Spring 2024
Introduction to Artificial Intelligence

Homework 4: Reinforcement Learning

Due Date: 2024/5/13 23:59

Introduction

Reinforcement learning is an important topic in the field of machine learning. It helps agents learn the policy to achieve their goal by interacting with the environment, and has achieved many outstanding performances in computer games. In this assignment, you will utilize OpenAI Gym environments (i.e., Taxi-v3 and CartPole-v0) and implement a RL algorithm, Q learning. Your agent will learn a policy by interacting with the environment. Then, you will analyze the performance and results and answer some questions.

The goal of this programming assignment is:

1) Implement Q learning in different ways based on the environment and the requirements
2) Get a taste of a complete reinforcement learning process.
3) Obtain hands-on experience in Python packages for Deep Learning (e.g., PyTorch in this assignment)
4) Learn the difference between RL and DRL

Before you start

Please make sure you understand the concept of **Q learning** and **DQN**. If you are not familiar with PyTorch or DQN, please read the **tutorial** in the reference. In this assignment, we will use Taxi-v3 and CartPole-v0, which are provided by OpenAI Gym, as our environments. Please refer to the **documentation** in the reference we provide.

Setup

We recommend you to use python **3.7** and all the packages you need are listed in the requirements.txt. Please run the command to install the packages:

```
pip install -r requirements.txt
```

File structure

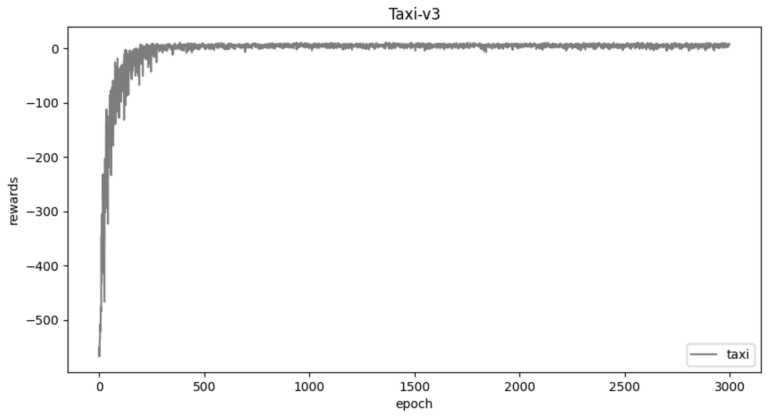
```
📂AI_HW4
 ├ 📜requirements.txt      (for setup)
 ├ 📜plot.py               (for experiment)
 ├ 📜taxi.py               (part1 source code)
 ├ 📜cartpole.py           (part2 source code)
 └ 📜DQN.py                (part3 source code)
```

Implementation (50%)

You will implement some key sections of the Q learning algorithm and its variants. These sections are specified with # Begin your code and # End your code. Please read all the comments to comprehend the source code before implementation. Note that **do not modify** the rest of the code.

Part 1: Q learning in Taxi-v3 (10%)

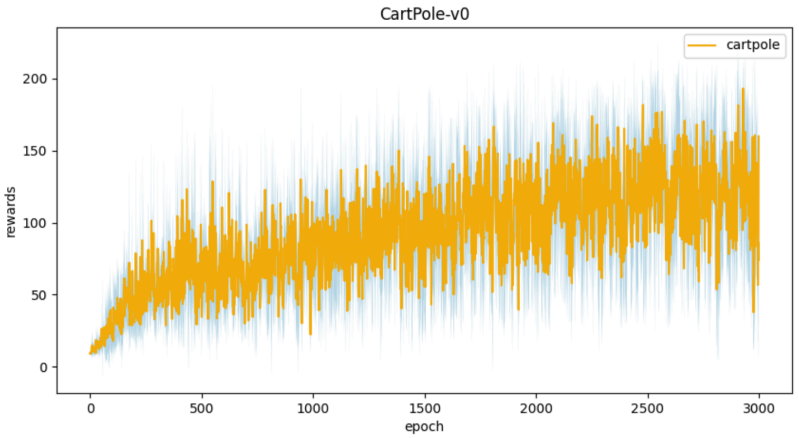
- Implement the "choose_action" function in taxi.py. The function will choose an action according to the given state.
- Implement the "learn" function in taxi.py, one of the most essential parts in a RL algorithm, to decide how an agent updates its Q-table.
- Implement the "check_max_Q" function in taxi.py to calculate the maximum Q value of the initial state.
- The resulting learning curve should look like this:



Part 2: Q learning in CartPole-v0 (15%)

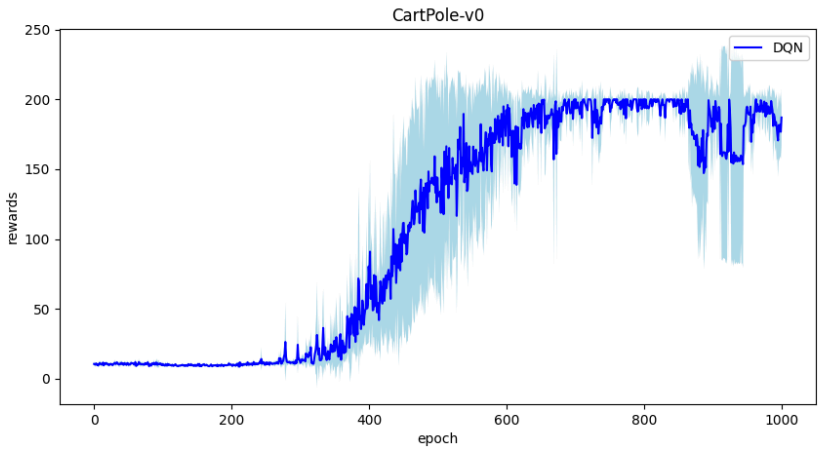
- In this part, we use a method called "discretization" to enable Q learning in CarPole-v0.
- Implement the "init_bins" function in cartpole.py to initiate the bins for discretization.

