

National University of Singapore
School of Computing

Semester 1, AY2024-25

CS4246/CS5446

AI Planning and Decision Making

Issued: 22 October 2024

Due: Thursday 7th November 2024 23:59

Rush Hour Adventures: Episode II - Reinforcement Learning

Guidelines

Please complete the Assignment in a 2-person TEAM. **ONE person from each team** should submit ALL the answers. Grading will be done team-wise.

On collaboration

You are encouraged to discuss solution ideas. However, each team *must write up the solutions independently*. It is considered plagiarism if the solution write-up is highly similar to other teams' write-ups or other sources.

Assignment Solutions Submission

- You can download the programming package from Canvas or from [this GitHub repo](#).
- We are using [Google Colab](#) to code and show instructions. You can set up and use the notebook environment on Colab [here](#).
- No late submissions are allowed.
- Your results will be submitted to the AICON (AI Contest) platform cs5446.comp.nus.edu.sg, see detailed instructions in the following sections.
- Please write the following on the top of the submission:
 - *Name* and *metric number* of **all team members** as they appear on Canvas.
 - Collaborators (write **None** if no collaborators)
 - Sources, if any, or if you obtained the solutions through research, e.g., through the web.

Background: The Elevator Domain

Welcome to the Elevator Challenge, episode II! Your mission, should you choose to accept it, is to design and implement efficient planning solutions to manage the movement of an elevator that serves impatient office workers eager to reach the ground floor. This seemingly simple scenario brings a host of complex planning challenges that require careful thought and strategy.

We are going to study the elevator domain used in Assignment 1 further. Suppose the building has 5 floors and 1 elevator, each floor has a maximum of 3 people waiting. The elevator can carry a maximum of 10 people. The task is to pick up passengers from different floors and deliver them to the first floor. New people may come to each floor to wait for the elevator at each time. The elevator can move up, move down, pick up, and drop off passengers. It can only pick up or drop off passengers if the door is open, and it can only move if the door is closed.

In this task, the state space is very large and complex, making it impossible to use tabular reinforcement learning (RL) algorithms to solve. Therefore, we turn to deep reinforcement learning (DRL). Your task is to complete a DRL algorithm (Proximal Policy Optimization, PPO) to train the agent and evaluate its performance.

To understand more about the domain, please explore the provided interactive Jupyter notebook, `ElevatorEnv.ipynb`, which details the state space, action space and interaction with the environment.

Instructions on Solution

The assignment can be worked on the Jupyter notebooks, and we encourage you to use Google Colab, which provides free access to high-performance GPU resources. There are more detailed instructions in the `Deep Reinforcement Learning.ipynb` notebook. All blocks you should complete are indicated by comments like (where N indicates different tasks):

```
### ----- TASK N ----- ###
### ----- YOUR CODES START HERE ----- ###

### ----- YOUR CODES END HERE ----- ###
```

Prerequisites: Numpy and PyTorch (Optional)

If you are unfamiliar with Numpy and PyTorch for machine learning, you may refer to [CS285 Lecture 3](#) and the corresponding [CS285 Lecture 3 Google Colab notebook](#) for a technical and hands-on tutorial on how Numpy and PyTorch works for Deep Reinforcement Learning.

Assignment: Proximal policy optimization (PPO)

In this assignment, we're going to implement the Proximal Policy Optimization (PPO) algorithm to solve the Elevator task. PPO is an actor-critic-style algorithm that uses a clipped surrogate objective to ensure that the policy update does not deviate too far from the current policy. [1]

[1] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347. <https://arxiv.org/abs/1707.06347>.

PPO maintains two main components: an actor (the policy) and a critic (the value function). In DRL, the actor and critic are both implemented as neural networks. The actor represents the policy, i.e., a neural network mapping from a state to a distribution of actions, in our case, we use the categorical distribution: $\pi(a|s; \theta)$; while the critic represents the value function, i.e., a neural network mapping from a state to a value: $V(s; \phi)$, the θ and ϕ are parameters respectively.

