**Balancing a Pole on a Cart**

**23S2 SC3000/CZ3005 Assignment 1**



* **Problem description**

As shown in the figure below, a pole is attached by an un-actuated joint to a cart, which moves along a frictionless track. The pendulum is placed upright on the cart and the goal is to balance the pole by applying forces in the left and right direction on the cart. In this project, you will need to develop a Reinforcement Learning (RL) agent. The trained agent makes the decision to push the cart to the left or right based on the cart position, velocity, and the pole angle, angular velocity.

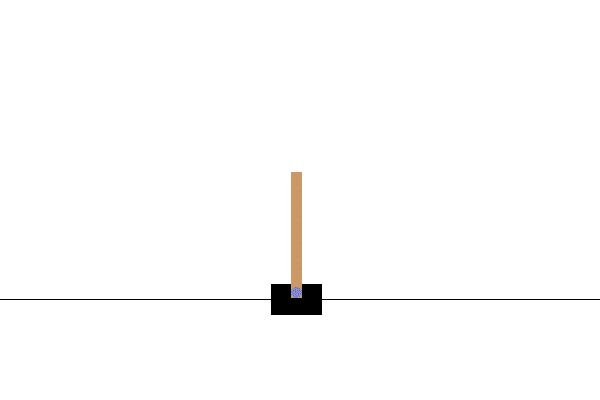


Figure 1. Balancing a pole on a cart.

* **Problem instance**

You are given an instance of the cart pole environment implemented by the gym library. A step-by-step tutorial on installing, loading, and using the CartPole environment is included in <https://github.com/yue-zhongqi/cartpole_colab>. You will use *Jupyter notebook*, an interactive, and what-you-see-is-what-you-get python script environment, which is widely used in AI research community. Before starting this project, please go to <https://docs.jupyter.org/en/latest/> and [https://colab.research.google.com/notebooks/basic\_features\_overview.ipynb#scrollTo=KR921S\_OQSHG](https://colab.research.google.com/notebooks/basic_features_overview.ipynb) for some hands-on examples.

A summary of the pole cart environment is given below:

**Action Space**

The action is an ndarray with shape (1,) which can take values {0, 1} indicating pushing the cart to the left or right, respectively. Note that the velocity that is reduced or increased by the applied force is not fixed and it depends on the angle the pole is pointing. The center of gravity of the pole varies the amount of energy needed to move the cart underneath it.

**Observation Space**

The observation is an ndarray with shape (4,) with the values corresponding to the following positions and velocities:

|  |  |  |  |
| --- | --- | --- | --- |
| Num | Observation | Min | Max |
| 0 | Cart Position | -4.8 | 4.8 |
| 1 | Cart Velocity | -Inf | Inf |
| 2 | Pole Angle | ~ -0.418 rad (-24°) | ~ 0.418 rad (24°) |
| 3 | Pole Angular Velocity | -Inf | Inf |

**Reward**

Since the goal is to keep the pole upright for as long as possible, a reward of +1 for every step taken, including the termination step, is allotted.

**Starting State**

All observations are assigned a uniformly random value in (-0.05, 0.05).

**Episode End**

The episode ends if any one of the following occurs:

Termination: Pole Angle is greater than ±12°

Termination: Cart Position is greater than ±2.4 (center of the cart reaches the edge of the display)

Truncation: Episode length is greater than 500.

* **Requirements and guidelines**

**2.1 Tasks and marking criteria**

You will need to solve three tasks that are listed below. You can refer to the Jupyter notebook in <https://github.com/yue-zhongqi/cartpole_colab> for sample codes.

**Task 1:** Development of an RL agent. Demonstrate the correctness of the implementation by sampling a random state from the cart pole environment, inputting to the agent, and outputting a chosen action. Print the values of the state and chosen action in Jupyter notebook.

(30 marks)

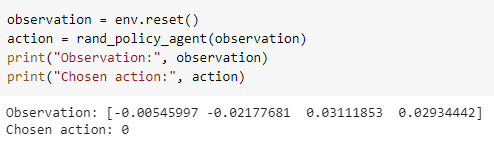


Figure 2. Sample code for Task 1. Open the sample code link for more details.

**Task 2:** Demonstrate the effectiveness of the RL agent. Run for 100 episodes (reset the environment at the beginning of each episode) and plot the cumulative reward against all episodes in Jupyter. Print the average reward over the 100 episodes. The average reward should be larger than 195.

(40 marks)

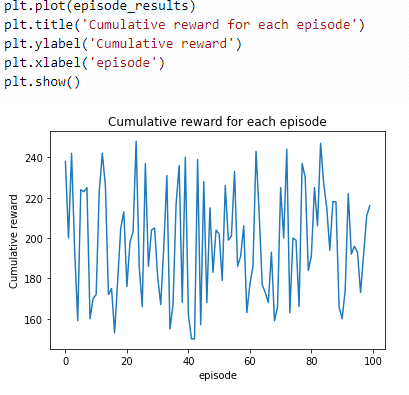
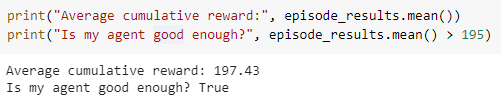


Figure 3. Sample code and output for Task 2.

**Task 3:** Render one episode played by the developed RL agent on Jupyter. Please refer to the sample code link for rendering code.

(10 marks)

**Task 4:** Format the Jupyter notebook by including step-by-step instruction and explanation, such that the notebook is easy to follow and run (refer to the tutorial section in the sample notebook). Include text explanation to demonstrate the originality of your implementation and your understanding of the code. For example, for each task, explain your approach and analyze the output; if you improve an existing approach, explain your improvements.

(20 marks)