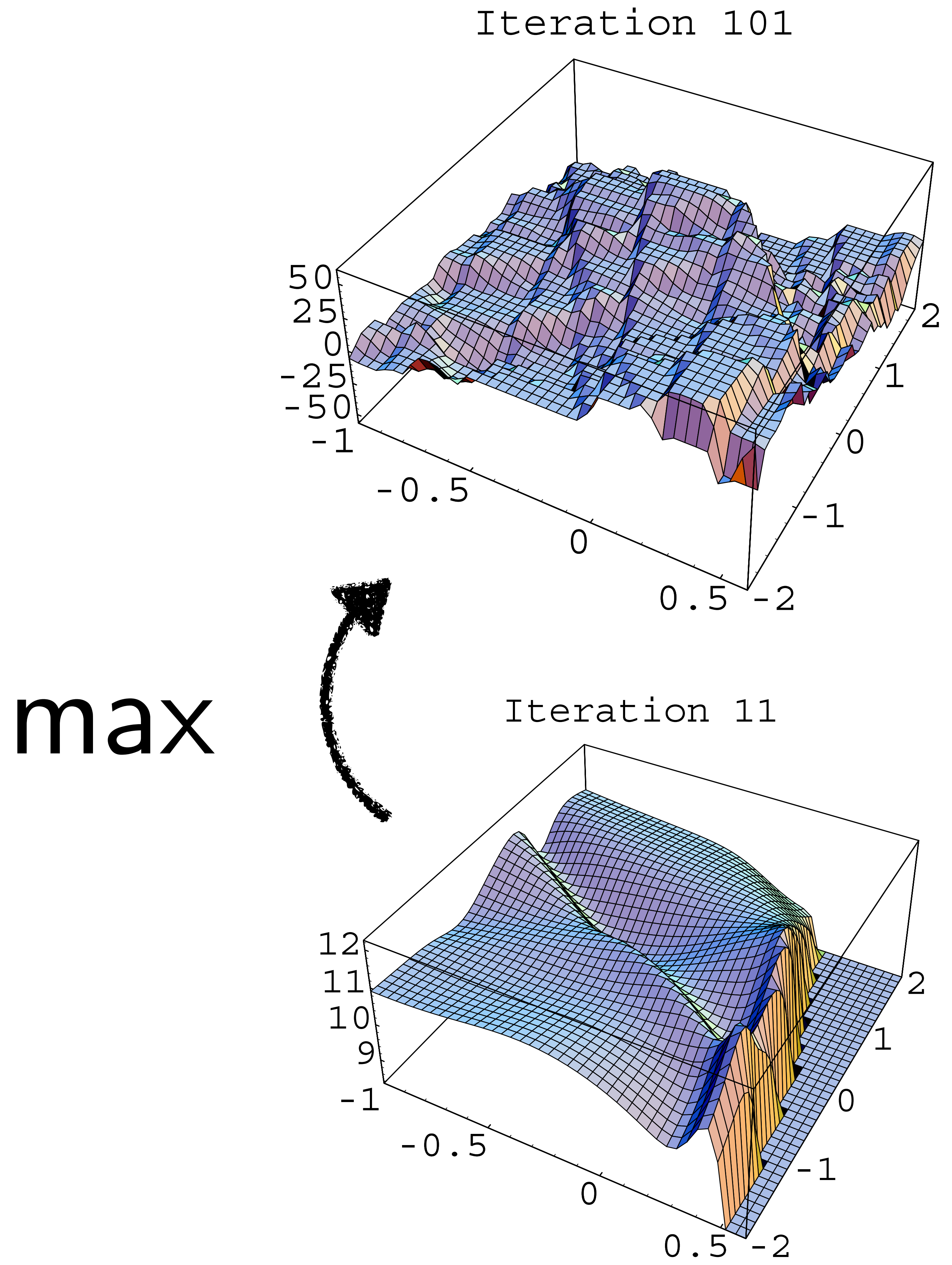


Boostrapping

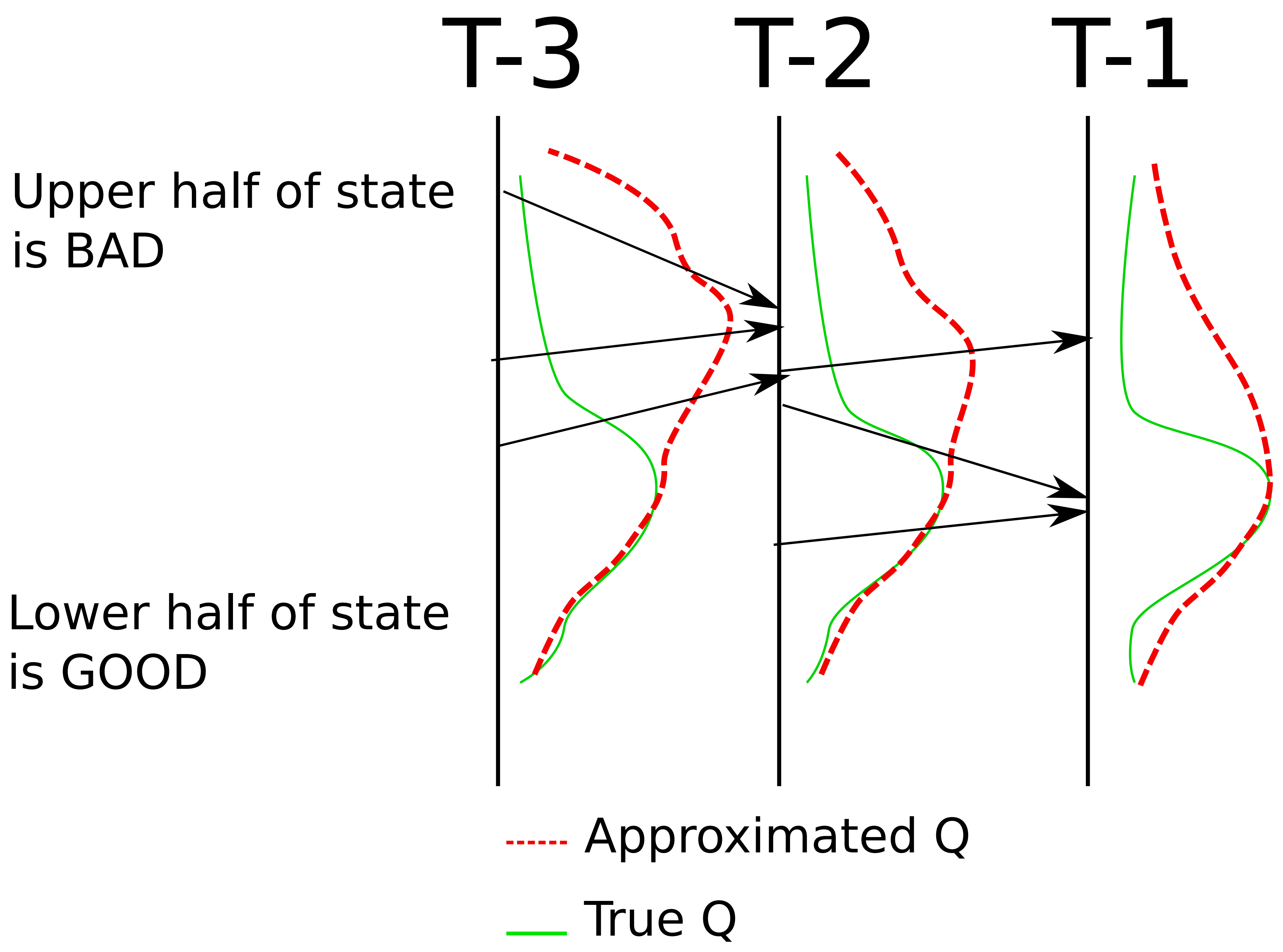
