

# Artificial Intelligence

CS 165A

May 11, 2023

Instructor: Prof. Yu-Xiang Wang

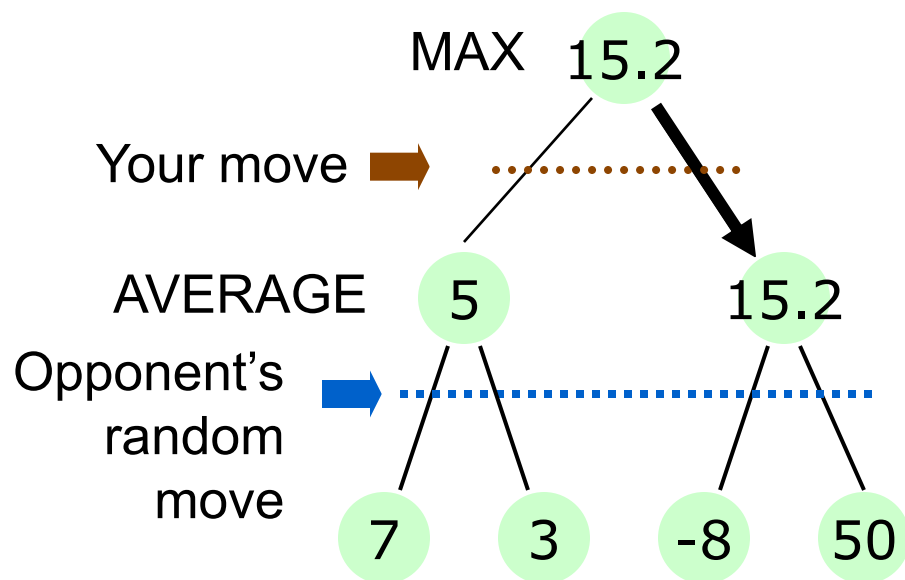
T  
o  
d  
a  
y

- Intro to RL
- Markov Decision Processes

# Announcement

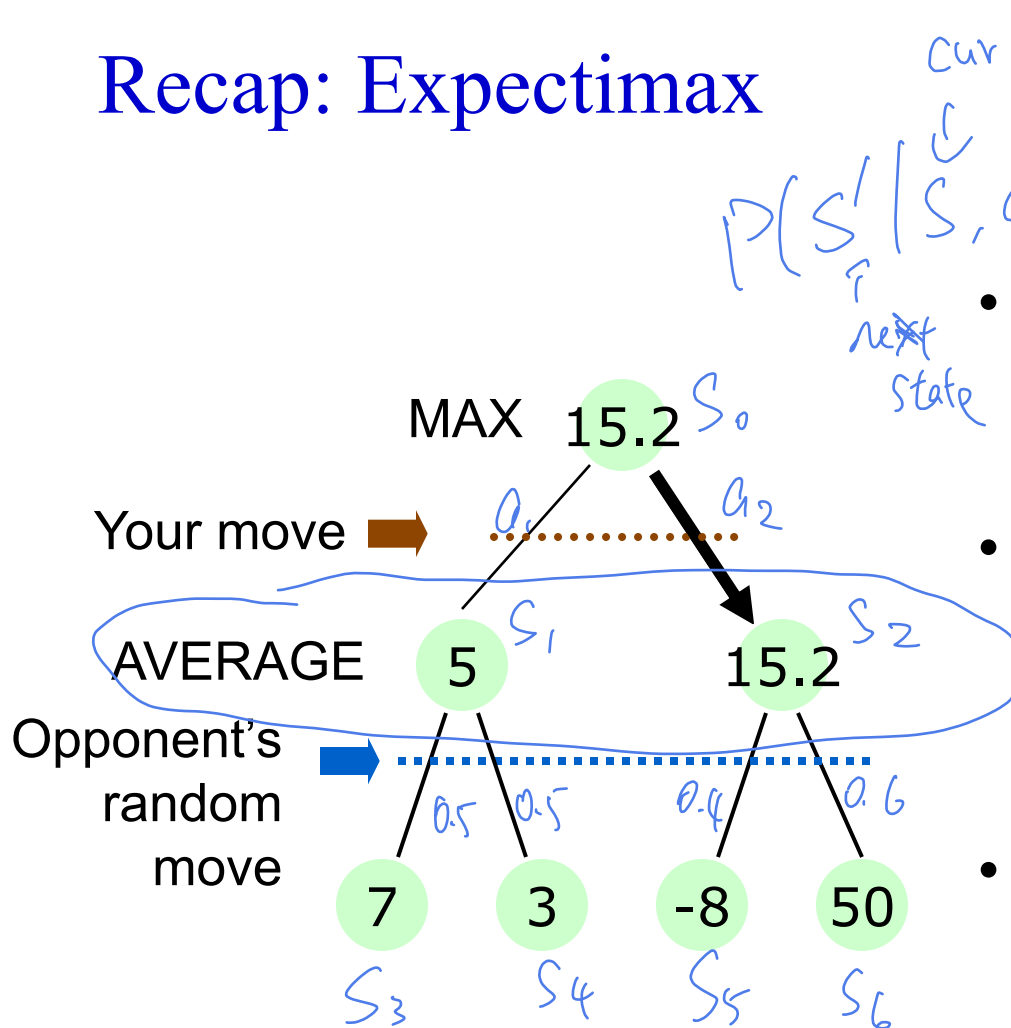
- Don't wait for the last minute to work on Project 2
  - Due next Thursday.
- Start early and get help early
- Please read and attempt the optional HW3 before going to the discussion class

# Recap: Expectimax



- Your opponent behaves randomly with a given probability distribution,
- If you move left, your opponent will select actions with probability  $[0.5, 0.5]$
- If you move right, your opponent will select actions with  $[0.6, 0.4]$

# Recap: Expectimax



- Your opponent behaves randomly with a given probability distribution,
- If you move left, your opponent will select actions with probability [0.5, 0.5]
- If you move right, your opponent will select actions with [0.6, 0.4]

From MAX point of view, she is playing against a stochastic environment.

