AI & ML 2022/2023 - Project Report

Rosapia Laudati - 1894372

July 21, 2023

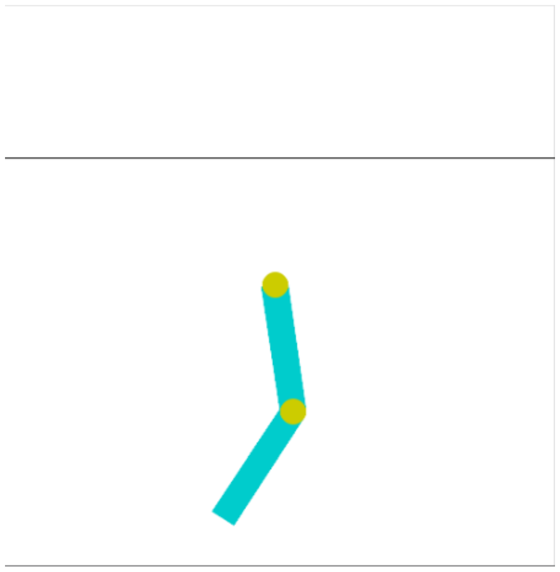
1 Introduction

For my Machine Learning project I implemented a Feedforward neural network to solve *Acrobot*, one of the ”Classic Control” environments from Gymnasium. This project demonstrates the effectiveness of deep reinforcement learning in enabling an agent to learn how to play a complex game like Acrobot.

2 Acrobot

2.1 The environment

The Acrobot environment is a simulated control problem in the field of machine learning and robotics. The system 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.

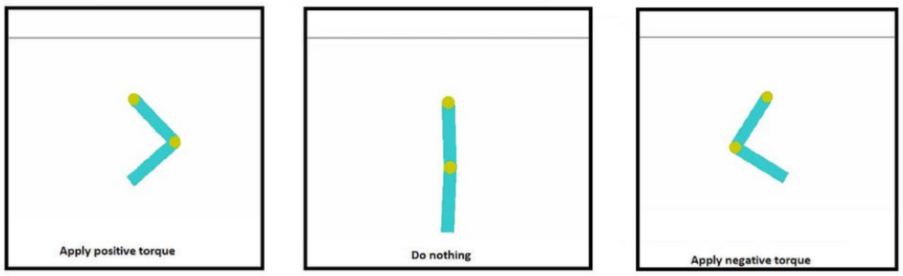
The **state** of the Acrobot environment is defined by the positions and velocities of the two joints of the robotic arm. Specifically, the state consists of 4 continuous variables:

Joint 1 angle	The rate of rotation of the first joint.
Joint 1 angular velocity	The angle of the first joint relative to the vertical.
Joint 2 angle	The rate of rotation of the second joint.
Joint 2 angular velocity	The angle of the second joint relative to the first joint.

Each parameter in the underlying state (*theta1*, *theta2*, and the two angular velocities) is initialized uniformly between -0.1 and 0.1. This means both links are pointing downwards. As we said, the goal is to swing the free end of the outer-link to reach the target height (black horizontal line above system) by applying torque on the actuator.

When the robotic arm chooses and performs an action, it may receive a reward or not. The **action** is deterministic, and represents the torque applied on the actuated joint between the two links.

Action	Description
0	Apply negative torque to the actuated joint (-1)
1	Apply 0 torque to the actuated joint
2	Apply positive torque to the actuated joint (+1)



The **reward** in the Acrobot environment is -1 for each time step, until the agent successfully makes the lower end of the arm reach the topmost position of the swing. Once the agent achieves the goal, it receives a reward of 0, and the episode ends.

The episode **terminates** when the agent successfully makes the lower end of the arm reach the topmost position of the swing (i.e., when the second joint surpasses a certain height) or when the maximum number of time steps is exceeded (500 time steps). The target height is constructed as following:

$$-cos(theta1) - cos(theta2 + theta1) > 1.0$$

.

2.2 The solution implemented

The solution I implemented for the Acrobot environment makes use of a neural network with back-propagation to approximate the Q-function and, therefore, to estimate the optimal policy.
The feed-forward neural network was implemented using pytorch and has three fully connected layers:

- **Input layer**: receives the state observations, which consists of four continuous variables representing the angles of the pendulum joints, and applies a linear transformation followed by ReLU activation function. The input layer has the same number of unit as the input dimensionality, which is four in this case and has 128 units in output.

- **Hidden layer**: receives the input from the input layer and performs computations using the received information. It has same input and output of 128 units and also applies a linear transformation and ReLU activation.

- **Output layer**: receives the input form the second layer and performs the final linear transformation to produce the output representing the Q-values for each possible action. It has 128 units in input and 3 units in output, since the number of possible action is 3. The output layer does not have an activation function applied.

To improve the efficiency and stability of training I used the *experience replay* technique encapsulated within the *ReplayMemory* class. This class stores the agent's experiences in the form of transitions that consists of the current state, the action taken, the next state, and the corresponding reward. Then, during the optimization, we randomly sample a batch of 128 transitions and use them to optimize our model.

