# Balancing a Pole on a Cart

**24S2 SC3000/CZ3005 Assignment 1**

**Submission deadline: 11:59pm, 4 April 2025**

## 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.

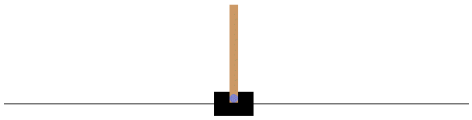


Figure 1. Balancing a pole on a cart.

### 1.1 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 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.

# 2. 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)

```
observation = env.reset()
action = rand_policy_agent(observation)
print("Observation:", observation)
print("Chosen action:", action)
```

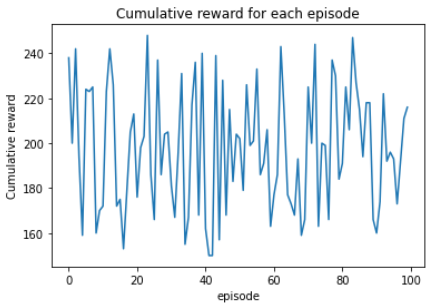
```
Observation: [-0.00545997 -0.02177681  0.03111853  0.02934442]
Chosen action: 0
```

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)

```
plt.plot(episode_results)
plt.title('Cumulative reward for each episode')
plt.ylabel('Cumulative reward')
plt.xlabel('episode')
plt.show()
```




```
print("Average cumulative reward:", episode_results.mean())
print("Is my agent good enough?", episode_results.mean() > 195)
```

```
Average cumulative reward: 197.43
Is my agent good enough? True
```

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)

## 2.2 Output format

All codes are to be included in a single Jupyter notebook written in Python (i.e.,ipynb file).

1. Include *all* codes for Task 1-4. Note that the submission is invalid if it only contains the outputs and plots without codes to obtain it.
2. Run the notebook before submission to save the output in the notebook, i.e., by opening the ipynb file (without running it), one can see the outputs and plots for Task 1-3
3. Make sure the Jupyter notebook is runnable, i.e., by running each code block sequentially from top to bottom, one can get the results for Task 1-3. The TAs may run your notebook.
4. Unless you are experienced with Jupyter, it is recommended to modify from the provided Jupyter notebook sample, rather than creating a new one.
5. If the developed RL agent is a trainable neural network, submit a .zip file by zipping the trained model parameters (e.g., .pth for PyTorch) and the ipynb file. In this case, your notebook must include the training code and model loading code.
6. Contribution: Please clearly state the contribution of each team member in the beginning of the Jupyter notebook if you have more than one member in your team.

## 2.3 Programming language

For the sake of running your codes, you are required to use Python and Jupyter notebook package. The TAs may run your codes by executing the code blocks in the notebook from top to bottom. All the algorithms must be implemented by you.

# 3. Teaming Formation

You are required to form a team of up to 3 people to complete this project. A team with only a single person is also allowed. For team size that is larger than one, you need to clearly state the contribution of each team member.

# 4. Honor Code

If you use any existing code, libraries, etc. and consult any papers, books, online references, etc. for your project**, you must cite your sources in your report and clearly indicate which parts of the project are your contributions and which parts were implemented by others.** Using others' code/ideas without acknowledgement will lead to failure of your project.

# 5. Deliverables

A single .ipynb file if your agent does not require training. Otherwise submit a single zip file containing the .ipynb file and trained model parameters. Use your team name as the name of the submitted file.

# 6. Submission

Please email your file (*<team name>.ipynb* or *<team name>.zip*) to the TA for your lab group by the ***deadline 11:59pm 4th April 2025***. Note that it is a hard deadline, no extension is allowed.

You can find the TA and TA's email in the Information tab on NTULearn. If you have questions, please ask questions during the lab session or email your TA.