# Artificial Intelligence

CS 165A

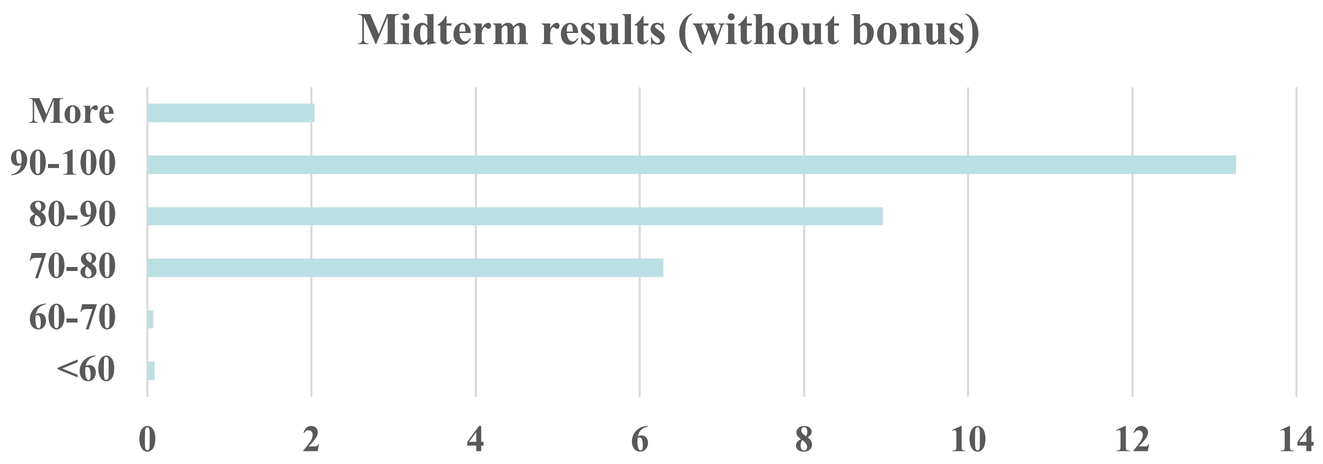
Nov 12, 2020

Instructor: Prof. Yu-Xiang Wang

T
o
d
a
y

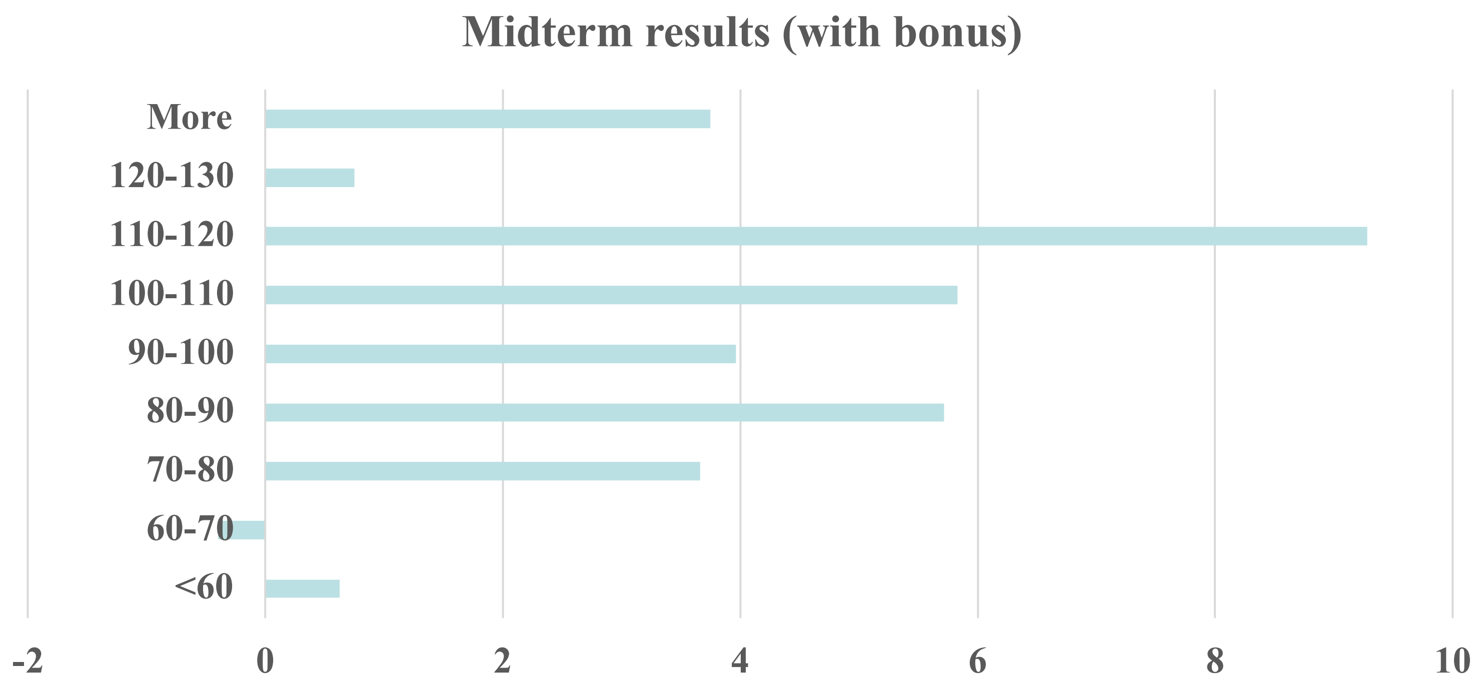
→ Markov Decision Processes

# Midterm Results




(Histogram is sanitized using Differential Privacy)

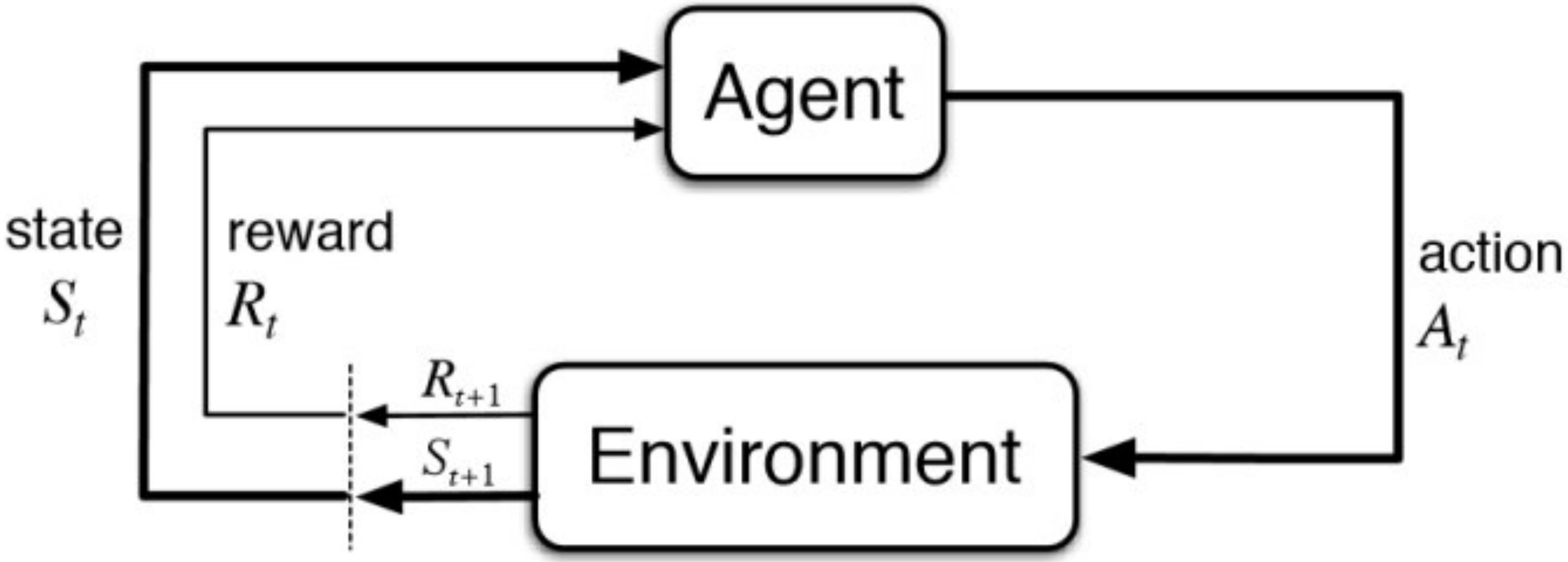
# Midterm Results (with bonus)




(Histogram is sanitized using Differential Privacy)

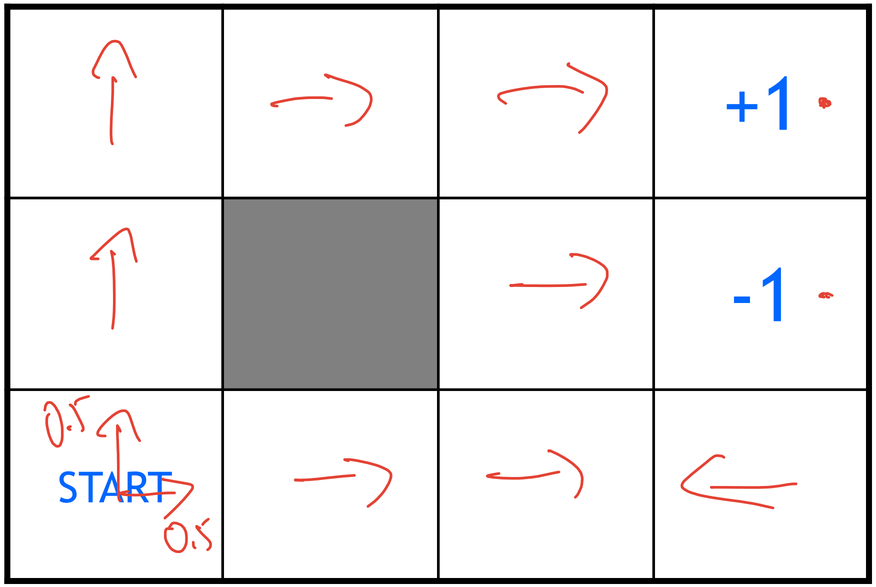
# Recap: Reinforcement learning problem setup

- State, Action, Reward
- Unknown reward function, unknown state-transitions.
- Agents might not even observe the state



# Recap: Robot in a room.

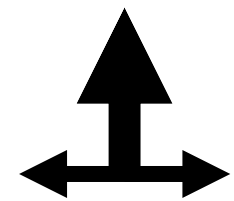
Policy : $\pi : S \to A$
$: S \to P_A$




actions: UP, DOWN, LEFT, RIGHT

e.g.,

UP



State-transitions with action **UP**:

| | |
|---|---|
| 80% | move up |
| 10% | move left |
| 10% | move right |

*If you bump into a wall, you stay where you are.

- reward +1 at [4,3], -1 at [4,2]
- reward -0.04 for each step
- what's the strategy to achieve max reward?
- what if the transitions were deterministic?

# Recap: Tabular MDP

- Discrete State, Discrete Action, Reward and Observation

$$S_t \in \mathcal{S} \quad A_t \in \mathcal{A} \quad R_t \in \mathbb{R} \quad \mathcal{O}_t \in \mathcal{O}$$

- Policy:
  - When the state is observable: $\pi : \mathcal{S} \rightarrow \mathcal{A}$
  - Or when the state is not observable

$$\pi_t : (\mathcal{O} \times \mathcal{A} \times \mathbb{R})^{t-1} \rightarrow \mathcal{A}$$

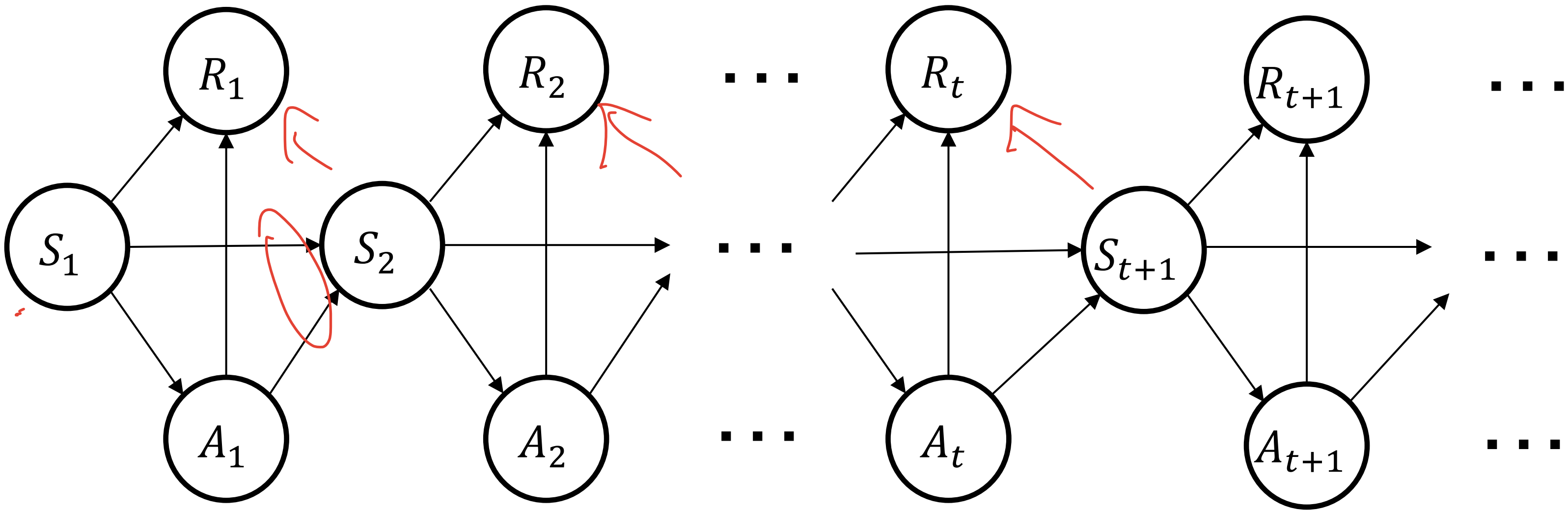
- Learn the best policy that maximizes the expected reward
  - Finite horizon (episodic) RL: $\pi^* = \arg\max_{\pi \in \Pi} \mathbb{E}[\sum_{t=1}^{T} R_t]$
  - Infinite horizon RL: $\pi^* = \arg\max_{\pi \in \Pi} \mathbb{E}[\sum_{t=1}^{\infty} \gamma^{t-1} R_t]$

