

*“The rotten tree-trunk, until the very moment when the storm-blast breaks it in two, has all the appearance of might it ever had.”*


Isaac Asimov, *Foundation*



# **CMPUT 655**

## **Introduction to RL**

# Plan

- Note on the proof last week
- Course project 
- Markov decision processes
  - The problem formulation (and then a solution)

**Please, interrupt me at any time!**



# Chapter 3

# Finite Markov Decision Processes

# Markov Decision Processes – Why?

- “MDPs are a classical formalization of sequential decision making, where actions influence not just immediate rewards, but also subsequent situations, or states, and through those future rewards.”
- “Thus MDPs involve delayed reward and the need to trade off immediate and delayed reward.”
- “Whereas in bandit problems we estimated the value  $q_*(a)$  of each action  $a$ , in MDPs we estimate the value  $q_*(s,a)$  of each action  $a$  in each state  $s$ , or we estimate the value  $v_*(s)$  of each state given optimal action selections.”
- MDPs are a mathematically idealized form of the reinforcement learning problem for which precise theoretical statements can be made.

# Markov Decision Processes – Why?

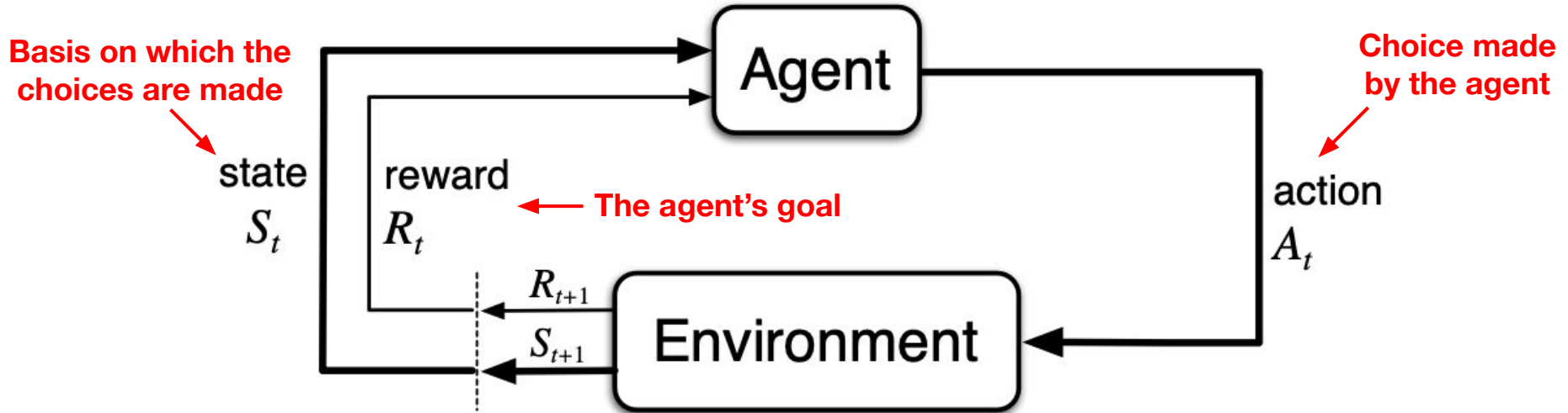
- “MDPs are a classical formalization of sequential decision making, where actions influence not just immediate rewards, but also subsequent situations, or states, and through those future rewards.”

**“In this chapter we introduce the formal problem of finite Markov decision processes, or finite MDPs, which we try to solve in the rest of the book.”**

MDPs we estimate the value  $q_*(s,a)$  of each action  $a$  in each state  $s$ , or we estimate the value  $v_*(s)$  of each state given optimal action selections.”

- MDPs are a mathematically idealized form of the reinforcement learning problem for which precise theoretical statements can be made.

