*National University of Singapore*

*School of Electrical and Computer Engineering*

# CA2 Project 2: Q-Learning for World Grid Navigation

*Student:*
Wanry Lin

*Matriculation Number:*
A0000001X

*Email address:*

*EE5904 Neural Network*

May 10, 2023

# $\varepsilon$-greedy Q-learning

I imply the Q-learning algorithm, using the reward function and with the $\varepsilon$ -greedy exploration algorithm by setting specific $\varepsilon_k, \alpha_k, \gamma$. The Q-function serves as a metric to evaluate the value of a state-action pair $(s,a)$ in relation to the rewards obtained when an agent performs a task. An optimal policy is achieved by maximizing the Q-function values, which are computed for all possible $(s,a)$ pairs with respect to the task at hand. In order to obtain the Q-function, the following steps are taken:

(1) **Initialize parameter:** Discount factor $\gamma$; exploration probability $\varepsilon_k$; learning rate $\alpha_k$.

(2) Initialize Q-function.

(3) Determine the initial state $s_0$.

(4) For time step $k$, select action $\alpha_k$ according to:

$$a_k = \begin{cases} a \in \arg\max_{\hat{a}} Q_k(s_k, \hat{a}) & \text{with probability } 1-\varepsilon_k \\ \text{an action uniformly randomly selected from all actions available at state } s_k & \text{with probability } \varepsilon_k \end{cases}$$

(5) Apply action $a_k$, receive reward $r_{k+1}$, then observe next state $s_{k+1}$.

(6) Update Q-function using Bellman equation:

$$Q_{k+1}(s_k, a_k) = Q_k(s_k, a_k) + \alpha_k(r_{k+1} + \gamma \max_{a'} Q_k(s_{k+1}, a') - Q_k(s_k, a_k));$$

(7) Set $k = k+1$ and repeat for next loop.

(8) Break the loop if one of these conditions are reached:

- Robot reaches goal state $s_k = 100$
- $\alpha_k < 0.005$ (optional)
- Maximum number of time step $k_{max} = 3000$ is reached.

(9) Run 10 times from (2) to (8) and obtain a result of a set of fixed parameter.

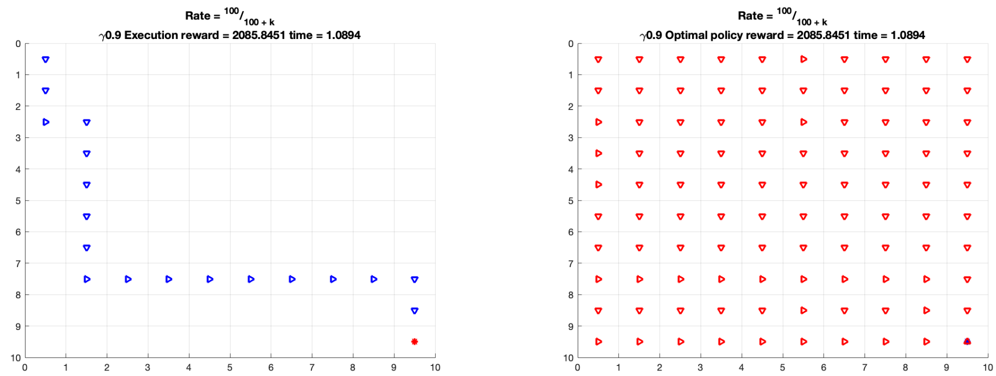
# Task 1

Here, I set the random seed as 5904. The performance of task 1 is show:

Table 1: Parameter values and performance of Q-Learning

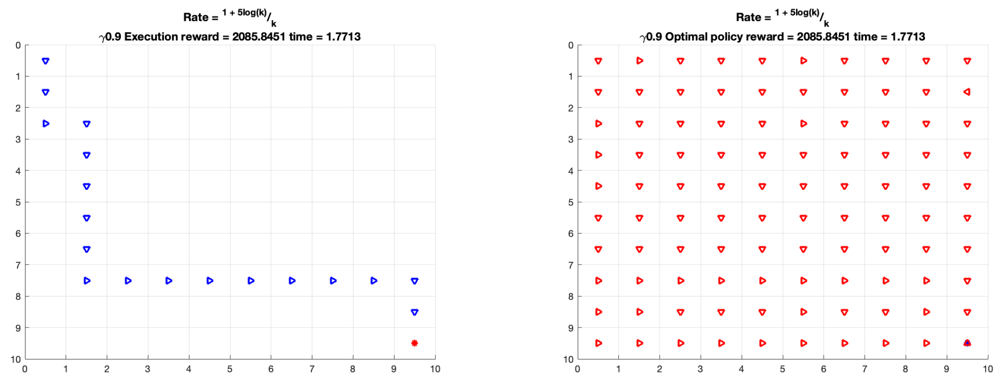
| $\varepsilon_k$, $\alpha_k$ | **No. of goal-reached runs** | | **Execution time (sec.)** | |
|---|---|---|---|---|
| | $\gamma = 0.5$ | $\gamma = 0.9$ | $\gamma = 0.5$ | $\gamma = 0.9$ |
| $\frac{1}{k}$ | 0 | 0 | N.A. | N.A. |
| $\frac{100}{100+k}$ | 0 | 9 | N.A. | 1.089 |
| $\frac{1+\log(k)}{k}$ | 0 | 0 | N.A. | N.A. |
| $\frac{1+5\log(k)}{k}$ | 0 | 10 | N.A. | 1.7712 |

The optimal policies and optimal paths are shown in Fig. 1-2.




