FC2:Reinforcement Learning

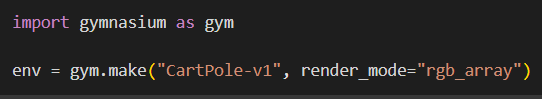
Diego Pardo Gonzalez ID:170531

**Introduction**

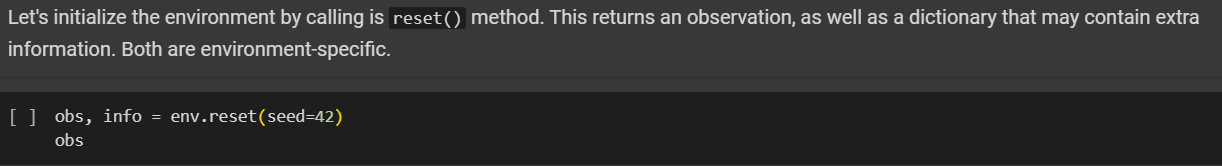
The code is based on the cart pole experiment; this experiment is very similar to the motion of a pendulum that is really common to solve equilibrium problems. The code sets a cart and a pole that is vertically on top of the cart. The environment sets basically that the cart can move from left to right as a one-dimension change. The objective of the cart is to move when it is needed for the pole to stay in equilibrium and don’t fall.

**Running the code as it was given**

We can observe in the beginning of the code that the project requires python 3.7 or above and we need to import series of libraries like tensorflow, sys, matplotlib.animation, matplotlib.pyplot, pathlib and others. Then the code imports gymnasium where we can find our “cart pole” experiment.



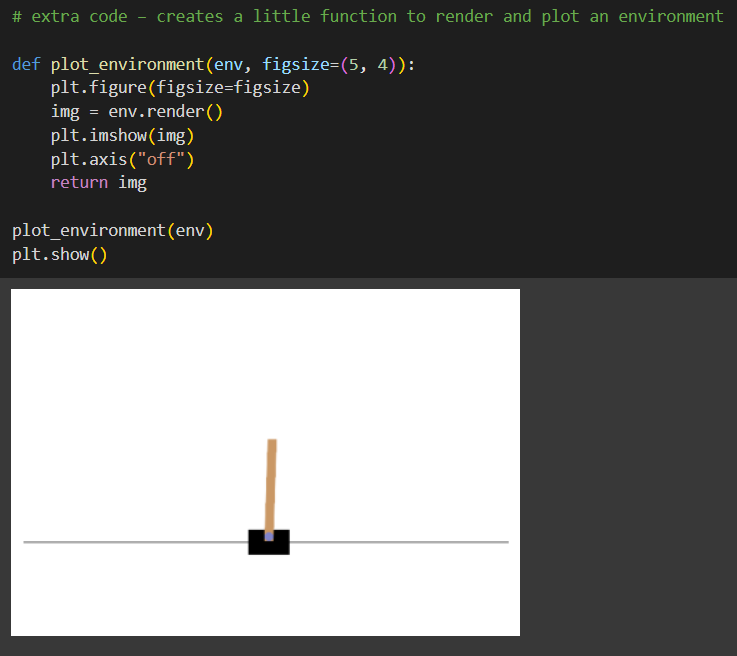
As we can see in the google colab code, we are going to use the version 1 of the CartPole, the environment of this version is really similar as the explanation above; the cart can only move in one dimension (left-right) and we have a pole that is set vertically on top of our cart and the objective of the problem is keeping the pole upright. As is mentioned in the code, each observation is a 1D NumPy array composed of 4 floats that represent the following aspects in the following order: carts horizontal position, velocity, angle of the pole and the angular velocity.



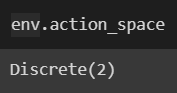
In the code we can appreciate that we are setting a seed t make that the initial state of the environment after calling reset() will be the same everytime.

Here we have the output:



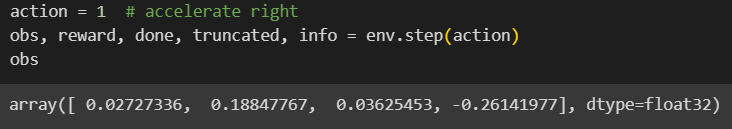


Let's see how to interact with an environment. Your agent will need to select an action from an "action space" (the set of possible actions). Let's see what this environment's action space looks like:



Yep, just two possible actions: accelerate towards the left (0) or towards the right (1).

Since the pole is leaning toward the right (obs[2] > 0), let's accelerate the cart toward the right:



Notice that the cart is now moving toward the right (obs[1] > 0). The pole is still tilted toward the right (obs[2] > 0), but its angular velocity is now negative (obs[3] < 0), so it will likely be tilted toward the left after the next step.

**Simple hard Coded Policy**

This is a pretty simple code, hard code a simple strategy: if the pole is tilting to the left, then push the cart to the left, and vice versa.

Well, as expected, this strategy is a bit too basic: the best it did was to keep the poll up for only 63 steps. This environment is considered solved when the agent keeps the poll up for 200 steps.