T: horizon

$\lim_{T \to \infty} \mathbb{E}[\frac{1}{T} \sum_{t=1}^{\infty} R_t]$

γ: discount factor

# Recap: Parameters of an MDP are the CPTs




- Initial state distribution $P(S_1) =: d_1 \in \mathbb{R}^{|S|}$
- Transition dynamics $P(S_{t+1} | S_t, A_t) =: \mathbb{P} \in \mathbb{R}^{|S| \times |S||A|}$
- Reward distribution $\mathbb{E}[R_t | S_t, A_t] =: r(s_t, a_t)$

$\mathbb{P}(R_t | S_t, A_t)$

$\mathbb{E}[R_t | S_t, A_t, S_{t+1}] =: r(s_t, a_t, s_{t+1})$

↑ AIMA book

# Recap: Reward function and Value functions

- Immediate reward function r(s,a,s')
  - expected **immediate** reward

$$r(s,a,s') = \mathbb{E}[R_1 | S_1 = s, A_1 = a, S_2 = s']$$

$$r^{\pi}(s) = \mathbb{E}_{a \sim \pi(a|s)}[R_1 | S_1 = s]$$

- state value function: $V^{\pi}(s)$
  - expected **long-term** return when starting in *s* and following π

$$V^{\pi}(s) = \mathbb{E}_{\pi}[R_1 + \gamma R_2 + ... + \gamma^{t-1} R_t + ... | S_1 = s]$$

- state-action value function: $Q^{\pi}(s,a)$
  - expected **long-term** return when starting in *s*, performing *a*, and following π

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}[R_1 + \gamma R_2 + ... + \gamma^{t-1} R_t + ... | S_1 = s, A_1 = a]$$

# Recap: Bellman equations – the fundamental equations of MDP and RL

- An alternative, recursive and more useful way of defining the V-function and Q function

$$V^\pi(s) = \sum_a \pi(a|s) \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma V^\pi(s')] = \sum_a \pi(a|s) Q^\pi(s,a)$$

# Recap: Bellman equations – the fundamental equations of MDP and RL

- An alternative, recursive and more useful way of defining the V-function and Q function

$$V^\pi(s) = \sum_a \pi(a|s) \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma V^\pi(s')] = \sum_a \pi(a|s) Q^\pi(s,a)$$

- Quiz:
  - Prove Bellman equation from the definition in the previous slide.
  - Write down the Bellman equation using Q function alone.

$$Q^\pi(s,a) = ?$$

$$Q^\pi(s,a) = E_\pi[R_1 + \gamma R_2 + \dots + \gamma^{t-1} R_t + \dots | S_1 = s, A_1 = a]$$

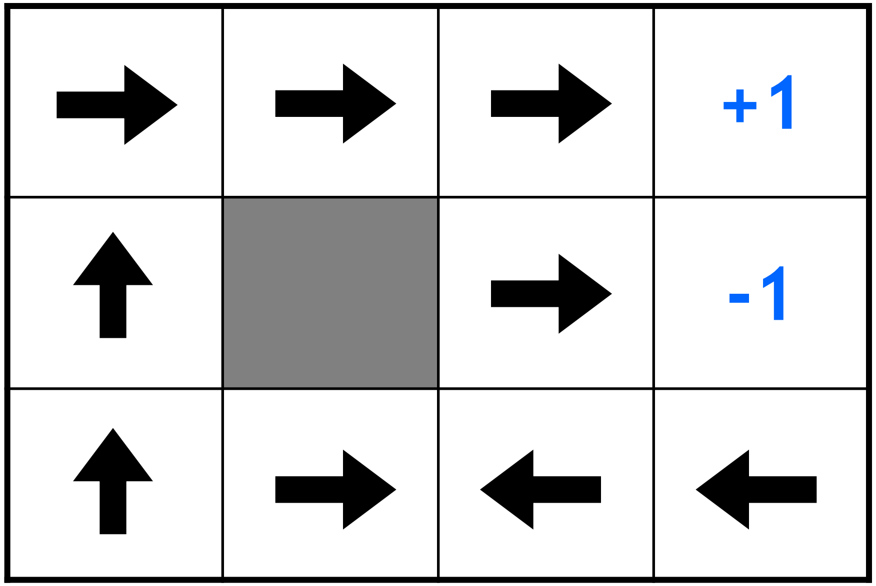
$$= E_\pi[R_1 | S_1 = s, A_1 = a] + \gamma E_\pi[R_2 + \gamma R_3 + \dots + \gamma^{t-1} R_{t+1} \dots]$$

$$= E_\pi[R_1 + \gamma E_\pi[Q(S_2, A_2)] | S_1 = s, A_1 = a]$$

# This lecture

- Bellman equations
- Algorithms for solving MDPs
  - Value iterations / Policy Iterations
- Exploration and Bandit problem

# Let’s work out the Value function for a specific policy




actions: UP, DOWN, LEFT, RIGHT

e.g., **UP**

state-transitions with action **UP**:

| | |
|---|---|
| 80% | move UP |
| 10% | move LEFT |
| 10% | move RIGHT |

*If you bump into a wall, you stay where you are.

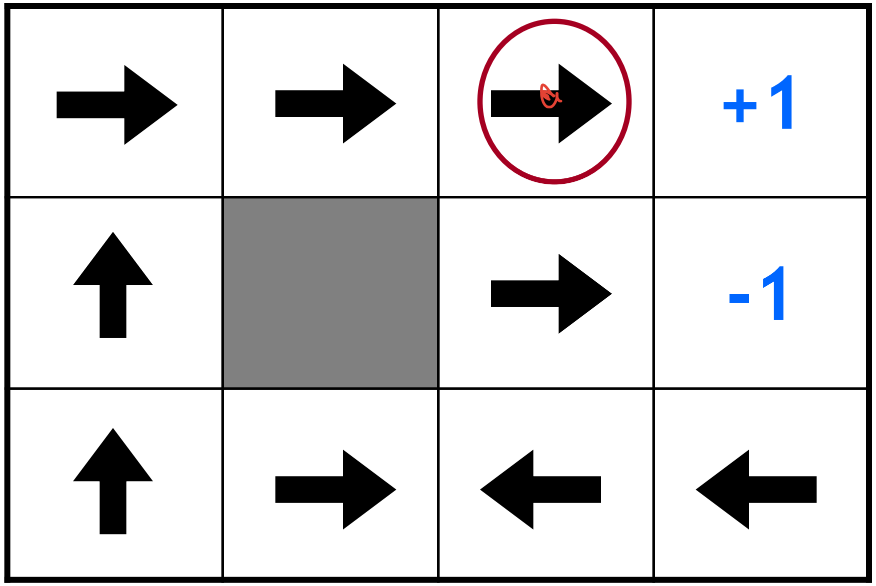
- reward +1 at [4,3], -1 at [4,2]
- reward -0.04 for each step

$$V^\pi(s) = \sum_a \pi(a|s) \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma V^\pi(s')] = \sum_a \pi(a|s) Q^\pi(s,a)$$

+

+

# Let's work out the Value function for a specific policy




actions: UP, DOWN, LEFT, RIGHT

e.g., **UP**

state-transitions with action **UP**:

| | |
|---|---|
| 80% | move UP |
| 10% | move LEFT |
| 10% | move RIGHT |

*If you bump into a wall, you stay where you are.

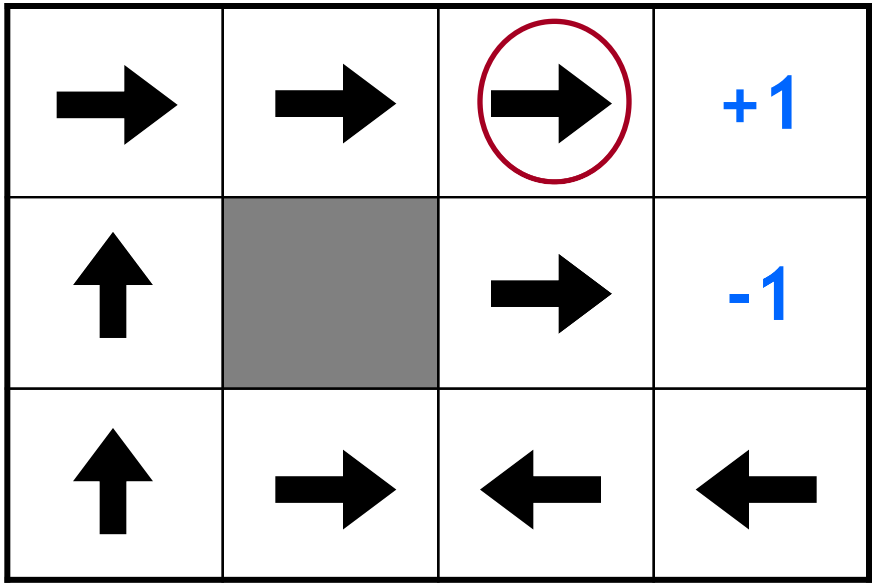
- reward +1 at [4,3], -1 at [4,2]
- reward -0.04 for each step

$$V^\pi(s) = \sum_a \pi(a|s) \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma V^\pi(s')] = \sum_a \pi(a|s) Q^\pi(s,a)$$

+

+

# Let's work out the Value function for a specific policy




actions: UP, DOWN, LEFT, RIGHT

e.g., **UP**

state-transitions with action **UP**:

80% move UP

10% move LEFT

10% move RIGHT

*If you bump into a wall, you stay where you are.

- reward +1 at [4,3], -1 at [4,2]
- reward -0.04 for each step

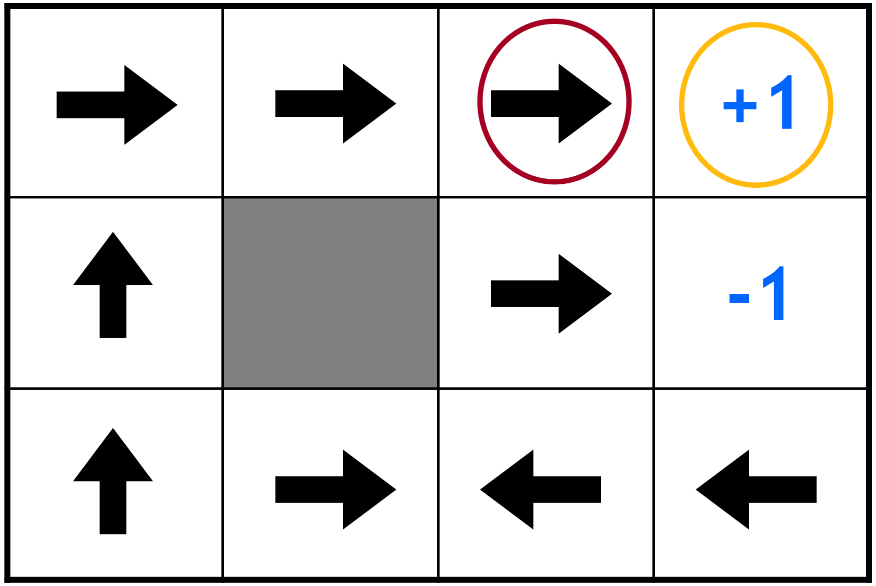
$$V^\pi(s) = \sum_a \pi(a|s) \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma V^\pi(s')] = \sum_a \pi(a|s) Q^\pi(s,a)$$

1.0

+

+

# Let's work out the Value function for a specific policy




actions: UP, DOWN, LEFT, RIGHT

e.g., **UP**

state-transitions with action **UP**:

80% move UP

10% move LEFT

10% move RIGHT

*If you bump into a wall, you stay where you are.

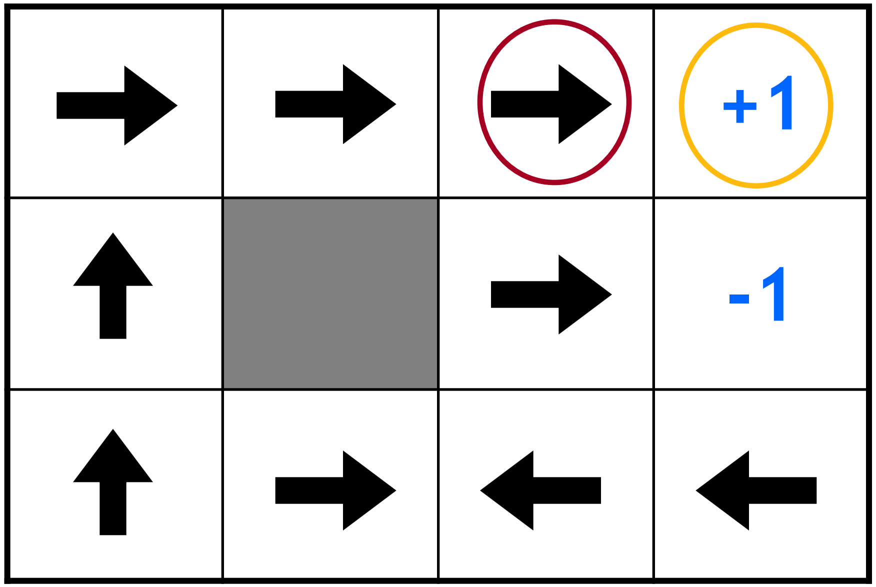
- reward +1 at [4,3], -1 at [4,2]
- reward -0.04 for each step

$$V^\pi(s) = \sum_a \pi(a|s) \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma V^\pi(s')] = \sum_a \pi(a|s) Q^\pi(s,a)$$

1.0 + 0.8 *

+

# Let's work out the Value function for a specific policy




actions: UP, DOWN, LEFT, RIGHT

e.g., **UP**

state-transitions with action **UP**:

80% move UP

10% move LEFT

10% move RIGHT

*If you bump into a wall, you stay where you are.