# Recap: Games: Modelling, Inference, Learning

- Modelling:
  - Formulating games as a search problem
  - Modeling your opponent
- Inference:
  - How to search for a strategy
  - Minimax, Expectimax (and Expectiminimax)
  - Pruning
  - Heuristic function and cut-off search
- Learning:
  - Learning heuristic functions
  - Modeling your opponent from data

# Recap: Games: Modelling, Inference, Learning

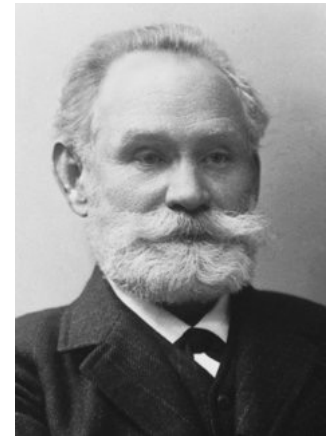
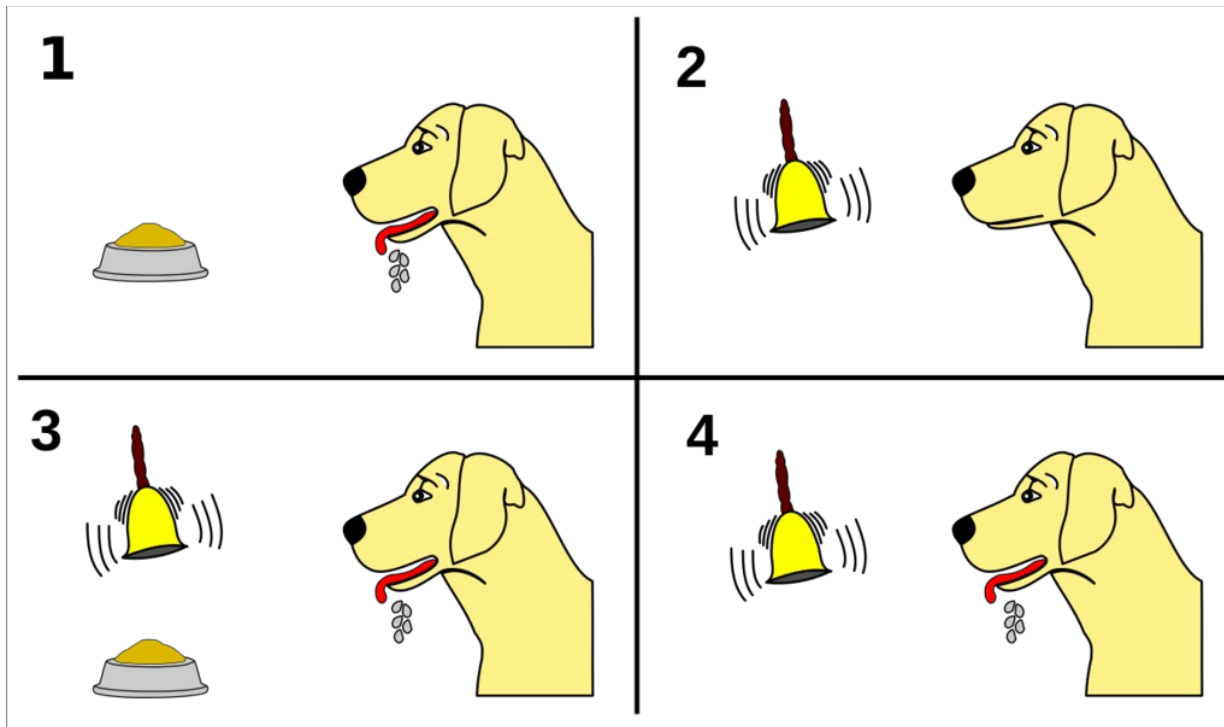
- Modelling:
  - Formulating games as a search problem
  - Modeling your opponent
- Inference:
  - How to search for a strategy
  - Minimax, Expectimax (and Expectiminimax)
  - Pruning
  - Heuristic function and cut-off search
- Learning:
  - Learning heuristic functions
  - Modeling your opponent from data

(Where are the data coming from?)

# Reinforcement Learning Lecture Series

- Overview (Today)
- Markov Decision Processes (Today)
- Bandits problems and exploration
- Reinforcement Learning Algorithms

# Reinforcement learning in the animal world

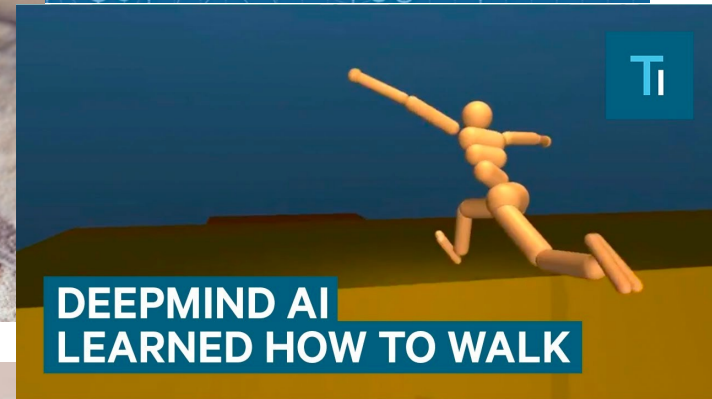
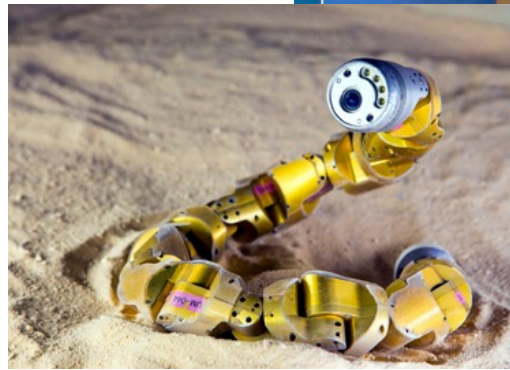
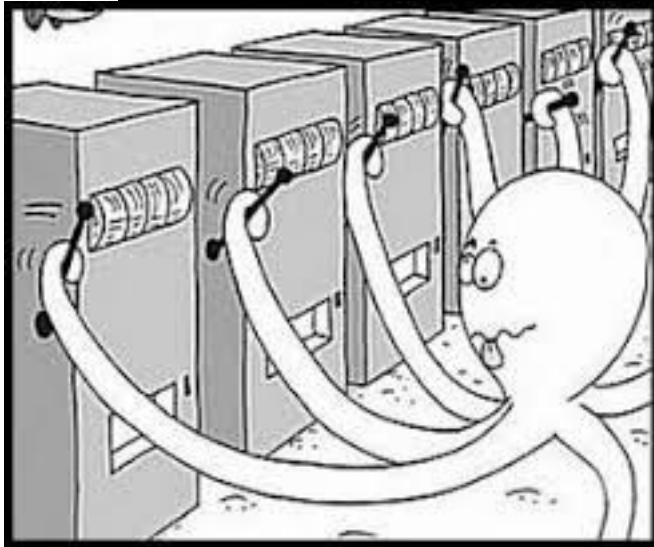
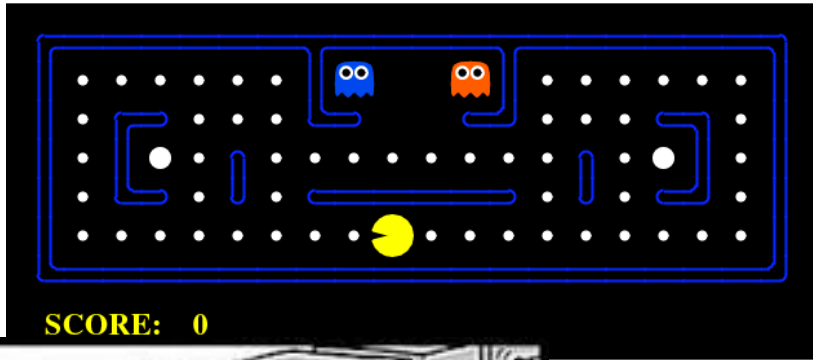