There are three tasks to complete in the assignment:

1. Complete the forward functions in the actor-critic-style agent.
2. Given the rollout data, compute the advantages and returns.
3. Update the actor and critic networks: implement the loss functions for actor and critic modules.

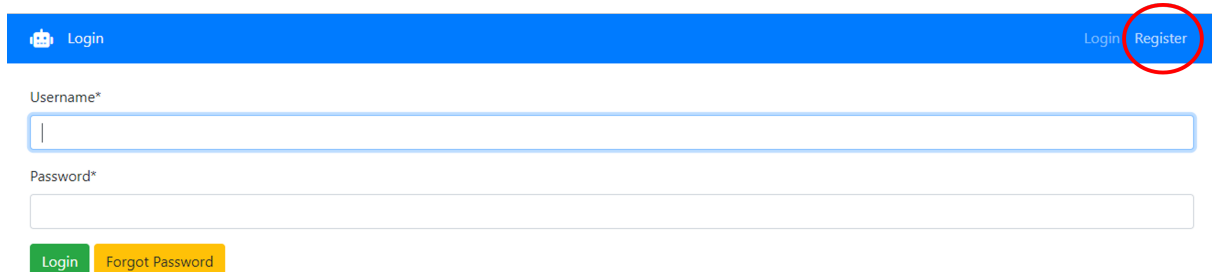
For further detailed instructions, please follow the `Deep Reinforcement Learning.ipynb`.

Submission Instructions

After finishing your assignment on Google Colab or locally, **EVERY INDIVIDUAL STUDENT** will need to submit the saved model weights and the code derived by your group. We will be using the AICON (AI Contest) platform: cs5446.comp.nus.edu.sg. You will need to first register for the platform before submitting your solutions. There are two Tasks in total, the first is to assess the final performance of your learnt agent, the second is to evaluate the correctness of your code.

1. Registration


1. Click on 'Register' on the top right corner of the site and enter your details



Username*

Password*

Login Forgot Password

 Create Account Login Register

Student ID (Axxxxxxx)*

Required: 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Email*

First name*

Last name*


Password



- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

Password confirmation


Enter the same password as before, for verification.

2. Click into CSxx46

 A12345678W

Course	Group	Tasks	
CSxx46 2024-2025 - Semester 1	None	7	
Test1 2099 - Semester 1	None	0	

3. Enter the invitation key: wI2Q7uOMw8Og6v06Dxf6k6ycI2-2W7sEGHIGrNigz3M

 CSxx46 - 2024-2025 Semester 1 / Join A12345678W

Invitation Key*

2. Submission

1. After registration, you should see two Tasks:

- Assignment 2: Deep Reinforcement Learning
- Assignment 2: Deep Reinforcement Learning - Components

Click the task you wish to submit.

CSxx46 - 2024-2025 Semester 1 Tasks A12345678W			
Task	Deadline	Opened At	Status
Assignment 2: Deep Reinforcement Learning	None	None	Open
Assignment 2: Deep Reinforcement Learning - Components	None	None	Open

2. Click 'New Submission' and then 'Code'

CSxx46 - 2024-2025 Semester 1 / Assignment 2: Deep Reinforcement Learning
Tasks A12345678W