For the optimization I used the AdamW optimizer, with learning rate 0,001, and the HuberLoss to calculate the error between the estimated Q-values and the target Q-values, with discount factor γ that is 0,99.

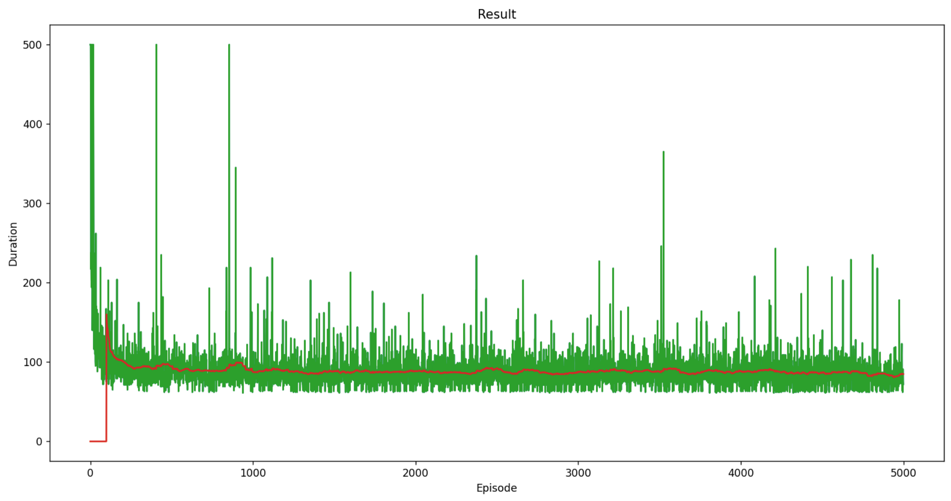
The constants related to the exploration-exploitation trade-off are:

EPS_START	0.9
EPS_END	0.05
EPS_DECAY	1000

In this case, the exploration rate decreases gradually over 1000 steps. The epsilon-greedy policy is used for action selection, where the agent chooses a random action with probability ϵ and chooses the action with the highest expected reward with probability $1 - \epsilon$.

2.3 Training

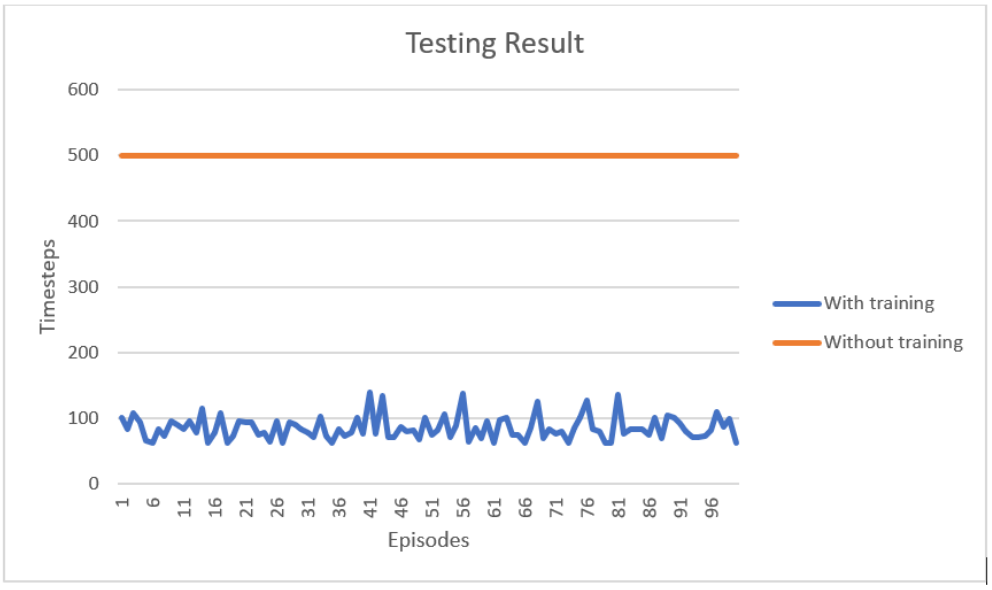
I have trained the model for 5000 episodes, and in the following picture we can see the cumulative averages of the episode results (red line).



By looking at the graph obtained from the training we can see that:

- The graph shows fluctuations in the duration of episodes, which may be indicative of an agent's exploration and learning phase, in which he or she is testing different strategies and trying to overcome any challenges or obstacles in the environment;
- The duration of the episodes decreases, which indicates that the agent is learning to reach the goal faster
- The duration of episodes stabilizes over time, which indicates that the agent has achieved a form of convergence and learned an effective strategy to complete the environment.

I attach below the results obtained from the 100-episode test on the trained model and the untrained model:



As we can see and expect, choosing completely at random is a worthless strategy, which mean that the link never reaches the target almost every time, which means that the episode lenght is greater than 500. Instead, from the testing on the trained model, we can see that out of 100 episodes, the average number of steps needed to reach the target is about 86, a far lower value.