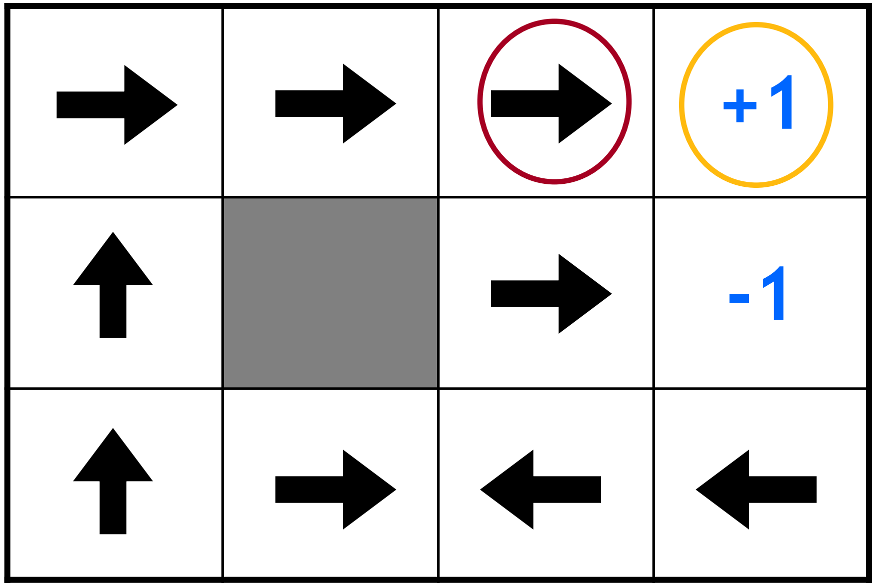
- reward +1 at [4,3], -1 at [4,2]
- reward -0.04 for each step

$$V^\pi(s) = \sum_a \pi(a|s) \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma V^\pi(s')] = \sum_a \pi(a|s) Q^\pi(s,a)$$

1.0 + 0.8 * ( +1-0.04

+

# Let's work out the Value function for a specific policy




actions: UP, DOWN, LEFT, RIGHT

e.g., **UP**

state-transitions with action **UP**:

| | |
|---|---|
| 80% | move UP |
| 10% | move LEFT |
| 10% | move RIGHT |

*If you bump into a wall, you stay where you are.

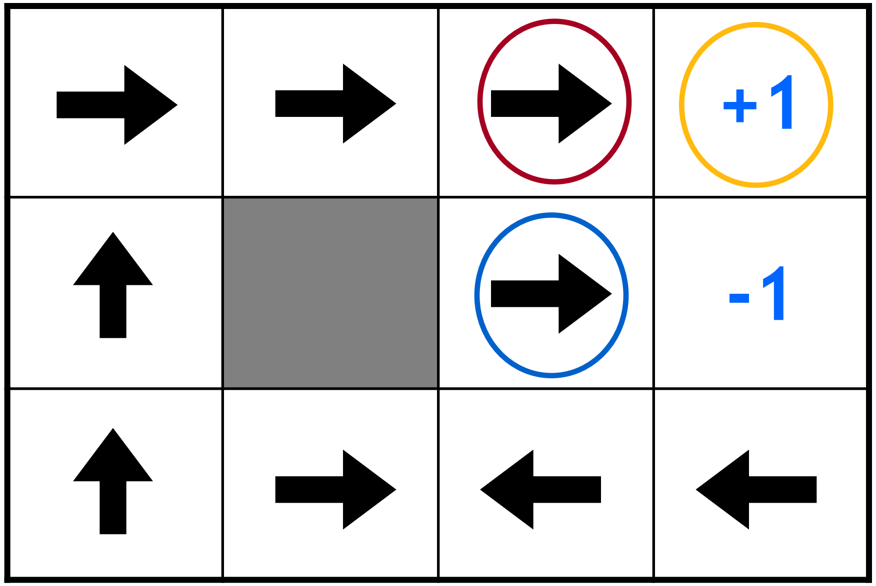
- reward +1 at [4,3], -1 at [4,2]
- reward -0.04 for each step

$$V^\pi(s) = \sum_a \pi(a|s) \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma V^\pi(s')] = \sum_a \pi(a|s) Q^\pi(s,a)$$

1.0   +   0.8 * ( +1-0.04 + 0 )

+

# Let's work out the Value function for a specific policy




actions: UP, DOWN, LEFT, RIGHT

e.g., **UP**

state-transitions with action **UP**:

80% move UP

10% move LEFT

10% move RIGHT

*If you bump into a wall, you stay where you are.

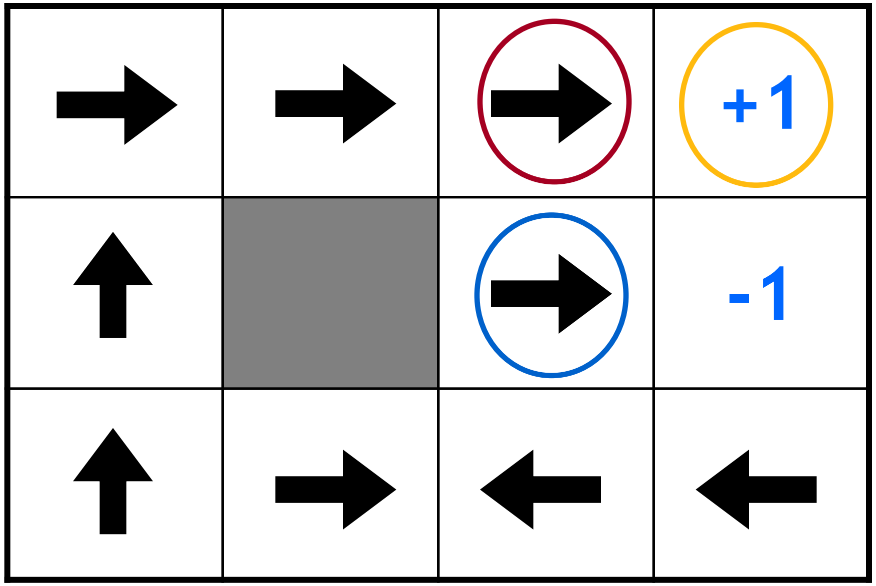
- reward +1 at [4,3], -1 at [4,2]
- reward -0.04 for each step

$$V^{\pi}(s) = \sum_{a} \pi(a|s) \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma V^{\pi}(s')] = \sum_{a} \pi(a|s) Q^{\pi}(s,a)$$

1.0 + 0.8 * ( +1-0.04 + 0 )

+

# Let's work out the Value function for a specific policy




actions: UP, DOWN, LEFT, RIGHT

e.g., **UP**

state-transitions with action **UP**:

80% move UP

10% move LEFT

10% move RIGHT

*If you bump into a wall, you stay where you are.

- reward +1 at [4,3], -1 at [4,2]
- reward -0.04 for each step

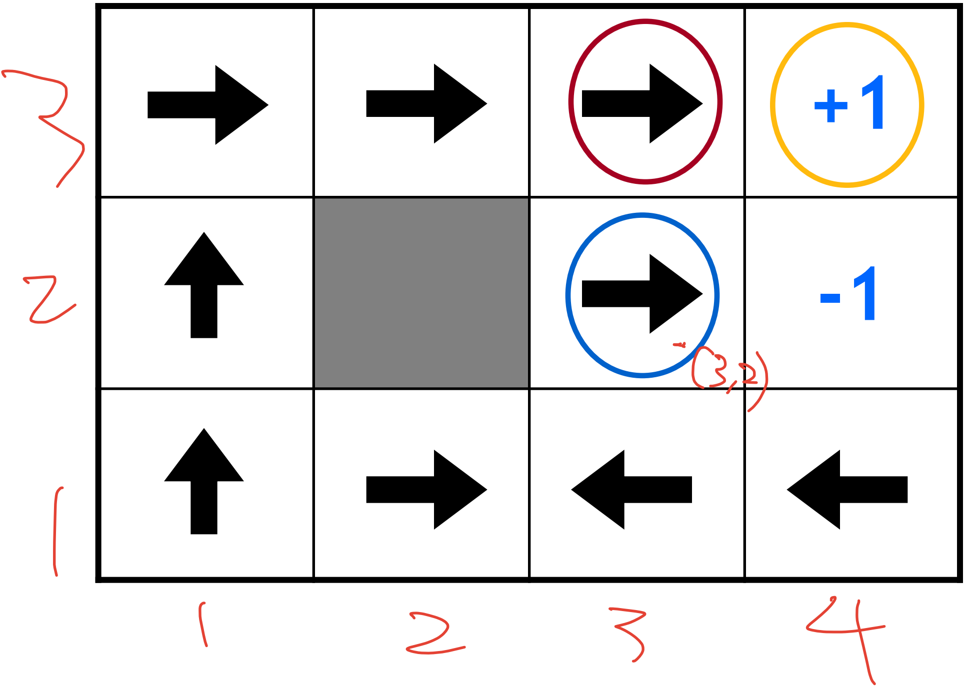
$$V^{\pi}(s) = \sum_a \pi(a|s) \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma V^{\pi}(s')] = \sum_a \pi(a|s) Q^{\pi}(s,a)$$

1.0 0.8 * ( +1-0.04 + 0 )

\+ 0.1 *

\+

# Let's work out the Value function for a specific policy




actions: UP, DOWN, LEFT, RIGHT

e.g., **UP**

state-transitions with action **UP**:

| | |
|---|---|
| 80% | move UP |
| 10% | move LEFT |
| 10% | move RIGHT |

*If you bump into a wall, you stay where you are.

- reward +1 at [4,3], -1 at [4,2]
- reward -0.04 for each step

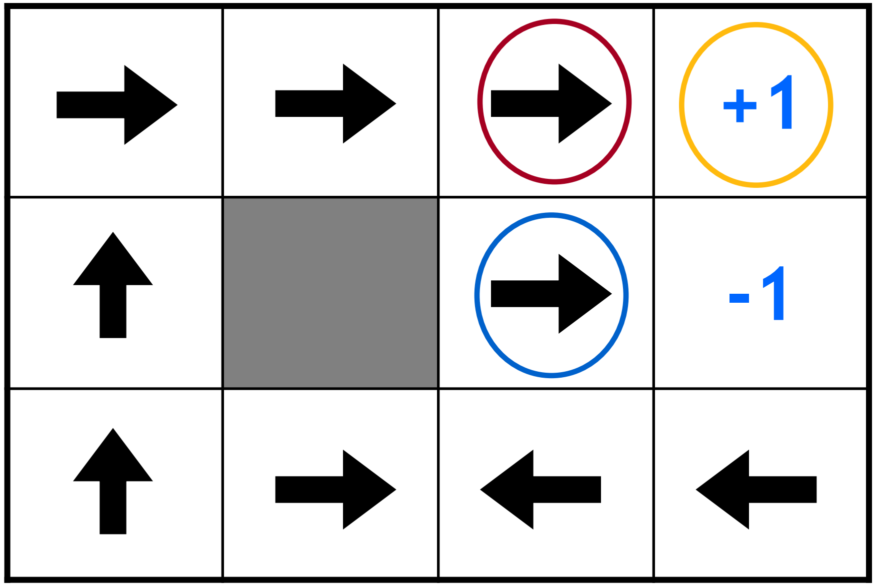
$$V^{\pi}(s) = \sum_a \pi(a|s) \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma V^{\pi}(s')] = \sum_a \pi(a|s) Q^{\pi}(s,a)$$

1.0 + 0.8 * ( +1-0.04 + 0 )

+ 0.1 * (-0.04 + $V^{\pi}$([3,2]) )

+

# Let's work out the Value function for a specific policy




actions: UP, DOWN, LEFT, RIGHT

e.g., **UP**

state-transitions with action **UP**:

| | |
|---|---|
| 80% | move UP |
| 10% | move LEFT |
| 10% | move RIGHT |

*If you bump into a wall, you stay where you are.

- reward +1 at [4,3], -1 at [4,2]
- reward -0.04 for each step

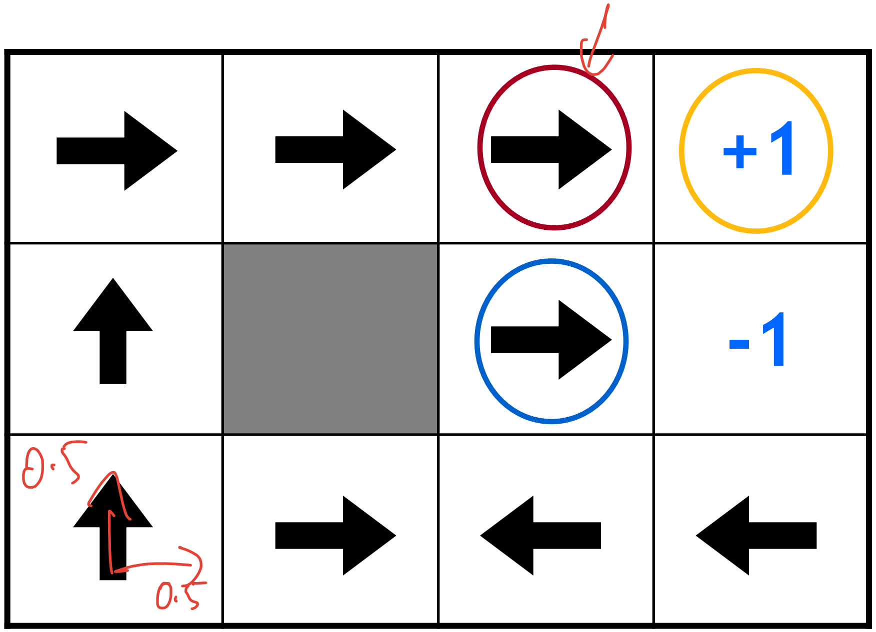
$$V^{\pi}(s) = \sum_{a} \pi(a|s) \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma V^{\pi}(s')] = \sum_{a} \pi(a|s) Q^{\pi}(s,a)$$

1.0 0.8 * ( +1-0.04 + 0 )

\+ 0.1 * (-0.04 + $V^{\pi}$([3,2]) )

\+ 0.1 *

# Let's work out the Value function for a specific policy




Choose "up" with prob 0.5

actions: UP, DOWN, LEFT, RIGHT

e.g., **UP**

state-transitions with action **UP**:

| | |
|---|---|
| 80% | move UP |
| 10% | move LEFT |
| 10% | move RIGHT |

*If you bump into a wall, you stay where you are.