Distribution Shift

Two sides of the same coin

Bootstrapping

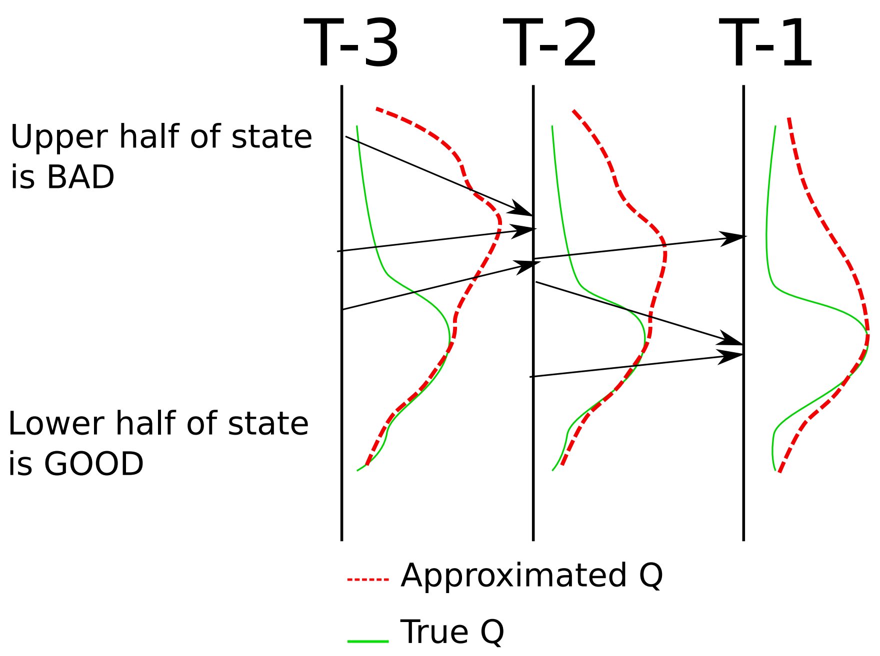