Ivan Pavlov  
(1849 - 1936)  
Nobel Laureate

- Learn from rewards
- Reinforce on the states that yield positive rewards



# Reinforcement learning: Applications



Recommendations

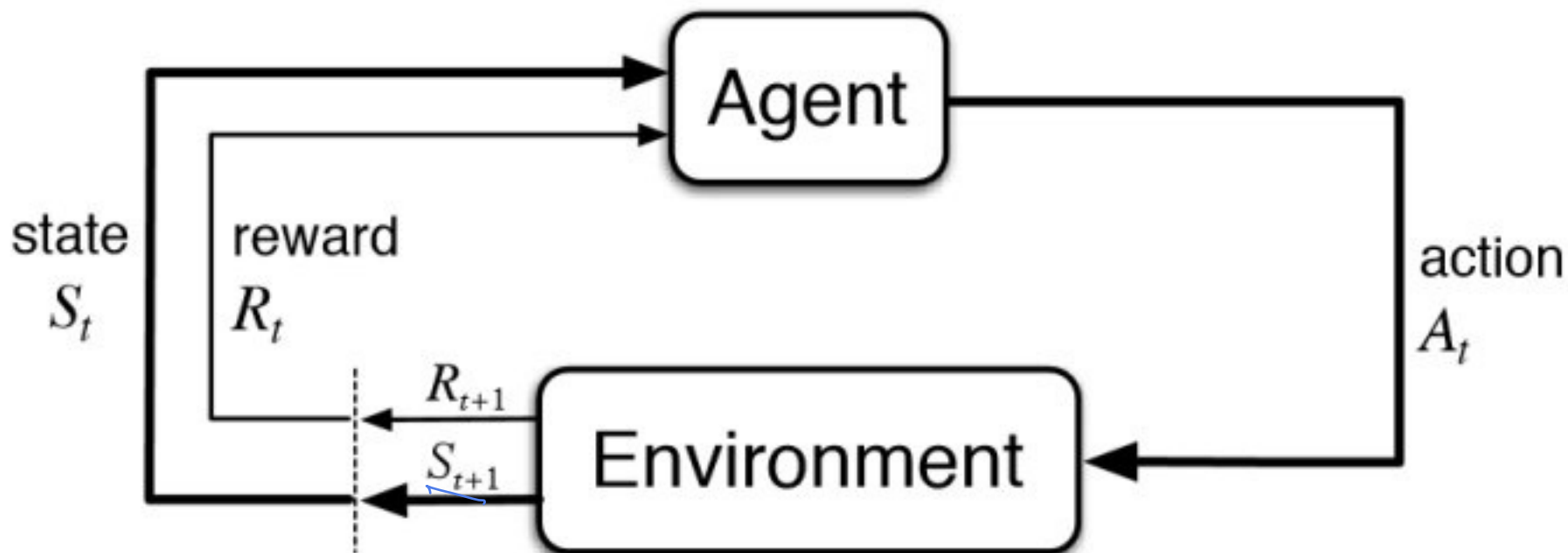


buy or not buy



# Reinforcement learning problem setup

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



# Reinforcement learning problem setup

# Reinforcement learning problem setup

- State, Action, Reward and Observation

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

# Reinforcement learning problem setup

- State, Action, Reward and Observation

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

- Policy:

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

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

# Reinforcement learning problem setup

$t$

- State, Action, Reward and Observation

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

- 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:  $\pi^* = \arg \max_{\pi \in \Pi} \mathbb{E} \left[ \sum_{t=1}^T R_t \right]$

- Infinite horizon RL:  $\pi^* = \arg \max_{\pi \in \Pi} \mathbb{E} \left[ \sum_{t=1}^{\infty} \gamma^{t-1} R_t \right]$

$T \approx \frac{1}{1-\gamma}$

$0 \leq \gamma < 1$

↑ discounted

# Reinforcement learning problem setup

- State, Action, Reward and Observation

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

- 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:  $\pi^* = \arg \max_{\pi \in \Pi} \mathbb{E} \left[ \sum_{t=1}^T R_t \right]$  **T: horizon**

- Infinite horizon RL:  $\pi^* = \arg \max_{\pi \in \Pi} \mathbb{E} \left[ \sum_{t=1}^{\infty} \gamma^{t-1} R_t \right]$

**$\gamma$ : discount factor**

# RL for robot control



- States: The physical world, e.g., location/speed/acceleration and so on.
- Observations: camera images, joint angles
- Actions: joint torques
- Rewards: stay balanced, navigate to target locations, serve and protect humans, etc.

