Assignment 1

Zirong Chen

# Question 1

Value after 100 iterations (python gridworld.py -a value -i 100 -k 10):

From the values after 100 iterations, we can tell that the top right one has the biggest value, which means reaching this spot gets highest value, so the agent is encouraged to reach that spot. In this case, the top right spot can be the “exit” or “end point” in this game. And on the opposite, the one below the top right one has the negative value and is likely to discourage the agent from reaching it. So, the greener one state gets, the more the agent will be encouraged to reach.

Chart, treemap chart

Description automatically generated

Q Value after 100 iterations (python gridworld.py -a value -i 100 -k 10 + enter):

From the Q-value results, each Q-value can only be gotten by giving (state, action) pairs. In this case, each state has four possible actions, thus, each state is supposed to have four Q-value corresponding to four different actions (except ending states). So any action points to the red state is supposed to have negative (or relatively low Q-value), since taking that action means to reach the ‘worst’ state.

Shape

Description automatically generated

Avg return after 10 episodes:

Text

Description automatically generated

# Question 2

Here is my setting: answerDiscount = 0.9; answerNoise = 0.002

From the results, the agent will be encouraged to take the green path over red ones.

A picture containing graphical user interface

Description automatically generated

# Question 3

**a. Prefer the close exit (+1), risking the cliff (-10)**

answerDiscount: 0.01; answerNoise: 0.0; answerLivingReward = 0.0

**b. Prefer the close exit (+1), but avoiding the cliff (-10)**

answerDiscount: 0.1; answerNoise: 0.1; answerLivingReward = 0.5

Several attempts made to avoid being subject to any noise, but none of them turns out to be effective, so there I just reported parameter settings that work under this requirement.

**c. Prefer the distant exit (+10), risking the cliff (-10)**

answerDiscount: 0.9; answerNoise: 0.0; answerLivingReward = -0.1

**d. Prefer the distant exit (+10), avoiding the cliff (-10)**

answerDiscount: 0.9; answerNoise: 0.1; answerLivingReward = 0.5

Several attempts made to avoid being subject to any noise, but none of them turns out to be effective, so there I just reported parameter settings that work under this requirement.

**e. Avoid both exits and the cliff (so an episode should never terminate)**

answerDiscount: 0; answerNoise: 0; answerLivingReward = 100

# Question 4

To get the same result shown in the requirement, go to the top left and turn to the top right. Do it three times.

# Question 5

From top to bottom:

python gridworld.py -a q -k 100

python gridworld.py -a q -k 100 –noise 0.0 -e 0.3

python gridworld.py -a q -k 100 –noise 0.0 -e 0.1

python gridworld.py -a q -k 100 –noise 0.0 -e 0.9

|  |  |
| --- | --- |
| Shape  Description automatically generated | Chart  Description automatically generated |
| Shape  Description automatically generated | Chart  Description automatically generated |
| Shape  Description automatically generated | Chart, treemap chart  Description automatically generated |
|  |  |
|  |  |
|  |  |

# Question 6

1. Step delay: control the delay time interval between steps
2. Epsilon: control the epsilon-greedy exploration.
   1. If the epsilon is set too high, it might make some unreasonable steps to make the crawling harder.
   2. If the epsilon is set too low, the crawler might be trapped in some strange position for some time.
3. Discount: control the effects of reward-to-go on current decisions.
   1. If the discount is set too high, the crawler might be trapped in some position for a while. However, in some way, the agent will be more ‘insightful’.
   2. If the discount is set too low, the crawler might be considered as ‘short-sighted’ and would prefer to make decisions have higher current rewards.
4. Learning rate: the step size in gradient descent.
   1. If the learning rate is set too high, it might miss some minimum/maximum and the results will zigzag a lot, which means a more unstable learning process.
   2. If the learning rate is set too low, it might take a longer time to train.

# Question 7

python gridworld.py -a q -k 50 -n 0 -g BridgeGrid -e 1

Epsilon=1: The agent will keep making random decisions despite a better one is given (explore too much).

A picture containing diagram

Description automatically generated

Epsilon=0: The agent will keep returning to the nearest exit despite higher rewards might be achieved after crossing the bridge (no exploration).

Diagram

Description automatically generated

answerEpsilon = None; answerLearningRate = None => NOT POSSIBLE

# Question 8

2000 episodes of training took around 15s on my PC and the actual play time on smallGrid took around 40s. Here is the result:

Text

Description automatically generated

2000 episodes of training took around 30s on my PC and the actual play time on mediumGrid took around 65s. Here is the result:

Text

Description automatically generated

Comparison between two grids with same number of eposides:

|  |  |  |
| --- | --- | --- |
|  | smallGrid | mediumGrid |
| Episode took [.] seconds | ~0.6s | ~1s |
| Avg rewards over all training(eventually) | -82.42 | -451.16 |
| Avg rewards for last 100 episodes | 206.94 | -456.68 |
| Play records | 10W-0L | 1W-9L |

However, I also tried to extend the training process to make the agent learn better in mediumGrid. If the number of episodes reaches around 15,000, the agent will get a similarly competitive result as smallGrid one.

Comparison between two grids until better policy is generated(see image next page):

|  |  |  |
| --- | --- | --- |
|  | smallGrid | mediumGrid |
| Episode took [.] seconds | ~0.6s | ~0.8s |
| Avg rewards over all training(eventually) | -82.42 | -200.10 |
| Avg rewards for last 100 episodes | 206.94 | 171.72 |
| Total training time | ~1min | ~2.5min |
| Play records | 10W-0L | 10W-0L |

Text

Description automatically generated