AU 342 Principles of Artificial Intelligence
Homework 2 Due OCT 25th 11:59pm

Adhere to the Code of Academic Integrity. You may discuss background issues and general strategies with others and seek help from course staff, but the implementations that you submit must be your own. In particular, you may discuss general ideas with others but you may not work out the detailed solutions with others. It is never OK for you to see or hear another student's code and it is never OK to copy code from published/Internet sources. If you feel that you cannot complete the assignment on you own, seek help from the course staff.

When submitting your assignment, follow the instructions summarized in Section 4 of this document.

1 Reinforcement Learning in Maze Environment

In this assignment, you will implement a Dyna-Q learning agent to search for the treasure and exit in a grid-shaped maze. The agent will learn by trail and error from interactions with the environment and finally acquire a policy to get as high as possible scores in the game. Please finish this part *individually*.

1.1 Game Description

Suppose a 6×6 grid-shaped maze in Figure 1. The red rectangle represents the start point and the green circle represents the exit point. You can move upward, downward, leftward and rightward and you should avoid falling into the traps, which are represented by the black rectangles. Finding the exit will give a reward +1 and falling into traps will cause a reward -1, and both of the two cases will terminate current iteration. You will get a bonus reward +3 if you find the treasure, which shown as golden diamond.

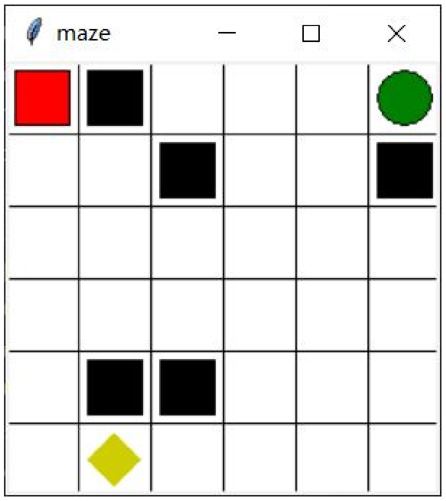

Figure 1: Maze environment

The environment is implement in **maze_env.py**. The state space and action space are briefly described as follows and you may learn more details in the code.

State(5-dimension): The current position of the agent (4D) and a bool variable (1D) that indicates whether the treasure has been found.

Action(1-dimension): A discrete variable and 0, 1, 2, 3 respectively represent move upward, downward, rightward and leftward.

Note: Please *do not* revise the **maze_env.py**.

1.2 Dyna-Q Learning [50 points]

You are asked to implement an agent based on Dyna-Q learning. Please design the agent in **agent.py** and complete the training process in **main.py**. An agent with random policy is provided in **main.py**. You can learn how to interact with the environment through the demo and then write your own code.

1.3 Epsilon Greedy [20 points]

Considering the explore-exploit dilemma, complete your Dyna-Q learning agent by implementing ε-greedy action selection, meaning it chooses random actions in an epsilon fraction of the time, and follows its current best Q-values otherwise. Note that choosing a random action should not choose a random sub-optimal action, but rather any random legal action.
You should balance the exploration with exploitation. In the early stage of the training, exploration should be encouraged as we hope that the agent can find better policy (e.g. getting the treasure). Then with the advancement of training, exploration may be reduced for the convergence of the policy. Our goal is to get high score with as few as possible interactions.

2 Reinforcement Learning on Atari Game

In this part, you will implement a DQN agent to play atari game *pacman*. You are asked to read the given code and complete the remaining part. The performance of the agent may not be satisfactory and you have to tune it to get higher scores. You will finish this part ***in a group of two***.

2.1 Game Description

Pacman is one of the classic and leading games. You need to guide the pac-Man to eat all the dots and avoid the ghosts. In this assignment, you are asked to design a DQN agent to learn control policies directly from the visual information of the game.