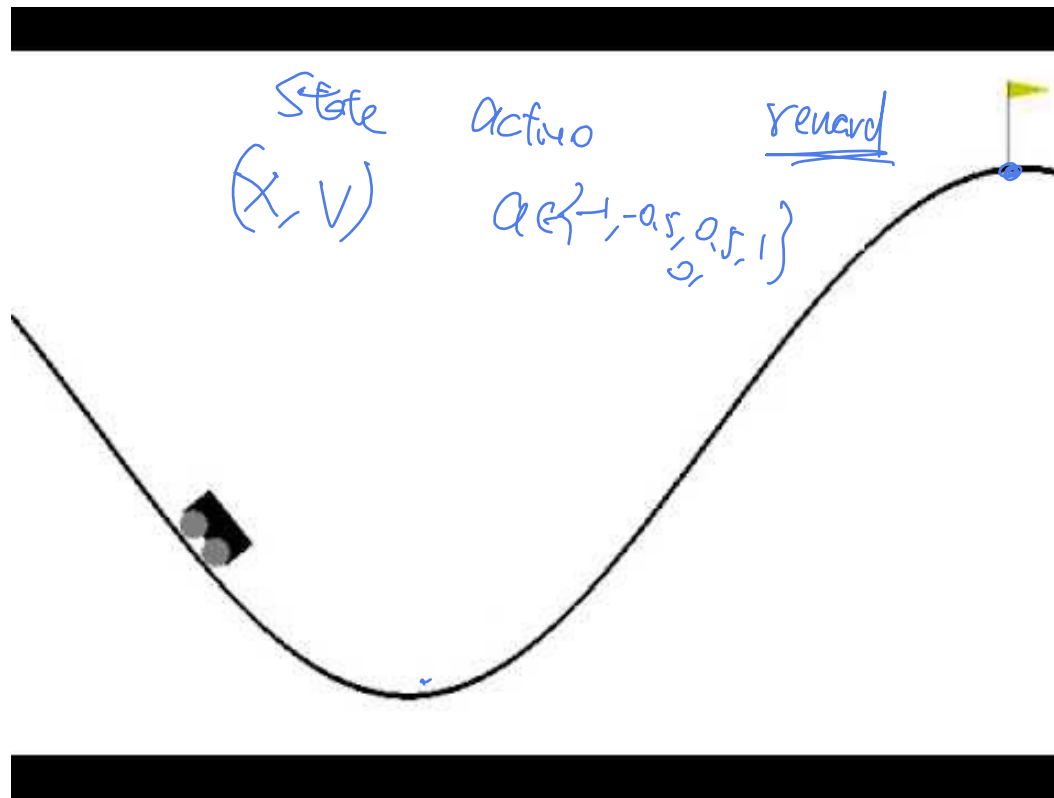


# RL for Inventory Management



- State: Inventory level, customer demand, competitor's inventory
- Observations: current inventory levels and sales history
- Actions: amount of each item to purchase
- Rewards: profit

# Demonstrating the learning process



- Mountain car:

<https://www.youtube.com/watch?v=U5w9PoKCOeM>

# Reading materials for RL

- Introduction:
  - Sutton and Barto: Chapter 1
- Markov Decision Processes
  - AIMA Section 17.1, Sutton and Barto: Ch 3
- Policy iterations / value iterations
  - AIMA Chapter 17.2-17.3, Sutton and Barto Ch 4.
- Bandits
  - Sutton and Barto Ch 2, AIMA Ch. 22.4
  - RL Algorithms: Sutton and Barto Ch 4, Ch 5, Ch 6, Ch 13

# Reinforcement learning is, arguably, the most general AI framework.

- RL: State, Action, Reward, Nothing is known.
- Simplified RL models:
  - iid state → Contextual bandits
  - No state, tabular action → Multi-arm bandits
  - iid state, no reward → Supervised Learning
  - Known dynamics / reward → Markov Decision Processes (Control/Cybernetics)
  - No reward / Unknown dynamics → System Identification

# Reinforcement learning is very challenging

- The agent needs to:
  - Learn the state-transitions ----- How the world works
  - Learning the costs / rewards ----- Cost of actions
  - Learning how to search ----- Come up with a good strategy

# Reinforcement learning is very challenging

- The agent needs to:
  - Learn the state-transitions ----- How the world works
  - Learning the costs / rewards ----- Cost of actions
  - Learning how to search ----- Come up with a good strategy
- All at the same time

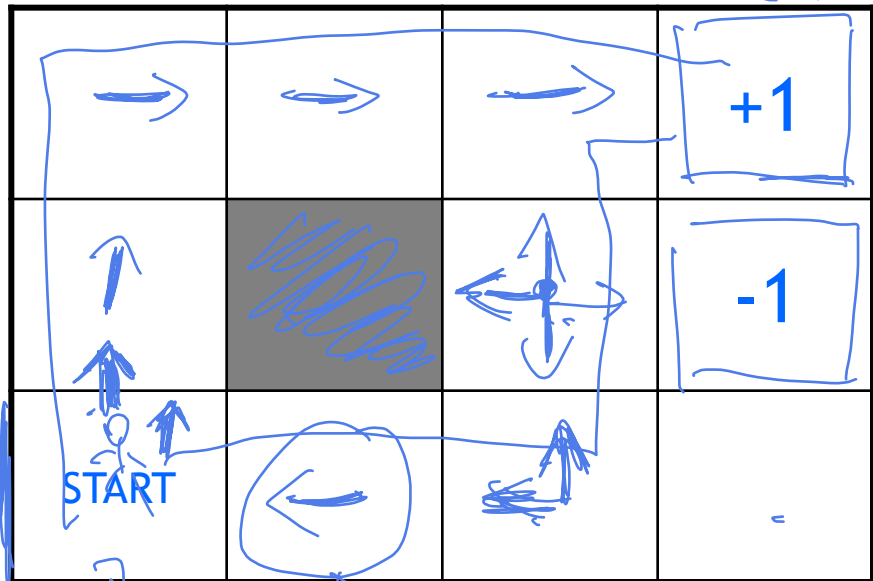
# Let us tackle different aspects of the RL problem one at a time

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

# Frozen Lake. (3 min discussion)

$S = [$

$(4,3)$

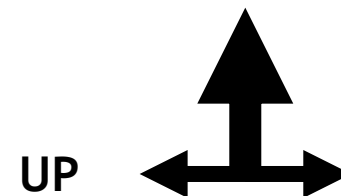


- reward +1 at  $[4,3]$ , -1 at  $[4,2]$

- reward -0.04 for each step
- what's the strategy to achieve max reward?
- what if the transitions were deterministic?

actions: UP, DOWN, LEFT, RIGHT

e.g.,



State-transitions with action UP:

100%	80%	move up
	10%	move left
	10%	move right

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



# Is this a solution?

→	→	→	+1
↑	■	·	-1
↑	·	·	·

- only if transitions are deterministic
  - not in this case (transitions are stochastic)
- solution/policy
  - mapping from each state to an action

