Reinforcement Learning & Autonomous systems

MBD Sept 24

Lecture 3
LAB 1
Gymnasium and Russell's World

jmanero@faculty.ie.edu

MBD-EN2024ELECTIVOS-MBDMCSBT_37E89_467614

The Gymnasium playground

Gymnasium environment

In this course we will use the gymnasium environment, which is a tool to test reinforcement learning environments

- It provides diverse set of environments that simulate various tasks ranging from simple control problems to complex robotics simulations, allowing researchers and practitioners to test the performance and generalization of their RL agents.
- The API to Gymnasium is a Standardized API which helps to simplify the development and allows to reproducible results and comparison between different approaches
- The community and ecosystem is developing new environments that expand the range of environments and contribute to the Reinforcement Learning community

OpenAI and Gym

GYM was a project created by OpenAI. In 2017 the package was not developed any more. The Farama Foundation took it over and now it maintains and evolves it with the name of Gymnasium

Even though you'll find many code with gym, don't use it as it is not maintained anymore, use Gymnasium instead

https://farama.org/

https://gymnasium.farama.org/

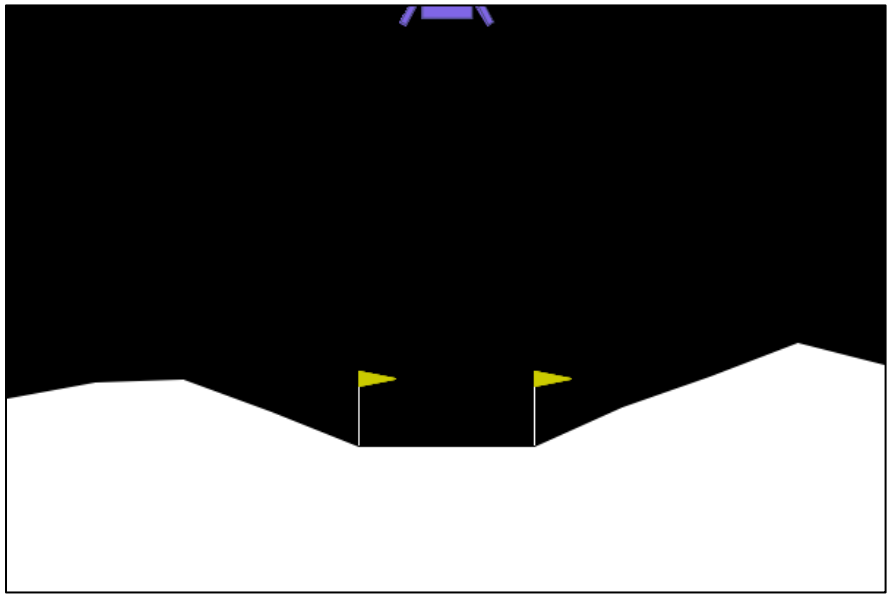
Gymnasium & Gym

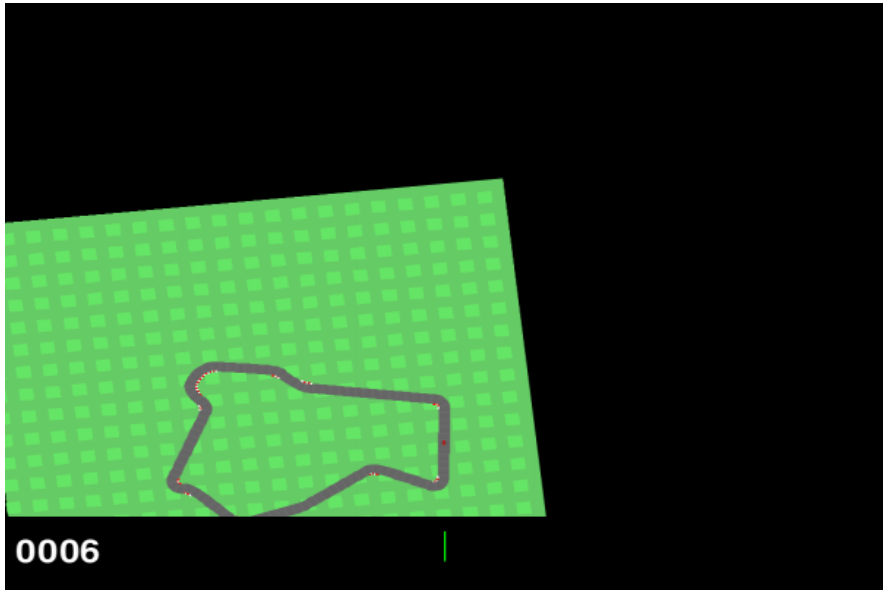
- In internet (github, medium, …) there are many projects made with GYM, avoid them or upgrade them to gymnasium
- Be aware that GYM and gymnasium are different, gymnasium is a more advanced version of gym with several differences
  - New environments
  - New approach with ATARI environments
  - Gymnasium is actively maintained. Gym is a legacy package
  - Much better integration with pygame
  - There are differences invoking the main methods

