"All have their worth," said Yavanna,

"and each contributes to the worth of the others".

J.R.R. Tolkien, *The Silmarillion*

CMPUT 365
Introduction to RL

Marlos C. Machado

Class 10/35

Brought to you by NeurAlbertaTech

natHACKS 2024

NOVEMBER 14-17

explore neurotech / expand your skills / network / win prizes

apply at nathacks.ca

join our newsletter

support@neuralberta.tech

Plan

- Dynamic programming
  - Finally, a solution method (albeit limited)!

Reminder I

You **should be enrolled in the private session** we created in Coursera for CMPUT 365.

I **cannot** use marks from the public repository for your course marks.

You **need** to **check**, **every time**, if you are in the private session and if you are submitting quizzes and assignments to the private section.

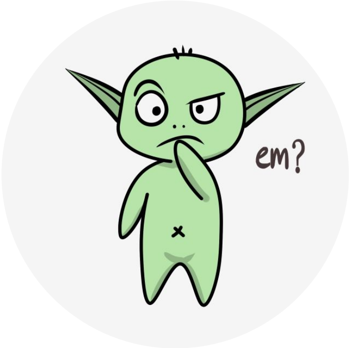
The deadlines in the public session **do not align** with the deadlines in Coursera.

If you have any questions or concerns, **talk with the TAs** or email us `cmput365@ualberta.ca`.

Reminder II

- Practice quiz for Coursera's Dynamic Programming module is due today.
  Fundamentals of RL: Dynamic Programming – Week 4.
- Progr. assign. for Coursera's Dynamic Programming module is due Friday.
  Fundamentals of RL: Dynamic Programming – Week 4.

Please, interrupt me at any time!



https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

Dynamic Programming – Why?

- "DP provides an essential foundation for the understanding of the methods presented in the rest of this book".
- … but "classical DP algorithms are of limited utility in reinforcement learning both because of their assumption of a perfect model and because of their great computational expense".
- "all of these [RL] methods can be viewed as attempts to achieve much the same effect as DP, only with less computation and without assuming a perfect model of the environment".

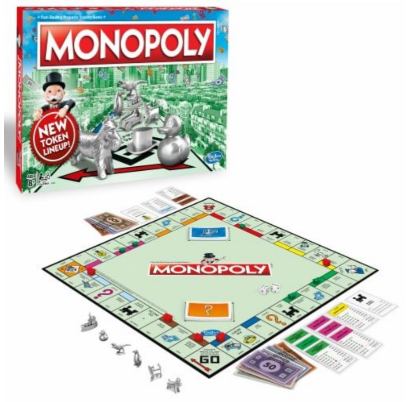
Models and Planning

- How should we think about p(s’, r | s, a)? **It is a model. It tells us everything that is possible and impossible to happen (and their probability!)**
- Is dynamic programming different from what we did in bandits?

Figuring out how to act

Imagine the universe consists of you playing Monopoly against a computer. Your goal is to win the game.

There are two ways you can do so:



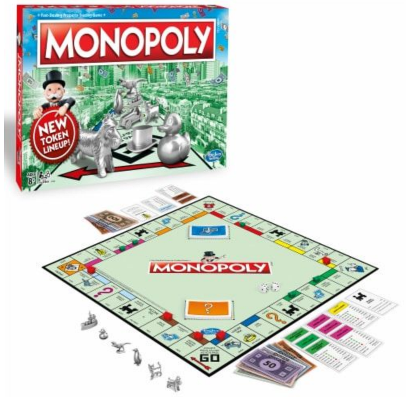
https://www.kroger.com/product/images/large/front/001951661463

Figuring out how to act

Imagine the universe consists of you playing Monopoly against a computer. Your goal is to win the game.

There are two ways you can do so:

1. **Trial and error learning.** Play against it over and over, figure out the game rules and the computer's strategy.



https://www.kroger.com/product/images/large/front/001951661463

Figuring out how to act

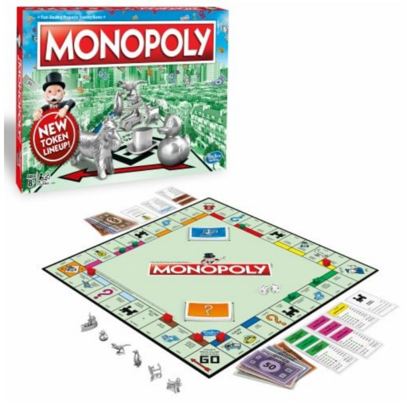
Imagine the universe consists of you playing Monopoly against a computer. Your goal is to win the game.

There are two ways you can do so:

1. **Trial and error learning.** Play against it over and over, figure out the game rules and the computer's strategy.
2. **Planning.** You could be given access to the game's rulebook as well as the code implementing the AI playing against you. You would then sit and there and **think** about how to win. You could **reason** about the rules and the AI, and **plan** how to win.

There's no interaction!

The game's rulebook and the code implementing the AI would allow you to compute p(s', r | s, a).



https://www.kroger.com/product/images/large/front/001951661463

Key Idea Behind Dynamic Programming

“To use value functions to organize and structure the search for good policies.”

We use the same equations as before, but we replace an = by a ←, that’s it (we turn Bellman equations into assignments).

