Artificial Intelligence Section3

Wendi He 贺文迪 517030910315

1.Q1

This problem is to find the V_{opt} in iteration 0,1 and 2. Firstly, we can use the formula to show the mathematics relationship between each variable.

$$V_{opt}=\begin{cases} 0 & (if\ IsEnd(s)) \\ max_a\, Q_{opt}(s,a) & otherwise \end{cases}$$

$$Q_{opt}=\sum_{s'} T(s,a,s') * [Reward(s,a,s') + \gamma * V_{opt}(s')]$$

At the beginning, $V_{opt}(-2) = V_{opt}(-1) = V_{opt}(0) = V_{opt}(1) = V_{opt}(2) = 0$.

Besides, in every iteration, $V_{opt}(-2) = V_{opt}(2) = 0$ will always hold.

So, we can derive the formula of Q_{opt} for each state and action.

$Q_{opt}(0,-1)$=0.8*[-5+$V_{opt}(-1)$]+ 0.2*[-5+$V_{opt}(1)$]=$0.8V_{opt}(-1) + 0.2V_{opt}(1) - 5$

$Q_{opt}(0,+1)$=0.7*[-5+$V_{opt}(1)$]+ 0.3*[-5+$V_{opt}(-1)$]=$0.3V_{opt}(-1) + 0.7V_{opt}(1) - 5$

$Q_{opt}(1,-1)$=0.8*[-5+$V_{opt}(0)$]+ 0.2*[200+$V_{opt}(2)$]=$0.8V_{opt}(0) + 16$

$Q_{opt}(1,+1)$=0.7*[100+$V_{opt}(2)$]+ 0.3*[-5+$V_{opt}(0)$]=$0.3V_{opt}(0) + 68.5$

$Q_{opt}(-1,-1)$=0.8*[20+$V_{opt}(-2)$]+ 0.2*[-5+$V_{opt}(0)$]=$0.2V_{opt}(0) + 15$

$Q_{opt}(-1,+1)$=0.7*[-5+$V_{opt}(0)$]+ 0.3*[20+$V_{opt}(-2)$]=$0.7V_{opt}(0) + 2.5$

Then, we can derive the formula of V_{opt} for each state.

$V_{opt}(-1)$=max$\{0.2V_{opt}(0) + 15, 0.7V_{opt}(0) + 2.5\}$

$V_{opt}(0)$=max$\{0.8V_{opt}(-1) + 0.2V_{opt}(1) - 5, 0.3V_{opt}(-1) + 0.7V_{opt}(1) - 5\}$

$V_{opt}(1)$=max$\{0.8V_{opt}(0) + 16, 0.3V_{opt}(0) + 68.5\}$

In iteration 0,$V_{opt}(-2) = 0, V_{opt}(-1) = 0, V_{opt}(0) = 0, V_{opt}(1) = 0, V_{opt}(2) = 0.$

In iteration 1,$V_{opt}(-2) = 0, V_{opt}(-1) = 15, V_{opt}(0) = -5, V_{opt}(1) = 68.5, V_{opt}(2)$=0

In iteration 2,$V_{opt}(-2) = 0, V_{opt}(-1) = 14, V_{opt}(0) = 47.45, V_{opt}(1) = 67, V_{opt}(2)$=0

2.Q2

We can use Sarsa learning method to solve this cliff walk problem. Q-value matrix is one of the most important part in this algorithm, which records the q-value for every directions of each states and it is also keep updating after each cycle. The self.map matrix records the cliff, start, terminal, barrier and reward in the graph.

Sarsa.learning function is the key part of the whole code. First, we define gamma for the discount factor and max_episode_num for total episode num. In each episode, the agent is learning to find its way to reach the maximum reward and look for the optimal policy for every state, also the q-value will be updated in the same time.

Since we are facing the dilemma of exploration and exploitation, so we need to explore more at the beginning while exploit more in the end. So, we sdjust the value of epsilon and use random number to solve this dilemma. Besides, to avoid the agent keep bouncing to the wall or lead to a place that it is not allowed to go to, such as barriers and boundaries, I design the code to ban such action, which means if this direction is barriers and boundaries, then the action to this direction will not be possible.

The agent will keep moving on the graph until it each the terminal state. In each movement, a instant reward or punishment will be received. And the agent will find its next action by

comparing the largest q-value for the four direction (if it is allowed) according to the q-value matrix. So, the next state of the agent will be determined by its current location and the action it choose. Besides, the q-value matrix will be updated according to the Bellman equation.

The picture “qtable.png” is the final result of the program, which directly shows the q-value for every directions of each state after several times of iteration. Besides, it also marks the optimal direction policy in color gray for every state.

The detail of the algorithms can be seen in the python file.