# The Agent-Environment Interface



**Figure 3.1:** The agent–environment interface

$S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, \dots$

# Example 1: Navigating a maze

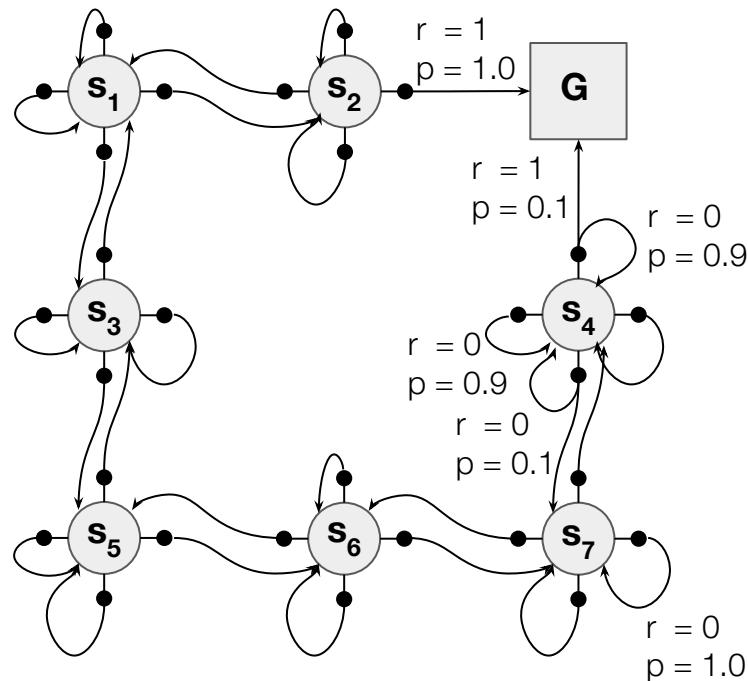
$s_1$	$s_2$	G
$s_3$		$s_4$
$s_5$	$s_6$	$s_7$

*States:* cell #

*Actions:* [up, down, left, right]

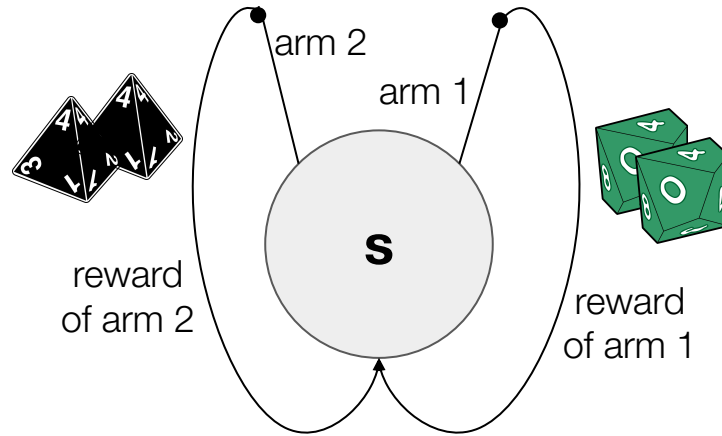
*Reward:* +1 upon arrival to G  
0 otherwise

*Dynamics:* deterministic outside mud puddle  
at the mud puddle you can get stuck  
with probability 0.9.





# Example 2: Bandits



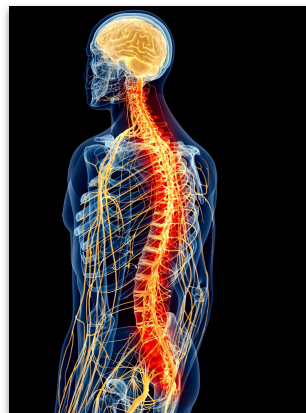


# Where's the boundary between agent and environment?

It depends!

And it is often much closer than you think!

“The agent-environment boundary represents the limit of the agent’s *absolute control*, not of its knowledge.”



## Formalizing the Agent-Environment Interface

$$p(s', r | s, a) \doteq \Pr\{S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a\}$$

$$\sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r | s, a) = 1, \text{ for all } s \in \mathcal{S}, a \in \mathcal{A}(s)$$

$$p(s' | s, a) \doteq \Pr\{S_t = s' \mid S_{t-1} = s, A_{t-1} = a\} = \sum_{r \in \mathcal{R}} p(s', r | s, a)$$

## Formalizing the Agent-Environment Interface

$$r(s, a) \doteq \mathbb{E}[R_t \mid S_{t-1} = s, A_{t-1} = a] = \sum_{r \in \mathcal{R}} r \sum_{s' \in \mathcal{S}} p(s', r \mid s, a)$$

$$r(s, a, s') \doteq \mathbb{E}[R_t \mid S_{t-1} = s, A_{t-1} = a, S_t = s'] = \sum_{r \in \mathcal{R}} r \frac{p(s', r \mid s, a)}{p(s' \mid s, a)}$$

**Can you show this?**

# The Markov Property

“The future is independent of the past given the present”

$$\Pr(S_{t+1} | S_t) = \Pr(S_{t+1} | S_1, \dots, S_t)$$

This should probably be seen as a restriction on the state, not on the decision process.

