Reinforcement Learning course project

Prof. Gian Antonio Susto, Prof. Ruggero Carli
Niccolò Turcato, Alberto Sinigaglia

December 2024

1 TL;DR

You can submit whenever you want until Sept 2025. You will have to submit code, weights and a report. Most of the code is already provided, you just need to fill some gaps. **The report has to use the latex template provided**.

2 Introduction

This document aims to give a complete description of what you are asked to do for the second part of your grade (project) for the Reinforcement Learning course 2024/2025. Recall that:

- if you **DON'T** do the project, your grade will be:

$$\min(28, \text{your grade on the written part})$$

- if you do the project, your grade will be:

$$\max(\min(28, \text{your grade on the written part}), \text{average}(\text{your grade on the written part, project}))$$

The project will be graded with a scale out of 32, and in case of decimals in the final grade, it will be rounded up... note that ***by submitting the project, the final grade will never be worse than the grade you get without submitting the project***.

The proposed projects are 4 (standard), two in the realm of games, one in the realm of robotics, and one for autonomous driving. In this cases, you'll be required to submit the following materials:

1. Heuristic / Baseline policy
2. Code, and weights of the final RL agent
3. Report

Advanced projects are also welcome, however we have limited capacity, and such project has to be discussed directly with the professors. Please refer to the recording of the last lecture for advanced project proposals.

3 Heuristic / Baseline policy

In order to fully appreciate the RL agent, you're asked to design and code an **heuristic/baseline policy**, and to evaluate it on the environment. Such baseline that you will define **should be an educated guess on how to solve the problem**, thus you are free to use everything in your toolbox to design and code it, given that such tools don't involve any learning.

In other words, you will have to define a set of rules / use some algorithms / exploit some other tricks from other courses, in order to define a policy that is reasonable.

Such baseline will serve as a benchmark for your RL agent.
Don't spend too much time on this part, but nor use a too-naive-approach, this baseline should motivate you to develop a good RL agent, it's not here to make your life easier, nor to make it become a nightmare.

4 Code, and weights of the final agent

After you have developed a informative baseline, you are asked to code the actual RL agent, and to do so, you can use your favourite auto-diff package (between TensorFlow and PyTorch).

For this part, you are asked to develop the following pieces:

- Training: define the agent, simulate on the environment with it, train it, and store the final trained agent on memory
- Evaluation: import the weight of the agent from memory, simulate on the environment for some time with it, and print the resulting performances

As per the algorithm to use, you are absolutely free to use whatever you find reasonable, it being presented during the course or not, you can use any algorithm proposed in literature, given a correct justification.

5 Task goal

Refer to the material uploaded in the related Moodle section for environment code and utilities.
You **can** use StableBaselines if you want (**apart from the Autonomous Driving project**), but you are encouraged to implement the algorithm you want by yourself (this won't make any difference from the evaluation standpoint) Task requirements follow.

5.1 Double Pendulum

The task to be carried out is the swing-up and stabilization of the underactuated 2-link System Acrobot (and Pendubot optionally).

`https://github.com/turcato-niccolo/double_pendulum/tree/main`

The swing-up is to be carried out from the initial position which is the robot pointing straight down. The task can be carried out in either system (or both). Use the parameters provided in the Moodle section (files pendubot_parameters.yml and acrobot_parameters.yml).

Example of acrobot swingup: `https://ijcai-23.dfki-bremen.de/wp-content/uploads/2023/03/sim_video_gif.gif`.

Example of pendubot swingup: `https://ijcai-23.dfki-bremen.de/wp-content/uploads/2023/03/sim_video_pendubot-1.gif`.

5.2 Snake

The environment will already hand you the reward of the transition that your agent will do. For this reason, you can just limit yourself to maximize such reward, however this is not mandatory, as described in the section **Bonus**.
Thus, both your heuristic and your agent just have to maximize the amount of "fruits" over a time-frame, limiting the number of times it hits the walls (you might also try to code an agent that manually mask out the actions that might lead to a wall hit). Highly suggested to do a comparison between the fully observable and the partially observable environment.