- reward +1 at [4,3], -1 at [4,2]
- reward -0.04 for each step

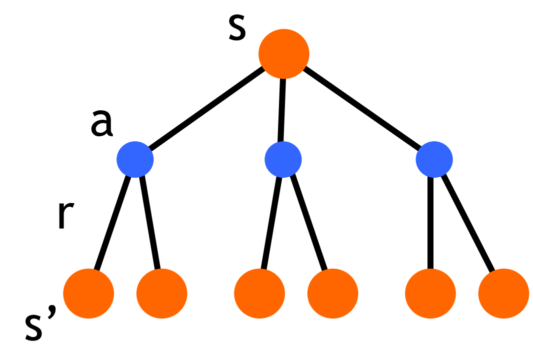
$$V^\pi(s) = \sum_a \pi(a|s) \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma V^\pi(s')] = \sum_a \pi(a|s) Q^\pi(s,a)$$

1.0 + 0.8 * ( +1-0.04 + 0 )

+ 0.1 * (-0.04 + $V^\pi$([3,2]) )

+ 0.1 * (-0.04 + $V^\pi$([3,3]) )

# Optimal value functions




- there's a set of *optimal* policies
  - $V^\pi$ defines partial ordering on policies
  - they share the same optimal value function

$$V^*(s) = \max_\pi V^\pi(s)$$

∀ s: $\arg\max_\pi V^\pi(s)$ is the same $=: \pi^*$

- Bellman optimality equation

$$V^*(s) = \max_a \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma V^*(s')]$$

future reward

  - system of n non-linear equations
  - solve for V*(s)
  - easy to extract the optimal policy

$V^* = V^{\pi^*}$

- having Q*(s,a) makes it even simpler

$$\pi^*(s) = \arg\max_a Q^*(s,a)$$

# Inference problem: given an MDP, how to compute its optimal policy?

- It suffices to compute its Q* function, because:

$$\pi^*(s) = \arg\max_a Q^*(s,a)$$

- It suffices to compute its V* function, because:

$$Q^*(s,a) = \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma V^*(s')]$$

# Summary of Bellman equations – the fundamental equations of MDP and RL

$\in \mathbb{R}^{|S|}$ $\in \mathbb{R}^{|S|\times|A|}$

- V-function and Q function
  - $V^\pi$ function Bellman equation

$$V^\pi(s) = \sum_a \pi(a|s) \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma V^\pi(s')]$$

  - $Q^\pi$ function Bellman equation

$$Q^\pi(s,a) = \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma \sum_{a'} \pi(a'|s') Q^\pi(s',a')]$$

  - V* function Bellman (optimality) equation

$$V^*(s) = \max_a \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma V^*(s')]$$

  - Q* function Bellman (optimality) equation

$$Q^*(s,a) = \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma \max_{a'} Q^*(s',a')]$$

# Algorithms for calculating the V* function

- Policy evaluation, policy-improvement

- Policy iterations

- Value iterations

# Dynamic programming

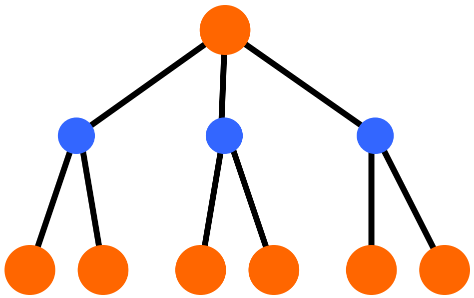
- main idea
  - use value functions to structure the search for good policies
  - need a known model of the environment
- two main components
  - policy evaluation: compute $V^{\pi}$ from $\pi$
  - policy improvement: improve $\pi$ based on $V^{\pi}$
  - start with an arbitrary policy
  - repeat evaluation/improvement until convergence

# Policy evaluation/improvement

- policy evaluation: π -> $V^\pi$
  - Bellman eqn's define a system of n eqn's
  - could solve, but will use iterative version

$$V^\pi_{k+1}(s) \leftarrow \sum_a \pi(a|s) \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma V^\pi_k(s')]$$

  - start with an arbitrary value function $V_0$, iterate until $V_k$ converges

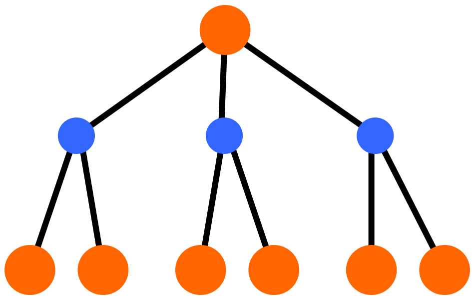


# Policy evaluation/improvement

- policy evaluation: $\pi$ -> $V^{\pi}$
  - Bellman eqn's define a system of n eqn's
  - could solve, but will use iterative version

$$V_{k+1}^{\pi}(s) \leftarrow \sum_{a} \pi(a|s) \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma V_k^{\pi}(s')]$$

  - start with an arbitrary value function $V_0$, iterate until $V_k$ converges



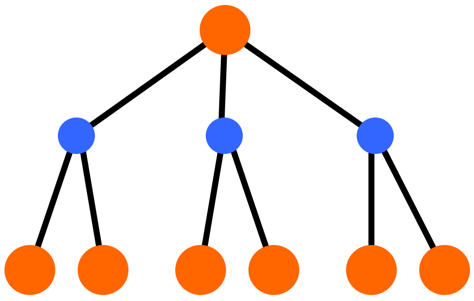
- policy improvement: $V^{\pi}$ -> $\pi'$

# Policy evaluation/improvement

- policy evaluation: $\pi \rightarrow V^{\pi}$
  - Bellman eqn's define a system of n eqn's
  - could solve, but will use iterative version

$$V_{k+1}^{\pi}(s) \leftarrow \sum_{a} \pi(a|s) \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma V_k^{\pi}(s')]$$

  - start with an arbitrary value function $V_0$, iterate until $V_k$ converges



- policy improvement: $V^{\pi} \rightarrow \pi'$

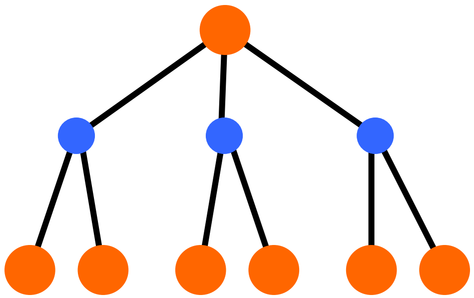
$$\pi'(s) = \arg\max_{a} Q^{\pi}(s,a)$$
$$= \arg\max_{a} \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma V_k^{\pi}(s')]$$

# Policy evaluation/improvement

- policy evaluation: $\pi$ -> $V^\pi$
  - Bellman eqn's define a system of n eqn's
  - could solve, but will use iterative version

$$V_{k+1}^{\pi}(s) \leftarrow \sum_a \pi(a|s) \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma V_k^{\pi}(s')]$$

  - start with an arbitrary value function $V_0$, iterate until $V_k$ converges



- policy improvement: $V^\pi$ -> $\pi'$

$$\pi'(s) = \arg\max_a Q^{\pi}(s,a)$$
$$= \arg\max_a \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma V_k^{\pi}(s')]$$

  - $\pi'$ either strictly better than $\pi$, or $\pi'$ is optimal (if $\pi = \pi'$)

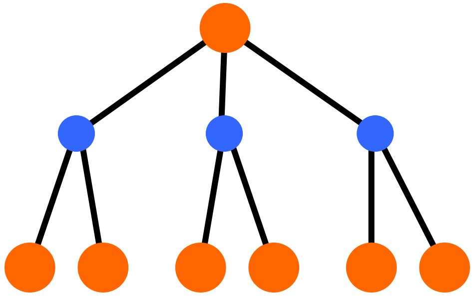
$\pi^*$ is a fixed point of PI

# Policy/Value iteration

- Policy iteration

$$\pi_0 \rightarrow^E V^{\pi_0} \rightarrow^I \pi_1 \rightarrow^E V^{\pi_1} \rightarrow^I \ldots \rightarrow^I \pi^* \rightarrow^E V^*$$

  - two nested iterations; too slow
  - don't need to converge to $V^{\pi_k}$
    - just move towards it

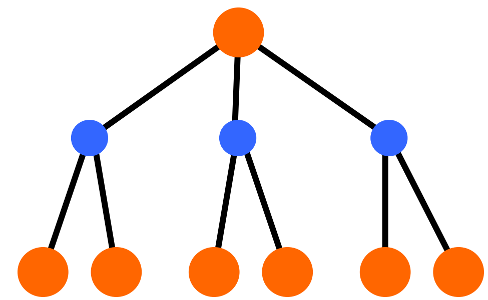


# Policy/Value iteration

- Policy iteration

$$\pi_0 \rightarrow^E V^{\pi_0} \rightarrow^I \pi_1 \rightarrow^E V^{\pi_1} \rightarrow^I \ldots \rightarrow^I \pi^* \rightarrow^E V^*$$

  - two nested iterations; too slow
  - don't need to converge to $V^{\pi_k}$
    - just move towards it

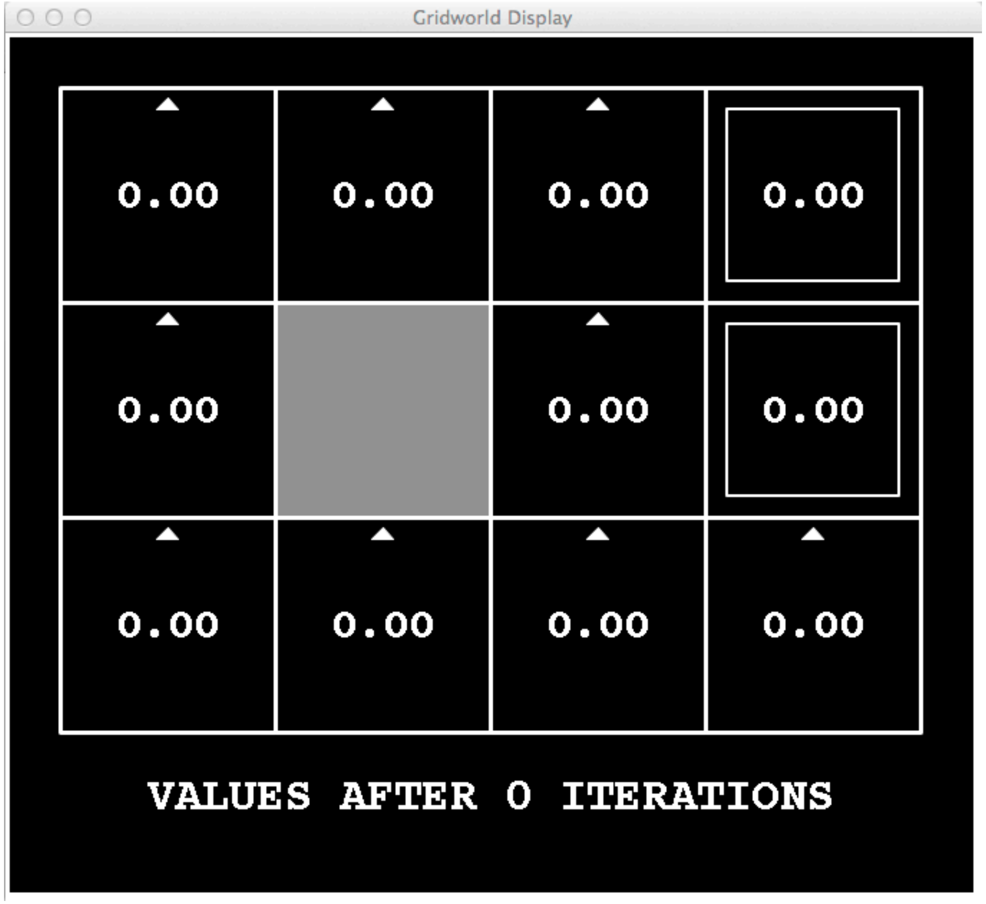


- Value iteration    Similarity $V^*$ eqn.