## A note on the Markov property

**Definition:** We say that  $(S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, \dots)$  has the *Markov property* if for any  $t \geq 0$ ,  $s_0, s_1, \dots, s_{t+1} \in \mathcal{S}$ ,  $a_0, a_1, \dots, a_t \in \mathcal{A}$ , and  $r_1, r_2, \dots, r_{t+1} \in \mathcal{R}$ , it holds that

$$\Pr(R_{t+1} = r_{t+1}, S_{t+1} = s_{t+1} \mid S_0 = s_0, A_0 = a_0, R_1 = r_1, \dots, R_t = r_t, S_t = s_t, A_t = a_t)$$

only depends on the values of  $s_t$ ,  $a_t$ ,  $s_{t+1}$  and  $r_{t+1}$ . In particular, none of the other past values matter when calculating probabilities of the form above. That is:

$$\begin{aligned} &\Pr(R_{t+1} = r_{t+1}, S_{t+1} = s_{t+1} \mid S_0 = s_0, A_0 = a_0, R_1 = r_1, \dots, R_t = r_t, S_t = s_t, A_t = a_t) \\ &= \Pr(R_{t+1} = r_{t+1}, S_{t+1} = s_{t+1} \mid S_t = s_t, A_t = a_t). \end{aligned}$$

## A note on the Markov property

Definition: We say that  $(S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, \dots)$  has the *Markov property* if for any  $t \geq 0$ ,  $s_0, s_1, \dots, s_{t+1} \in \mathcal{S}$ ,  $a_0, a_1, \dots, a_t \in \mathcal{A}$ , and  $r_1, r_2, \dots, r_{t+1} \in \mathcal{R}$ , it holds that

**The Markov property does not mean that the state representation tells all that would be useful to know, only that it has not forgotten anything that would be useful to know.**

$$\begin{aligned} \Pr(R_{t+1} = r_{t+1}, S_{t+1} = s_{t+1} \mid S_0 = s_0, A_0 = a_0, R_1 = r_1, \dots, R_t = r_t, S_t = s_t, A_t = a_t) \\ = \Pr(R_{t+1} = r_{t+1}, S_{t+1} = s_{t+1} \mid S_t = s_t, A_t = a_t). \end{aligned}$$





# Reward Hypothesis

*“That all of what we mean by goals and purposes can be well thought of as the maximization of the expected value of the cumulative sum of a received scalar signal (called reward).”*

## The ultimate goal: Maximize Returns

$$G_t \doteq R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T$$

End of an episode

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

Continuing task

$$\begin{aligned} G_t &\doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \\ &= R_{t+1} + \gamma (R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + \cdots) \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned}$$

# Unifying Notation

$$G_t \doteq \sum_{k=0}^T R_{t+k+1}$$

$$G_t \doteq \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

- We can't use the same notation for episodic and continuing tasks because:
  - We are not specifying the episodes in the indices of an episodic task, we should actually have  $R_{t,i}$ .
  - In continuing tasks we have a sum over infinite numbers and in episodic tasks we sum over finite numbers.

# Unifying Notation

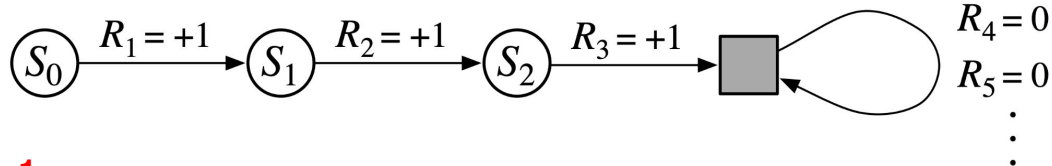
$$G_t \doteq \sum_{k=0}^T R_{t+k+1}$$

$$G_t \doteq \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

- We can't use the same notation for episodic and continuing tasks because:
  - We are not specifying the episodes in the indices of an episodic task, we should actually have  $R_{t,i}$ .
  - In continuing tasks we have a sum over infinite numbers and in episodic tasks we sum over finite numbers.
- Solution:
  - It is mostly fine to drop the episode number.
  - We create an absorbing state!

$$G_t \doteq \sum_{k=t+1}^T \gamma^{k-t-1} R_k$$

**$T = \infty$  or  $\gamma = 1$   
(but not both)**

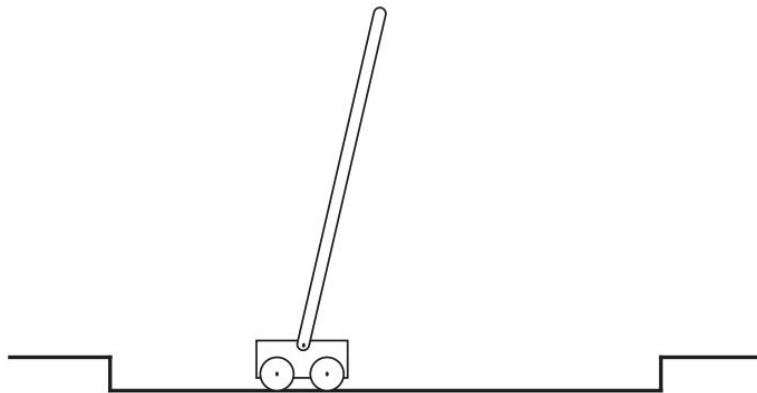




# Example

## Example 3.4: Pole-Balancing

The objective in this task is to apply forces to a cart moving along a track so as to keep a pole hinged to the cart from falling over: A failure is said to occur if the pole falls past a given angle from vertical or if the cart runs off the track. The pole is reset to vertical after each failure. This task could be treated as episodic, where the natural



episodes are the repeated attempts to balance the pole. The reward in this case could be +1 for every time step on which failure did not occur, so that the return at each time would be the number of steps until failure. In this case, successful balancing forever would mean a return of infinity. Alternatively, we could treat pole-balancing as a continuing task, using discounting. In this case the reward would be  $-1$  on each failure and zero at all other times. The return at each time would then be related to  $-\gamma^{K-1}$ , where  $K$  is the number of time steps before failure (as well as to the times of later failures). In either case, the return is maximized by keeping the pole balanced for as long as possible. ■