- Implement the "discretize_value" function in cartpole.py. The function will discretize the value with given bins.
- Implement the "discretize_observation" function in cartpole.py to discretize the observation in continuous space into a discrete state.
- Implement the "choose_action" and "learn" functions in cartpole.py. These functions are similar to those you implemented in part1.
- Implement the "check_max_Q" function in cartpole.py to calculate the maximum Q value of the initial state.
- The resulting learning curve should look like this:



Part 3: DQN in CartPole-v0 (25%)

- Implement the "choose_action" function in DQN.py. The function will choose an action according to the given state.
- Implement the "learn" function in DQN.py, one of the most important parts in an RL algorithm, to decide how an agent updates its networks.
- Implement the "check_max_Q" function in DQN.py to calculate the maximum Q value of the initial state.
- The resulting learning curve should look like this:



Experiment:

After finishing the implementation, you can run the following command to train your agent:

```
python taxi.py / python cartpole.py / python DQN.py
```

During the training process, a .npy/.pt file of the Q table/network will be generated, which will be overwritten every single step. After the training, there will be a test to evaluate your agent, the "test" function will load the .npy/.pt file to test its performance. For the CartPole task, if you want to visualize the process, you can try to add "`env.render()`" in your code.

After testing, a reward record of the training process will be saved as a .npy file for you. You need to use the following command to execute plot.py in order to produce four graphs (taxi.png, cartpole.png, DQN.png, compare.png) showing your training processes:

```
python plot.py [-h] [--taxi] [--cartpole] [--DQN] [--compare]
```

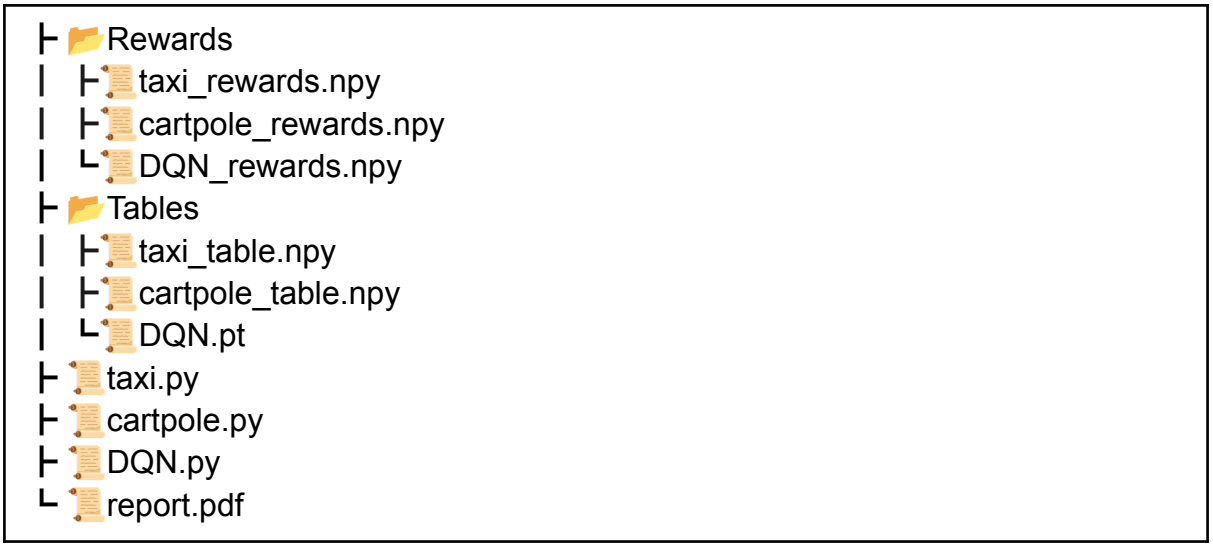
Report (50%)

- A report is required.
- The report should be written in **English**.
- Please save the report as a **.pdf** file. (font size: 12)
- Answer the questions in the report template **in detail**.

Submission

Please prepare your source code, tables, networks, reward records, graphs, and report (.pdf) in `STUDENTID_hw4.zip`, following the file structure. There should **not** be a {student_id}_hw4 folder in the zip file.

```
📂{student_id}_hw4.zip
 ├ 📂Plots
 |  ├ 📜taxi.png
 |  ├ 📜cartpole.png
 |  ├ 📜DQN.png
 |  └ 📜compare.png
```



e.g. 110123456_hw4.zip

Wrong submission format leads to -10 points.

Late Submission Policy

20% off per late day

Reference

1. Taxi-v3 environment 2. CartPole-v0 environment 3. Pytoch DQN tutorial