# Optimal policy

→	→	→	+1
↑		↑	-1
↑	←	←	←

Reward for each step: -2

→	→	→	+1
↑	■	→ ↓	-1
→	→	→	↑

0.5

-1 0000

# Markov Decision Process (MDP)

- set of states  $S$ , set of actions  $A$ , initial state  $S_0$

- transition model  $P(s' | s, a)$

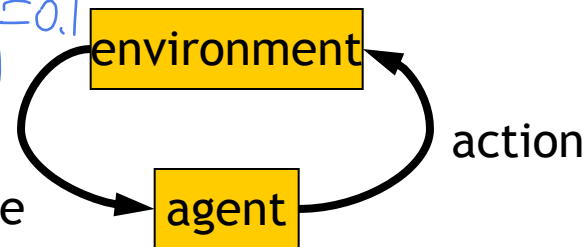
–  $P([1,2] | [1,1], \text{up}) = 0.8$

$P([1,1] | [1,1], \text{up}) = 0.1$   
 $P([2,1] | [1,1], \text{up}) = 0.1$   
 reward

- reward function  $r(s')$

–  $r([4,3]) = +1$  (Sometimes also depend on  $s, a$ )

$r(s, a, s')$   
 new state



- goal: maximize cumulative reward in the long run

- policy: mapping from  $S$  to  $A$

– Overloading notation:  $\pi(s)$  outputs an actions (for deterministic policy), or a probability distribution of actions (for stochastic policy).

– We also use  $\pi(a|s)$  as a short hand for  $P_\pi(a|s)$  --- the conditional probability table under policy  $\pi$

# Tabular MDP $\text{MDP} = (\mathcal{S}, \mathcal{A}, r, s_0, P)$

$P(s'|s,a)$ ,  $P(s_0)$ ,  $\mathbb{E}[R|S,A]$

- **Discrete** State, **Discrete** Action, Reward and Observation

$$S_t \in \mathcal{S} \quad A_t \in \mathcal{A} \quad R_t \in \mathbb{R} \quad \text{---} \quad O_t \in \mathcal{O}$$

- 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:  $\pi^* = \arg \max_{\pi \in \Pi} \mathbb{E} \left[ \sum_{t=1}^T R_t \right]$  **T: horizon**

– Infinite horizon RL:  $\pi^* = \arg \max_{\pi \in \Pi} \mathbb{E} \left[ \sum_{t=1}^{\infty} \gamma^{t-1} R_t \right]$

**$\gamma$ : discount factor**

# What is Markovian about MDPs?

- “Markov” generally means that given the present state, the future and the past are independent
- For Markov decision processes, “Markov” means action outcomes depend only on the current state

$$P(S_{t+1} = s' | S_t = s_t, A_t = a_t, \cancel{S_{t-1} = s_{t-1}}, \cancel{A_{t-1}}, \dots, \cancel{S_0 = s_0})$$
$$=$$
$$P(S_{t+1} = s' | \underline{S_t = s_t}, \underline{A_t = a_t})$$

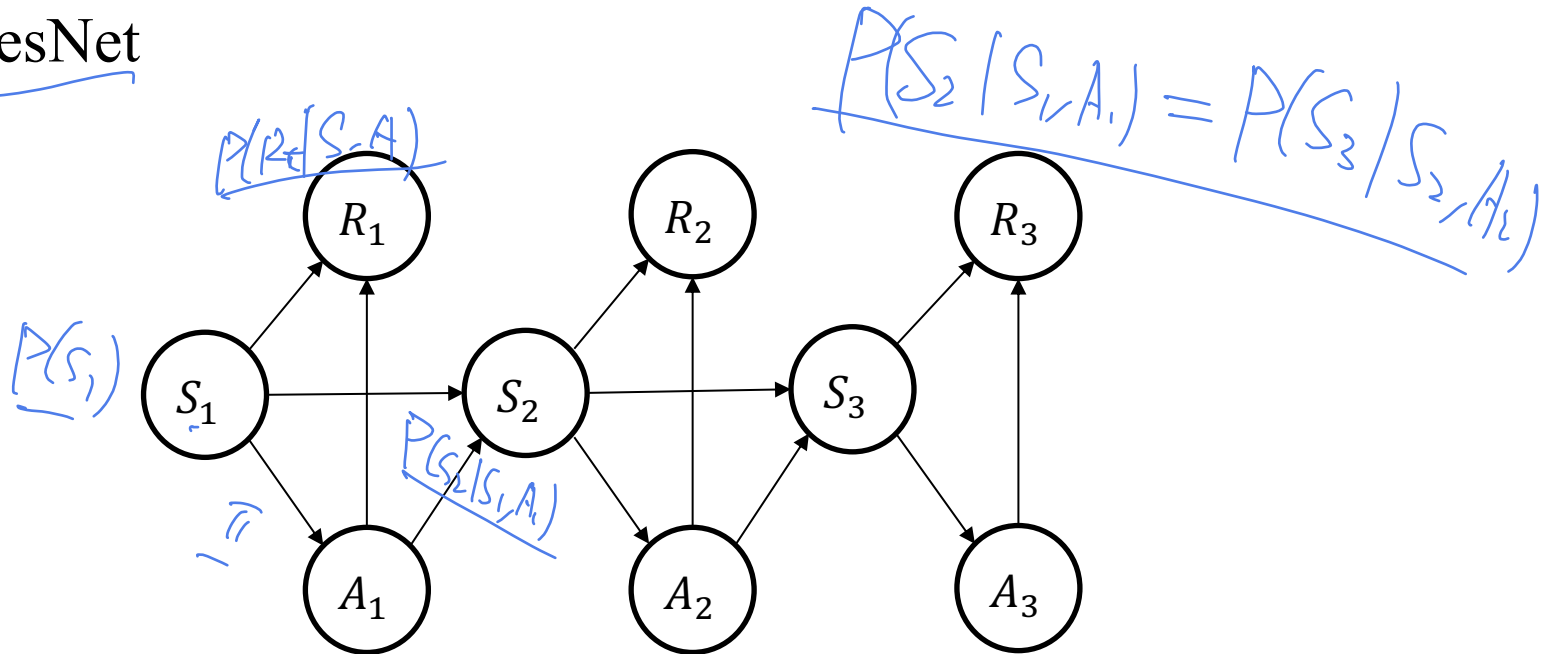
- This is just like search, where the future (available actions, states to transition to) could only depend on the current state (not the history)