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma V_k(s')]$$

  - use Bellman optimality equation as an update
  - converges to V*

# k=0

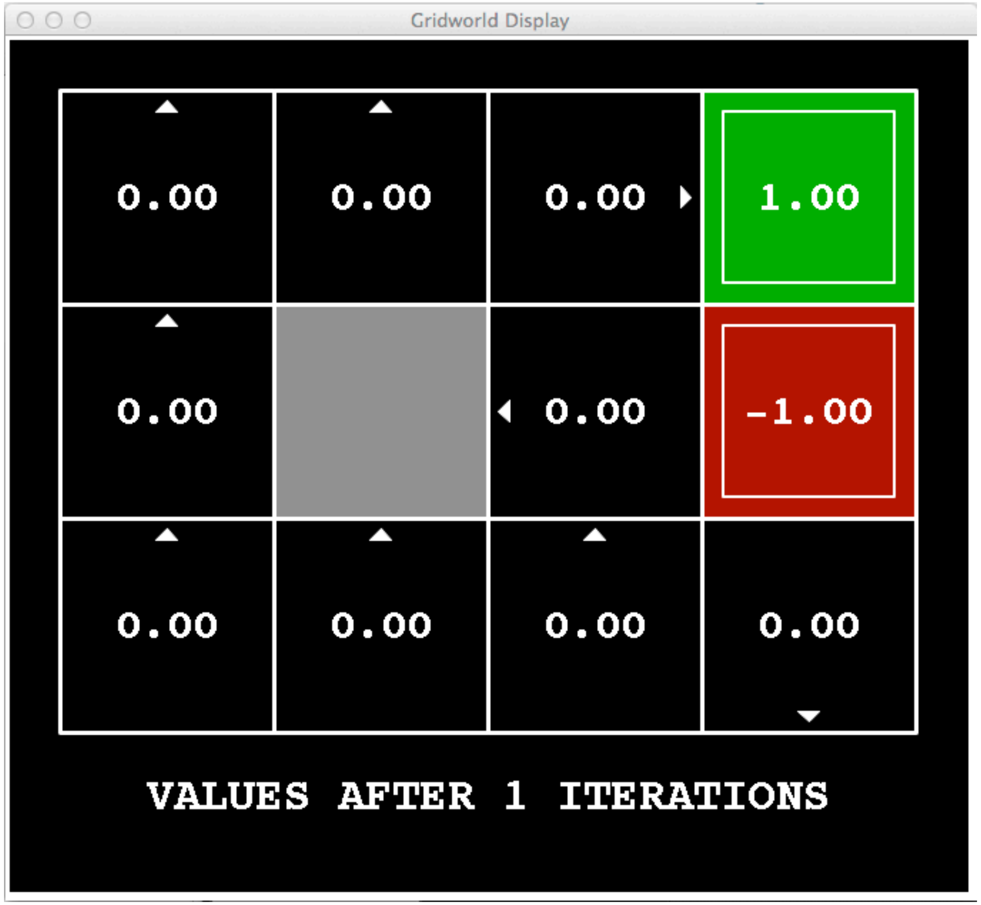



Noise = 0.2

Discount = 0.9

Living reward = 0

# k=1




Noise = 0.2

Discount = 0.9

Living reward = 0

# k=2


Noise = 0.2

Discount = 0.9

Living reward = 0

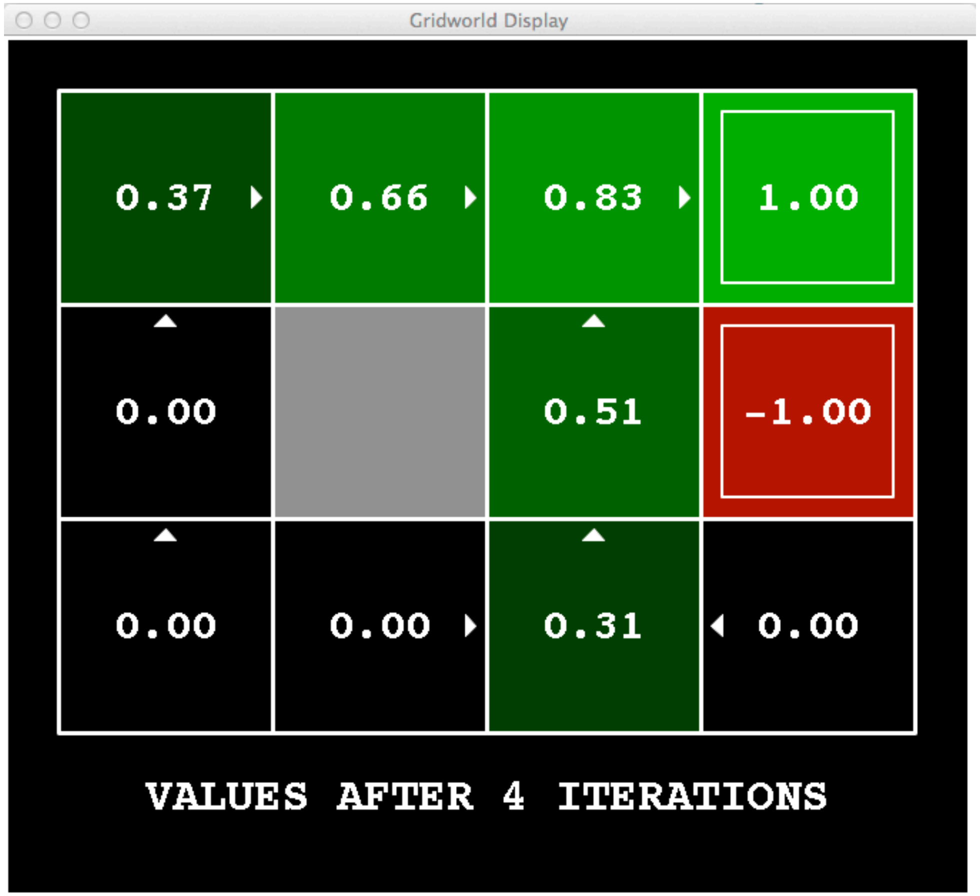
# k=3


Noise = 0.2

Discount = 0.9

Living reward = 0

# k=4

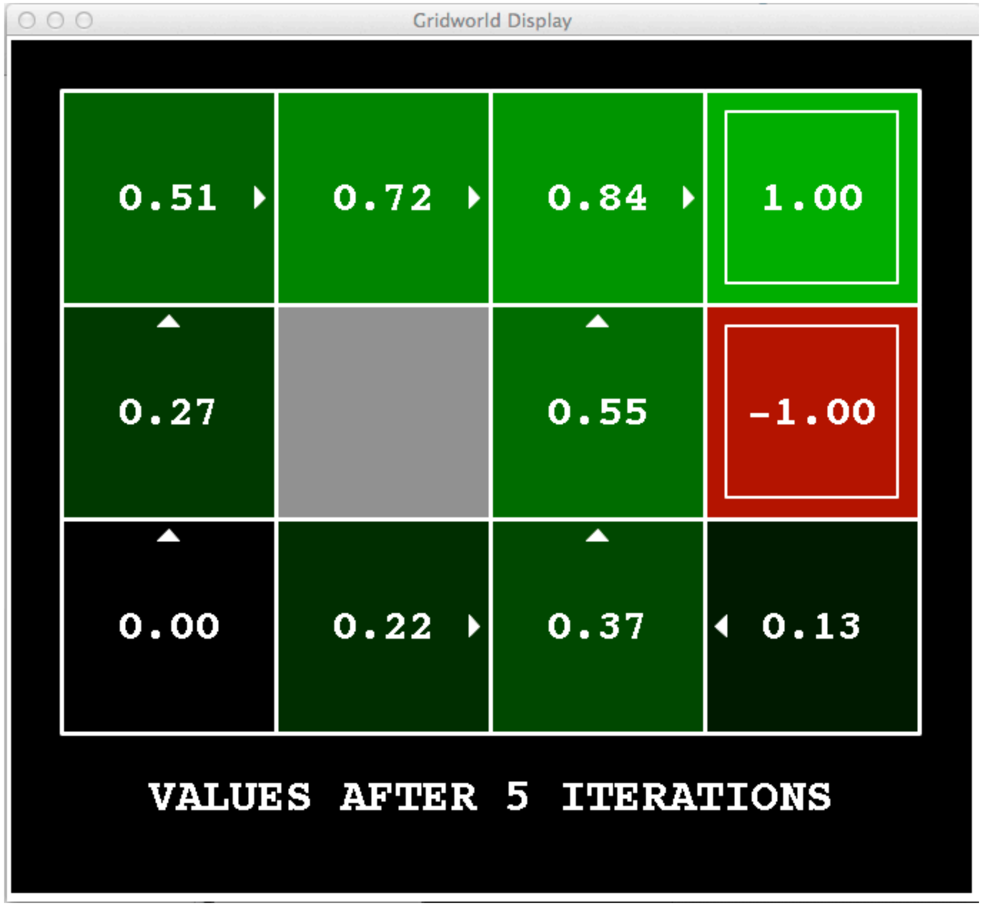



Noise = 0.2

Discount = 0.9

Living reward = 0

# k=5

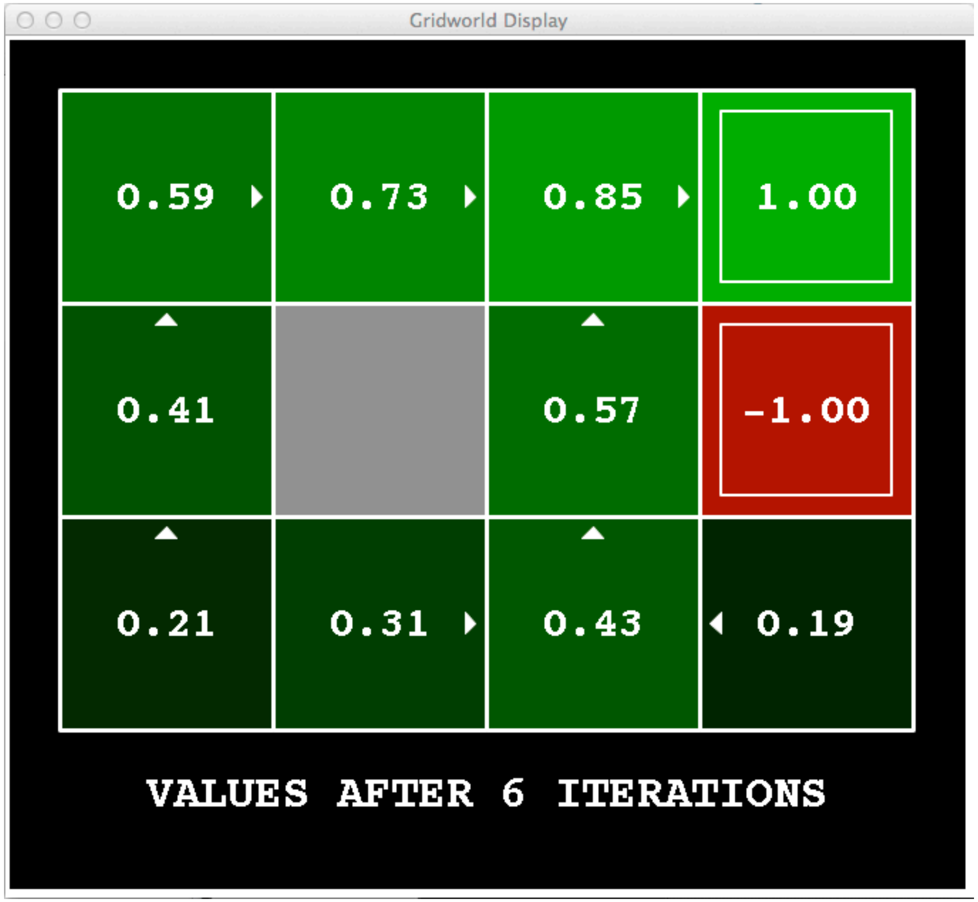



Noise = 0.2

Discount = 0.9

Living reward = 0

# k=6

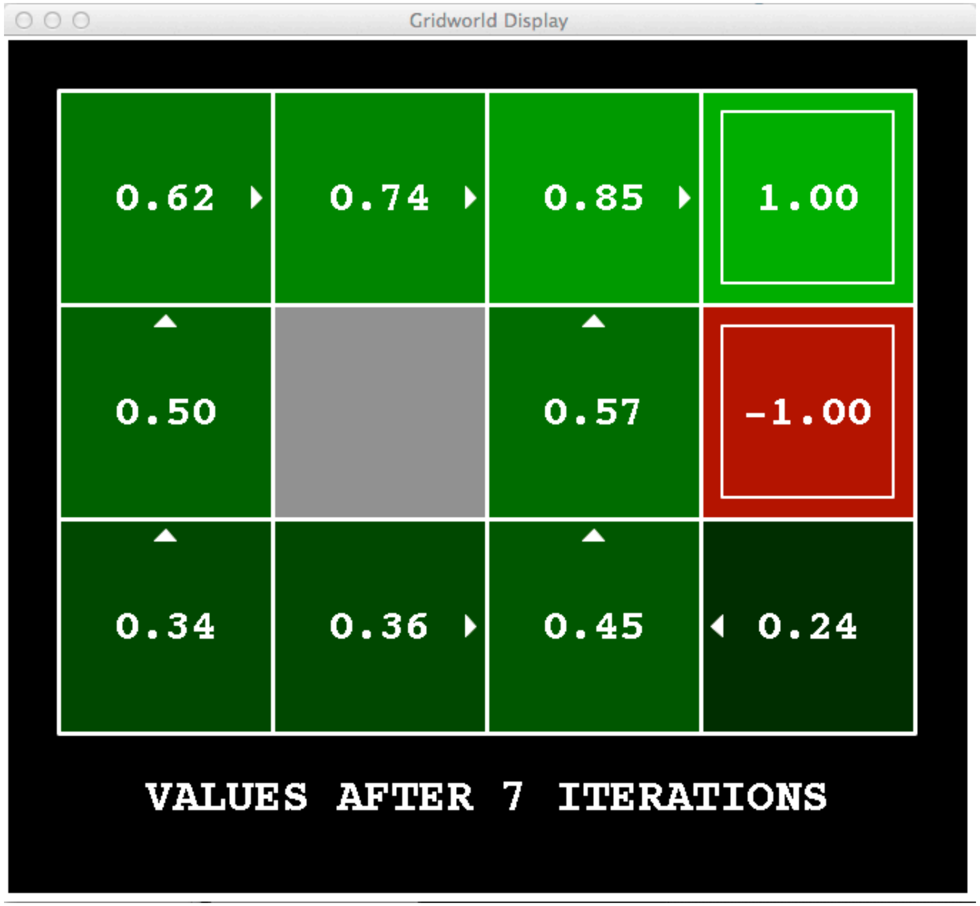



Noise = 0.2

Discount = 0.9

Living reward = 0

# k=7

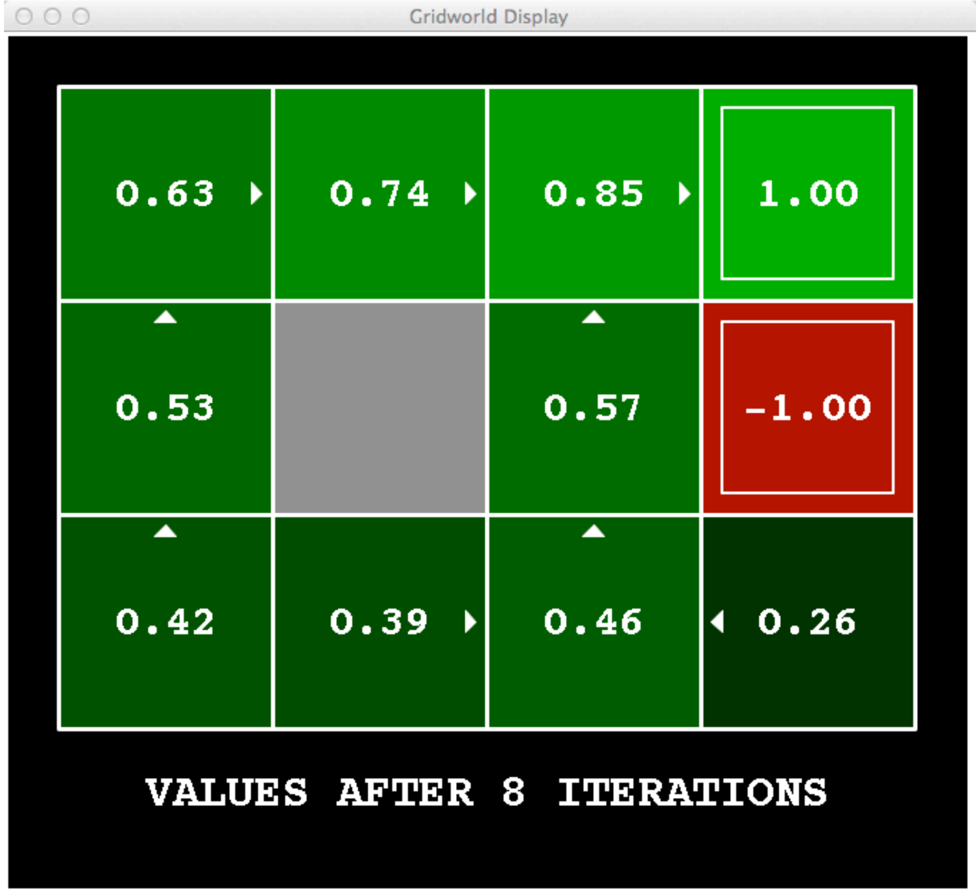



Noise = 0.2

Discount = 0.9

Living reward = 0

# k=8

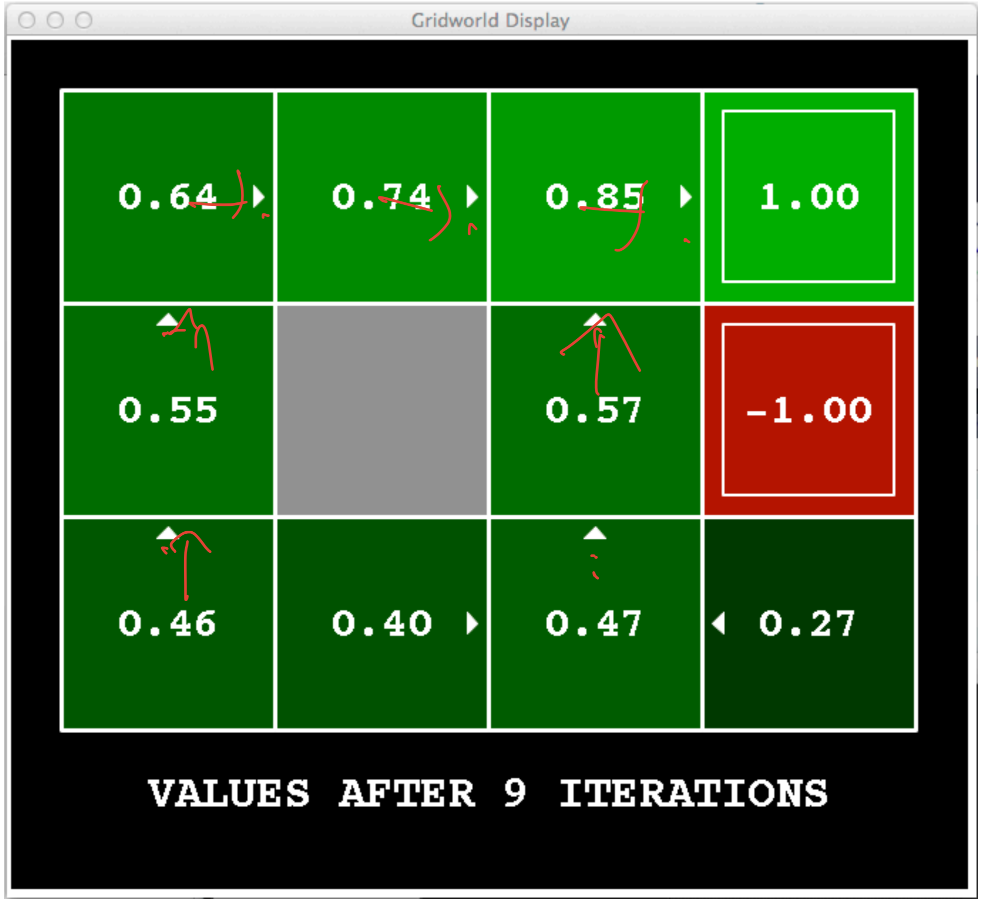



Noise = 0.2

Discount = 0.9

Living reward = 0

# k=9

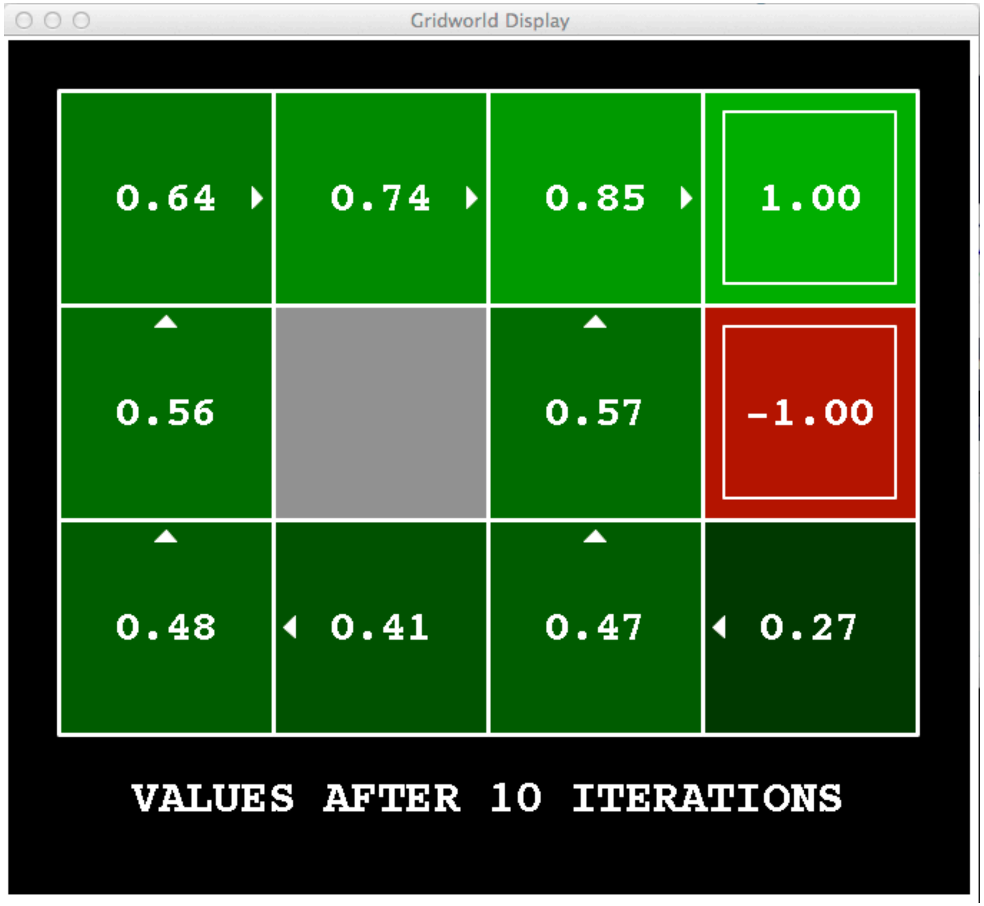