## Exercise 3.6 of the Textbook

*Exercise 3.6* Suppose you treated pole-balancing as an episodic task but also used discounting, with all rewards zero except for  $-1$  upon failure. What then would the return be at each time? How does this return differ from that in the discounted, continuing formulation of this task? □



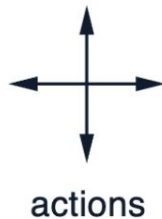
## Exercise 3.7 of the Textbook

*Exercise 3.7* Imagine that you are designing a robot to run a maze. You decide to give it a reward of +1 for escaping from the maze and a reward of zero at all other times. The task seems to break down naturally into episodes—the successive runs through the maze—so you decide to treat it as an episodic task, where the goal is to maximize expected total reward (3.7). After running the learning agent for a while, you find that it is showing no improvement in escaping from the maze. What is going wrong? Have you effectively communicated to the agent what you want it to achieve?

## Exercise 3.8 of the Textbook

*Exercise 3.8* Suppose  $\gamma = 0.5$  and the following sequence of rewards is received  $R_1 = -1$ ,  $R_2 = 2$ ,  $R_3 = 6$ ,  $R_4 = 3$ , and  $R_5 = 2$ , with  $T = 5$ . What are  $G_0, G_1, \dots, G_5$ ? Hint: Work backwards. □

# Practice Exercise



	1	2	3
4	5	6	7
8	9	10	11
12	13	14	

$R_t = -1$   
on all transitions

$$p(6, -1 | 5, \text{right}) =$$

$$p(7, -1 | 7, \text{right}) =$$

$$p(10, r | 5, \text{right}) =$$

## Exercise 3.10 of the Textbook

*Exercise 3.10* Prove the second equality in (3.10).

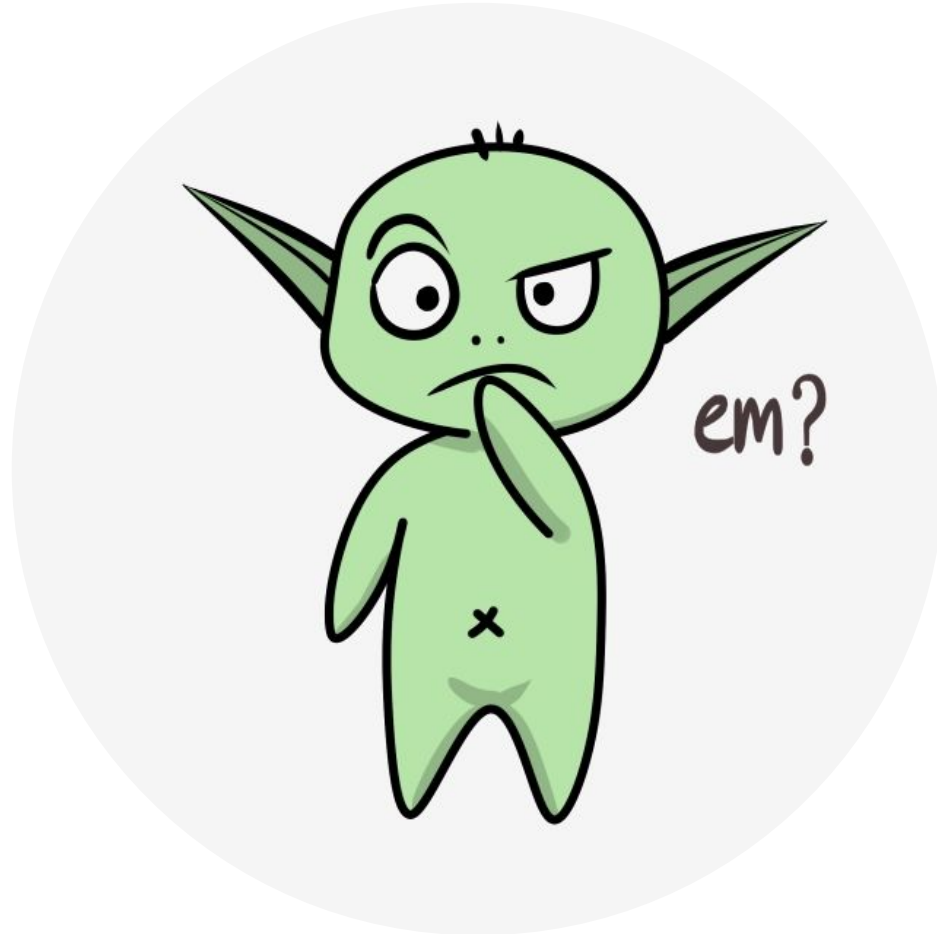
$$G_t = \sum_{k=0}^{\infty} \gamma^k = \frac{1}{1 - \gamma}$$

