Reinforcement Learning Assignment Report

Li Kaiyu (S220048)

AI6101 School of Computer Science and Engineering Nanyang Technological University

2023

Background

In this assignment, we are required to choose one of RL algorithms to solve a specific CliffBoxPushing grid-world game. The game map is shown in figure 1. In this game, the agent 'A' tries to 'push' the box 'B' to the goal 'G' as quick as possible while avoiding the cliff which is denoted as 'x' on the map.

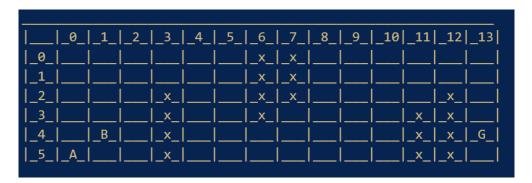


Figure 1: Game Map

Algorithm Implementation

Q-Learning

We implement the Q-Learning algorithm to solve the game. Q-learning is one of the fundamental RL algorithms that allows agents to learn optimal policies without the need of transition probability distribution. It works based on the concept of a Q-table, keeping updating $Q_{\pi}(s,a)$ (which is the expected reward if action a is taken at state s under policy π for every states and corresponding action sets.

Now we briefly describe the Q-Learning method. Before training, we initialize an all-zero Q-table and a ε -soft policy as follow:

$$\pi(s,a) = \begin{cases} 1 - \varepsilon + \varepsilon/|A(s)|, & \text{if } a = a^*. \\ \varepsilon/|A(s)|, & \text{if } a \neq a^*. \end{cases}$$
 (1)

where |A(s)| is the size of action set at state s. Then at each iteration, an episode is generated using π , and at each step in each episode, we update the Q-table using the following updating rule:

$$Q_{new}(S_t, A_t) = Q_{old}(S_t, A_t) + \alpha (R_{t+1} + \gamma max_a Q_{old}(S_{t+1}, a) - Q_{old}(S_t, A_t))$$
(2)

where α is the learning rate, and γ is the discount factor to reduce future rewards to help reaching convergence.

Epsilon Decay

During the experiments we find out that the algorithm finds out a policy that leads to the goal at around 6000-7000 episodes, so we set the total episode number to be 8000.

However, with our ε -soft policy, the agent may still choose wrong policies even the global optimal policy has been found thus make the result unstable. In other words, at early stage of training, we prefer relatively large ε to help find potentially better policies; while at the late stage of training, we prefer our policy to converge at optimal. Therefore, we apply a **linear decay coefficient** to gradually reduce ε during training and acquire much better convergence.

Result Visualization

Learning Process

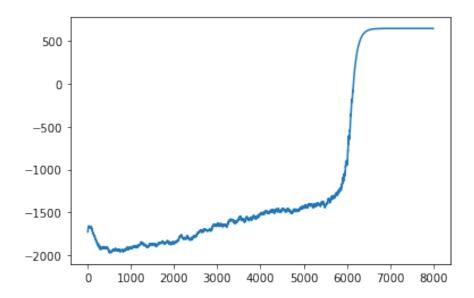


Figure 2: Smoothed learning curve: episode rewards vs. episodes

V-table

```
V table:
                                                                             10
                                                                                    11
                                                                                            12
  -39.55 -39.91 -40.24 -40.48 -38.10 -43.86
                                              0.00
                                                      0.00 -36.36 -31.09 -30.40 -29.28 -25.32 -23.81
 -37.69 -35.42 -36.11 -42.27 -36.68 -42.32
                                              0.00
                                                      0.00 -36.37 -28.92 -28.09 -27.24 -30.99 -23.40
                         0.00 -44.94 -42.65
                                                      0.00 -37.55 -29.94 -29.41 -40.89
 -38.03 -35.79 -43.36
                                              0.00
                                                                                         0.00 -27.65
                                              0.00 -44.31 -34.11 -33.55 -34.70
  -40.29 -35.67 -42.88
                         0.00 -41.98 -41.78
                                                                                          0.00 -29.44
  -40.31 -35.46 -42.25
                         0.00 -40.58 -35.22 -39.79 -33.15 -32.05 -31.38 -35.54
                                                                                  0.00
  -38.19 -38.24 -44.31
                         0.00 -43.83 -37.85 -37.25 -36.24 -35.31 -31.94 -36.10
                                                                                  0.00
                                                                                         0.00 -27.18
```