New Submission
Template
Task Info

Code

Package

Submission

Size

Verdict

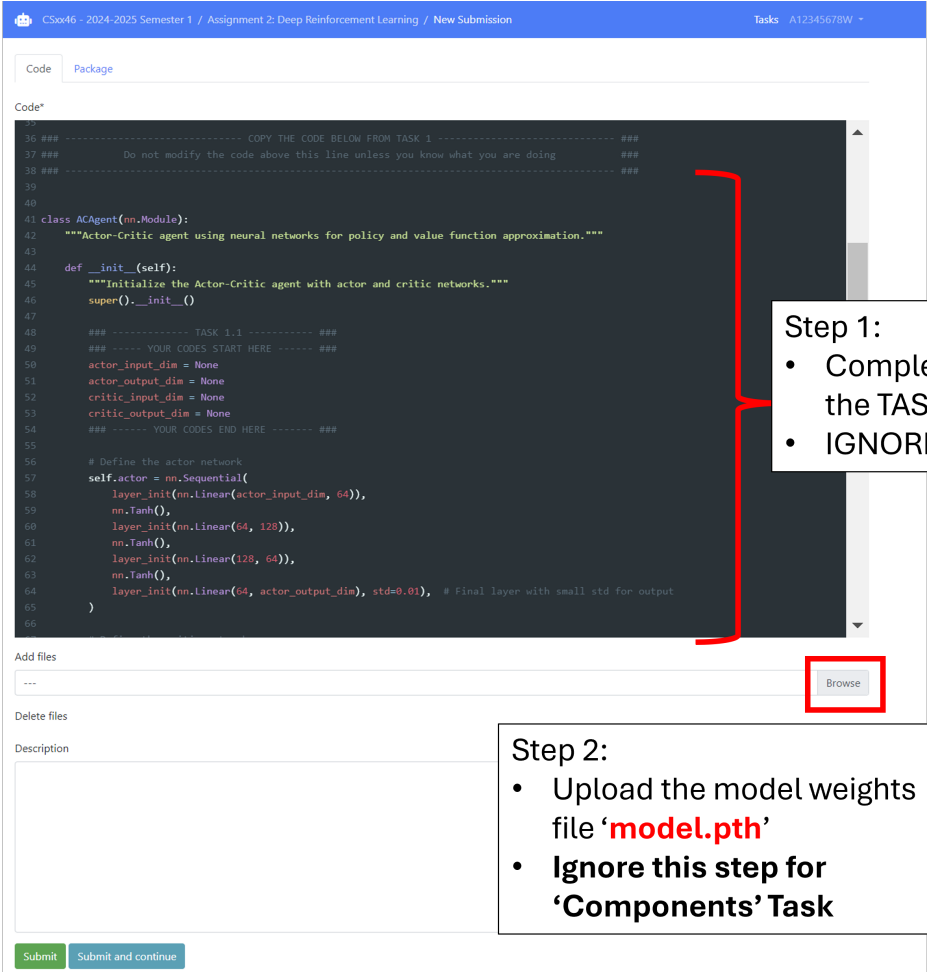
Created At

No submission.

« 1 »

Per Page 10

3. You should see a code template. Complete the code in the TASK code blocks and ignore all other code
4. Upload the model weights file 'model.pth' for the first task. You do NOT need to upload the weights for the 'Components' task.
5. Click Submit.



CSx46 - 2024-2025 Semester 1 / Assignment 2: Deep Reinforcement Learning / New Submission

Code Package

Code*

```
36 ### ----- COPY THE CODE BELOW FROM TASK 1 ----- ###
37 ### Do not modify the code above this line unless you know what you are doing ###
38 ### -----
39
40
41 class ACagent(nn.Module):
42     """Actor-Critic agent using neural networks for policy and value function approximation."""
43
44     def __init__(self):
45         """Initialize the Actor-Critic agent with actor and critic networks."""
46         super().__init__()
47
48         ### ----- TASK 1.1 ----- ###
49         ### ----- YOUR CODES START HERE ----- ###
50         actor_input_dim = None
51         actor_output_dim = None
52         critic_input_dim = None
53         critic_output_dim = None
54         ### ----- YOUR CODES END HERE ----- ###
55
56         # Define the actor network
57         self.actor = nn.Sequential(
58             layer_init(nn.Linear(actor_input_dim, 64)),
59             nn.Tanh(),
60             layer_init(nn.Linear(64, 128)),
61             nn.Tanh(),
62             layer_init(nn.Linear(128, 64)),
63             nn.Tanh(),
64             layer_init(nn.Linear(64, actor_output_dim), std=0.01), # Final layer with small std for output
65         )
66
```

Add files

Browse

Delete files

Description

Step 1:

- Complete the code in the TASK code blocks
- IGNORE all other code

Step 2:

- Upload the model weights file '**model.pth**'
- **Ignore this step for 'Components' Task**

Step 3:

- Submit

Submit Submit and continue

Good luck, and have fun exploring the fascinating world of Reinforcement Learning!