Assignment 3

Reinforcement Learning - A.Y. 2024/2025

Assigned: November 22nd, 2024
Deadline: December 7th, 2024

Rules

The assignment is due on December 7th, 2024. Students may discuss assignments, but **each student must code up and write up their solutions independently**. **Students must also indicate on each homework the names of the colleagues they collaborated with** and what online resources they used.

The theory solutions must be submitted in a pdf file named "XXXXXXX.pdf", where XXXXXXX is your matricula. We encourage you to type the equations on an editor rather than uploading a scanned written solution. **In the pdf** you have to hand over **the answers to the theory questions (not just the numerical results, but also the derivations)** and a **small report of the practice exercises**.

The practice exercises must be uploaded in a zip file named "XXXXXXX.zip", where XXXXXXX is your matricula. **The zip file must have the same structure of the assignment.zip** that you find in the attachments, but with the correct solution. You are only allowed to type your code in the files named "student.py". Any modification to the other files will result in penalization. You are not allowed to use any other python library that is not present in python or in the "requirements.txt" file. You can use as many functions you need inside the "student.py" file. The zip file must have the same structure of the assignment.zip

All the questions must be asked in the Classroom platform but it is forbidden to share the solutions on every forum or on Classroom.

Theory

Suppose you have an environment with 2 possible actions and a 2-d state representation ($x(s) \in R^2$). Consider the 1-step Actor-Critic Algorithm with the following policy and action-state value function approximators:

$$\pi_\theta(a = 1|s) = \sigma(\theta^T x(s)) = \frac{1}{1 + e^{-(\theta^T x(s))}}$$

$$Q_w(s, a = 0) = w_0^T x(s)$$

$$Q_w(s, a = 1) = w_1^T x(s)$$

Given

$$w_0 = (0.5, 0)^T, w_1 = (1, 0)^T$$

$$\theta_0 = (0.6, 1)^T$$

$$\alpha_w = \alpha_\theta = \alpha = 0.5$$

$$\gamma = 0.8$$

and the following transition:

$$x(s_0) = (0, 1)^T, a_0 = 1, r_1 = 1, x(s_1) = (1, 0)^T, a_1 = 0$$

Compute new values of w_0, w_1 and θ after the transition.

Practice

Solve the CarRacing-v2 gym environment using one of the following algorithms:

- Double DQN with Prioritized Experience Replay (PER) (https://arxiv.org/pdf/1511.05952.pdf)
- World Models (https://arxiv.org/pdf/1803.10122.pdf)
- Advantage Actor-Critic (A2C) (https://arxiv.org/pdf/2205.09123.pdf)
- TRPO (https://arxiv.org/pdf/1502.05477.pdf)
- PPO (https://arxiv.org/pdf/1707.06347.pdf)

If you choose to implement World Models, you are allowed to use the CMA library https://pypi.org/project/cma/.

In the folder "car racing" you find three files:

- "main.py" that contains the main script to evaluate your solution. Don't modify this file!
- "student.py" is the file you have to modify, by implementing the algorithm you have chosen.
- "requirements.txt" contains the name of the libraries needed for this part of the assignment.

For this exercise, in addition to the python source code, you must also submit a trained model. Make sure the model can be loaded correctly using the load method in student.py. You can freely modify the save and load methods as long as their signature remains the same as the template. Do **not** also include additional datasets if they are not strictly required to execute the policy (e.g. for World Models), especially if they have a very large size. To train the model you submit, you can use any strategy that makes use of the algorithm you chose to implement.

You may add some requirements, but they need to be authorized on Classroom. In order to request them, place a private comment under the assignment post with the **name and version** of the libraries you want to add. **Stable-baselines and any other library that already implements the algorithm are banned**. You need to use **pytorch** as the deep learning framework.

For this exercise, **3 bonus points** (valid for the final exam grade) will be awarded to the 3 students delivering the 3 best performing models.

Note: the algorithms among which you can choose have different implementation complexities. In ascending order (from less to most complex): A2C, PPO, DDQN+PER, TRPO, World Models. Overall, we will assign grades by considering both the performance of the final model, the complexity of the algorithm that you chose and any improvements/additions that you made (implementation details, reward shaping, observation preprocessing, etc.). For example, even if your model is performing poorly, you can still receive a high grade if you correctly implemented a complex algorithm or you implemented well-motivated improvements/additions to it.