Distribution shift

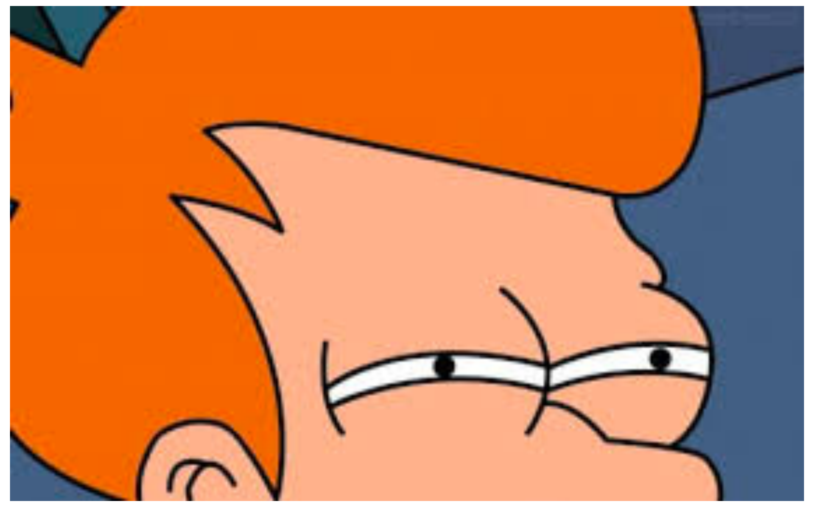


Ideas for fixing this?