Andrey Markov  
(1856-1922)

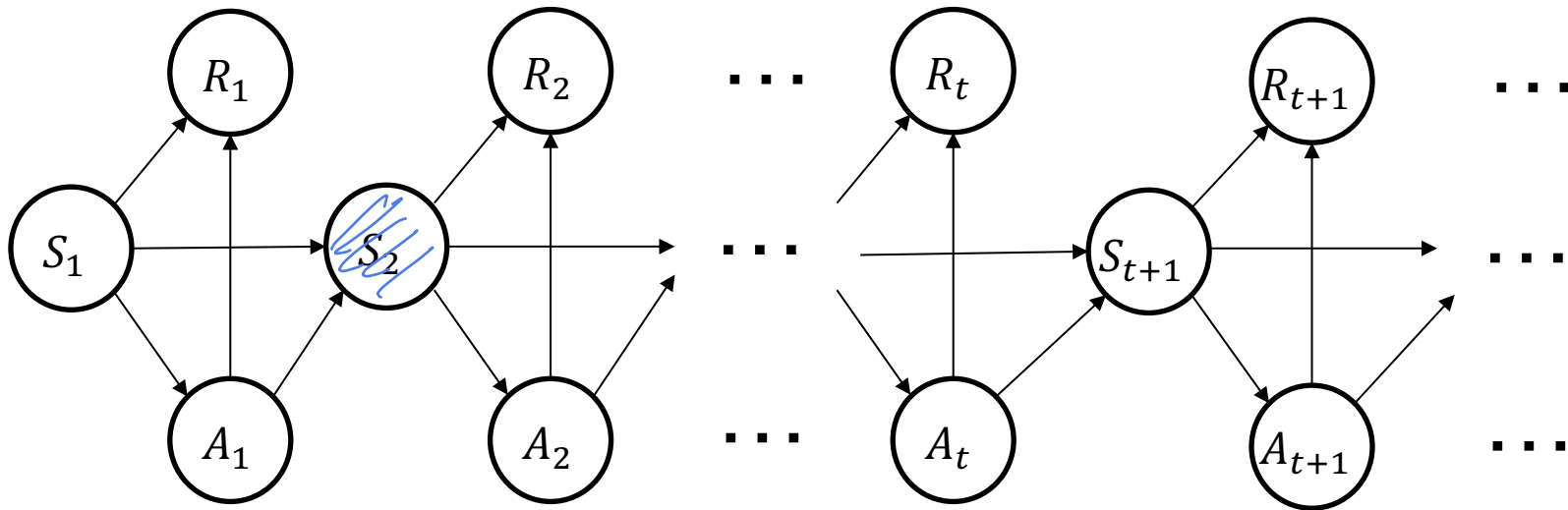
# This is a **conditional independence** assumption!

- Example of a finite horizon MDP with  $H = 3$ , as a BayesNet



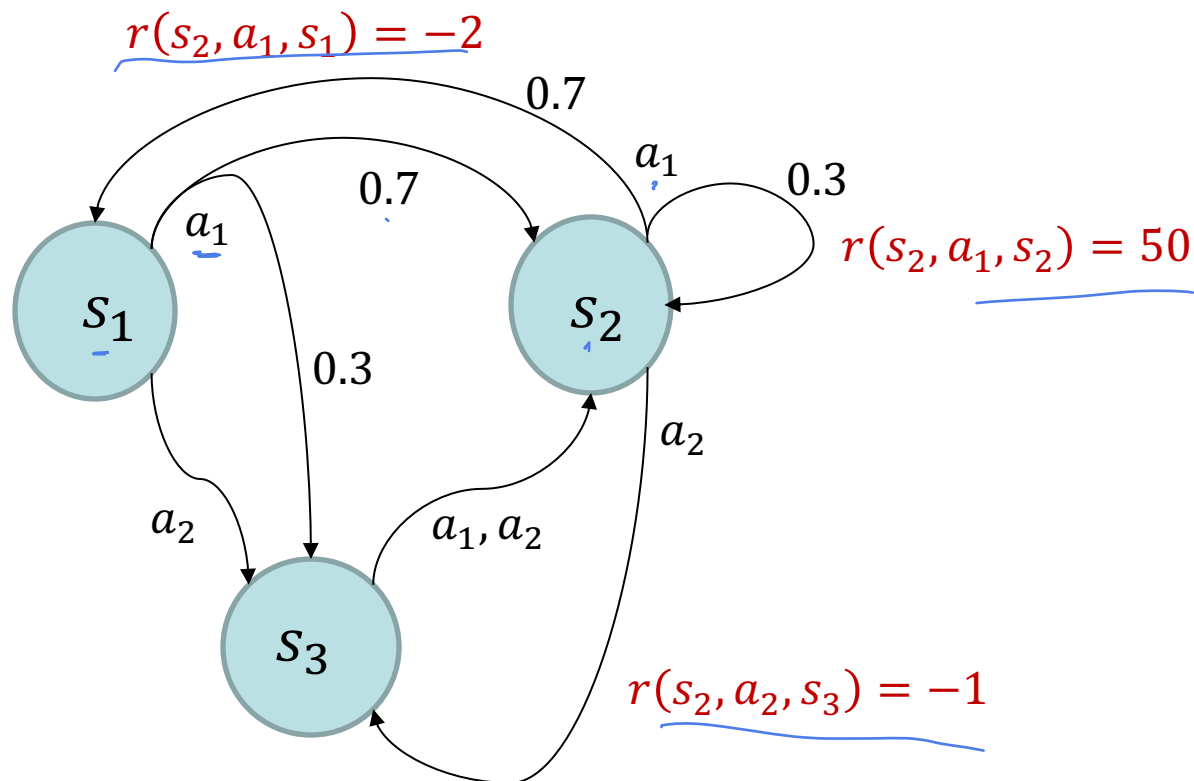
# This is a **conditional independence** assumption!

