1. Bellman Equation

1.

2.

=

=

=

3.

4.

5. a.

|  |  |  |
| --- | --- | --- |
| Q-values | a1 | a2 |
| S1 | -5 | 0 |
| S2 | 0 | 0 |

b.

|  |  |  |
| --- | --- | --- |
| Q-values | a1 | a2 |
| S1 | -5 | -5 |
| S2 | 0 | 0 |

c.

|  |  |  |
| --- | --- | --- |
| Q-values | a1 | a2 |
| S1 | -5 | -5 |
| S2 | 9.25 | 0 |

d.

|  |  |  |
| --- | --- | --- |
| Q-values | a1 | a2 |
| S1 | -5 | -7.6875 |
| S2 | 9.25 | 0 |

Therefore: Thus, Both are policy a1 for S1 and S2.

2. Combinational Lock

1. Since ,

.

2.

, for i<n

, for i<n

3. For i<n, although both a1 or a2 yields no immediate rewards, but a1 pushes agent toward the target, so it will accumulate more rewards in the future. Thus, the agent should always choose a1 for i<n. Therefore, under the greedy policy, the expected number of steps to reach sn from s1 is exactly n-1 steps.

3. Q-value Initialization

1. For Q1, since the choice are purely random at first, the steps of reaching sn will be very large and random. As learning progresses, the policy will favorite actions that leads toward sn, and thus the agent will learn the optimal path.

For Q2, the steps of reaching sn should be less than that of Q1, since the optimum reward can promote the agent to find the best policy toward the target.

2. Since each state’s initial reward is 0 and if the agent makes the optimal action at each step, the expected number of steps of reaching sn starting from s1 is n-1.

3. This time, the agent might experience more steps when reaching sn for first several times or episodes. After that, the agent will follow the learned optimal policy for the following episodes and reach sn with less steps.

4. Replay buffer technique can store the past or experienced states or process into the buffer and the agent in the new episode can use them to help it choosing the best policy, which is helpful to reduce the episodes number for reaching the optimal policy.

5. The minimum step is determined by the shortest path between s1 and sn if the learning conditions or policies are optimal and ideal from the start state. However, the maximum states are very large since the agent might consider each actions for each states from s1 to sn.

4. Policy Gradient

1.

.

Since ,

.

Therefore, .

2. Since ,

.

3. Since the expectation of log-probability weighted by any functions of actions that integrates to 1 over all actions equals to 0, , and thus:

.

Hence,

5. Tabular Q-learning:

1.

a. States in Taxi-v3 represents the combination of the taxi’s location, passenger’s location and destination of the passenger. There are 500 possible states and the action space is:

0: move south

1: move north

2: move east

3: move west

4: pickup passenger

5: drop off passenger

b. env.step() will return a tuple of (next state, reward, done, info), and env.reset() will return an random resetting state as a starting state. States are represented by the combination of the taxi’s location, passenger’s location and destination of the passenger.

c. valid render moves including “human” and “ansi”. “human” renders the environment on the current screen and “ansi” renders the environment as a string that can be printed or logged.

d. self.\_render\_text(self.s).

2.

A graph of blue lines

Description automatically generated

3. It will negatively affect the performance of the Q-learning since it will render agent spending a longer time to reach the expected-reward.

4. The rate becomes very low now which is 0.0466 for my agent.

A graph with blue lines

Description automatically generated

5.

A graph with blue lines

Description automatically generated

6. The percentage becomes higher than before but still not too high.

A graph of blue lines

Description automatically generated

7.

A graph with blue lines

Description automatically generated