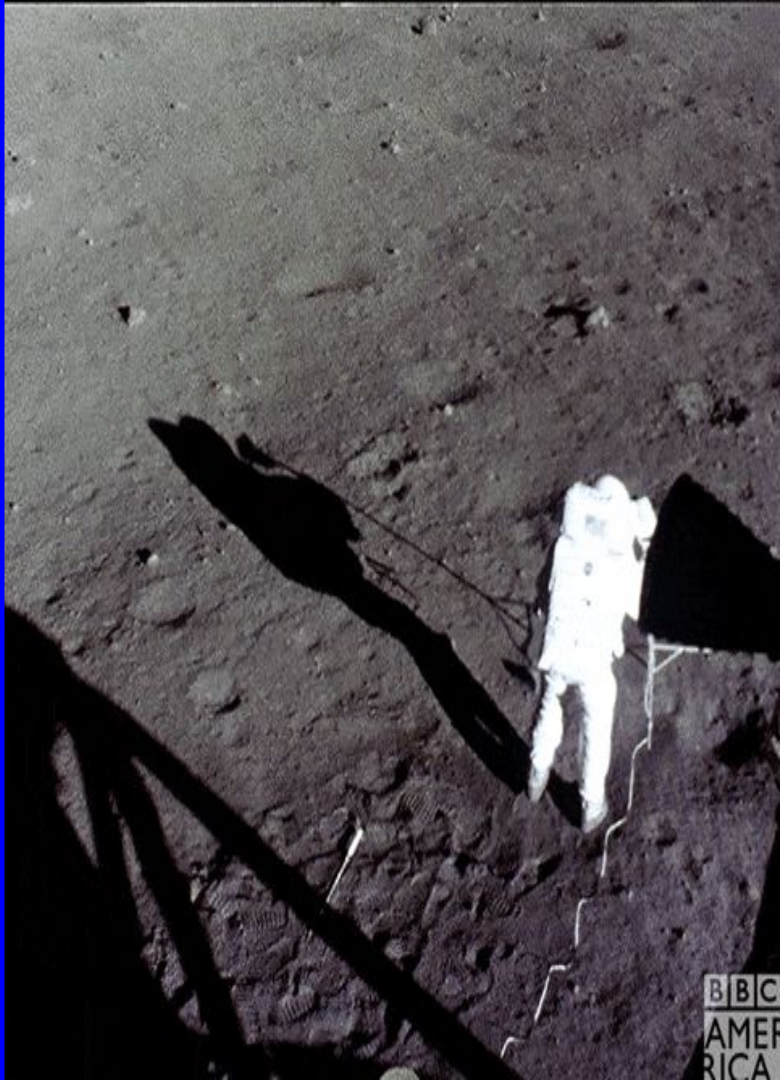
Have-a-go series:

Reinforcement Learning 101



What is your favourite (video) game?

Agenda

- Introduction to RL
- The “So What” and potential for Nesta
- Foundational concepts and libraries
- Introducing the environment
- Hands-on single agent example
- Other examples and use cases
- Further reading

nesta

Introduction

What is reinforcement learning?

- Type of ML where one or more agents learn how to behave (what actions to take) in an environment by performing actions and seeing the results

- Agents learn by interacting with their environment through trial and error, maximising their rewards

- Reward hypothesis: all goals/strategies can be summarised as the maximisation of the expected return (cumulative reward)

- Markov property: agent is focused on present information, not historical states and actions

- Reinforcement learning vs deep reinforcement learning: an agent comes up with the best action to 'maximise rewards' and stores that information (the knowledge space) vs a deep neural network that predicts rewards given some input variables

nesta

The “So What”

- Helps understand and address problems which involve acquiring information by interacting with the environment:
    - Create next best action/alternative recommendation systems

- Works well with complex environments and systems for which it is possible to collect a lot of data (e.g. games, video games, self driving cars etc.), but perhaps difficult to obtain labelled data

- Multi-agent reinforcement learning (MARL) can be used to simulate how individuals interact with society (competitive vs cooperative), so we can simulate coordination/collaboration issues:
    - Share common resources/create better institutions
    - Avoid existential disasters (climate change, pandemics, nuclear wars)
    - Policy design

- For Nesta: cooperative MARL problems may be worth looking into, as an alternative to ABMs

nesta

Foundational concepts

Building blocks

An example of a state from a chess game environment

nesta

Some basic ideas

- **Environment** - first frame of our game, S_0, “state 0”
- **Observations/states spaces** - information that agent gets from the environment, in a video game this may be a “frame” (screenshot), in a trading context, e.g. value of a stock. The difference is that state spaces have **complete information**
- **Action space** - set of all possible actions in an environment, can be **discrete** or **continuous**
- The **cumulative reward** (the expected return) is the only feedback mechanism for the agent to regulate action, so it is very important
- **Episodic and continuous tasks** - the former has a starting point and ending, so an episode is a list of state, action, reward and next state. Continuous tasks need to be interrupted (e.g. trading)