GYM	Gymnasium
`state = env.reset()`	`state, _ = env.reset()`
`next_state, reward, terminated, truncated = env.step(action)`	`next_state, reward, terminated, truncated, _ = env.step(action)`
`render modes “human”, “rgb_array”, “rgb_array_list”`	`Render modes = “human”, “rgb_array”, “ansi”`

DO NOT USE GYM

The main environments





Classic Control

- Acrorobot
- Cart Pole
- Mountain Car
- Pendulum

Box2D

- Bipedal walker
- Moon Lander
- Car Racing

Toy Text

- Blackjack
- Taxi
- Cliff Walking
- Frozen Lake

MuJoCo

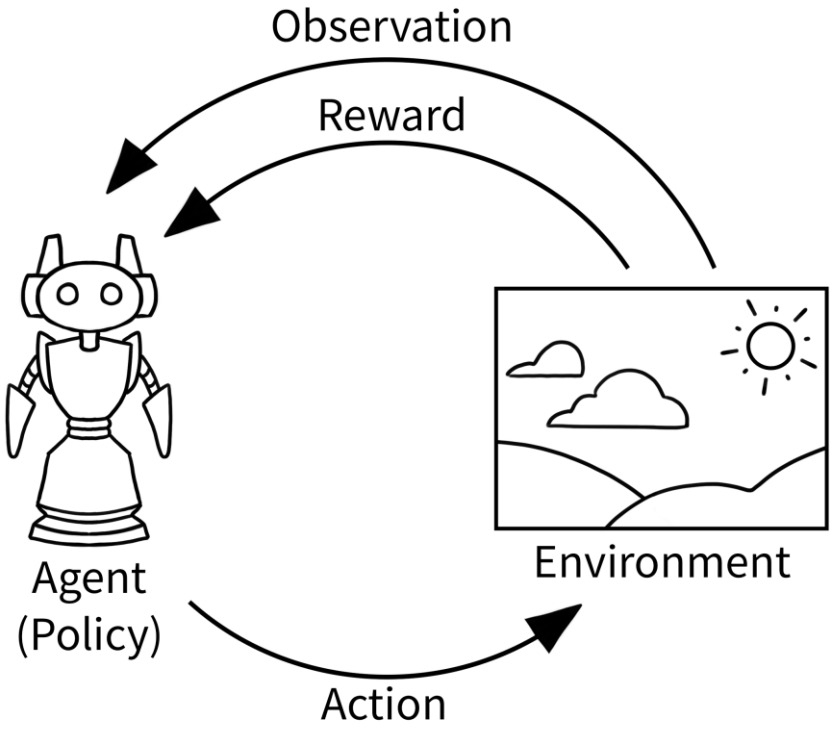
- Ant
- Humanoid

….