There's lots to decide

- What should we compute? v_π, q_π, v_*, q_*, π^*?
- How should we select states to imagine about? And in what order?
- How much computation do we need to figure out the optimal policy, π^*, using the function p: $\mathscr{S} \times \mathcal{R} \times \mathscr{S} \times \mathscr{A} \rightarrow [0, 1]$?
- How many times do we need to iterate over this imagining / planning process?

Obtaining value functions and π^* from π and p (with no interaction) is called ***Dynamic Programming****.*

Policy Evaluation (Prediction)

Given a policy and an MDP, what's the corresponding value function?

$$v_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s]$$
$$= \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)\Big[r + \gamma v_\pi(s')\Big]$$

↓

$$v_{k+1}(s) \doteq \mathbb{E}_\pi[R_{t+1} + \gamma v_k(S_{t+1}) \mid S_t = s]$$
$$= \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)\Big[r + \gamma v_k(s')\Big]$$

expected update

Policy Evaluation (Prediction)

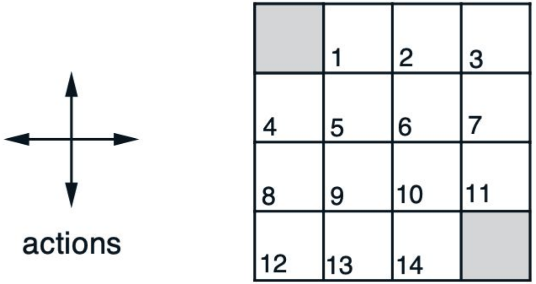
Iterative Policy Evaluation, for estimating $V \approx v_\pi$

Input π, the policy to be evaluated
Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$ arbitrarily, for $s \in \mathcal{S}$, and $V(terminal)$ to 0

Loop:
  $\Delta \leftarrow 0$
  Loop for each $s \in \mathcal{S}$:
    $v \leftarrow V(s)$
    $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

"in-place" update

Policy Evaluation – Example



$R_t = -1$
on all transitions

v_k for the random policy

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0

→

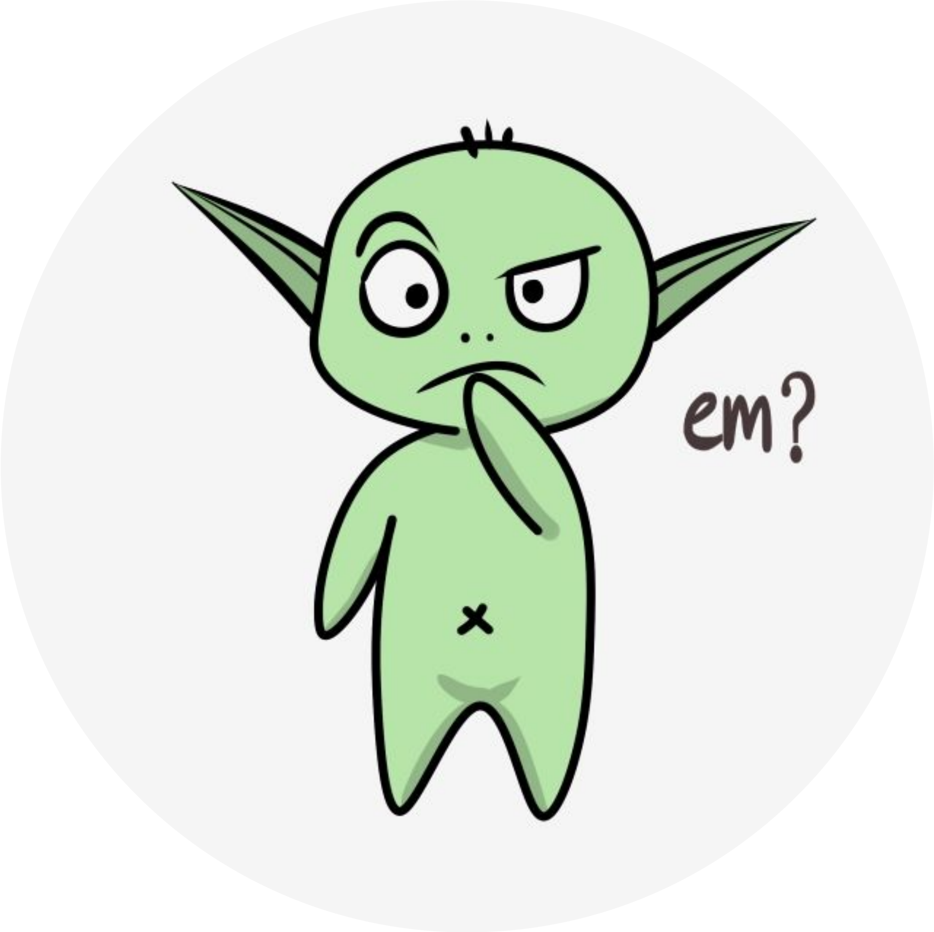
0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0

→

0.0	-1.7	-2.0	-2.0
-1.7	-2.0	-2.0	-2.0
-2.0	-2.0	-2.0	-1.7
-2.0	-2.0	-1.7	0.0

→ … →

0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0



https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht