Reinforcement Learning Lab

Lesson 1: MDP and Gym Environments

Davide Corsi and Alberto Castellini

University of Verona
email: davide.corsi@univr.it

Academic Year 2022-23

UNIVERSITÀ
di VERONA
Dipartimento
di INFORMATICA

Initial Setup Process - 1

The first step for the setup of the laboratory environment is to install miniconda, a light version of the Conda package manager.

- Download the *Miniconda* package manager:
  `https://docs.conda.io/en/latest/miniconda.html`
- Install *Miniconda*
  - On *Linux* run *Miniconda3-latest-Linux-xx.sh*, remember to provide the executions permissions to the file.
  - On *Windows*, double-click the installer; in the installation phase, ensure to install *Anaconda Prompt* and use it for the next steps.

Virtual Environment

For python virtual environments' users (*venv*), the Miniconda installation can be avoided.

Initial Setup Process - 2

The second step is downloading the official laboratory repository and creating the working environment. The following commands must be run in the *Linux Shell* or in the *Anaconda Prompt* on Windows.

- Clone the official Lab repository:
  git clone https://github.com/d-corsi/RL-Lab
- Create the environment and install the required packages:
  conda env create -f RL-Lab/tools/rl-lab-environment.yml
- Before running the scripts, remember to activate the environment:
  conda activate rl-lab

Git Installation

The installation of Git may be required. For Linux users, just run *sudo apt-get install git*. For Windows users, refer to the official page `https://git-scm.com/downloads`.

First Tutorial

The README file on the repository (link) contains a simple tutorial on the basic libraries and tools we will use in the next lessons, in particular:

Numpy

NumPy is the fundamental package for scientific computing with Python. Think of it as a MATLAB-like environment for Python.

OpenAI Gym

Gym is a toolkit for developing and comparing reinforcement learning algorithms, a standardized set of tools and instructions to work with environments.

Keras

Keras is a high-level neural networks APIs that implements simple functions to create, train and modify neural networks.

Assignments Structure

In general, the requirement for a laboratory lesson is to complete the partial python code provided, for example:

`RL—Lab/lessons/lesson_n_code.py`

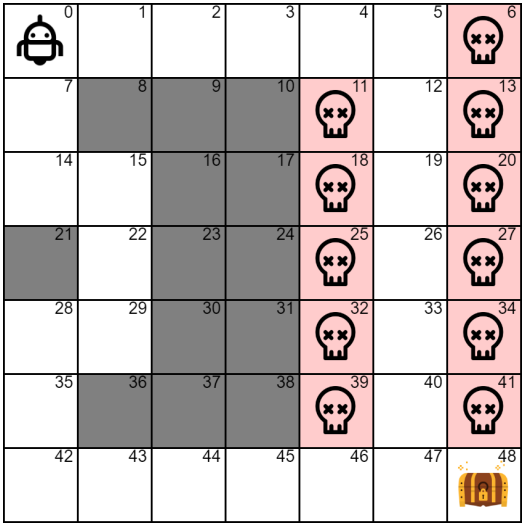
The **main** function is always provided in a Python script. Once the code is completed, students have to run the script to obtain all the results in the output on the console. The expected results can be found in the *.txt* file corresponding to the lesson, for example:

`RL—Lab/results/lesson_n_results.txt`

Disclaimer

Given the stochasticity of the algorithms that we will run in the course, the results can be slightly different from the ones reported in the *.txt* file. The relevant point is the general trend.

Environment Description



The *DangerousGridWorld* environment. The goal for the robot is to reach the target (treasure) while avoiding the terminal states (death). Dark cells represent walls that can not be crossed.

All cells return a reward:

- **+5** for the target position (state 48)
- **-1** for the terminal cells (death)
- **-0.1** for each empty step

Today Assignment

In today's lesson, we experience gym environments. We implement a function to make random actions inside the given environment *DangerousGridWorld* generating a trajectory; after that, we implement an environment from scratch: *Recycling Robot*. In particular, the file to complete is:

```
RL-Lab/lessons/lesson_1_code.py
```

Inside the file, a *python function* and a *python class* are only partially implemented. The objective of this lesson is to complete them.

- **def random_dangerous_grid_world()**
- **class RecyclingRobot()**

Expected results can be found in:

```
RL-Lab/results/lesson_1_results.txt
```

Recycling Robot (a)

s	a	s'	$p(s' \mid s, a)$	$r(s, a, s')$
high	search	high	α	$r_{\texttt{search}}$
high	search	low	$1-\alpha$	$r_{\texttt{search}}$
low	search	high	$1-\beta$	-3
low	search	low	β	$r_{\texttt{search}}$
high	wait	high	1	$r_{\texttt{wait}}$
high	wait	low	0	-
low	wait	high	0	-
low	wait	low	1	$r_{\texttt{wait}}$
low	recharge	high	1	0
low	recharge	low	0	-

Figure: Description of the environment *Recycling Robot*, reporting the transition table from one state to the next and the reward for each action. A detailed description of the environment is freely available (here).

Recycling Robot (b)

Hint:

The first tutorial (environments) presents and describes all the functions and tools necessary to complete this first lesson.

Minimal Gym Environment

A minimal gym environment must implement the following functions: *init*, *reset*, *step*, and *render*. All these functions must be implemented in the class Recycling Robot. A complete example can be found (here).

DangerousGridWorld Disclaimer

DangerousGridWorld is not a *standard* gym environment. It implements different functions and simplified methods for learning purposes.