Atari

- Breakout
- Boxing
- DonkeyKong
- Pong

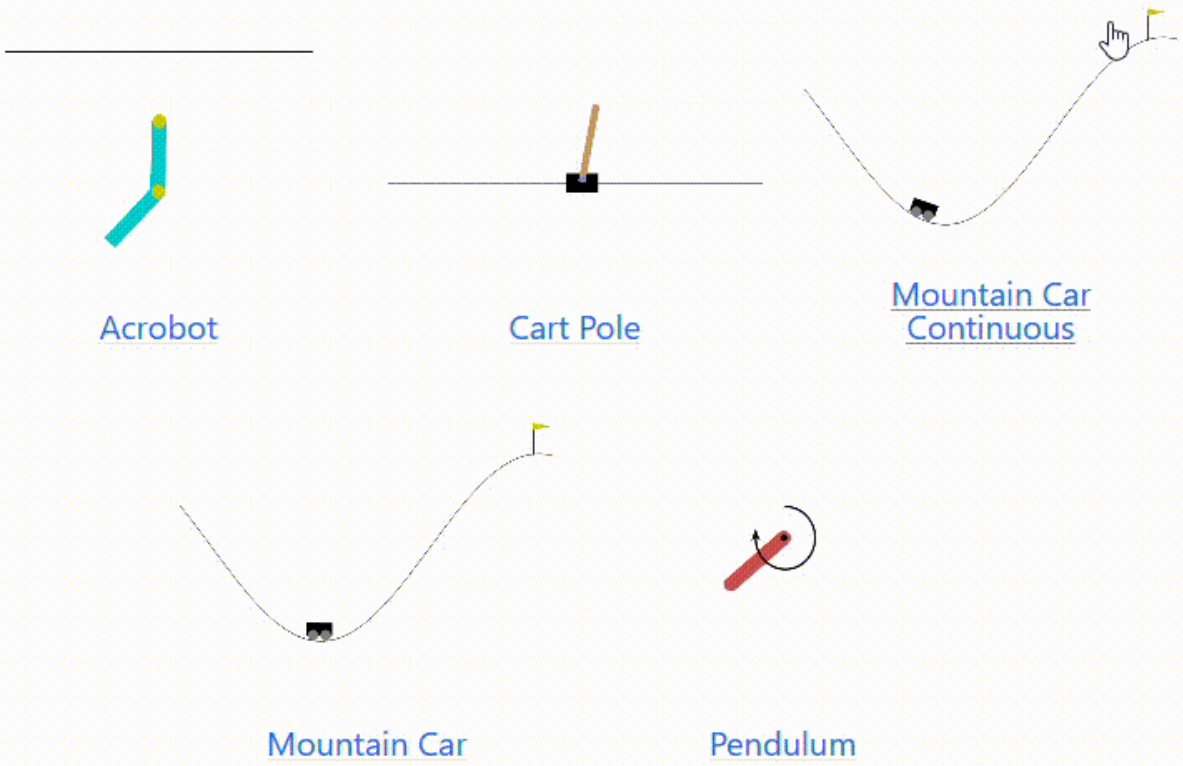
Main features of Gymnasium



All environments have the same structure defined by 5 elements
These elements vary environment to environment

- Environment: It is the world or universe where the agent exists and must perform a task
- Agent: The agent is the program that must perform actions in the environment
- Observation: An observation is the perception of the environment by the agent
- Reward: Is the value that the reward function offers to the agent following an action
- Policy: Is the ….