Building blocks

The Reinforcement Learning Process

Today we'll be looking at environments with episodic tasks, and this is what an episode is made up of:

Source: [R. Sutton, A. Barto, Reinforcement Learning: An Introduction]()

nesta

The Reinforcement Learning Process

And the RL loop is generated…

Source: [R. Sutton, A. Barto, Reinforcement Learning: An Introduction]

Building blocks

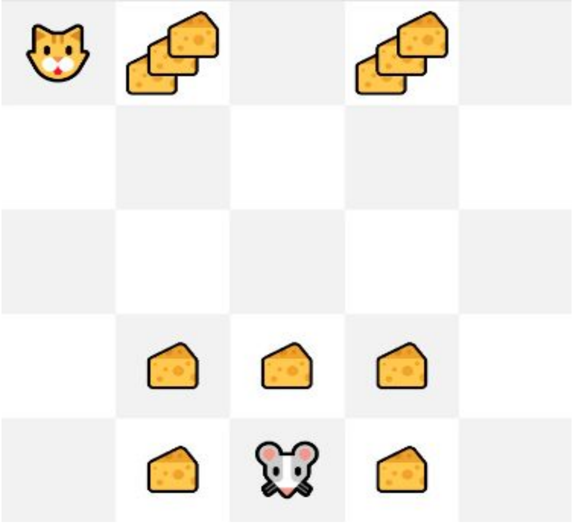
Some basic ideas … continued

$$R(\boxed{\tau}) = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} + \ldots$$

Return: cumulative reward

Gamma: discount rate

Trajectory (read Tau)
Sequence of states and actions



Source: Hugging Face Deep RL Intro

Building blocks

Some basic ideas … continued

- **The exploration-exploitation trade-off -** often the "strategy" that will give you the best results is a combination of the two
  - Trying random actions to gather more information about the environment
  - Exploiting known information to maximise reward

Source: [Tear Along The Dotted Line, Zerocalcare]

nesta

Building blocks

Our goal is to find the ***optimal policy*** *- the one that* ***maximises expected return*** *when the agent follows it. We need to do this by training our agent.*

nesta

Some basic ideas … continued

- **Policy** - what we referred to as "strategy" earlier, or you can think of it as the agent's ``brain`' or thinking process, it is the function that tells us what action to take given a state

- Two avenues:
  - **Policy-based methods**: directly teaching the agent which action to take given a state

$$a = \pi(s) \quad \text{or} \quad \pi(a|s) = \boxed{P[A|s]}$$

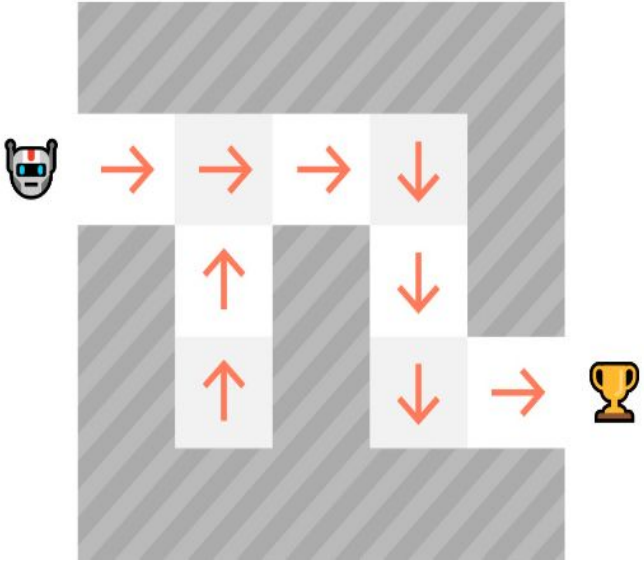
Probability Distribution over the set of actions given the state