Noise = 0.2

Discount = 0.9

Living reward = 0

# k=10

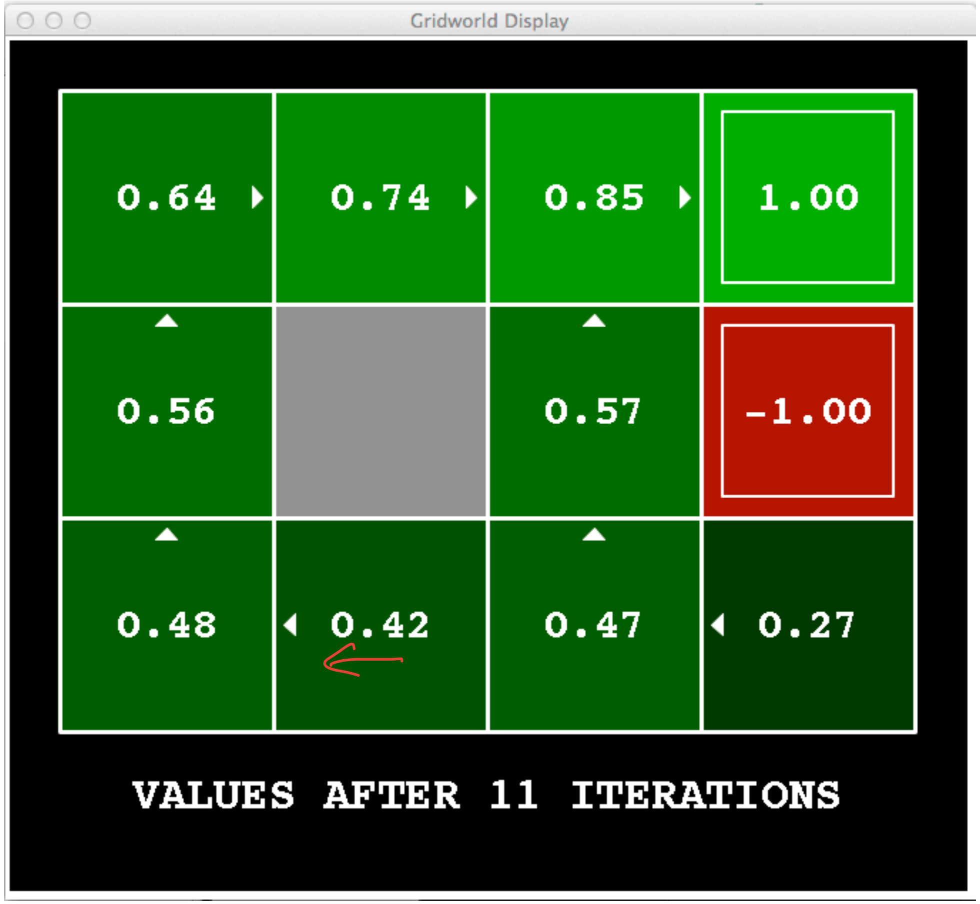



Noise = 0.2

Discount = 0.9

Living reward = 0

# k=11

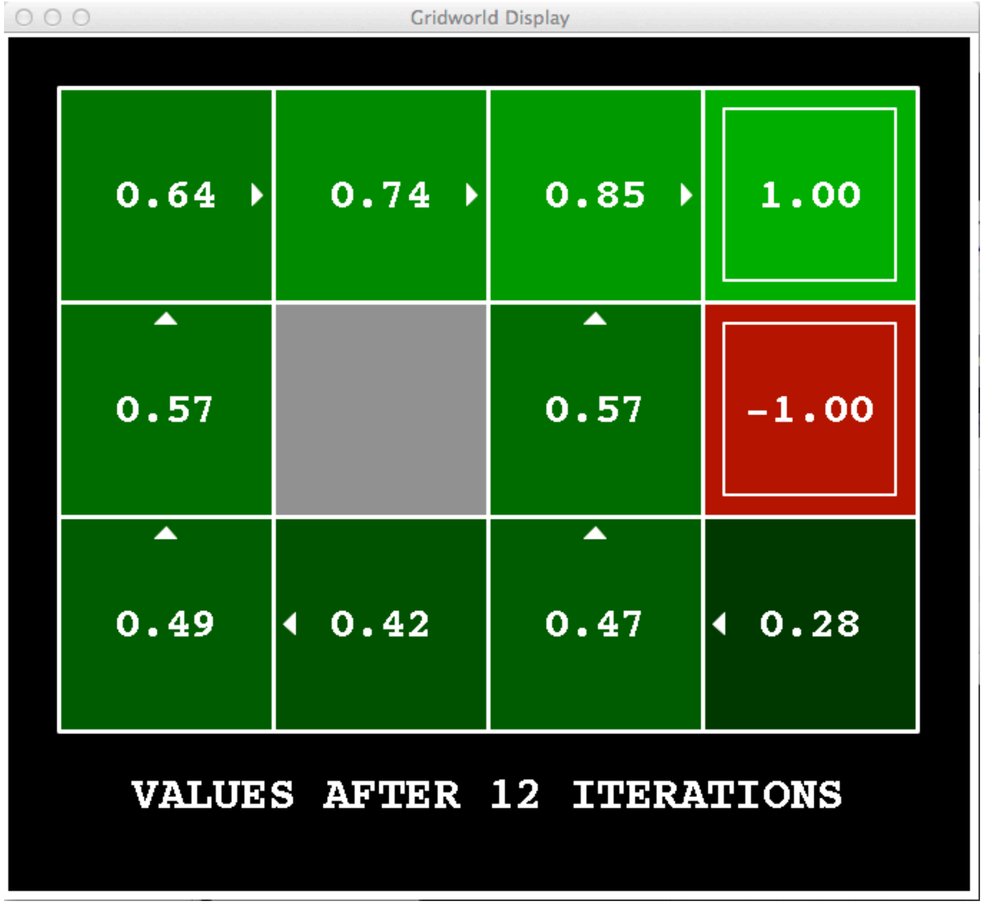



Noise = 0.2

Discount = 0.9

Living reward = 0

# k=12




Noise = 0.2

Discount = 0.9

Living reward = 0

# k=100


Noise = 0.2

Discount = 0.9

Living reward = 0

# Q-iteration

- Updating Q functions instead of V functions

$$Q_{k+1}(s,a) \leftarrow \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma \max_{a'} Q_k(s',a')]$$

- Quiz: What is the difference from the following extended version of value iteration?

$$V_{k+1}(s) \leftarrow \max_{a} \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma V_k(s')]$$

$$Q_{k+1}(s,a) \leftarrow \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma V_{k+1}(s')]$$

# Q-iteration

- Updating Q functions instead of V functions

$$Q_{k+1}(s,a) \leftarrow \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma \max_{a'} Q_k(s',a')]$$