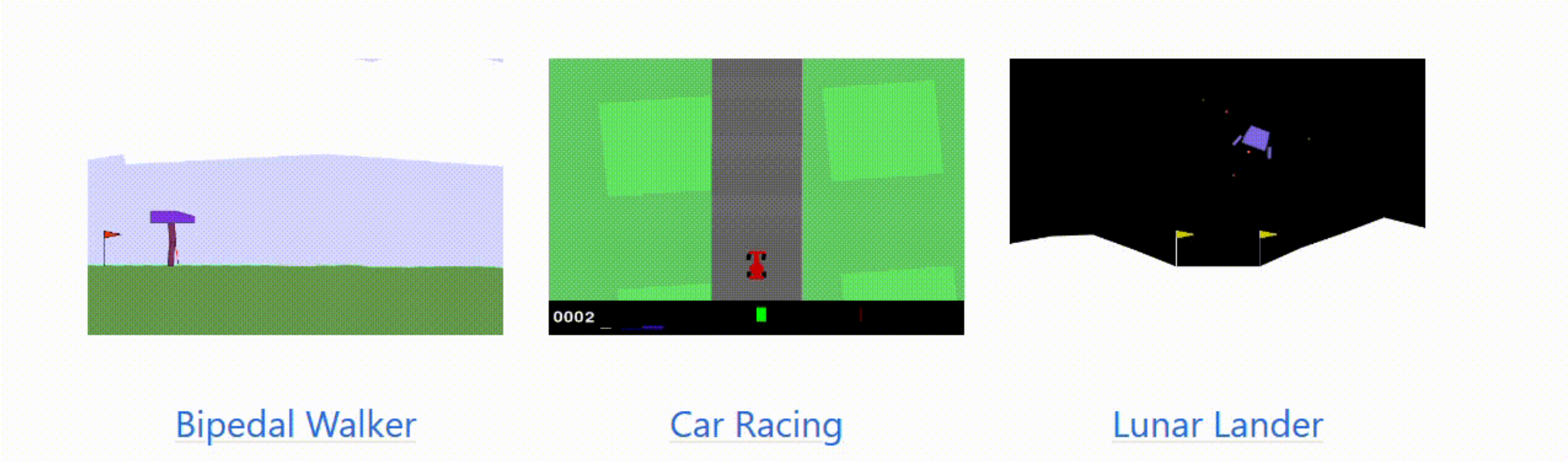
Gymnasium
Classic Control



All of these environments are stochastic in terms of their initial state, within a given range. In addition, Acrobot has noise applied to the taken action. Also, regarding both mountain car environments, the cars are underpowered to climb the mountain, so it takes some effort to reach the top.

Among Gymnasium environments, this set of environments can be considered easier ones to solve by a policy.

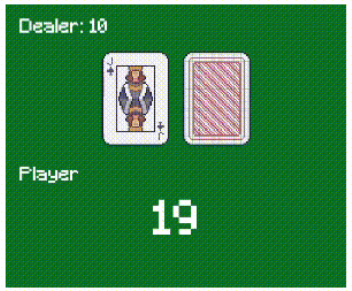
Gymnasium
Box2D



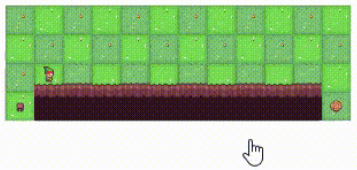
These environments consist of toy games focused on physics-based control, using Box2D physics and PyGame rendering. Originally contributed by Oleg Klimov in the early days of OpenAI Gym, they have since become popular benchmark tasks

Box2D is a 2D rigid body simulation library for games. Programmers can use it in their games to make objects move in realistic ways and make the game world more interactive. From the game engine's point of view, a physics engine is just a system for procedural animation.

