CSE 326/426 (Fall 2021) Project 4

Due on 11:55pm, Dec 2, 2021

Goal: Implement Q-learning using a table and function approximation to operate a cart.

Instruction:

- Before you start, you will need the following materials: the reinforcement learning lecture notes; the documentation of the gym package for the basic API and the description of the CartPole environment. The latter can be satisfied by reading through this blog:

  https://blog.paperspace.com/getting-started-with-openai-gym/.

  You can install gym on your machine according to

  https://gym.openai.com/docs/

- Now start coding. Decompress the file project_4.zip. There is no input data needed (the reinforcement learning agent collects data by itself). You should have the following file structure:

```
.
|-- data
`-- src
    |-- problem1.py
    |-- problem2.py
    |-- problem3.py
    |-- test1.py
    |-- test2.py
    `-- visualization.ipynb
```

- **Environment**: The environment for the agent to explore and learn from is called "CartPole". A very good video introducing the environment can be found here:

  https://towardsdatascience.com/how-to-beat-the-cartpole-game-in-5-lines-5ab4e738c93f

- Your codes will be added to the following files.

  - problem1.py: implementing Q-learning using a table as the Q function.
  - problem2.py: implementing Q-learning using linear function $Q(s, a) = \mathbf{w}_a^\top \phi(s)$, with $\phi(s)$ a vector representation of the state s.
  - problem3.py: implementing off-policy TD control using Q-learning.

  More detailed instructions are given in the comments of the functions.

- Files test1.py and test2.py will unit-test the correctness of your implementations in the corresponding problem1.py and problem2.py files. For example, after you implement problem1.py file, run

```
nosetests -v test1
```

to test the correctness of your implementation of problem1.py. Note that passing the tests does not mean your implementations are entirely correct: the test can catch only a limited number of mistakes.

If you use Anaconda (highly recommended), there should NOT be any packages you need to install. Otherwise, you will need to install nosestest for such unit test.

- The output of running problem3.py should look similar to the following:

```
************** Environment Info **************
Observation space: Box(4,)
Observation space high values: [4.8000002e+00 3.4028235e+38 4.1887903e-01 3.4028235e+38]
Observation space low values: [-4.8000002e+00 -3.4028235e+38 -4.1887903e-01 -3.4028235e+38]
Action space: Discrete(2)

episode=0
episode=100
episode=200
episode=300
episode=400
episode=500
episode=600
episode=700
episode=800
episode=900
episode=1000
episode=1100
episode=1200
episode=1300
episode=1400
episode=1500
episode=1600
episode=1700
episode=1800
episode=1900
```

After running problem3.py, you can visualize the training and test rewards by running the Jupyter notebook

```
src/visualization.ipynb
```

The rewards and the change in the learning rate and ϵ parameter can be plotted and look like the following figures.

Grading: This project has 100 points in total for all students (graduates and undergraduates). The number of points for each functions in problem1.py and problem2.py are printed below:

```
(10 points) encode_state() ... ok
(10 points) problem1: epsilon_greedy() ... ok
(20 points) problem1:learn() ... ok
(10 points) problem2: epsilon_greedy() ... ok
(20 points) problem2:learn() ... ok
----------------------------------------------------------------------
Ran 5 tests in 0.001s

OK
```

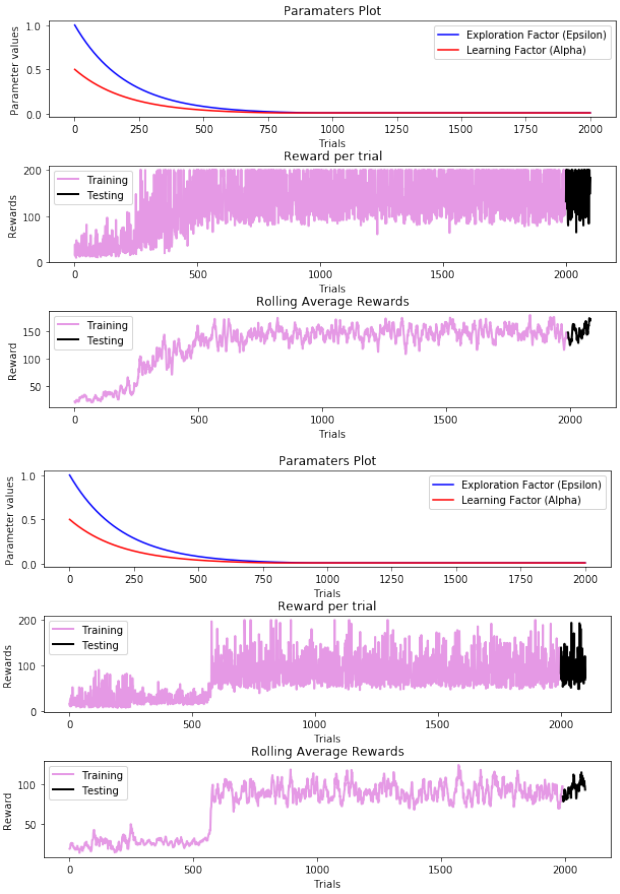

Figure 1: *Top*: results of tabular Q-learning. *Bottom*: results of Q-learning using linear models.

20 points go to the problem3.py implementation and 10 points go to the performance, such as training **speed** and test **total reward**. The project total counts towards 10% of the final grade (40% for all projects).

Submitting: There is no hand-written report required, and your submission should include the ONLY file

`<your_LIN>_P4.zip`

which is the zipped folder that you created when unzipping project4.zip. Retain the src and data folders. Submit the compressed file to Coursesite.