Remedies



Bootstrapping



When doing min(), don't trust value estimate

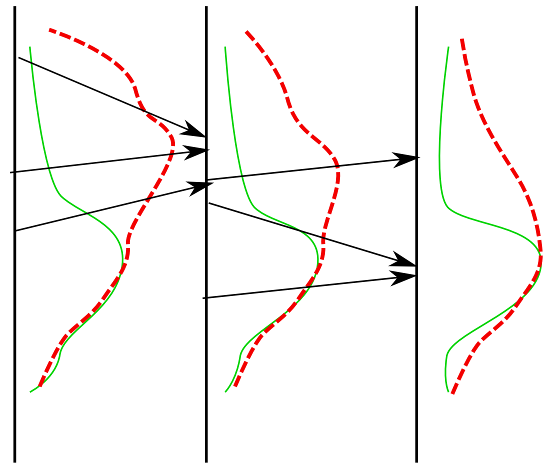
Execute policy and trust actual returns

Distribution shift

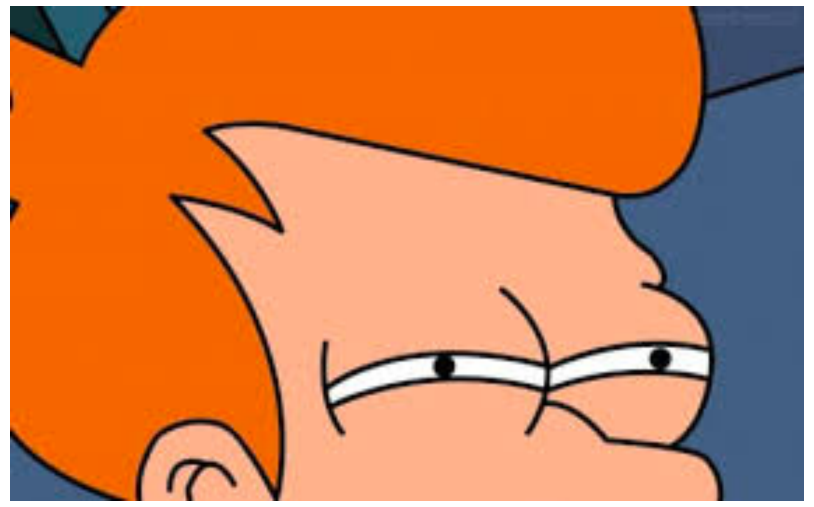
Minimize the distribution shift

Be conservative, change policy slowly

Remedies



Bootstrapping



When doing min(),
don't trust value estimate

Execute policy
and trust actual returns

Distribution shift

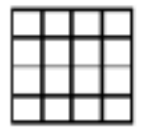
Minimize the
distribution shift

Be conservative,
change policy slowly

tl;dr

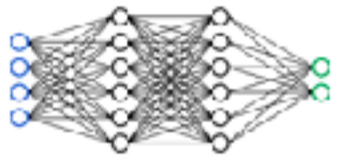
Approximate (Fitted) Value Iteration

Q-iteration



$Q(s,a) \leftarrow 0$
while *not converged* **do**
  for $s \in S, a \in A$
    $Q^{new}(s,a) = c(s,a) + \gamma \mathbb{E}_{s'} \min_{a'} Q(s',a')$
  $Q \leftarrow Q^{new}$
return Q

