Ciências ULisboa | Informática

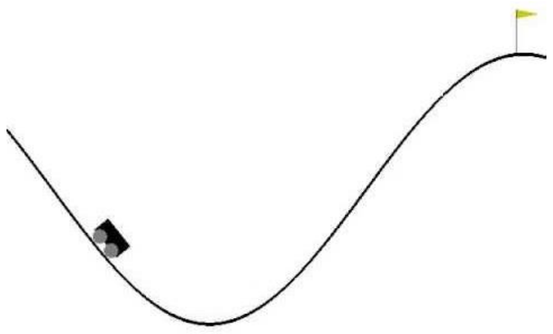
Advanced Machine Learning

Project 3 – Reinforcement Learning

Introduction

Mountain Car is a classic reinforcement learning problem, often used as a benchmark for testing the performance of reinforcement learning algorithms. In this problem, an underpowered car must climb a steep hill to reach a goal located at the top of the hill. The car is subject to the laws of physics, which means that it cannot simply drive straight up the hill. Instead, it must build up speed by accelerating back and forth across the hill.

The state of the Mountain Car environment is represented by two continuous variables, the position and velocity of the car. The goal of the agent is to learn how to control the car's acceleration to climb the hill and reach the goal as quickly as possible while using the least amount of energy. The agent receives a negative reward for every time step it takes to reach the goal, so the goal is to minimize the number of time steps required to reach the goal.



Objective

The main goal of this project is to implement two agents to solve the Mountain Car problem: the Q-learning and the deep learning agent. To accomplish this, some main sub-goals are underlying: understanding the algorithms to train one agent and the challenges associated with it. In that sense, you can find many implementations of the problem online, however, not all of them are good implementations and you won’t learn enough if you copy one of those implementations. Try to solve it by yourself and don’t forget to describe properly your agents in the report (simply putting the code is not enough)! Also, beware that the training of the agent can take some time, so start working on the project as soon as possible.

Environment

Before starting to work on the project, you should read the documentation in Gymnasium about this problem. There you can find the description of the problem, the observation space, the action space, and the reward. See this link.

Work plan

This project has two main parts:

1. Train an agent with Q-learning.
2. Train an agent with deep learning.

To fulfill those two parts, you need to perform the following steps:

- For the Q-learning,
  - Since the space is continuous, you need to discretize the position and velocity of the observation space for it to work with Q-tables. Notice that in this case, you will have a matrix of dimensions PxVxA, where P corresponds to the number of discrete positions, V to the number of discrete velocities, and A to the number of actions. Be careful with the dimensionality of the problem as it will take longer to train.
  - You need to choose the parameters and implement the epsilon-greedy. You can decide to implement an epsilon decay as the episodes evolve.
  - You need to implement the Q-learning algorithm as learned in classes.
  - You need to validate your agent, so you can measure, for instance, the total reward obtained in each episode and/or the number of steps in each episode required to complete the task. You may present the result as plots in function of the number of episodes performed.
- For the deep learning,
  - Study the following tutorials so you can understand how to implement an agent using deep learning. Those tutorials implement deep learning agents for the Cart Pole problem found here.
    - Train a Deep Q Network with TF-Agents
    - Solving Open AI's CartPole Using Reinforcement Learning Part-2
  - Implement an agent using deep learning and train it to solve the problem.
  - Again, you need to validate your agent, so you can use the same measures as for Q-learning.

- Compare the two developed agents (Q-learning and the deep learning one). The first tutorial about deep learning shows a way to make videos of the trained agents. Optionally, you can try to create videos to see the difference between the two agents.

Submission

Each group should submit the report written in Jupyter Notebook with the name **AAA2324_P3_xx_yy.ipynb**, where xx and yy should be replaced by each of the student numbers of the group. The report should include (but not limited to):

- The identification of the members of the group.
- Implementation of both agents (you don't need to present all your experiments for each agent, you can briefly summarize what you did, and the results obtained).
- Comparison and discussion of the results obtained by the agents.

The report should also include your **explanations and justifications for your decisions**, as well as the **code and corresponding outputs** obtained.

Beware that **I will not run the code of all groups, but I might run some groups selected at random**, so I need to have explicit outputs in the report to confirm your conclusions.

Deadline

The deadline for this project is **January 3rd at 23:59 in Moodle**. The groups without access to Moodle to submit the project should send the project by email.