## Practice Exercise

Prove that the discounted sum of rewards is always finite, if the rewards are bounded:  $|R_{t+1}| \leq R_{\max}$  for all  $t$  for some finite  $R_{\max} > 0$ .

$$\left| \sum_{i=0}^{\infty} \gamma^i R_{t+1+i} \right| < \infty \text{ for } \gamma \in [0, 1). \quad \text{Hint: Recall that } |a + b| < |a| + |b|.$$

# Solution Practice Exercise



# Value Functions and Policies

- *Value functions are “functions of states (or state-action pairs) that estimate how good it is for the agent to be in a given state”.*
- “How good” means expected return.
- Expected returns depend on how the agent behaves, that is, its *policy*.



# Policy

- A policy is a mapping from states to probabilities of selecting each possible action:

$$\pi : \mathcal{S} \rightarrow \Delta(\mathcal{A})$$

in other words,  $\pi(a|s)$  is the probability that  $A_t = a$  if  $S_t = s$ .

*Exercise 3.11* If the current state is  $S_t$ , and actions are selected according to a stochastic policy  $\pi$ , then what is the expectation of  $R_{t+1}$  in terms of  $\pi$  and the four-argument function  $p$  (3.2)? □

# Value Function

- The value function of a state  $s$  under a policy  $\pi$ , denoted  $v_\pi(s)$  is the expected return when starting in  $s$  and following  $\pi$  thereafter.

state-value  
function for  
policy  $\pi$

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s] = \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s \right]$$

$$q_\pi(s, a) \doteq \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] = \mathbb{E}_\pi \left[ \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a \right]$$

action-value  
function for  
policy  $\pi$

## Exercises from the Textbook

*Exercise 3.12* Give an equation for  $v_\pi$  in terms of  $q_\pi$  and  $\pi$ .

*Exercise 3.13* Give an equation for  $q_\pi$  in terms of  $v_\pi$  and the four-argument  $p$ .

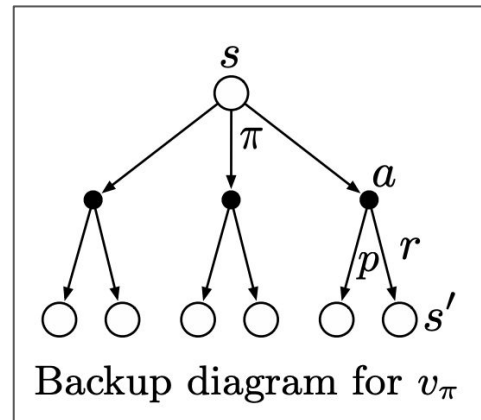
# Value Functions Satisfy Recursive Relationships

$$v_{\pi}(s) \doteq \mathbb{E}_{\pi}[G_t \mid S_t = s]$$

# Value Functions Satisfy Recursive Relationships

$$\begin{aligned}
 v_{\pi}(s) &\doteq \mathbb{E}_{\pi}[G_t \mid S_t = s] \\
 &= \mathbb{E}_{\pi}[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \\
 &= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r | s, a) \left[ r + \gamma \mathbb{E}_{\pi}[G_{t+1} | S_{t+1} = s'] \right] \\
 &= \sum_a \pi(a|s) \sum_{s', r} p(s', r | s, a) \left[ r + \gamma v_{\pi}(s') \right]
 \end{aligned}$$

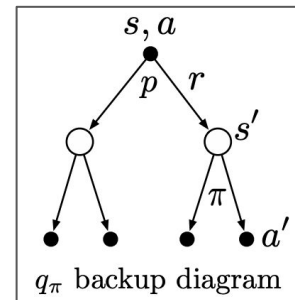
**This is a system of linear equations!**