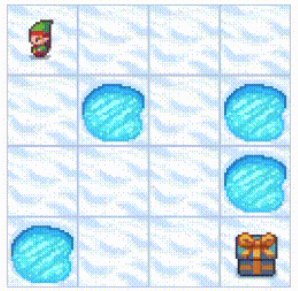
Gymnasium
ToyText



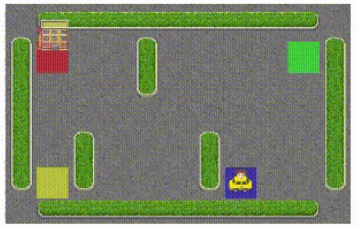
Blackjack



Cliff Walking



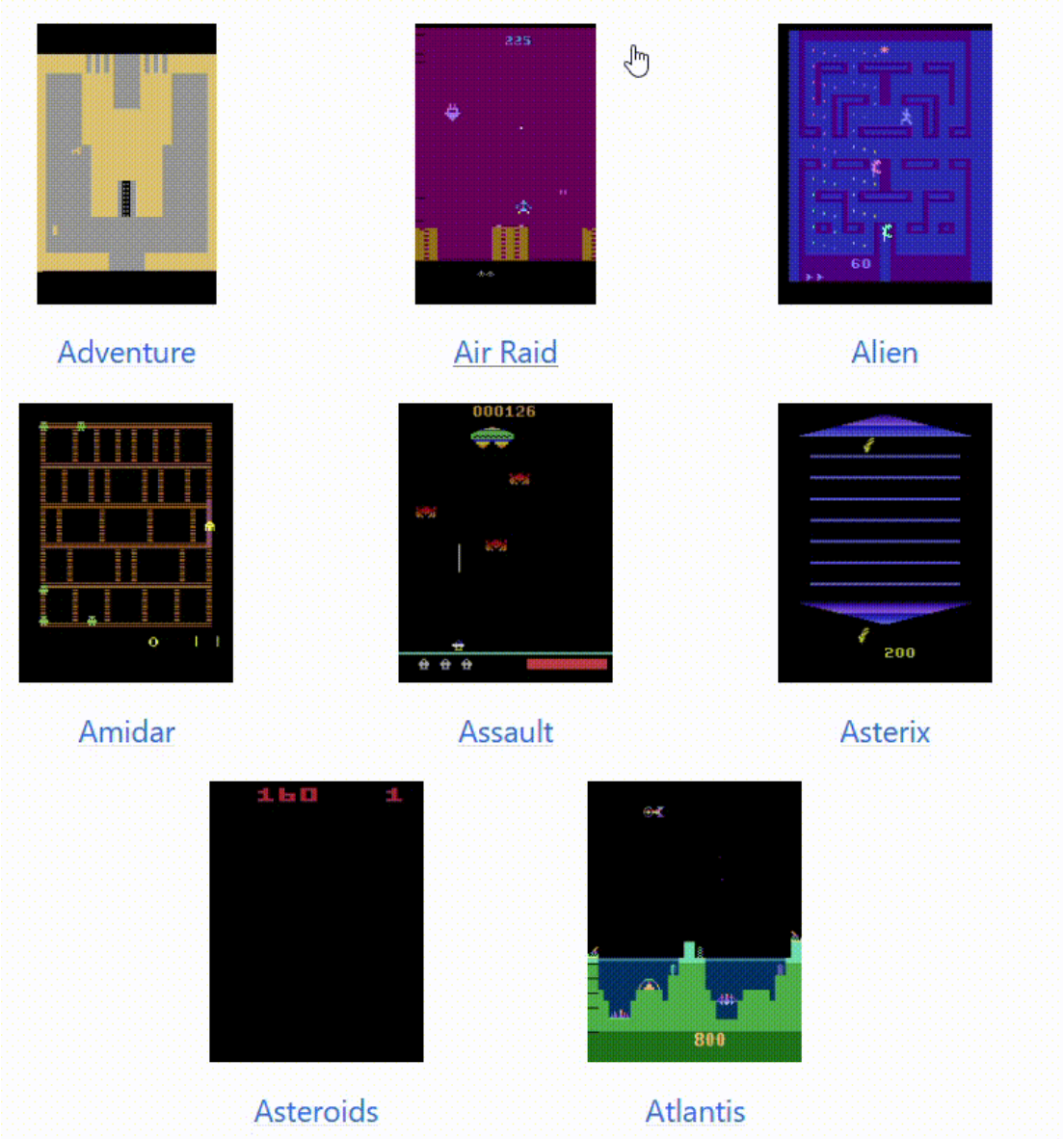
Frozen Lake



Taxi

Simple, with small discrete state and action spaces, and hence easy to learn.

ATARI



- The David Silver article on Atari was an important milestone in the RL evolution
- Replicating some of the conclusions of this paper is a task that many students around the world have been trying
- Using the Atari environments this task has become much easier allowing to focus on the RL part and not in the game display or construction part.

(Silver, Mnih 2013) Playing with Atari with Deep Reinforcement Learning. Arxiv

Understanding the API

00_Cartpole_render.ipynb

Import gymnasium package

```
import gymnasium as gym
import matplotlib.pyplot as plt
```

How to see the environment

Create environment in our program

```
env = gym.make('CartPole-v1', render_mode="rgb_array")
env.reset()
for _ in range(1):
   frame = env.render()
   action = env.action_space.sample()
   next_state, reward, terminated, truncated, _  = env.step(action)

env.close()
plt.imshow(frame)plt.show()
```

Random action from space

Apply action to environment

https://github.com/castorgit/RL_course/blob/main/00_Cartpole_render.ipynb

Recommendation!

GYM is the old version of Gymnasium.

Do not use Gym, you'll find in internet many examples with GYM, you can transform them into gymnasium by fixing some sentences.

To avoid confusion don't install gym in your environment, in this way you'll be gym-free.