(a) Optimal path (b) Optimal policy

Figure 1: Performance of $\gamma = 0.9$ and $\varepsilon_k = \frac{100}{100+k}$




(a) Optimal path (b) Optimal policy

Figure 2: Performance of $\gamma = 0.9$ and $\varepsilon_k = \frac{1+5\log(k)}{k}$,

## Comments

(1) Only when $\gamma = 0.9$, the robot can find the terminal. Only when $\varepsilon_k = \frac{100}{100+k}$ or $\varepsilon_k = \frac{1+5\log(k)}{k}$, the robot with exploration action can reach the goal.

(2) The result may be not reproducible because of exploration. Therefore, I set the random seed at 5904. The execution time fluctuates because of computing resource status unstable.

(3) The reason why $\varepsilon_k = \frac{100}{100+k}$ and $\varepsilon_k = \frac{1+5\log(k)}{k}$ are able to find the terminal is that their decreasing speed is more slow according to the Fig.3

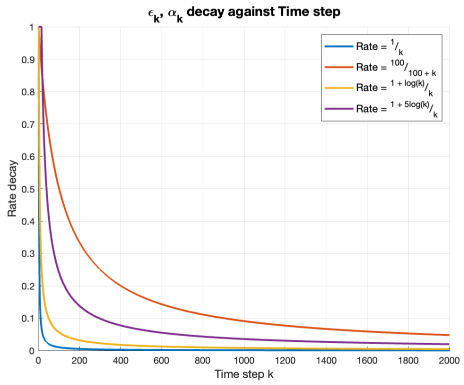



Figure 3: Decreasing speeds of different $\varepsilon_k,\ \alpha_k$ expression

If the $\varepsilon_k$ drops too fast, the exploration will be less. The robot is more easily stuck in the trap of $\arg\max_{\hat{a}} Q_k(s_k, \hat{a})$.

# Task 2

To design a Q-learning using my own parameter. I firstly test some new $\varepsilon_k$ and $\gamma$.

Table 2: $\gamma = [0.7,\ 0.8,\ 0.9]$ and performance of Q-Learning

| $\varepsilon_k,\ \alpha_k$ | **No. of goal-reached runs** | | | **Execution time (sec.)** | | |
|---|---|---|---|---|---|---|
| | $\gamma = 0.7$ | $\gamma = 0.8$ | $\gamma = 0.9$ | $\gamma = 0.7$ | $\gamma = 0.8$ | $\gamma = 0.9$ |
| $\frac{100}{100+k}$ | 10 | 10 | 10 | 0.65 | 0.92 | 1.08 |
| $\frac{1+5\log(k)}{k}$ | 9 | 9 | 7 | 0.82 | 1.06 | 1.04 |
| $\frac{100}{100+\sqrt{k}}$ | 10 | 10 | 10 | 0.16 | 0.16 | 0.17 |
| $\frac{1+10\log(k)}{k}$ | 10 | 10 | 10 | 0.69 | 0.63 | 0.82 |
| $\exp(-0.001k)$ | 10 | 10 | 10 | 0.09 | 0.11 | 0.13 |
| $\frac{1}{k^{0.1}}$ | 10 | 10 | 10 | 0.06 | 0.07 | 0.07 |

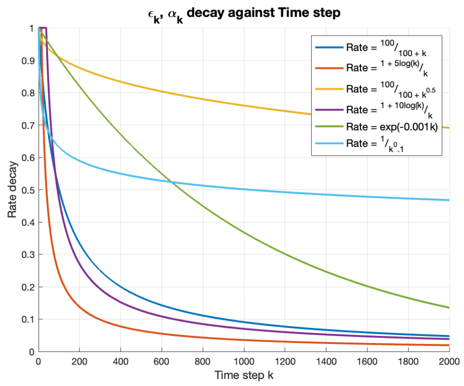



Figure 4: Decreasing speeds of different $\varepsilon_k,\ \alpha_k$ expression

From the Table 2, I choose the best performance parameter :$\varepsilon_k = \frac{1}{k^{0.1}}$ and $\gamma = 0.7$.

## Conclusion

From this project, I review the structure of Reinforcement Learning and learn the influence on the mode with difference $\varepsilon_k$ and $\gamma$. It is not always the complex $\varepsilon_k$ the best. It depends on the task and a balance of damping speed and computing complexity should be reached . I also learn the Bellman equation is really important for Q-learning. In the 2nd task, I design a new $\varepsilon_k$ inspired from the given. It decreases faster in the first 200 attempts and slower in the last attempts than other functions. Then I find the best $\gamma$ via test different value on the given reward. In my opinion, this may derive to the model overfitting to task 1 dataset and perform weak in evaluation set.