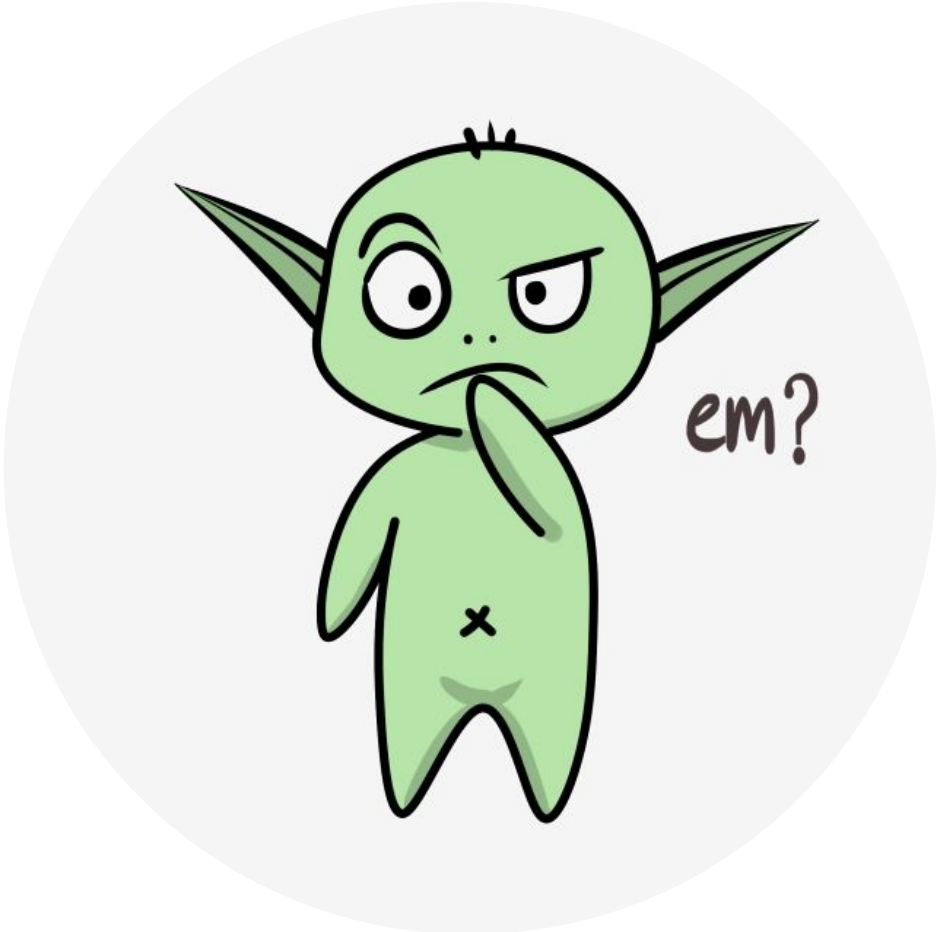




# State-Action Value Functions Satisfy Recursive Relationships

*Exercise 3.17* What is the Bellman equation for action values, that is, for  $q_\pi$ ? It must give the action value  $q_\pi(s, a)$  in terms of the action values,  $q_\pi(s', a')$ , of possible successors to the state–action pair  $(s, a)$ . Hint: The backup diagram to the right corresponds to this equation. Show the sequence of equations analogous to (3.14), but for action values. □







# Value Functions in Matrix Notation

$$\mathbf{v}_\pi \in \mathbb{R}^{|\mathcal{S}|}$$

Reward vector:

$$\mathbf{r} \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{A}| \times 1}$$

$$\mathbf{r}[sa] = \mathbb{E}[r|s, a]$$

Transition probability matrix:

$$\mathbf{P} \in [0, 1]^{|\mathcal{S}| \times |\mathcal{A}| \times |\mathcal{S}|}$$

$$\mathbf{P}[sa][s'] = p(s'|s, a) \geq 0 \quad \sum_{s'} p(s'|s, a) = 1$$

# Value Functions in Matrix Notation

$$\mathbf{v}_\pi \in \mathbb{R}^{|\mathcal{S}|}$$

Policy matrix:

$$\mathbf{\Pi} \in [0, 1]^{|\mathcal{S}| \times |\mathcal{S}| \times |\mathcal{A}|}$$

$$\mathbf{\Pi} = \begin{bmatrix} \boldsymbol{\pi}(a|s_1)^\top & & & \\ & \boldsymbol{\pi}(a|s_2)^\top & & \\ & & \dots & \\ & & & \boldsymbol{\pi}(a|s_{|\mathcal{S}|})^\top \end{bmatrix}$$

$$\boldsymbol{\pi}(a|s_1) = [\pi(a_1|s_1) \quad \pi(a_2|s_1) \quad \dots \quad \pi(|\mathcal{A}||s_1)]$$

Notice:

$$\mathbf{\Pi P} \in [0, 1]^{|\mathcal{S}| \times |\mathcal{S}|}$$

$$\mathbf{P\Pi} \in [0, 1]^{|\mathcal{S}| \times |\mathcal{A}| \times |\mathcal{S}| \times |\mathcal{A}|}$$

