Homework 6

Collaborators:
Name: Chen Zhibo
Student ID: 3190100923

Problem 6-1. A Walk Through Reinforcement Learning

In this problem, you will implement some reinforcement learning algorithms, including Q-learning with table and Q-learning with approximators. You will also get touch with some popular RL techniques and tricks.

Here we use the gym benchmark to do experiments. It provides some handful environments to play with the agents/algorithms you design.

Skeleton code run.ipynb are provided for your convenience. Please see the documentation and comments in the notebook for more details. Also, please report critical results in this report. It should be made that we can judge your assignment without referring to your code (though we may check your code).

(a) Please implement the Q-Learning algorithm with a look-up table.

Answer: This program use Q-learning to solve two complex problem, freeze lake and N-Chain, based on gym environment.
To implement Q-learning, I first complete the qTable.py. Since the Q-learning have the policy that sample the action from the qtable and update the qtable according to the action that have taken. The update function is Bellman Equation,

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha\delta_Q = Q(s_t, a_t) + \alpha(R_{t+1} + \gamma max_a(Q(s_{t+1}, a)) - Q(s_t, a_t))$$

which is exactly what is in function bellman_equation_update.
And it take the action use the $\epsilon - greedy$ policy. It has a probability of ϵ to explore a random choice, and a probability of $1 - \epsilon$ to take a best choice from current qtable. And ϵ reduces at constant rate every step.
The eval part is simply take action just from qtable to see if the table is optimal enough to find a best policy.
After finishing the qTable.py, then it's the parameter adjusting. In the non-slippery scenario, it is quite simple. Just take a rather big epsilon decay rate, I choose epsilon-decay to be 0.9999999. Though in training the average reward is seemingly small, that is because it takes a great many random exploration due to the big epsilon. But it doesn't effect the final test result. Below is the result pic

```
Average reward is 0.98355, average step is 6.04315
[[0.73509189 0.77378094 0.6983373  0.73509189]
 [0.73509189 0.         0.66137577 0.6983299 ]
 [0.69763338 0.82087479 0.         0.        ]
 [0.53713267 0.         0.         0.        ]
 [0.77378094 0.81450625 0.         0.73509189]
 [0.         0.         0.         0.        ]
 [0.         0.90248725 0.         0.        ]
 [0.         0.         0.         0.        ]
 [0.81450625 0.         0.857375   0.77378094]
 [0.81450625 0.9025     0.9025     0.        ]
 [0.857375   0.95       0.         0.8559437 ]
 [0.         0.         0.         0.        ]
 [0.         0.         0.         0.        ]
 [0.         0.9025     0.95       0.857375  ]
 [0.9025     0.95       1.         0.9025    ]
 [0.         0.         0.         0.        ]]
In test, Average reward is 1.0, average step is 6.0
```

Figure 1: qTable and test result

Then is with slippery. Since it has a probability of 0.2 of random action, it is very sensitive to parameters. After several times of trying, I choose $\epsilon - decay - rate = 0.9999995$ $\gamma = 0.991$ $\alpha = 0.769$ and got a relatively good testing result.

```
Average reward is 0.1268, average step is 15.7283
[[2.64280096e-02 1.08664122e-02 1.23254526e-02 1.21887986e-02]
 [2.62357181e-03 1.85093665e-05 1.98750430e-03 7.04028927e-03]
 [6.96077653e-04 1.98487893e-03 1.41925757e-03 2.85698348e-03]
 [1.33181705e-04 6.81074811e-04 3.53269913e-04 1.66093760e-03]
 [3.05436335e-02 3.06868878e-03 4.59192932e-03 4.26647796e-03]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [4.79438837e-02 1.98152476e-06 1.84128003e-06 2.03030447e-05]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [2.96894697e-05 5.78117246e-03 4.94879277e-03 8.60307635e-02]
 [1.22129705e-02 8.73002079e-02 1.39325217e-02 7.14055248e-03]
 [6.06191180e-01 3.34384999e-05 3.37595054e-04 4.67584943e-04]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [4.03526536e-03 5.58241069e-03 1.84762039e-01 4.12026207e-02]
 [8.82969377e-02 8.41539261e-01 1.68699158e-02 7.27015319e-02]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]]
0.011
```

```
]:
total_episode = 1000

qTable.eval(env, total_episode)
#print(qTable.epsilon)
```

```
In test, Average reward is 0.753, average step is 44.379
```

Figure 2: slippery surface

Then is the N-Chain section. I take $\epsilon - decay - rate = 0.999999$ $\gamma = 0.94$ $\alpha = 0.4$ and got the result below:

```
Average reward is 253.86132, average step is 100.0
[[37.10409806 28.40088848]
 [39.2325401  29.49093888]
 [39.42308399 34.62648736]
 [52.93964129 32.64171856]
 [50.82287643 35.44980342]]
0.01
In test, Average reward is 357.054, average step is 100.0
```

Figure 3: N-Chain

(b) (Optional, Homework Bonus) Please implement the Q-Learning algorithm with an approximator with the CartPole environmant.

Answer: ..