

# 2019-1 Deep Learning Homework #6

Fabian Tan Hann Shen (ID 20185112)

June 19, 2019

## 1.

OpenAI gym provides an easy way for people to experiment with the learning agents. Here, FrozenLake environment is used, made up of 4x4 grid of blocks - with the implementation of starting block, goal block, safe block and dangerous block. The objective is to let me agent to learn how to navigate from the starting point to the final goal without getting to the hole. The agent has 4 available moves - up, down, left and right.

In the Q-learning algorithm, the table was initialized with all zeros, and the learning rate( $\alpha$ ) used was 0.85 and the future expected reward( $\gamma$ ) was set as 0.9. The Q-table will be updated based on the computation on the expected reward for given action on the particular state. Action is selected with greedy method from the Q-table, which denotes that the highest state value function is chosen. Lastly, it calculates the average reward per episode at the end of training.

Figure 2 and Figure 3 shows the rewards plot and the total steps taken in 2000 epochs respectively.

The estimated value function from the next action, Q-value equation:

$$Q(s_{t+1}, a) = r + \gamma(\max(Q(s', a'))$$

```
In [28]: print("Final Q-Table Values")
         print(Q)

Final Q-Table Values
[[5.06078105e-02 2.78129657e-03 1.32641419e-03 2.35501307e-03]
 [3.59300326e-04 1.03240259e-05 2.25238076e-04 1.27373092e-03]
 [5.10705052e-04 5.68781442e-04 1.57781192e-02 9.99600739e-04]
 [1.61099335e-04 2.43667299e-05 9.00441655e-05 1.88349323e-03]
 [2.42442141e-01 8.95554958e-04 8.97243814e-04 2.12354729e-03]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [1.27949625e-02 6.66963459e-06 3.46214589e-06 7.83828015e-06]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [4.29512784e-05 4.72794158e-05 2.54026561e-04 1.53492638e-01]
 [1.07764736e-03 3.07216841e-01 3.39677345e-05 4.03003152e-05]
 [8.39487659e-01 0.00000000e+00 5.42826861e-04 1.59717676e-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]
 [0.00000000e+00 0.00000000e+00 1.93253727e-01 0.00000000e+00]
 [0.00000000e+00 8.81276234e-01 3.64678501e-03 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]]
```

Figure 1: shows the Q value for the table

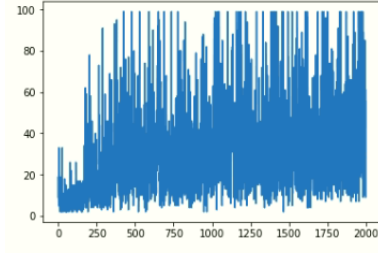


Figure 2: shows the rewards plot for 2000 epochs

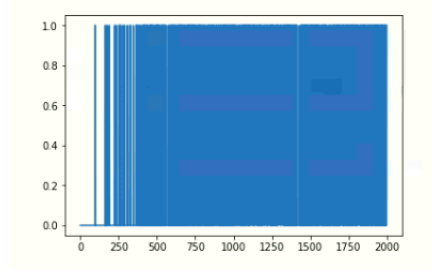


Figure 3: shows the total steps per epochs

Updated value function, Q-value :

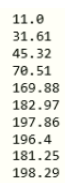
$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(Q(s_{t+1}, a) - Q(s_t, a_t))$$

whereby  $\alpha$  is the learning rate,  $r$  is the reward function,  $\gamma$  is the discount factor  
The loss function:

$$L = \sum (Q_{target} - Q)^2$$

## 2.

It is implemented on Cart-Pole with the policy-based agent. The agent is initialized with state of size 4, and action of 2, along with the learning rate  $10^{-2}$ . It ensures that the gradient can be computed for both possible actions that the network generates. An action is selected based on the probability from the computed network output. Then, the reward is attained from the selected action. The magnitude of each update is preserved by aggregating the gradient. Figure 4 shows the total scores every 500 epochs.



11.0
31.61
45.32
70.51
169.88
182.97
197.86
196.4
181.25
198.29

Figure 4: shows the tally of scores every 500 epochs