***Fitted* Q-iteration**

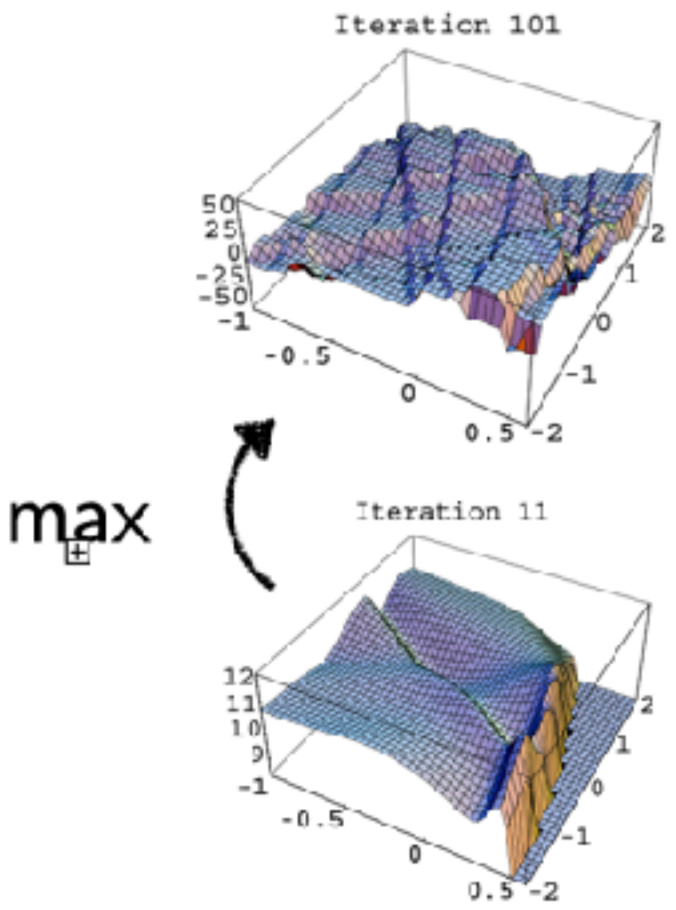


Given $\{s_i, a_i, c_i, s_i'\}_{i=1}^N$

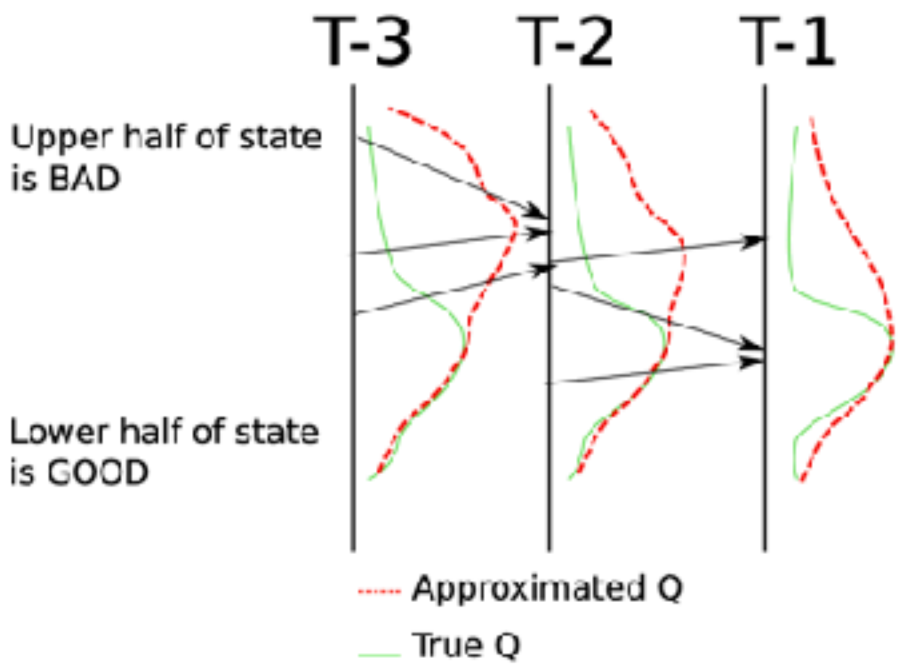
Init $Q_\theta(s,a) \leftarrow 0$
while *not converged* **do**
  $D \leftarrow \varnothing$
  for $i \in 1, \ldots, n$
    input $\leftarrow \{s_i, a_i\}$
    target $\leftarrow c_i + \gamma \min_{a} Q_\theta(s_i', a')$
    $D \leftarrow D \cup \{\text{input}, \text{output}\}$
  $Q_\theta \leftarrow$ **Train**(D)
return Q_θ

1

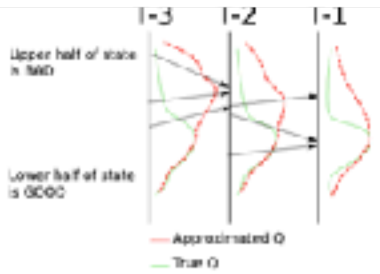
Bootstrapping



Distribution shift



Remedies



Bootstrapping

When doing min(), don't trust value estimate

Execute policy and trust actual returns

Distribution shift

Minimize the distribution shift

Be conservative, change policy slowly