Reinforcement Learning 2024
Assignment 3: Policy-based RL

1 Introduction

In the previous assignment, you worked with deep Q-learning, which aims to learn the values of actions in various states. Such a method falls under the umbrella of *value-based* reinforcement learning. In this assignment, in contrast, we are going to investigate the *policy-based* approach to reinforcement learning. In this approach, the policy is optimized directly (i.e., instead of deriving the policy from a representation of the value function).

2 Assignment

In this assignment, you will need to first choose your own environment, then implement variants of policy gradient methods to solve it.

Part 1: Environment selection In order to make the assignment more fun, in this assignment you will pick the task you want to solve. If you want, you can even design and implement your own task (bonus). We provide some commonly-used tasks below, you can pick one of them to work on.

- Mountain Car: A car is placed stochastically at the bottom of a sinusoidal valley, with the only possible actions being the accelerations that can be applied to the car in either direction. The goal is to strategically accelerate the car to reach the goal state on top of the right hill.
- Acrobot: It consists of two links connected linearly to form a chain, with one end of the chain fixed. The joint between the two links is actuated. The goal is to apply torques on the actuated joint to swing the free end of the linear chain above a given height while starting from the initial state of hanging downwards.
- Pendulum: It consists of a pendulum attached at one end to a fixed point, and the other end being free. The pendulum starts in a random position and the goal is to apply torque on the free end to swing it into an upright position, with its center of gravity right above the fixed point.
- Lunar Lander: It is a classic rocket trajectory optimization problem. The rocket needs to be landed within the landing pad.
- Other environments: If you want more challenges, you can find other public OpenAI gym environments to work on, for example, other environments from OpenAI gym. But please be aware that some environments might be difficult to set up.

Part 2: Algorithm variation Implement the following algorithms **from scratch** (it is not allowed to use existing libraries that implement these algorithms such as Stable-baselines).

- REINFORCE
- Actor-Critic with:
  - Bootstrapping

  - Baseline subtraction
  - Bootstrapping + baseline subtraction

For each of these algorithms, use **entropy regularization** as an exploration method. Make sure to **tune the strength of the entropy regularization term in the loss**. How does this affect performance? Do not forget to also (at least) tune the learning rate.

Detailed descriptions of all these algorithms can be found in the lecture notes.

Carefully think about how you can best visualize these questions. **Do not forget to explain/interpret your environment/observations!**

3 Goal

The goal of the assignment is to implement and investigate policy gradient techniques. Questions that you should aim to answer are:

- Do the policy gradients used by REINFORCE indeed suffer from high variance?
- What is the effect of bootstrapping and baseline subtraction, and their combination, within the actor-critic framework on the variance of the policy gradients?
- How do these techniques compare in terms of performance?
- What is the effect of entropy regularization on performance?

As always, make sure to think carefully about the experimental design (what network architecture, what hyperparameters do you use, how to account for randomness, etc.) that you use to answer these questions. Below is a checklist you might want to follow:

- ☐ Implement REINFORCE, different variants of Actor-Critic.
- ☐ Design experiments to compare agents you implemented.
- ☐ Answer all listed questions.

4 Bonus (optional)

In case you are up for an additional challenge, you can consider:

- Implement a genetic algorithm such as **(CMA-ES)** (Covariance Matrix Adaptation Evolution Strategy) to optimize the policy in a gradient-free way.
- Implement a state-of-the-art policy gradient loss such as the **Clip-PPO** (proximal policy optimization) loss. Search yourself for the correct loss term formulation, which is for example described here here.
- Design your own environment and run your implemented algorithms on it.
- Come up with your own idea! Anything you find interesting is worth exploring.

5 Submission

Make sure to nicely document everything that you do. Your final submission consists of:

- Source code with instructions (e.g., README) that allows us to **easily** (single command per experiment / sub task) rerun your experiments on a university machine booted into Linux (DSLab or computer lab).

- A self-contained scientific pdf report (using the ICML template) of at **at most 8 pages** including figures and references using the provided template. You are not allowed to deviate from this page limit or template. Every page over the limit will result in a loss of 1 point. This report contains an explanation of the techniques, your experimental design, results (performance statistics, other measurements,...), and overall conclusions, in which you briefly summarize the goal of your experiments, what you have done, and what you have observed / learned. Make sure you also list the contributions of your team members.

If you have any questions about this assignment, please visit our lab sessions where we can help you out. In case you cannot make it, you can post questions about the contents of the course on the Brightspace discussion forums, where other students can also read and reply to your questions. Personal questions (not about the content of the course!) can be sent to our email address: rl@liacs.leidenuniv.nl.

The deadline for this assignment is the **5th of May 2024 at 23:59 CET**. For each full 24 hours late, one full point will be deducted (e.g., if your work is graded with a 7, but you are two days too late, you get a 5).

Good luck and have fun! :-)

6 Useful resources

For this assignment you may find the following resources useful:

- Lecture notes by Thomas Moerland
- Definitely go to OpenAIs Spinning Up, and the code repo, which is here.
- Baseline implementations of PPO, and many other algorithms are here. Don't copy code from here, only use it as a reference.
- You might find some environments interesting from third-party environments.
- You could use wandb to log/manage your experiments.