ULiège
Prof. Damien Ernst

INFO8003-1
Optimal decision making
for complex problems

Project 3

Reinforcement Learning : Searching High-Quality Policies to Control an Unstable Physical System

INSTRUCTIONS

You need to deliver:

- a report containing your discussion of the different domains, algorithms and results,
- your code commented and clean (the algorithms must be implemented by yourself) with the associated weights and readme file,
- a `requirements.txt` file, in order to install your libraries (with the correct version), allowed libraries: `pytorch`, `numpy`, `sklearn`, `matplotlib`, `gym` (or equivalently gymnasium). Please use a minimum number of libraries. If you need any additional library please ask.

1 DOMAIN (5 POINTS)

We consider the following two environments:

1. Inverted pendulum[1]
2. Double Inverted pendulum [2]

The goal of the agent is to keep the pendulum at equilibrium the longest.

[1] `https://www.gymlibrary.dev/environments/mujoco/inverted_pendulum/`
[2] `https://www.gymlibrary.dev/environments/mujoco/inverted_double_pendulum/`

You do not need to code yourself the environments you can use `https://github.com/Farama-Foundation/Gymnasium` where both environments are available. Gymnasium is a newer implementation of the gym environment. We advise you to install Gymnasium and to use it with "`import gymnasium as gym`" in your scripts.
In your report, give a description of the domains as seen in the course. The equations of the dynamics are not required.

2 ALGORITHMS (15 POINTS)

Your implementations of the algorithms are required to work properly on both environments introduced in section 1. The first one is simpler than the second one. Therefore, we advise you to start with the simple inverted pendulum and then use the double inverted pendulum in order to implement you algorithms. The action spaces are continuous therefore you need to handle them in the appropriate manner in for each algorithm.
We ask you to implement:

- An adaptation of FQI [1].
- A policy Gradient Algorithm like reinforce[3].
- An actor critic method : choose between DDPG [3], PPO [4], Soft Actor Critic [2].

You should save the weights of your models and provide a rendering of the environment through a python script named `interface.py`. Use the function `render()` provided by the `gym` environment. When `python3 interface.py` is called your code should load the weights and control the gym environment in its rendering mode. Therefore, we should be able to see your agent controlling the environment. You should also provide a readme file with the utilization details in order to run each experiment.
Design a protocol to compare the algorithms and discuss the performance of your algorithms in both environments in your report.

[3] `https://fr.wikipedia.org/wiki/REINFORCE`

REFERENCES

[1] Damien Ernst, Pierre Geurts and Louis Wehenkel. “Tree-Based Batch Mode Reinforcement Learning.” In: *Journal of Machine Learning Research* 6 (Apr. 2005), pp. 503–556.

[2] Tuomas Haarnoja et al. *Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor*. 2018. arXiv: 1801.01290 [cs.LG].

[3] Timothy P. Lillicrap et al. *Continuous control with deep reinforcement learning*. 2019. arXiv: 1509.02971 [cs.LG].

[4] John Schulman et al. *Proximal Policy Optimization Algorithms*. 2017. arXiv: 1707.06347 [cs.LG].