5.3 Autonomous Driving

In this project you will have to make an autonomous vehicle safely and efficiently drive through a highway populated with other vehicles.

The environment will already hand you the reward of the transition that your agent will do. For this reason, you can just limit yourself to maximize such reward.
Thus, the objective of both your heuristic and your agent is to complete the scenario by progressing quickly on the road and avoid collisions with other road users.
Code can be found either on Moodle or here: `https://github.com/mcederle99/rl_project_ad`.

5.4 Briscola

The code present on Moodle is part of a bigger project, thus you are asked only to fill some gaps in it. In particular, all the required part are listed in the README file present in the zip file with the code, thus before starting, please consider reading it carefully.

The task is to develop an agent that maximizes the win-rate against a "Scripted Agent" (a manually crafted rule-based agent), that you can also find in the zip file with the rest of the code.

Regarding the baseline, here it's a bit harder, but still you are required to create something smart. In order to define such agent, please check the implementation of the `ScriptedAgent`, create a new class with the same structure, and change the logic of how it selects the action.

6 Report

In addition to the code, you have to submit a report (6 pages long) with the following information about your code:

- the critical issues you think the project you picked might have.
- an extensive and clear explanation of your baseline and why it should be reasonable: from such description, we should be able to fully characterize the baseline you used, thus please be exhaustive

- the RL algorithm you used and why you picked that specific one
- results: for this part, you are asked to plot the learning of the agent during time compared to the baseline (which should be just a straight horizontal line) and how you measure such learning/evaluation of the agent
- discussion on why the RL agent is or is not better than the baseline you have defined, and in case possible improvements.

The report will have to be done with the provided template (in latex, if you don't know it, it's a great chance to learn it :P) **Bear in mind that this will play a major role in the evaluation of your project, so spend some time curating it.** More info about the report directly in the provided template.

Bonus

As a bonus, you can play around with the problem definition, by varying parts of it, for example:

- use multiple algorithms
- play around with the state representation
- play around with the reward function

In case you do this part, please note that you only have to submit the ~~code~~ MODEL PARAMETERS for the optimal one, but feel free to report in the PDF essay all the results you obtained (the code used for the training should still be part of the submission).

Submission and evaluation

For the submission, as stated earlier, you have to submit this exact files:

- the PDF report
- the environment you used (most likely files we provide you)
- the weights of the trained final optimal agent
- a file that runs the baseline / heuristic policy on the environment
- a file that contains the code you used to train your agent
- a file that loads the model weights from memory, and evaluates such agent on the environment

To be clear, what we want, is to be able to download the zip of your files from Moodle, unzip it, go on the terminal and run `python evaluate.py`, and everything needs to be working.

For this reason, please don't use esoteric libraries, stick to Tensorflow, PyTorch, Numpy and Matplotlib, and in case you think you need to use any other major library, please ask before submission. You can use stable-baselines (**apart from the AD project**), anaconda is recommended.

PLEASE FIX THE SEED TO 0 TO MAKE YOUR RESULTS AS REPRODUCIBLE

AS POSSIBLE.

IF YOUR CODE DOESN'T RUN: we will send you an email letting you know the code can't run, reporting the error we got, and asking you to fix it and upload it back, FROM THEN ON, ANY RE-SUBMISSION WILL COST YOU 1 GRADE OF YOUR FINAL PROJECT GRADE

Communication

If you need to contact us, use the following emails:

- Prof. Susto: gianantonio.susto@unipd.it
- Prof. Carli: carlirug@dei.unipd.it
- Niccolò Turcato: niccolo.turcato@phd.unipd.it
- Alberto Sinigaglia: alberto.sinigaglia@phd.unipd.it

As pointed out at the beginning of the course, for email communications, please add at the beginning of the object of the email **"[RL2024-25]"**.

Deadline

For the project, there is no specific deadline. You can submit it anytime from January onwards. The only requirement is to submit it before the end of the academic year, thus **before October 2025** (October not included) and by the of December 2025 if you pass the written exam during the August exam.