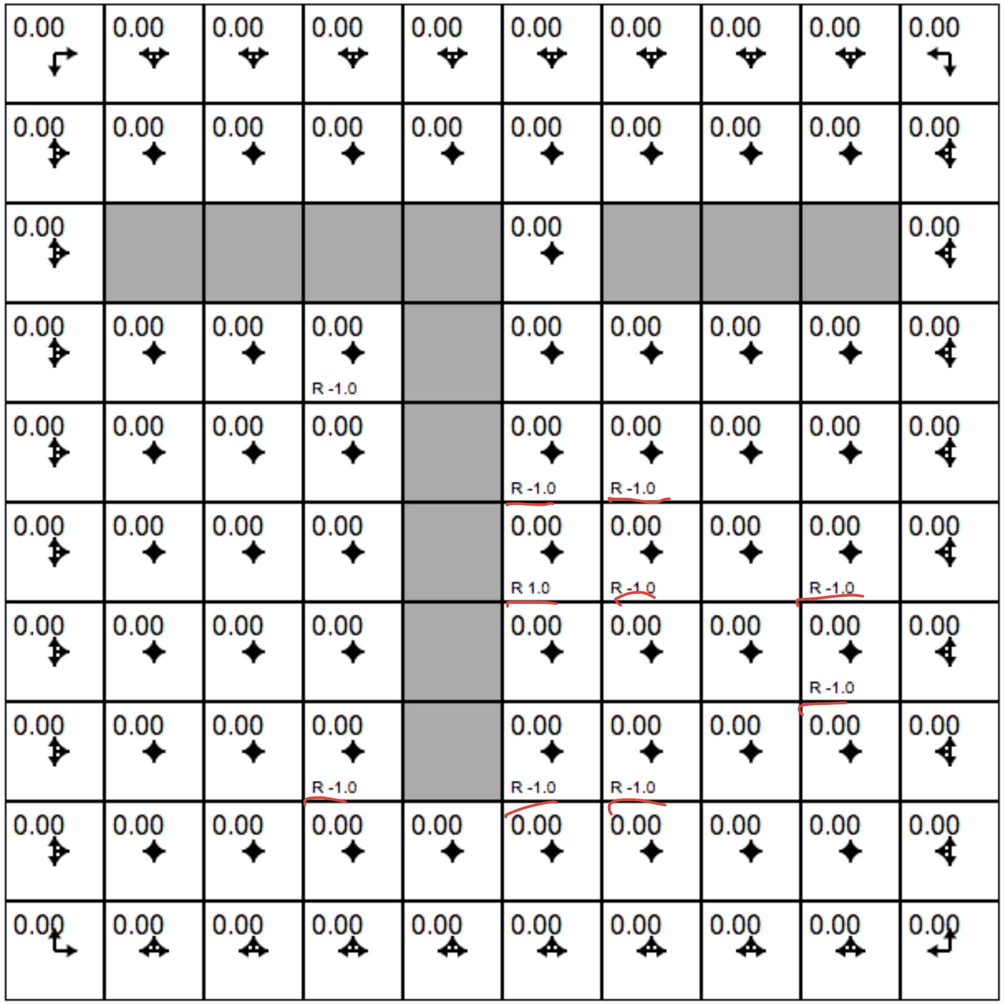
- Quiz: What is the difference from the following extended version of value iteration?

$$V_{k+1}(s) \leftarrow \max_{a} \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma V_k(s')]$$

$$Q_{k+1}(s,a) \leftarrow \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma V_{k+1}(s')]$$

**Ans: They are identical!**

# Demo: grid worlds




https://cs.stanford.edu/people/karpathy/reinforcejs/gridworld_dp.html

# MDP summary

- Tabular MDP
- Episodic vs. infinite horizon (discounted)
- Immediate reward vs long-term reward
- Value functions:  V functions, Q functions
- Bellman equations,  Bellman optimality equations
- How to solve MDP?  Policy iterations,  value iterations

# MDP Summary

Standard expectimax: $V(s) = \max_a \sum_{s'} P(s'|s,a)V(s')$

Bellman equations: $V(s) = \max_a \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma V(s')]$

Value iteration:

$$V_{k+1}(s) = \max_a \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma V_k(s')], \quad \forall s$$

$$Q_{k+1}(s,a) = \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma \max_{a'} Q_k(s',a')], \quad \forall s,a$$

$$V_{k+1}^{\pi}(s) = \sum_{s'} P(s'|s,\pi(s))[r(s,\pi(s),s') + \gamma V_k^{\pi}(s')], \quad \forall s$$

$$\pi_{new}(s) = \operatorname{argmax}_a \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma V^{\pi_{old}}(s')], \quad \forall s$$

# MDP Summary

Standard expectimax: $V(s) = \max_a \sum_{s'} P(s'|s,a)V(s')$

Bellman equations: $V(s) = \max_a \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma V(s')]$

Value iteration: $V_{k+1}(s) = \max_a \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma V_k(s')], \quad \forall\, s$

Q-iteration: $Q_{k+1}(s,a) = \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma \max_{a'} Q_k(s',a')], \quad \forall\, s,a$

Policy evaluation: $V_{k+1}^{\pi}(s) = \sum_{s'} P(s'|s,\pi(s))[r(s,\pi(s),s') + \gamma V_k^{\pi}(s')], \quad \forall\, s$

Policy improvement: $\pi_{new}(s) = \underset{a}{\operatorname{argmax}} \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma V^{\pi_{old}}(s')], \quad \forall\, s$

# Matrix-form of Bellman Equations and VI

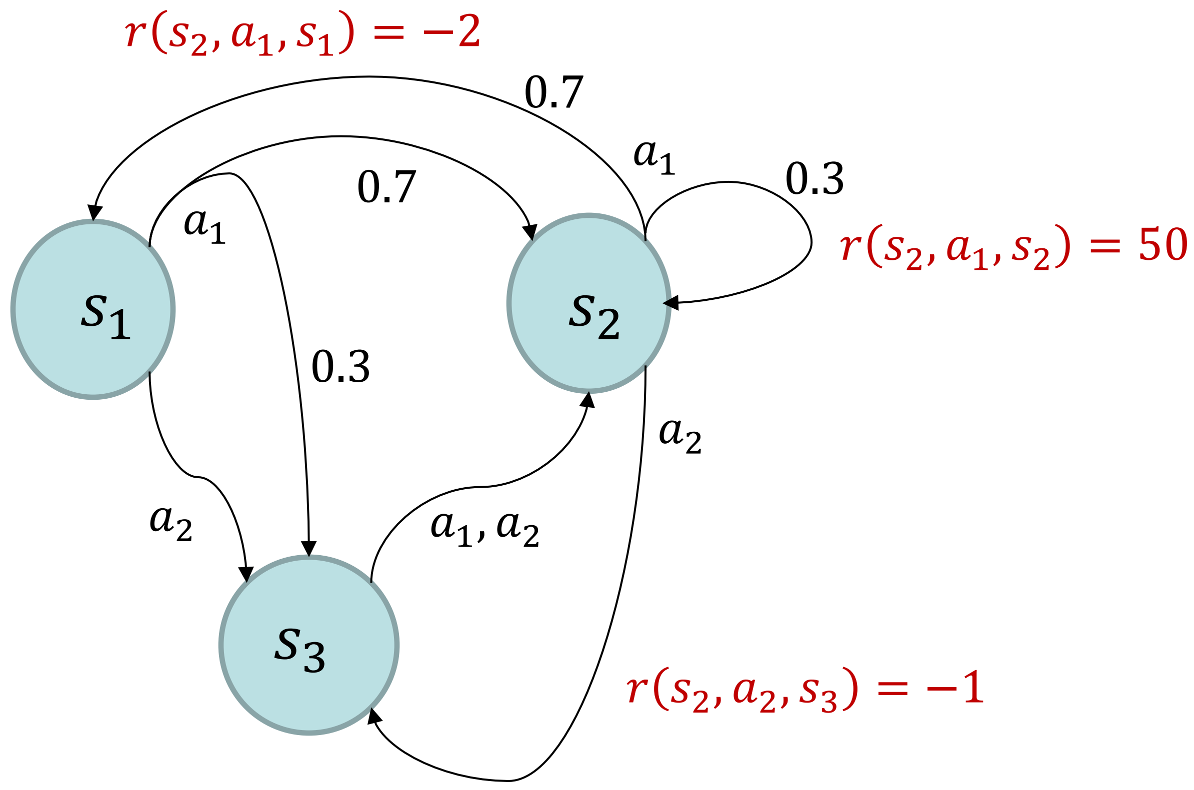
$$V^\pi(s) = \sum_a \pi(a|s) \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma V^\pi(s')]$$

$$V^\pi_{k+1}(s) \leftarrow \sum_a \pi(a|s) \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma V^\pi_k(s')]$$

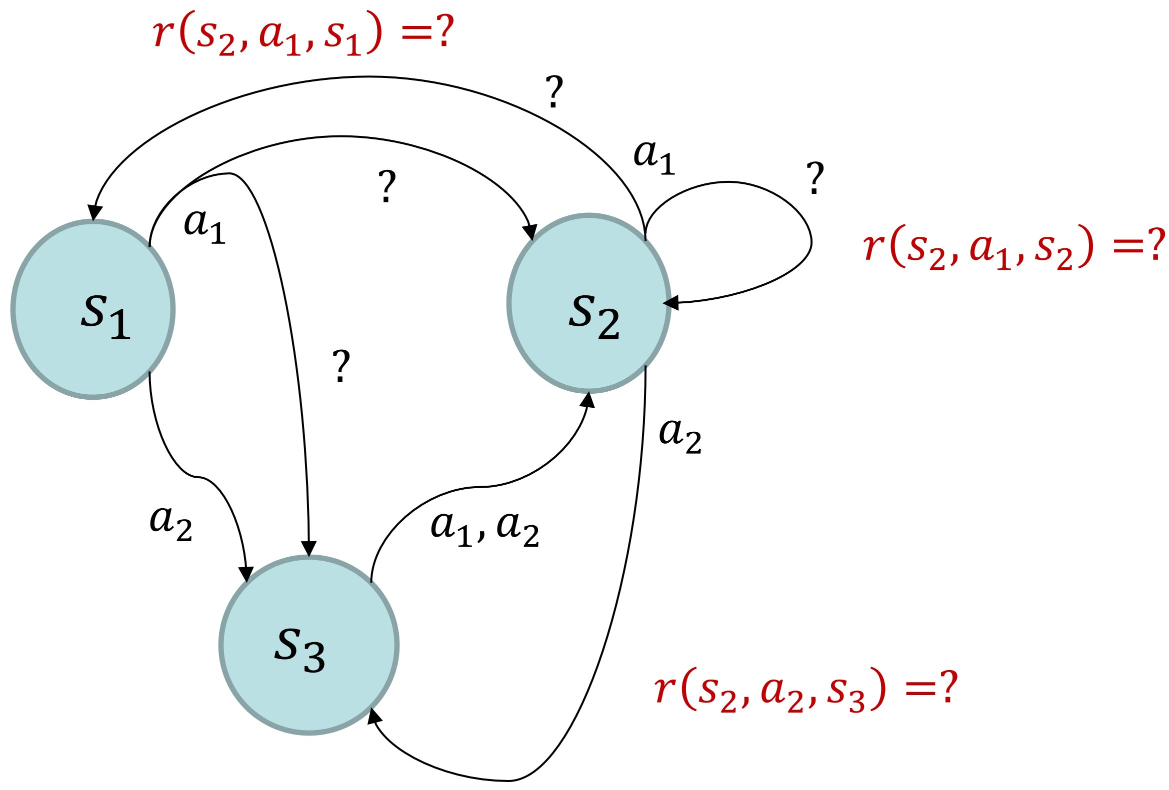
# Solving MDP with VI or PI is offline planning

- The agent is given how the environment works
- The agent works out the optimal policy in its mind.
- The agent never really starts to play at all.
- No learning is happening.

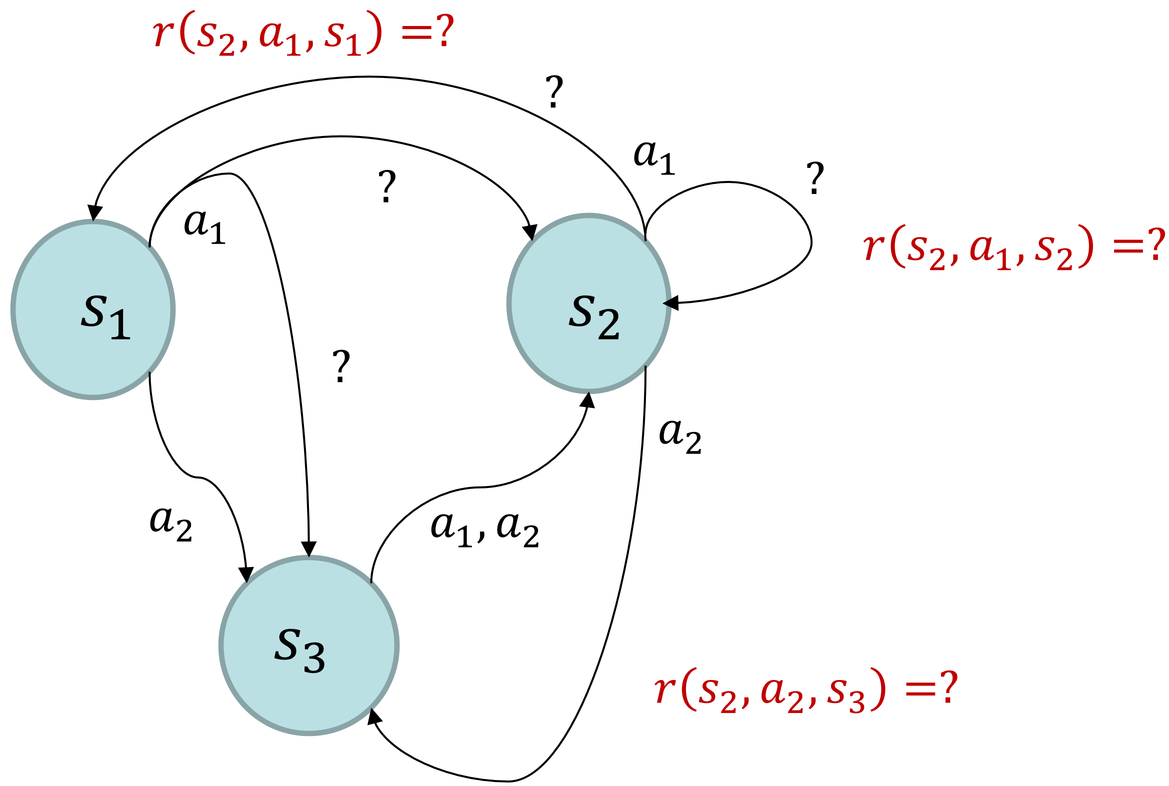
# **State-space diagram** representation of an MDP: An example with 3 states and 2 actions.