Transforming Gym program to Gymnasium

1. Gymnasium has new environments. Make sure you are using a gymnasium environment
2. Gymnasium env methods have additional outputs

Gym	Gymnasium

`state = env.reset()`	`state, info = env.reset()`
	
	`state, _ = env.reset()`

`state , reward, done, info = env.step(action)`	`state , reward, terminated, truncated, info = env.step(action)`
	
	`state , reward, terminated, truncated, _ = env.step(action)`

Exercise 1

https://github.com/castorgit/RL_course/blob/main/00_Cartpole_render.ipynb

1. Install gymnasium in your environment

```
$ pip install gymnasium
```

2. Create a notebook like the cartpole one but in this case use the Taxi environment

```
env = gym.make("Taxi-v3", render_mode='human')
```

3. Use it with render mode human, ansi and RGB

4. Answer this questions

    a. What is the environment space in the Taxi environment?
    b. How many actions can be performed in the Taxi environment?

The Assignments

4 assignments + 1 Group Practice

- 4 Labs
- 1 assignment each lab
- We start the assignment in class you finish at home
- There is a complete description of each assignment available
- There is some support code in the course repo.

https://github.com/castorgit/RL_course

- Assignments preferable in jupyter notebook + python
- Submit it in Blackboard
- There is a due date!!!

4 assignments + 1 Group Practice

Method	Model-Based / Model-Free	On-Policy / Off-Policy	Practice
Using Gymnasium - Russell's Grid	N/A	N/A	1
Dynamic Programming	Model-Based	N/A (requires a model)	2
Monte Carlo	Model-Free	On-Policy	2
Temporal Difference (TD)	Model-Free	On-Policy	3
SARSA and Q-Learning	Model-Free	On-Policy	3
DQN-Learning	Model-Free	Off-Policy	4

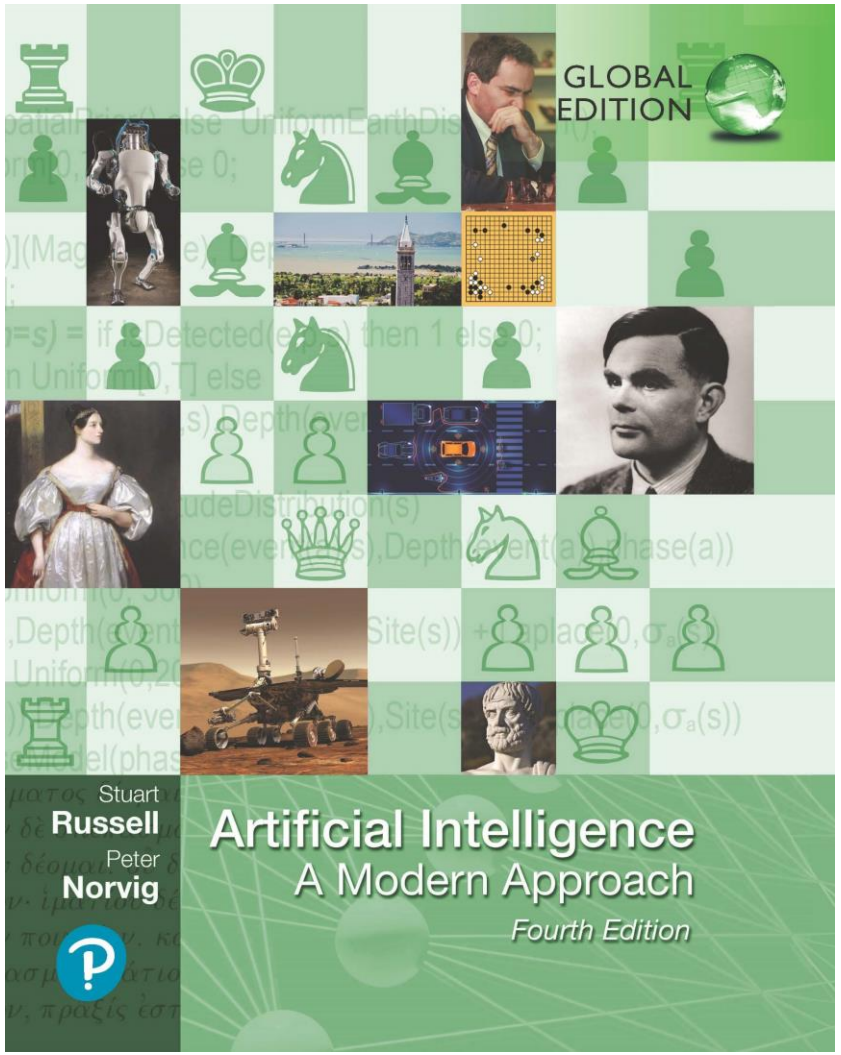
Table 1: Classification of RL Methods and Assignments/Practices