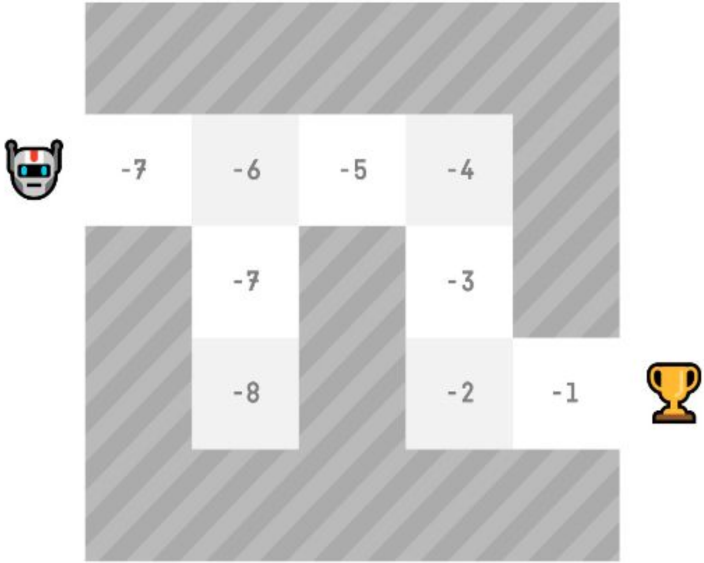
  - **Value-based methods**: training a value function mapping expected value to a given state

$$\underbrace{v_\pi(s)}_{\text{Value function}} = \mathbb{E}_\pi\big[\underbrace{R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots}_{\text{Expected discounted return}} \mid \underbrace{S_t = s}_{\text{Starting at state s}}\big]$$

Some basic ideas … continued

Which is which?





[Source: Hugging Face Deep RL Intro]

Libraries and tools - where to start

- Stable Baselines3 is the go to in Python, it also has integration with other tools, including a very useful Hugging Face integration. This is particularly useful as you can push trained agents on the hub and download trained agents as well.

- RLib - some great examples using RLib can be found in this repo belonging to this great course from Anyscale: Hands-on Reinforcement Learning with RLib

- Baselines3 Zoo is a great library for training and evaluating RL agents

For environment generation and rendering:

- Gym provides a set of environment options including the one we will be using today

Building blocks

You can find more info on the Stable Baselines 3 [documentation page on PPO]

Source: Hugging face PPO

nesta

Some notions about PPO

- Proximal Policy Optimisation (PPO) Is a combination of a value-based method and a policy-based method (find action to take based on value function + learning a policy based on probability distribution over actions)

- Improve performance by making small adjustments to the policy when training: this is more likely to converge to an optimal solution

- A 'clipped' ratio of current policy to past policy is used to limit drastic policy changes

- Other relatively simple state of the art algorithms you can choose are A2C and DQN

Hands-on example

nesta

Access the Colab notebook here. N.B.: Change runtime to GPU

nesta

Our task: landing safely on the moon

- The environment: Lunar Lander v2
- Discrete action space, and episode terminates when lander crashes or is not awake
- Observation space is made of 8 states:
  - Coordinates of the lander in x and y
  - Linear velocities in x and y
  - Angle
  - Angular velocity
  - 2 indicator variables to indicate whether left or right leg have touched the ground
- Rewards:
  - Coming down from top of the screen until rest: 100-140 points
  - Crash: -100
  - Rest: +100
  - Each leg ground contact is +10
  - Firing different engines -0.03 to -0.3
  - Solved: 200 points

Let’s play!

nesta

Use cases

Examples

- Reinforcement learning based multi-agent system for network traffic signal control. Researchers designed a traffic light control system using RL to help some the congestion and traffic injuries problem in a simulated environment. Results were superior to traditional methods.

- Energy-saving automation: AI agents by DeepMind cooling down Google Data Centres. The AI agents identify actions that will lead to lowest possible power consumption while complying with safety criteria

- Text summarisation, generation and translation. RL has been used to summarise long texts, to generate chatbot conversations and simultaneous machine translation, where agents have learned to wait for more input when the prediction has high uncertainty

- Optimizing Chemical Reactions with Deep Reinforcement Learning. A deep RL model was used to optimise reactions, saving time and improving performance of an otherwise trial-and-error process

- Healthcare treatments: using RL to predict viable treatment options. This falls under the category of dynamic treatment regimes (DTRs) where the AI predicts possible treatments at each state. Used for chronic diseases especially.

nesta

Examples

- Mastering 'super-human' performance in games using deep RL. Examples of this are AlphaGo and AlphaGo Zero.

- Have a look at examples that use Stable Baselines 3 on the Hugging Face Hub

Further reading

nesta

Further reading

- Deep RL Class by Hugging Face. A wonderfully accessible and fun course with both foundational concepts explained and full worked examples

- Anyscale course RLib examples. Example notebooks from the Anyscale RL course

- Basic coded examples from the Stable Baselines 3 documentation to get you started

- In the spirit of the past low-code/no-code session, here is a KNIME codeless workflow where an agent plays Tic-Tac-Toe

- A comprehensive youtube training video on doing codeless RL using KNIME

nesta