# Value Functions in Matrix Notation

$$\mathbf{v}_\pi \in \mathbb{R}^{|\mathcal{S}|}$$

$$\mathbf{r} \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}| \times 1} \quad \mathbf{P} \in [0, 1]^{|\mathcal{S}||\mathcal{A}| \times |\mathcal{S}|} \quad \mathbf{\Pi} \in [0, 1]^{|\mathcal{S}| \times |\mathcal{S}||\mathcal{A}|}$$

$$\mathbf{v} = \mathbf{\Pi} \mathbf{r} + \gamma \mathbf{\Pi} \mathbf{P} \mathbf{v}$$

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s', r} p(s', r | s, a) \left[ r + \gamma v_\pi(s') \right]$$



# Optimal Policies and Optimal Value Functions

- Value functions define a partial ordering over policies.
  - $\pi \geq \pi'$  iff  $v_\pi(s) \geq v_{\pi'}(s)$  for all  $s \in \mathcal{S}$ .
  - There is always at least one policy that is better than or equal to all other policies. The *optimal policy*.

$$v_*(s) \doteq \max_{\pi} v_{\pi}(s)$$

$$q_*(s, a) = \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a]$$

$$q_*(s, a) \doteq \max_{\pi} q_{\pi}(s, a)$$

# Optimal Policies and Optimal Value Functions

- Because  $v_*$  is the value function for a policy, it must satisfy the self-consistency condition given by the Bellman equation for state values.

$$v_*(s) = \max_{a \in \mathcal{A}(s)} q_{\pi_*}(s, a)$$

# Optimal Policies and Optimal Value Functions

- Because  $v_*$  is the value function for a policy, it must satisfy the self-consistency condition given by the Bellman equation for state values.

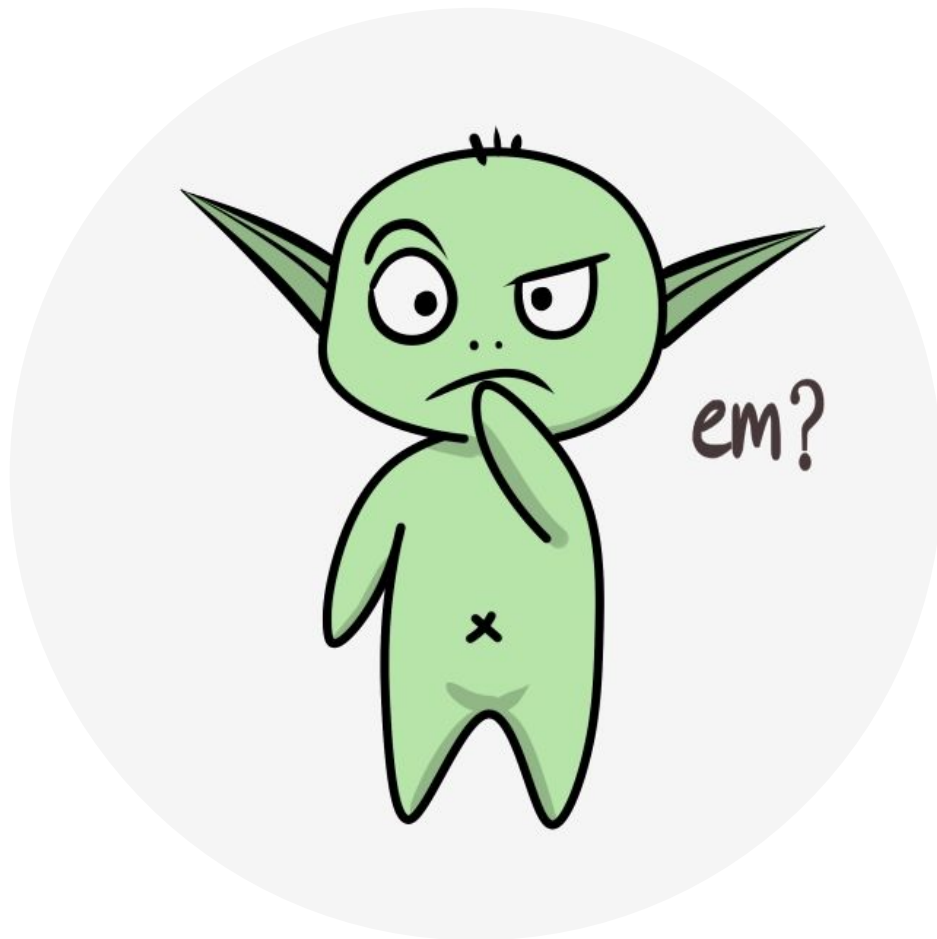
$$\begin{aligned}
 v_*(s) &= \max_{a \in \mathcal{A}(s)} q_{\pi_*}(s, a) \\
 &= \max_a \mathbb{E}_{\pi_*}[G_t \mid S_t = s, A_t = a] \\
 &= \max_a \mathbb{E}_{\pi_*}[R_{t+1} + \gamma G_{t+1} \mid S_t = s, A_t = a] \\
 &= \max_a \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a] \\
 &= \max_a \sum_{s', r} p(s', r \mid s, a) [r + \gamma v_*(s')].
 \end{aligned}$$

$$\begin{aligned}
 q_*(s, a) &= \mathbb{E}\left[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') \mid S_t = s, A_t = a\right] \\
 &= \sum_{s', r} p(s', r \mid s, a) \left[r + \gamma \max_{a'} q_*(s', a')\right].
 \end{aligned}$$

# Also...

I have highlighted a couple of exercises during the class, but there are more. The exercises in Chapter 3 of the book are great. I particularly encourage you to look at Exercises 3.25 – 3.29 as well.