Assignment 1 – Russell's World

Although greed is considered one of the seven deadly sins, it turns out that greedy algorithms often perform quite well

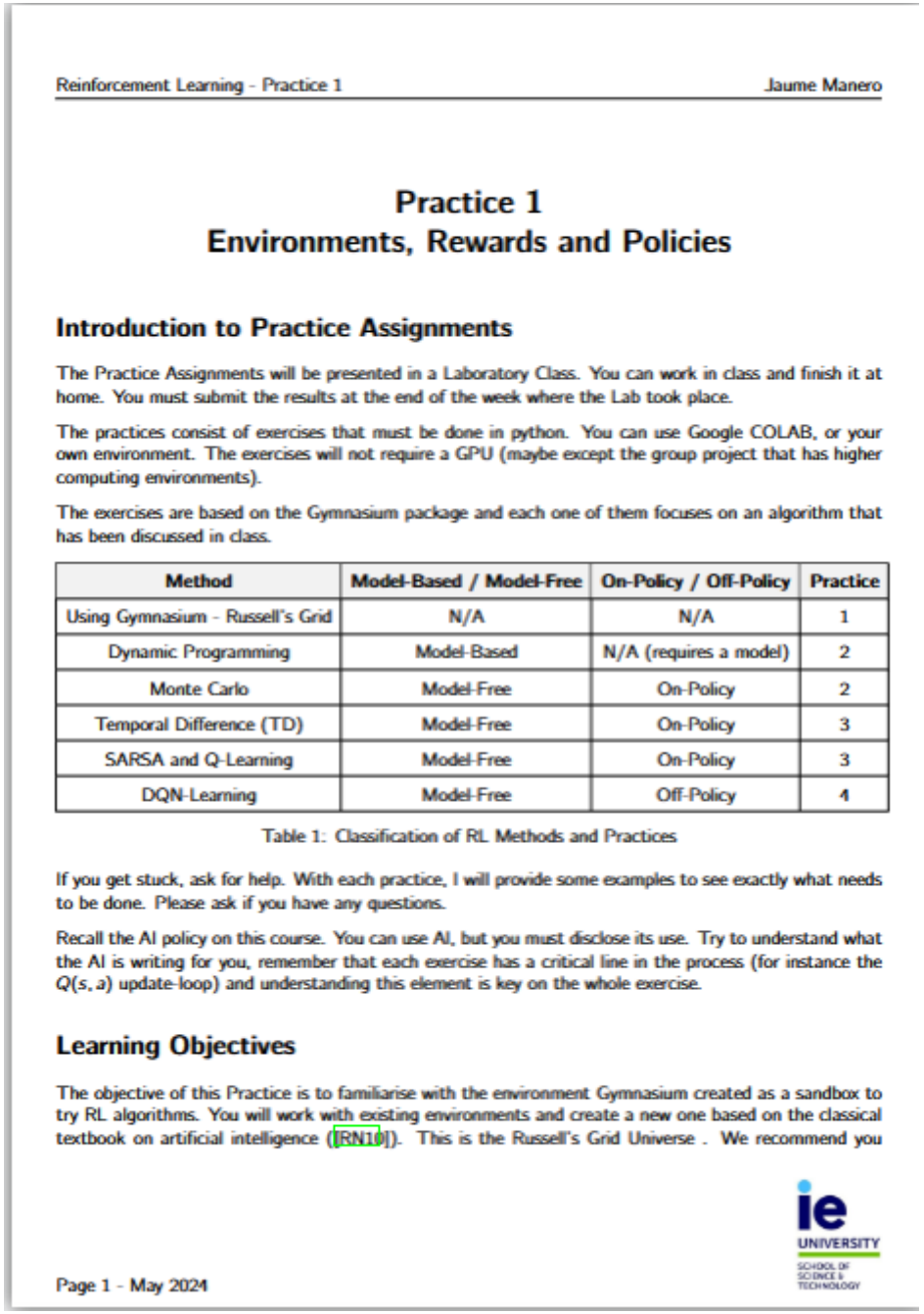
Stuart Russell, in Artificial Intelligence: A Modern Approach