- Example of an infinite horizon MDP (as a BayesNet)



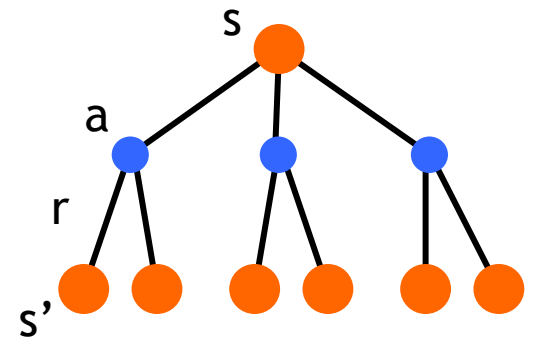


# State-space diagram representation of an MDP: An example with 3 states and 2 actions.



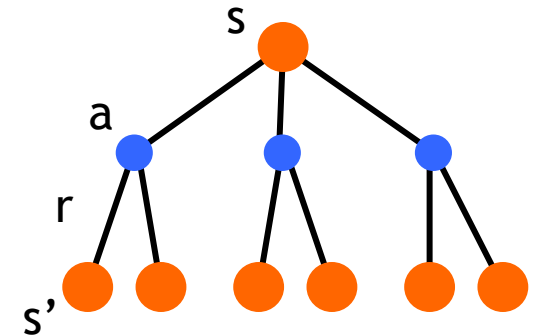
- \* The reward can be associated with only the state  $s'$  you transition into.
- \* Or the state that you transition from  $s$  and the action  $a$  you take.
- \* Or all three at the same time.

# Reward function and Value functions



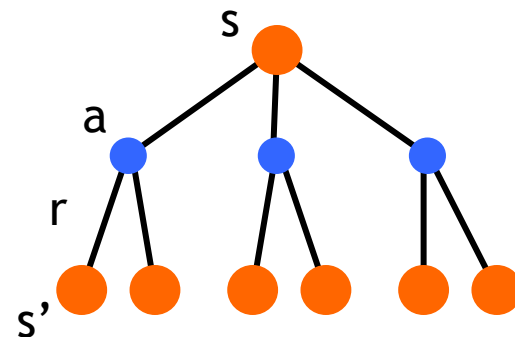
# Reward function and Value functions

- Immediate reward function  $r(s,a,s') = \mathbb{E}[R_t | S_1=s, A_t=a, S_2=s']$ 
  - expected **immediate** reward



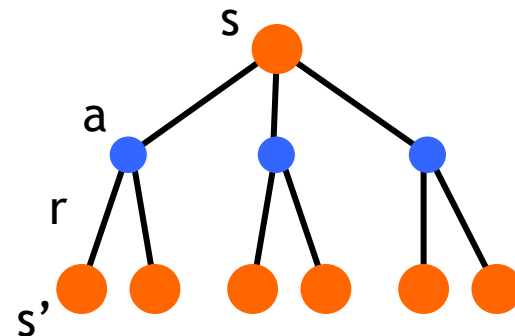
# Reward function and Value functions

- Immediate reward function  $r(s,a,s')$ 
  - expected **immediate** reward
- state value function:  $V^\pi(s)$ 
  - expected **long-term** return when starting in  $s$  and following  $\pi$



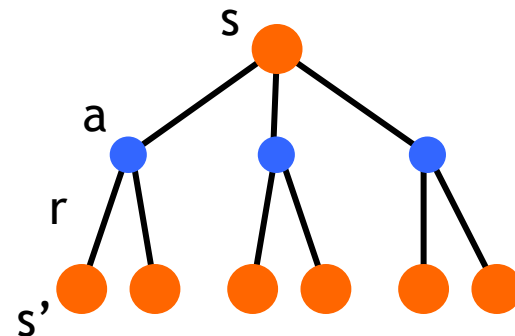
# Reward function and Value functions

- Immediate reward function  $r(s,a,s')$ 
  - expected **immediate** reward
- state value function:  $V^\pi(s)$ 
  - expected **long-term** return when starting in  $s$  and following  $\pi$
- state-action value function:  $Q^\pi(s,a)$ 
  - expected **long-term** return when starting in  $s$ , performing  $a$ , and following  $\pi$



# Reward function and Value functions

- Immediate reward function  $r(s,a,s')$ 
  - expected **immediate** reward
- state value function:  $V^\pi(s)$ 
  - expected **long-term** return when starting in  $s$  and following  $\pi$
- state-action value function:  $Q^\pi(s,a)$ 
  - expected **long-term** return when starting in  $s$ , performing  $a$ , and following  $\pi$
- useful for finding the optimal policy
  - can estimate from experience
  - pick the best action using  $Q^\pi(s,a)$



# Reward function and Value functions

# Reward function and Value functions

- Immediate reward function  $r(s,a,s')$

- expected **immediate** reward

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

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



# Reward function and Value functions

$\pi: S \rightarrow \mathcal{A}$

- 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  $\pi$

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

# Reward function and Value functions

- Immediate reward function  $r(s, a, s')$

- **expected immediate** reward

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

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

*Deterministic Policy  
 $\pi: S \rightarrow A$   
 Random Policy  
 $\pi: S \rightarrow P_A$   
 $A \sim P_A$*

- state value function:  $V^\pi(s)$

- **expected long-term** return when starting in  $s$  and following  $\pi$

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

*$A_1, A_2, \dots, A_t, \dots$   
 $S_2, S_3, \dots, S_t, \dots$*

- state-action value function:  $Q^\pi(s, a)$

- **expected long-term** return when starting in  $s$ , performing  $a$ , and following  $\pi$

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

*$Q = \pi(s)$*

# Bellman equations – the fundamental equations of MDP and RL

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

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

# Bellman equations – the fundamental equations of MDP and RL

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

$$V(s) = E_a [R + \gamma V(s') | s=s]$$

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

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

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

# Bellman equations – the fundamental equations of MDP and RL

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

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

- More quiz:
  - On AIMA textbook, reward is only a function of the state your transition into (Think about we collect a reward when we transition into  $s'$ ). What is the Bellman equation in this special case?
  - Sometimes, the reward is conditionally independent to  $s'$  given  $s, a$ . What is the Bellman equation in this special case?

