Deep Reinforcement Learning

Homework #2

Due: 2024/04/09 (Tue.) 23:59

Problem Description

Overview & Installation

Saving the World of Mario! The gym-super-mario-bros is an Reinforcement Learning environment for Super Mario Bros. & Super Mario Bros. 2 (Lost Levels). Your mission is to train an agent to successfully navigate through the vibrant world of Mario, aiming to maximize the expected rewards over epochs. The challenge encompasses 32 stages, and you are tasked with conquering all of them while starting with just 3 lives.

The package lists six optional environments in total, all featuring high-dimensional observation spaces (e.g., RGB image frames). Your assignment is to implement and train your agent in the **SuperMarioBros-v0** environment.

It is recommended to use virtual environments like conda for managing your codebase. Super Mario Bros for OpenAI Gym is compatible with a wide range of Python versions (from 3.6 to 3.8). Install gym-super-mario-bros directly using:

```
pip install gym-super-mario-bros
```

Environment Spaces

There are three types of action spaces with varying actions. For this assignment, to master the game and become the king of the Mario world, we will be using the **COMPLEX_MOVEMENT** action space. You are expected to initialize your environment with:

```
env = gym_super_mario_bros.make('SuperMarioBros-v0')
env = JoypadSpace(env, COMPLEX_MOVEMENT)
```

Filed	Type
Action	Discrete(12) | Actions Explained
Observation	Box(0, 255, [240,256,3]) | Numpy array of shape (240,256,3) - A single frame

Reward	Default Rewards in range (-15,15)

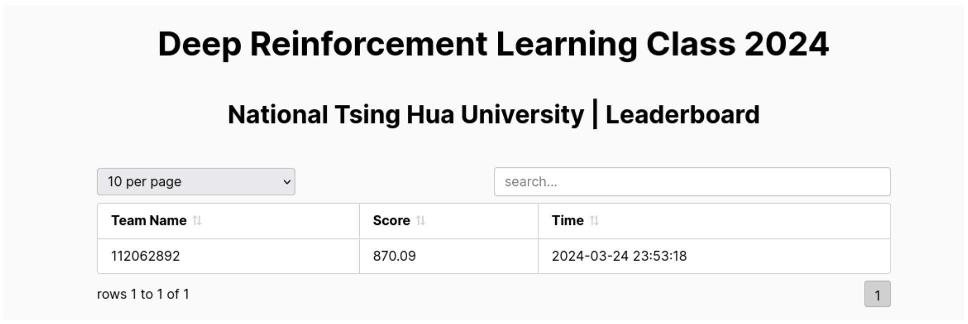
Programming Requirements

1. Implement and train your agent in **SuperMarioBros-v0** environment.
2. Use DQN as your core algorithm, but enhancements are permitted (DQN、Double DQN、DRQN、Dueling DQN、Prioritized Experience Replay, ...). Anything that is not fit with DQN is now allowed. (3C、PPO、Distributional RL...)
3. You must write all your training and testing code by yourself.
4. You may store your learned model in an external file "./<Student_ID>_hw2_data", and access it with your program (during testing).
5. You should implement a class called `Agent` with the member function `act(observation)` function in your testing code. See `random_agent.py` for an example. **[`Agent.__init__` should take NO OTHER ARGUMENT RATHER THAN `self`]**
6. If your program outputs invalid moves, you lose and the game ends immediately.
7. Time limit for each move is 2.5 second, and the memory limit is 4 GB. (Note that the 1-second duration may vary depending on different processors. If you use a DQN-based agent and it doesn't perform additional calculations during inference, you don't need to worry about the time limit.)
8. You are allowed to use the following Python package:
- numpy, scipy, gym, pandas, tensorflow, pytorch and the packages mentioned in the environment's repo.
- You are allowed to use Python default installed packages. (e.g., sys, time, pickle, random, etc.)
- If you need to use other packages, state your reasons and post on hackmd.
- You are not allowed to use any ready-made RL algorithms like stablebaseline3.

Other Requirements

A.Write a report to:
1. Detail your agent's design and any advanced DQN techniques used. Ensure your report includes sufficient detail and figures for grading.
2. Include screenshots of critical code snippets in your report as the report will be graded independently of the code.

B. Win against your classmates!



Deep Reinforcement Learning Class 2024

National Tsing Hua University | Leaderboard

Team Name	Score	Time
112062892	870.09	2024-03-24 23:53:18

rows 1 to 1 of 1

A leaderboard will be provided for comparison, significantly influencing your overall score based on your ranking.
Submit your agent to the leaderboard as instructed in the repository. Ensure your agent is set to **CPU mode** for leaderboard submission. The leaderboard will evaluate your agent across 50 epochs, with a maximum epoch limit of 120 seconds. Scores are calculated based on total_reward/50.
Submission to leaderboard is **compulsory**.

Program Submission

1. For each problem, please use Python to implement with a single source file.
Your files must be named as:
    - **"<Student_ID>_hw2_train.py"**
    - **"<Student_ID>_hw2_test.py"**
    - **"<Student_ID>_hw2_data"**
    - **"<Student_ID>_hw2_report.pdf"**

and please make sure that all characters of the filename are in lower case. For example, if your student id is 110062579, the name of your program file should be 110062579_hw2_train.py and so on.
2. Compress all your files directly (do not compress the folder containing your files) and name it as **"<Student_ID>_hw1.zip"**, then upload to the eeclass submission portal before the deadline. (Total 4 files).
3. Submit your code to the leaderboard and follow the specification.
4. 0 points will be given to **Plagiarism**. NEVER SHOW YOUR CODE to others and you must write your code by yourself. If the codes are similar to other people's and you can't explain your code properly, you will be identified as a plagiarist.
5. 0 points will be given if you violate the rules above. If you use modularized / OOP code and want to use multiple files to keep your code structured, please upload it along with the 3 files above.

Grading Policy

1. The project accounts for 15 points (tentative) of your total grade.
2. You must submit both your source code and report. Remember the submission rules mentioned above, or you will be punished on your grade. Late submission rules are specified in Lecture 1 Slides.

Detailed marking rubric is given in below:

Components	Marks
Checker File	20
DQN Implementation	40
Report	20
Leaderboard Ranking	20
TOTAL	100