Bibliography
A book to own



Assignment 1

- We will have 4 Assignments and 1 Group Practice in this course
- Each Practice will have a document with a full description of the practice, how to do it, and how to submit the result
- You just need to follow the instructions
- I will share some notebooks to help to solve the problems
- Ask if you need information, if you don't understand or if you just can't go ahead in one specific point



Reinforcement Learning - Practice 1 Jaume Manero

Practice 1
Environments, Rewards and Policies

Introduction to Practice Assignments

The Practice Assignments will be presented in a Laboratory Class. You can work in class and finish it at home. You must submit the results at the end of the week where the Lab took place.

The practices consist of exercises that must be done in python. You can use Google COLAB, or your own environment. The exercises will not require a GPU (maybe except the group project that has higher computing environments).

The exercises are based on the Gymnasium package and each one of them focuses on an algorithm that has been discussed in class.

Method	Model-Based / Model-Free	On-Policy / Off-Policy	Practice
Using Gymnasium - Russell's Grid	N/A	N/A	1
Dynamic Programming	Model-Based	N/A (requires a model)	2
Monte Carlo	Model-Free	On-Policy	2
Temporal Difference (TD)	Model-Free	On-Policy	3
SARSA and Q-Learning	Model-Free	On-Policy	3
DQN-Learning	Model-Free	Off-Policy	4

Table 1: Classification of RL Methods and Practices

If you get stuck, ask for help. With each practice, I will provide some examples to see exactly what needs to be done. Please ask if you have any questions.

Recall the AI policy on this course. You can use AI, but you must disclose its use. Try to understand what the AI is writing for you, remember that each exercise has a critical line in the process (for instance the $Q(s, a)$ update-loop) and understanding this element is key on the whole exercise.

Learning Objectives

The objective of this Practice is to familiarise with the environment Gymnasium created as a sandbox to try RL algorithms. You will work with existing environments and create a new one based on the classical textbook on artificial intelligence ([RN10]). This is the Russell's Grid Universe . We recommend you

ie UNIVERSITY SCHOOL OF SCIENCE & TECHNOLOGY

Page 1 - May 2024

First practice 2.2

- Create a .py from the 00_RussellGrid_new_environment.ipynb.
- Make sure this new .py file is in the same folder as your .ipynb
- Create a .ipynb in that folder
- Import the .py into your notebook
- Create a sample episode and render it
- You are done?
- How did it go?

Second practice 2.3.1 One Random Episode

- You basically did it already, but now, instead of sampling, use a randomizer to generate a random action
- Remember you need to know how many actions do you have
- How do you find the number of actions in an environment? – look at the code of Russell'sGrid

Third practice 2.3.2 Rewards

- Create a python list **lr** that stores all the rewards in your episode
- Implement two functions **longtermreward1** and **longtermreward2**
- Which totals do you obtain?
- Are they aligned with the expected result?

Third practice 2.3.3 Policy

- Create a policy array
- The array has the structure of the environment (careful with 2D coordinates to 1D coordinates)
- You must create the best policy by updating the values on each move

```
Optimal Policy:
→ → → G
↑ X ↑ N
↑ ← ↑ ←
```

End

Wrap-up your code and submit your notebook

Wrap-up

LAB 1

- Gymnasium is a python package to practice all the different RL learning strategies
- It is widely used in the field and will be used in this introductory course
- The methods are shared, making easy to move the algorithms between different environments
- It is important for this course to have a stable python environment in your laptops. If you don't manage to have this stable setup, use Google Colab instead.
- We will try to use Jupyter Notebooks for most of applications, as it is easier to share and work (but more difficult to troubleshoot)

END
Session 3