# What happens if you do not know the rewards / transition probabilities?



# What happens if you do not know the rewards / transition probabilities?




Then you have to learn by interacting with the unknown environment.

You cannot use only offline planning!

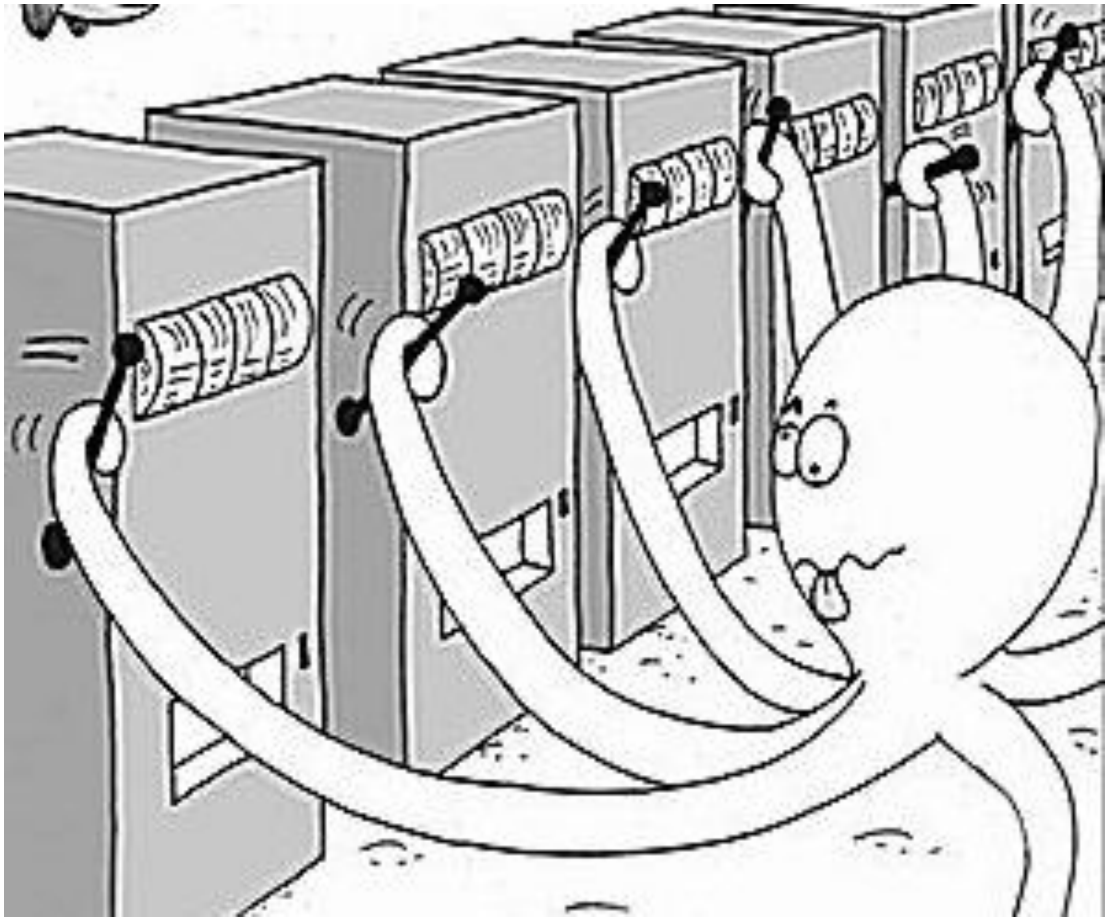
**Exploration:** Try unknown actions to see what happens.

**Exploitation:** Maximize utility using what we know.

# Let us tackle different aspects of the RL problem one at a time

- Markov Decision Processes:
  - Dynamics are given no need to learn
- **Bandits: Explore-Exploit in simple settings**
  - RL without dynamics
- Full Reinforcement Learning
  - Learning MDPs

# Slot machines and Multi-arm bandits



# Multi-arm bandits: Problem setup

- No state. k-actions $a \in \mathcal{A} = \{1, 2, ..., k\}$
- You decide which arm to pull in every iteration

$$A_1, A_2, ..., A_T$$

- You collect a cumulative payoff of $\sum_{t=1}^{T} R_t$
- The goal of the agent is to maximize the expected payoff.
  - For future payoffs?
  - For the expected cumulative payoff?

# Key differences from MDPs

- Simplified:
  - No state-transitions
- But:
  - We are not given the expected reward r(s, a, s')
  - We need to learn the optimal policy by trials-and-errors.

# A 10-armed bandits example

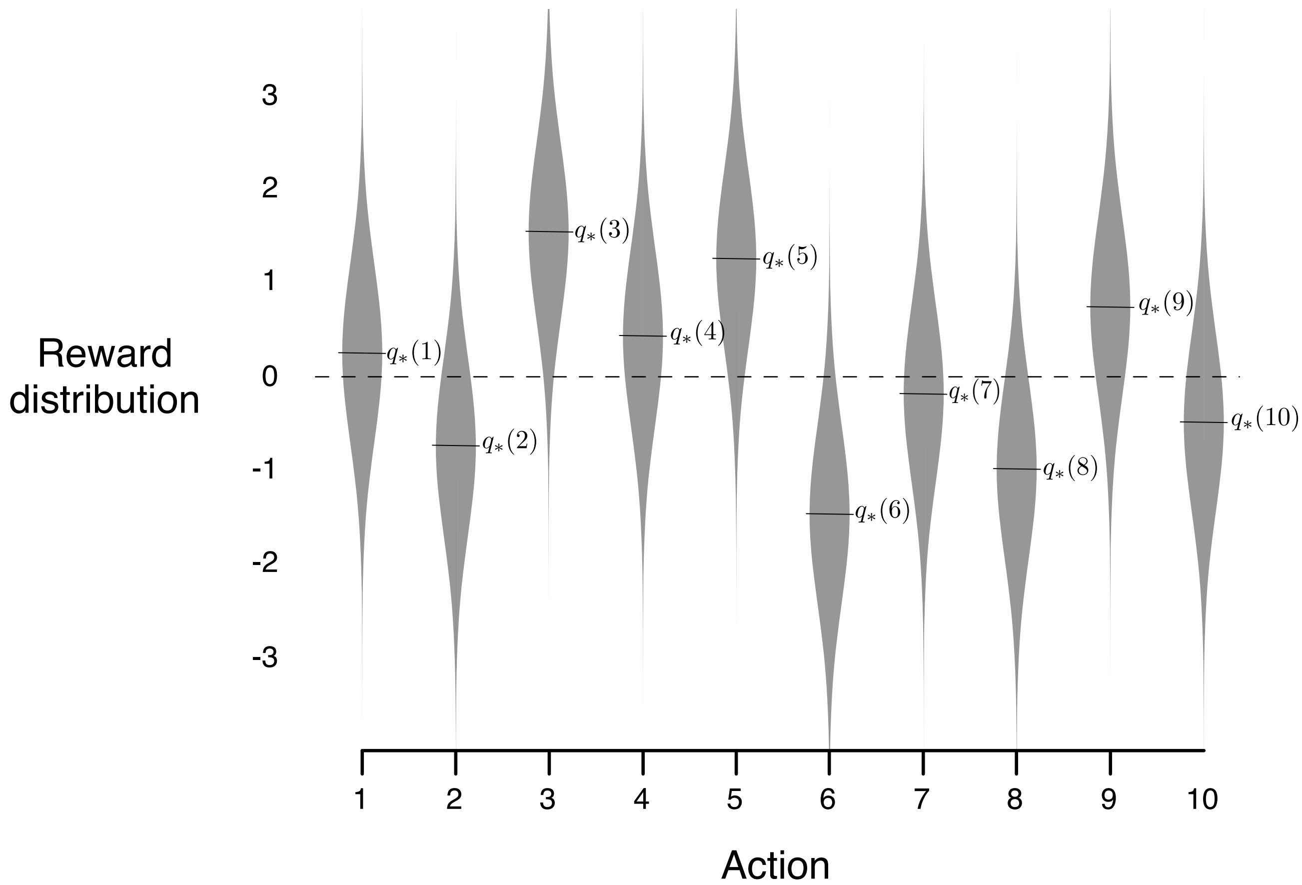



**Figure 2.1:** An example bandit problem from the 10-armed testbed. The true value $q_*(a)$ of each of the ten actions was selected according to a normal distribution with mean zero and unit variance, and then the actual rewards were selected according to a mean $q_*(a)$ unit variance normal distribution, as suggested by these gray distributions.

# How do we measure the performance of an **online learning agent**?

- The notion of "Regret":
  - I wish I have done things differently.
  - Comparing to the best actions in the hindsight, how much worse did I do.

- For MAB, the regret is defined as follow

$$T \max_{a \in [k]} \mathbb{E}[R_t|a] - \sum_{t=1}^{T} \mathbb{E}_{a \sim \pi} \left[ \mathbb{E}[R_t|a] \right]$$

# Greedy strategy

- Expected reward

$$q_*(a) \doteq \mathbb{E}[R_t \mid A_t = a] .$$

- Estimate the expected reward

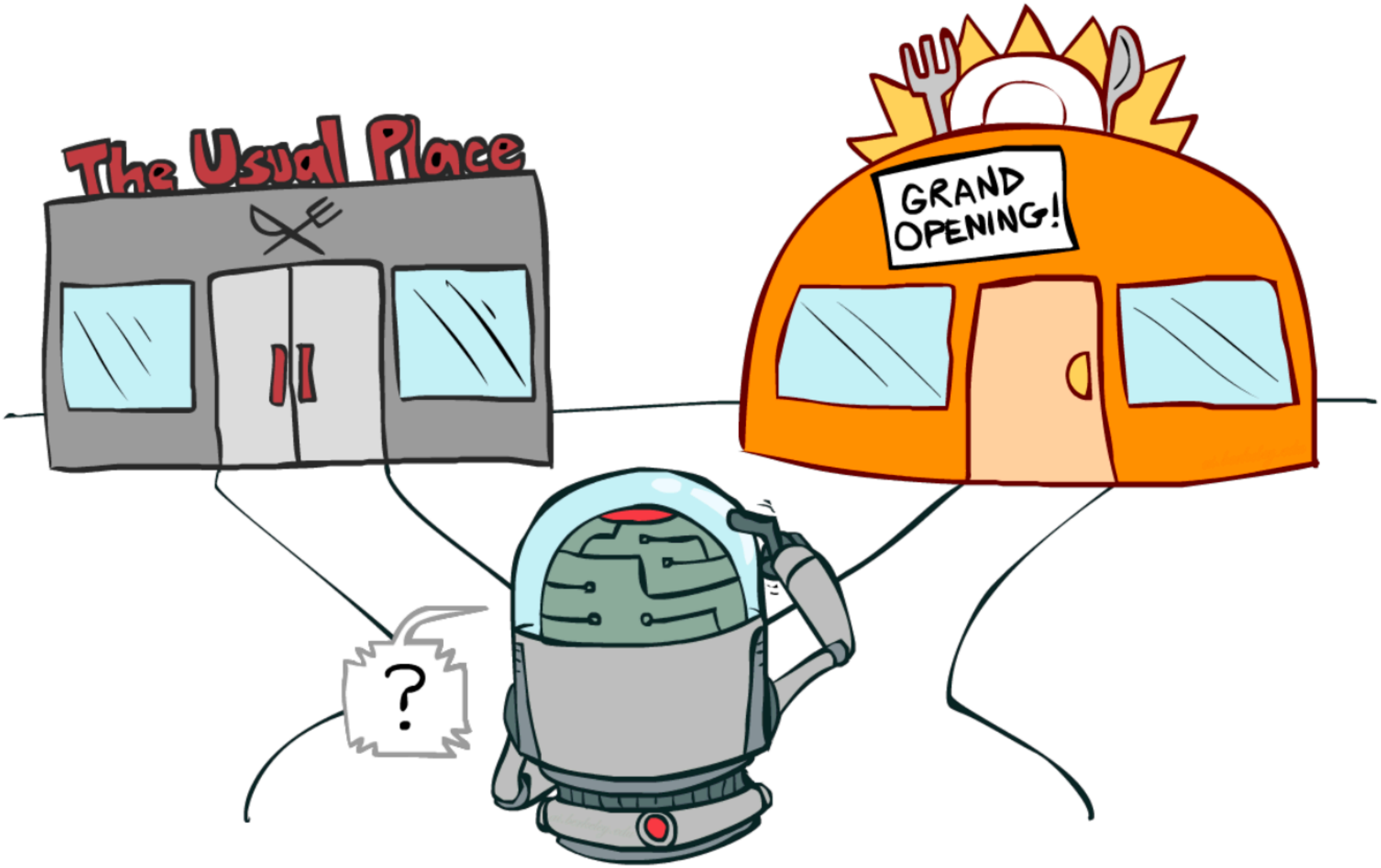
$$Q_t(a) \doteq \frac{\text{sum of rewards when } a \text{ taken prior to } t}{\text{number of times } a \text{ taken prior to } t} = \frac{\sum_{i=1}^{t-1} R_i \cdot \mathbb{1}_{A_i=a}}{\sum_{i=1}^{t-1} \mathbb{1}_{A_i=a}}$$

- Choose $A_t \doteq \underset{a}{\arg\max}\, Q_t(a),$



What is the issue with this strategy?

# Exploration vs. Exploitation




(Illustration from Dan Klein and Pieter Abbeel's course in UC Berkeley)

# Next Tuesday

- Bandits algorithms
  - Explore-first
  - epsilon-greedy
  - Upper confidence bound