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

→	→	→	+1
↑		→	-1
↑	→	←	←

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

→	→	→	+1
↑		→	-1
↑	→	←	←

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

→	→	→	+1
↑		→	-1
↑	→	←	←

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

→	→	→	+1
↑		→	-1
↑	→	←	←

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

→	→	→	+1
↑		→	-1
↑	→	←	←

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

→	→	→	+1
↑		→	-1
↑	→	←	←

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

→	→	→	+1
↑		→	-1
↑	→	←	←

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

→	→	→	+1
↑		→	-1
↑	→	←	←

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

→	→	→	+1
↑		→	-1
↑	→	←	←

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

$$\begin{aligned}
 V^\pi(s) &= \sum_a \pi(a|s) \sum_{s'} P(s'|s, a) [r(s, a, s') + \gamma V^\pi(s')] = \sum_a \pi(a|s) Q^\pi(s, a) \\
 &= 1.0 + 0.8 * (+1 - 0.04 + 0) + 0.1 * (-0.04 + V^\pi([3,2])) + \dots
 \end{aligned}$$

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

→	→	→	+1
↑		→	-1
↑	→	←	←

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

$$\begin{aligned}
 V^\pi(s) &= \sum_a \pi(a|s) \sum_{s'} P(s'|s, a) [r(s, a, s') + \gamma V^\pi(s')] = \sum_a \pi(a|s) Q^\pi(s, a) \\
 &= 1.0 * 0.8 * (+1 - 0.04) + 0.1 * (-0.04 + V^\pi([3,2])) + 0.1 * 0
 \end{aligned}$$

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

→	→	→	+1
↑		→	-1
↑	→	←	←

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

$$\begin{aligned}
 V^\pi(s) &= \sum_a \pi(a|s) \sum_{s'} P(s'|s, a) [r(s, a, s') + \gamma V^\pi(s')] = \sum_a \pi(a|s) Q^\pi(s, a) \\
 &= 1.0 * (+1 - 0.04 + 0) \\
 &+ 0.8 * (+1 - 0.04 + 0) \\
 &+ 0.1 * (-0.04 + V^\pi([3,2])) \\
 &+ 0.1 * (-0.04 + V^\pi([3,3]))
 \end{aligned}$$



# Optimal value functions

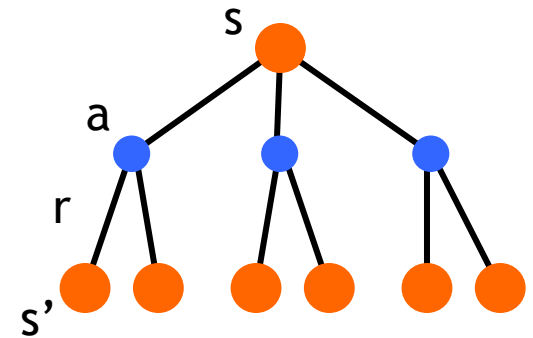
- there's a set of *optimal* policies
  - $V^\pi$  defines partial ordering on policies
  - they share the same optimal value function

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

- Bellman optimality equation

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

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

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

# 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



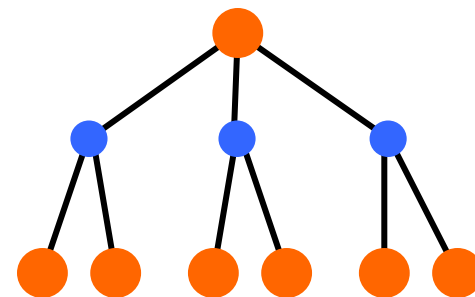
# Policy evaluation/improvement

- policy evaluation:  $\pi \rightarrow V^\pi$

- 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 evaluation/improvement

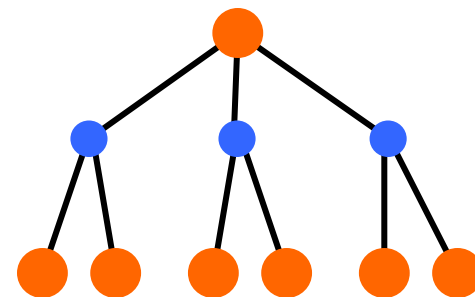
- policy evaluation:  $\pi \rightarrow V^\pi$

- 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:  $V^\pi \rightarrow \pi'$



# 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

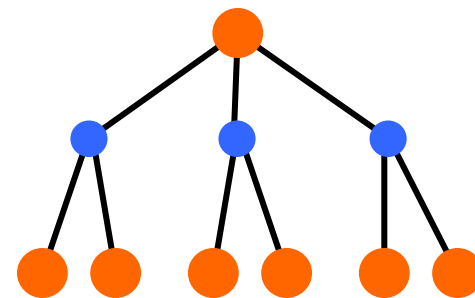
$$V_{k+1}^\pi(s) \leftarrow \sum_a \pi(a|s) \sum_{s'} P(s'|s, a) [r(s, a, s') + \gamma V_k^\pi(s')]$$

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

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

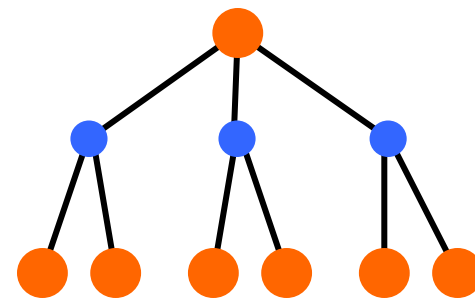
- start with an arbitrary value function  $V_0$ , iterate until  $V_k$  converges

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

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

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

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



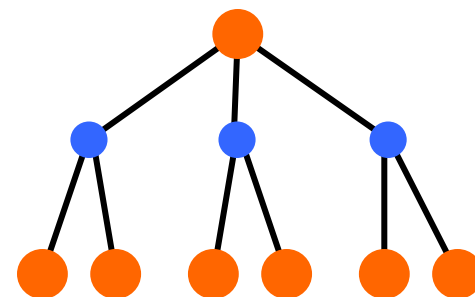


# Policy/Value iteration

- Policy iteration

$$\pi_0 \xrightarrow{E} V^{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} V^{\pi_1} \xrightarrow{I} \dots \xrightarrow{I} \pi^* \xrightarrow{E} V^*$$

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



# Policy/Value iteration

- Policy iteration

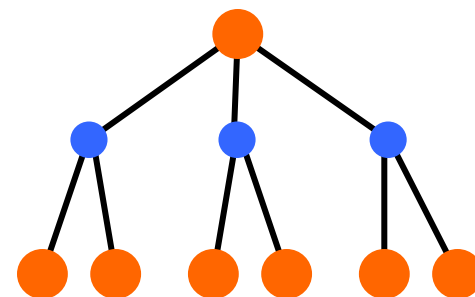
$$\pi_0 \xrightarrow{E} V^{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} V^{\pi_1} \xrightarrow{I} \dots \xrightarrow{I} \pi^* \xrightarrow{E} V^*$$

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

- Value iteration

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

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



## So far no learning at all. Next Tuesday

- More on MDPs
- MDP inferences
- Start bandits and exploration