## Reinforcement learning is very related to search algorithms

*“Heuristic search methods can be viewed as expanding the right-hand side of the equation below several times, up to some depth, forming a “tree” of possibilities, and then using a heuristic evaluation function to approximate  $v_*$ , at the “leaf” nodes.”*

$$v_*(s) = \max_a \sum_{s', r} p(s', r | s, a) [r + \gamma v_*(s')].$$

# Yay! We solved sequential decision-making problems

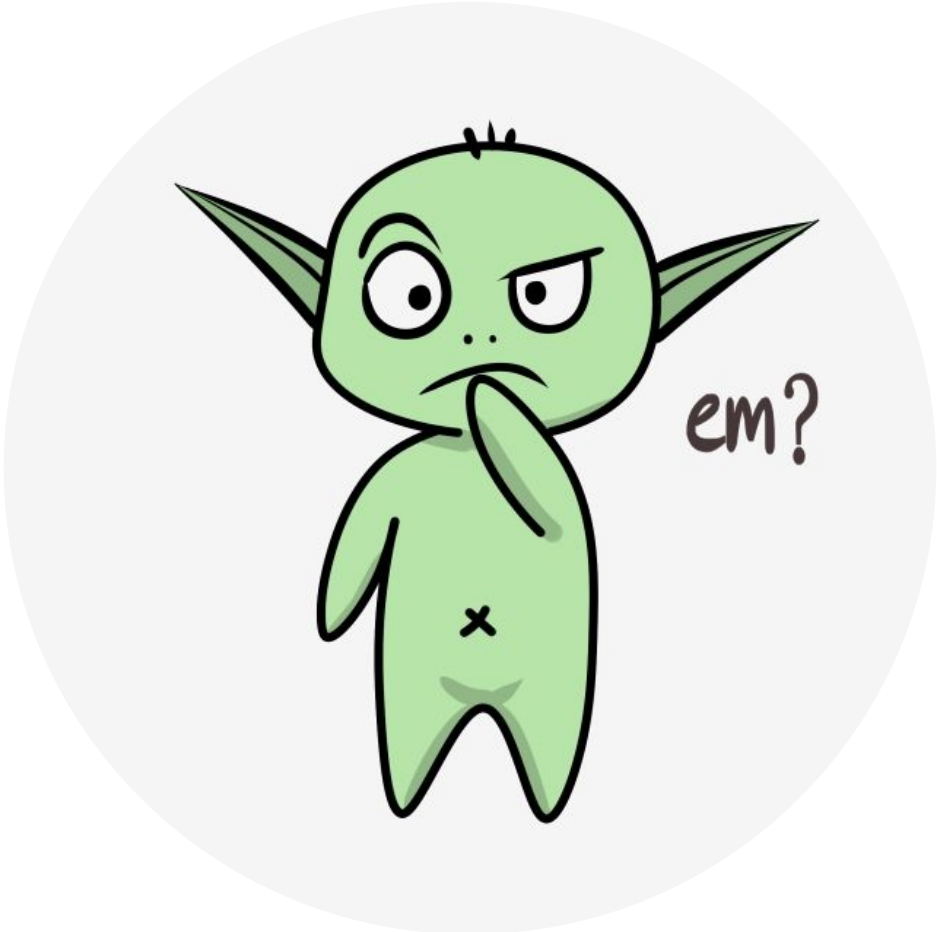
Except...

- 1.
- 2.
- 3.

# Yay! We solved sequential decision-making problems

Except...

1. we need to know the dynamics of the environment
2. we have enough computational resources to solve the system of linear eq.
3. the Markov property



# Next class

- What **I** plan to do:
  - Dynamic Programming / Bellman Equations
  - (Maybe) Monte-Carlo Methods
- What **YOU** need to do for next class:
  - Week 4 of Fundamentals of RL: Dynamic Programming (Practice quiz and Progr. assignment)
  - Week 2 of Sample-based Methods: Monte Carlo Methods for Prediction & Control
    - Week 1 of Sample-based Methods is just a Welcome video.
    - Some of you **are not** enrolled in the Sample-based Methods module on Coursera.