# AI6101 Reinforcement Learning Assignment

### Chen Lei G2202273D Nanyang Technological University

October 16, 2023

#### Abstract

This assignment implements a game agent to solve the CliffBoxPushing gridworld game, using Q-Learning algorithm. The Q-Learning algorithm adapt cosine annealing scheduler to decreased the  $\epsilon$  parameter, which proves to be effective to get a relative good result.

# **Q-Learning**

In this assignment, I choose Q-Learning algorithm to train the game agent. It is a model-free algorithm learning optimal policies based on reward observed when an action is taken at a specific state. The results(Q-values) are updated in a state-action table called Q-table. The Q-learning algorithm can be formulated as:

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

where  $Q_{new}(S_t,A_t)$  and  $Q_{old}(S_t,A_t)$ ) refers to the new and old estimation when taking action  $A_t$  at state  $S_t$ , and  $\alpha$  and  $\gamma$  is the learning rate and  $\gamma$  is the discount parameter.

Also, in order to encourage the agent to explore more available paths, the Epsilon Greedy Method is used to enable agent to take actions randomly. However, Q-value finally converges to a consistent value, so I uses cosine annealing scheduler, which can be formulated as:

$$\eta_t = \eta_{min}^i + \frac{1}{2} (\eta_{max}^i - \eta_{min}^i) (1 + \cos(\frac{T_{cur}}{T_i}\pi))$$
 (2)

where  $\eta^i_{max}$  is the largest  $\epsilon$  and  $\eta^i_{min}$  is the minimum  $\epsilon(0$  in this scenario) and  $T_{cur}$  is the current episode during training.

# **Implementation and Training**

In this assignment, the agent is trained for 10,000 episodes. I have tried to use scheduler on  $\epsilon$  and learning rate  $\alpha$ , which is shown in Figure 1. It seems that using cosine annealing scheduler can not improve the rewards, even worse, the curve keeps vibrating at a

quite reward of -1,602 while the default Q-Learning agent gains the reward of -889. However, the decay of  $\epsilon$  seems to be helpful as the agent succeeds to converge after trained for nearly 6,000 episodes and gain 642 rewards at last. The final rewards are shown in table 1.

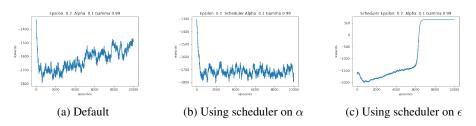


Figure 1: Episodes vs Rewards with three agents

Table 1: Rewards of three agents

Agent	Reward
Default	-889
Using $\alpha$ Scheduler	-1,602
Using $\epsilon$ Scheduler	642

After adapting cosine annealing scheduler on  $\epsilon$  parameter during the training, I try to find a proper initial  $\epsilon$  value. The experiments are conducted with 9 different  $\epsilon$  value varying from 0.1 to 0.9 adding 0.1 each time. The episode vs reward curves of these experiments has little difference as figure3, which may means that a random initial  $\epsilon$  value is suitable when using cosine annealing scheduler.

#### Result

After conducting the above experiments, I decide to train my agent for 10,000 episodes with initial  $\alpha=0.1, \gamma=0.99$  and  $\epsilon=0.2$ . I use V-table to show the best value of each state(here is the position of agent in this game) when applying the policy. The V-table is as the Figure2:

```
V table:
0 [-40.02, -40.31, -40.68, -40.94, -38.61, -44.45, 0.0, 0.0, -37.15, -31.7, -30.9, -29.78, -25.79, -24.35]
1 [-38.06, -35.75, -36.46, -44.64, -37.13, -43.3, 0.0, 0.0, -36.82, -29.51, -28.6, -27.73, -31.38, -23.94]
2 [-38.27, -36.83, -45.19, 0.0, -46.44, -44.03, 0.0, 0.0, -38.52, -30.51, -29.9, -41.43, 0.0, -28.37]
3 [-40.46, -35.84, -47.75, 0.0, -43.72, -43.6, 0.0, -44.51, -34.68, -34.11, -35.65, 0.0, 0.0, -30.22]
4 [-40.47, -37.27, -45.19, 0.0, -41.21, -35.8, -40.59, -33.72, -32.64, -31.97, -36.26, 0.0, 0.0, -28.71]
5 [-38.31, -38.35, -49.77, 0.0, -45.02, -38.48, -37.84, -36.82, -35.88, -32.53, -37.12, 0.0, 0.0, -27.63]
```

Figure 2: V-table

The agent can get the reward of 642 at last with the behavior history of [4, 1, 1, 1, 3, 1, 4, 4, 4, 1, 4, 2, 2, 2, 3, 2, 4, 4, 4, 4, 4, 2, 4, 1, 1, 1, 3, 1, 4, 4, 4, 1, 4, 2, 2, 2] where 1, 2, 3, 4 refers to left, right, down and up. The policy is shown in the appendix result policy.