Figure 3: V-table

Policy

```
step: 1, state: (5, 1, 4, 1), actions: 4, reward: -14
Action: 4
step: 2, state: (4, 1, 3, 1), actions: 1, reward: -15
Action: 1
step: 3, state: (3, 1, 2, 1), actions: 1, reward: -16
Action: 1
step: 4, state: (2, 1, 1, 1), actions: 1, reward: -17
Action: 1
step: 5, state: (2, 0, 1, 1), actions: 3, reward: -18
Action: 3
'b'_
        _' b'x' b'x' b'_
step: 6, state: (1, 0, 1, 1), actions: 1, reward: -17
Action: 1
```

```
step: 7, state: (1, 1, 1, 2), actions: 4, reward: -16
Action: 4
step: 8, state: (1, 2, 1, 3), actions: 4, reward: -15
Action: 4
step: 9, state: (1, 3, 1, 4), actions: 4, reward: -14
Action: 4
step: 10, state: (1, 4, 1, 5), actions: 4, reward: -13
Action: 4
step: 11, state: (0, 4, 1, 5), actions: 1, reward: -14
Action: 1
[[b' 'b' 'b' 'b' 'b'A' b' 'b'x'b'x'b' 'b' 'b' 'b' 'b' 'b' ']
```

```
step: 12, state: (0, 5, 1, 5), actions: 4, reward: -13
Action: 4
step: 13, state: (1, 5, 2, 5), actions: 2, reward: -12
Action: 2
[b'_' b'_' b'_' b'_' b'_' b'A' b'x' b'x' b'_' b'_' b'_' b'_' b'_']
[b' 'b' 'b' 'b'x'b' 'b'B'b'x'b'x'b' 'b' 'b' 'b' 'b' 'b' 'b'
step: 14, state: (2, 5, 3, 5), actions: 2, reward: -11
Action: 2
_' b'_' b'_' b'x' b'_' b'A' b'x' b'x' b'_' b'_' b'_' b'_' b'_'
[b' 'b' 'b' 'b'x'b' 'b'B'b'x'b' 'b' 'b' 'b' 'b' 'b'x'b'x'b'']
 step: 15, state: (3, 5, 4, 5), actions: 2, reward: -10
Action: 2
[b'_' b'_' b'_' b'x' b'_' b'A' b'x' b'_' b'_' b'_' b'_' b'x' b'x' b'_']
step: 16, state: (3, 4, 4, 5), actions: 3, reward: -11
Action: 3
[b' 'b' 'b' 'b'x'b' 'b'B'b' 'b' 'b' 'b' 'b' 'b' 'b'x'b'x'b'G']
step: 17, state: (4, 4, 4, 5), actions: 2, reward: -10
Action: 2
```

```
[b'_' b'_' b'_' b'x' b'A' b'B' b'_' b'_' b'_' b'_' b'_' b'x' b'x' b'G']
step: 18, state: (4, 5, 4, 6), actions: 4, reward: -9
Action: 4
' b'_' b'_' b'x' b'_' b'A' b'B' b'_' b'_' b'_' b'_' b'x' b'x' b'G']
step: 19, state: (4, 6, 4, 7), actions: 4, reward: -8
Action: 4
[b'_' b'_' b'_' b'x' b'_' b'A' b'B' b'_' b'_' b'_' b'x' b'x' b'G']
step: 20, state: (4, 7, 4, 8), actions: 4, reward: -7
Action: 4
[b' 'b' 'b' 'b'x'b'_'b'_'b'_'b'A'b'B'b'_'b'_'b'x'b'x'b'G']
step: 21, state: (4, 8, 4, 9), actions: 4, reward: -6
Action: 4
step: 22, state: (4, 9, 4, 10), actions: 4, reward: -5
Action: 4
step: 23, state: (5, 9, 4, 10), actions: 2, reward: -6
Action: 2
```

```
step: 24, state: (5, 10, 4, 10), actions: 4, reward: -5
step: 25, state: (4, 10, 3, 10), actions: 1, reward: -6
Action: 1
step: 26, state: (3, 10, 2, 10), actions: 1, reward: -7
Action: 1
step: 27, state: (2, 10, 1, 10), actions: 1, reward: -8
Action: 1
step: 28, state: (2, 9, 1, 10), actions: 3, reward: -9
Action: 3
[b' 'b' 'b' 'b'x'b' 'b'x'b'x'b' 'b'A'b' 'b'x'b'']
  _' b'_
step: 29, state: (1, 9, 1, 10), actions: 1, reward: -8
Action: 1
```

```
step: 30, state: (1, 10, 1, 11), actions: 4, reward: -7
Action: 4
step: 31, state: (1, 11, 1, 12), actions: 4, reward: -6
Action: 4
step: 32, state: (1, 12, 1, 13), actions: 4, reward: -5
Action: 4
step: 33, state: (0, 12, 1, 13), actions: 1, reward: -6
Action: 1
step: 34, state: (0, 13, 1, 13), actions: 4, reward: -5
Action: 4
```

```
step: 35, state: (1, 13, 2, 13), actions: 2, reward: -4
Action: 2
step: 36, state: (2, 13, 3, 13), actions: 2, reward: -3
Action: 2
step: 37, state: (3, 13, 4, 13), actions: 2, reward: 998
Action: 2
```