Tsinghua-Berkeley Shenzhen Institute
LEARNING FROM DATA
Fall 2019

Programming Assignment 4

Issued: Friday 22nd November, 2019 **Due:** Sunday 8th December, 2019

4.1. (Q learning) Now suppose there is a mouse who is hungry, thereby plans to take an exploration. The *Reward Matrix* it confronts is illustrated in Fig.1 where the blue squares are rat traps, and the red squares are cheese.

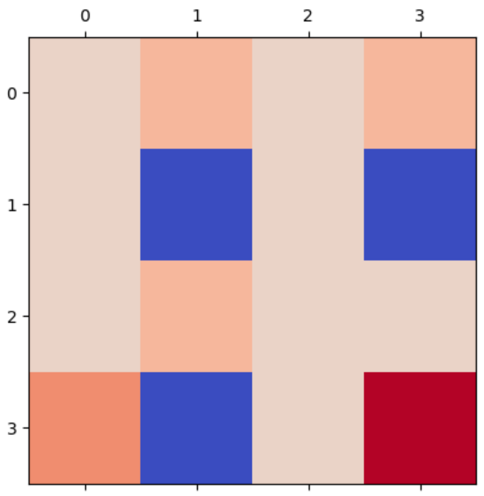

Figure 1: Reward result.

Now we are ought to help it learn from the reward matrix, thus it can eat the cheese on the $(3, 3)$ and avoid traps on $(1, 1)$, $(1, 3)$ and $(3, 1)$. Recall the *Q-learning* you learned from class, the mouse itself has to build a Q-table $Q(s, a)$ where s is the state and a is the action, then it can decide which action to take by this Q-table. Updating this Q-table is commonly performed as algorithm 1:

This algorithm includes hyper parameters as ϵ, α and γ. The ϵ here is used for ϵ-greedy strategy, which means during learning, the mouse has ϵ probability to randomly select an action, or it will select the action with best expected reward; γ, α are discounted rate and learning rate, respectively.

Algorithm 1 Q table learning

```
Input: Initialize $Q(s, a)$ arbitrarily
1: for for each episode do
2:     Initialize $s$
3:     while $s$ is not terminal do
4:         Choose $a$ from $s$ using policy derived from $Q$ (e.g., $\epsilon$-greedy)
5:         Take action $a$, observe $r$, $s'$
6:         $Q(s, a) \leftarrow Q(s, a) + \alpha\,(r + \gamma \max_{a'} Q(s', a') - Q(s, a))$
7:         $s \leftarrow s'$
```

In this homework, you are only required to achieve the *do* and *learn* functions of the mouse. **There are several rules need to be noted**:

1. The mouse at most has **4** alternative actions: left, rigth, up, down, and when it is at the boundary of the grid, it cannot jump out of it.
2. The terminal condition for learning at each step is that the mouse succeeds to eat the largest cheese (reward = 4) or the mouse is trapped (reward = -5).
3. Each cheese can at most be eaten once. That is, after the mouse passes by, the reward becomes 0.

You are encouraged to achieve the following 2 extra questions if you are interested in reinforcement learning (RL). Please save the code for the two extra questions in a different file from the code for the question 4.1.

4.2. (Extra Credits, 2 points) If you have compeleted the question 4.1, can you quantitatively analyze the value distribution of the learned Q-table, such as drawing a matrix which consists of average estimated reward of four actions at each point. Then, evaluate the mean squared error (MSE) between the learned Q-table and the real reward matrix, and analyze which place these two matrices are most different, and try to give your explanation.

4.3. (Extra Credits, 4 points) In practice, we often use what is called *Deep Q-learning* instead of simple Q-table learning. That is, we use a deep neural network to fit the Q-table, with the current state of the agent as its inputs. Then the decision is made by ranking the output of the deep neural network. Can you design a neural network, and perform Deep Q-learning on the given reward matrix named **reward_10_10.npy** ? After that, compare the performance of Deep Q-learning and Q-table learning on this 10×10 grid, show your results and analysis in a pdf and submit it. (Tip: You can use **pytorch** or **tensorflow** to build a deep neural network easily.)

Notice for the question 4.1:

1. You are ought to perform ϵ-greedy, forbid the mouse jumping out of grid as well as ensure the mouse succeeding to reach $(3, 3)$ at last in evaluation step. The correctness of implementation will be taken into account for scoring.
2. Submit your codes and the results in image or pdf.
3. **DO NOT** change other part of codes apart from the given spaces for you to fill, the parameters are carefully tuned in advance.
4. The final result would be like Fig.1 if your code is right.

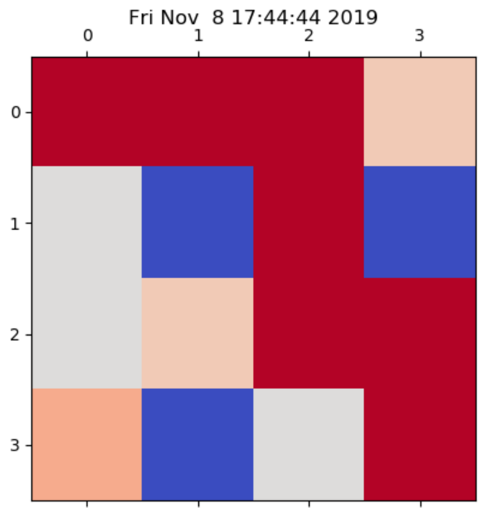

Figure 2: Trajectory result.