# **Appendix**

### Episode vs Rewards with different $\epsilon$ value

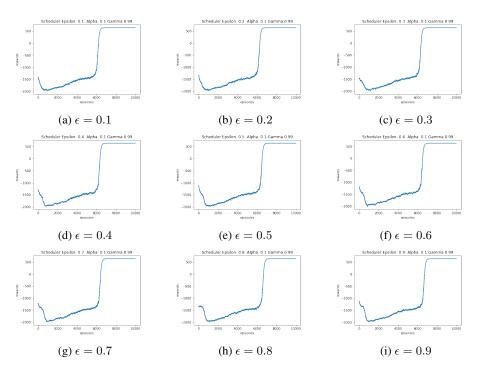


Figure 3: Episodes vs Rewards with different initial  $\epsilon$  value

### **Result Policy**

```
step: 2, state: (5, 1, 4, 1), actions: 1, reward: -15
Action: 1
_ ' b'A' b'_ ' b'x' b'_ ' b'x' b'X' b'G']
[b'
step: 3, state: (4, 1, 3, 1), actions: 1, reward: -16
Action: 1
step: 4, state: (3, 1, 2, 1), actions: 1, reward: -17
Action: 1
[b, ', p, B, p, ', p, ', p, ', p, ', p, x, p, x, p, ', p, ',
[b'
step: 5, state: (2, 1, 1, 1), actions: 3, reward: -18
Action: 3
step: 6, state: (2, 0, 1, 1), actions: 1, reward: -17
Action: 1
step: 7, state: (1, 0, 1, 1), actions: 4, reward: -16
Action: 4
```

```
step: 8, state: (1, 1, 1, 2), actions: 4, reward: -15
Action: 4
[b'.
step: 9, state: (1, 2, 1, 3), actions: 4, reward: -14
Action: 4
[b'_' b'_
 , b,
step: 10, state: (1, 3, 1, 4), actions: 4, reward: -13
Action: 4
step: 11, state: (1, 4, 1, 5), actions: 1, reward: -14
Action: 1
step: 12, state: (0, 4, 1, 5), actions: 4, reward: -13
[b'.
step: 13, state: (0, 5, 1, 5), actions: 2, reward: -12
Action: 2
```

```
[b'_' b'_' b'_' b'x' b'_' b'B' b'x' b'x' b'_' b'_' b'_' b'_' b'x' b'.
[b'.
step: 14, state: (1, 5, 2, 5), actions: 2, reward: -11
Action: 2
[b'_' 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'_'
step: 15, state: (2, 5, 3, 5), actions: 2, reward: -10
Action: 2
'b'_'b'
[b'_
step: 16, state: (3, 5, 4, 5), actions: 3, reward: -11
Action: 3
step: 17, state: (3, 4, 4, 5), actions: 2, reward: -10
Action: 2
step: 18, state: (4, 4, 4, 5), actions: 4, reward: -9
Action: 4
[b'_' b'_' b'_' b'x' b'_' b'A' b'B' b'_' b'_' b'_' b'_' b'x' b'x' b'G']
step: 19, state: (4, 5, 4, 6), actions: 4, reward: -8
```

```
Action: 4
[b'_' b'_' b'_' b'x' b'_' b'_' b'A' b'B' b'_' b'_' b'_' b'x' b'x' b'G']
step: 20, state: (4, 6, 4, 7), actions: 4, reward: -7
Action: 4
step: 21, state: (4, 7, 4, 8), actions: 4, reward: -6
Action: 4
step: 22, state: (4, 8, 4, 9), actions: 4, reward: -5
Action: 4
step: 23, state: (4, 9, 4, 10), actions: 2, reward: -6
Action: 2
step: 24, state: (5, 9, 4, 10), actions: 4, reward: -5
Action: 4
```

```
step: 25, state: (5, 10, 4, 10), actions: 1, reward: -6
Action: 1
[p, ', p, ', p, ', p, x, p, ', p, x, p, x,
step: 26, state: (4, 10, 3, 10), actions: 1, reward: -7
Action: 1
step: 27, state: (3, 10, 2, 10), actions: 1, reward: -8
Action: 1
[b'
step: 28, state: (2, 10, 1, 10), actions: 3, reward: -9
Action: 3
step: 29, state: (2, 9, 1, 10), actions: 1, reward: -8
Action: 1
step: 30, state: (1, 9, 1, 10), actions: 4, reward: -7
Action: 4
```

```
step: 31, state: (1, 10, 1, 11), actions: 4, reward: -6
Action: 4
_ ' b'_ ' b'_ ' b'x ' b'_ ' b'_ ' b'x ' b'x ' b'_ ' b'_ ' b'_ ' b'_ ' b'_ ' b'_ ']
[b'.
[b'-', b'-', b'
  step: 32, state: (1, 11, 1, 12), actions: 4, reward: -5
Action: 4
[b'_' b'_
 , b,
step: 33, state: (1, 12, 1, 13), actions: 1, reward: -6
Action: 1
step: 34, state: (0, 12, 1, 13), actions: 4, reward: -5
Action: 4
step: 35, state: (0, 13, 1, 13), actions: 2, reward: -4
[b'.
step: 36, state: (1, 13, 2, 13), actions: 2, reward: -3
Action: 2
```