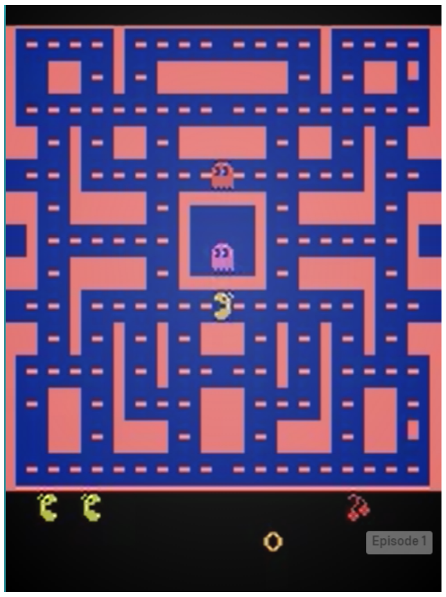

Figure 2: Atari MsPacman

As shown in the figure, the 'MsPacman-ram-v0' gym environment is utilized as the training environment. This environment provides the ram(128 bytes) of the atari console as model input. Each time, the agent should choose an action from 9 available actions, corresponding to the 8 buttons on the handle and "do nothing".

2.2 Deep Q-Learning [20 points]

Considering the complexity of the task and the limitation of hardware requirements, you will implement a simple DQN agent in this part and set a training limitation on the number of iterations (5000 episodes at most). The DQN agent and the training process are almost complete in the given code. You need to understand the code and complete the remaining part. Before running your code, you need to install the required environment for it (see Section 3).

2.3 Tune the Agent [10 points]

The agent in Section 2.2 may not (almost impossible to) get a high score in the game. You need to tune the agent to get as high as possible scores **within 5000 iterations** of the training. Here are some possible ways to improve the performance:

- **(requested)** Tuning the hyper-parameters of the agent.
- Give penalty when the agent dead.
  (Please don't add the penalty you designed to the final score of the DQN model.)
- Tuning the structure of the policy network.
- Implementing prioritized experience replay.
- Normalization of the model input.
- Utilize multiple continuous frames of the game instead of one frame each time.
- Your own ideas.

2.4 Bonus

You will be awarded extra score for the following works:

- Modify the structure of the DQN model (not the policy network) and improve its performance;
- Obtain better performance utilizing other DRL algorithms written by yourself.
- Improve the performance with other novel ideas. Please describe them in the report clearly.

3 Installation

To train the agent on atari game, you need to install Gym and Keras. (You also can choose tensorflow, pytorch or other libraries.) You can follow the tutorial in this section to install the environment on Linux, Windows or macOS, and we strongly recommend you use Linux system.

3.1 Install Anaconda

Open the address `https://www.anaconda.com/distribution/` and download the installer of **Python 3.x version**(3.6 recommended) for your system.

- For Linux

  ```
  bash Downloads/Anaconda3-2019.07-Linux-x86_64.sh
  ```

- For Windows

  Open the .exe file and follow the installer steps.

- For macOS

  Open the .pkg file and follow the installer steps.

3.2 Install Required Environment

After installing anaconda, open a terminal (Linux and macOS) or Anaconda Prompt (Windows) and create an environment for Gym:

```
conda create python=3.6 --name gym
```

Then activate the environment

```
conda activate gym
```

Install gym

```
pip install gym

pip install gym[atari]
```

Install tensorflow (1.x version recommended) and keras (2.3.1 recommended)

```
pip install tensorflow==1.13.1

pip install keras==2.3.1
```

Install matplotlib

```
pip install matplotlib
```

Install pandas

```
pip install pandas
```

Execute the code

```
cd

python atariDQN.py
```

4 Submission instructions

1. Zip all your python files **main.py**, **agent.py**, **atariDQN.py** and report file **HW2.pdf** to a folder called **homework2_studentID_name.zip**.
2. Upload the file to the jbox. **https://jbox.sjtu.edu.cn/l/PJl21u**