Department of Electrical & Computer Engineering
Faculty of Engineering

Q-Learning for World Grid Navigation
—— EE5904/ME5404 Neural Networks
Part II Project2 Report

Student Name: Gu Jiapan

Student Number: A0186538N

Email Address: e0320776@u.nus.edu

Task 1:

➢ Implementation:

The Q-learning algorithm using the reward function as given in "task1.mat" with the ε-greedy exploration algorithm is implemented in the M-file: "Task1.m". The model mainly involves three key parameters:

Discount factor γ, exploration probability ε_k and learning rate α_k.

First, a series of parameters and coefficients are set. According to the project description and requirement, the discount factor set as 0.5 or 0.9. There are 4 types of exploration probability with respect to the time step *k*. The learning rate is set the same value as exploration probability. In addition, the total number of runs is 10, the number of trials for each run is 3000 and the maximum number of steps for each trial is 100.

At the beginning of each run, we need to initialize the Q-function as 0. At the beginning of each trial, we need to reset the initial state (*s=1*). At the beginning of each step, the value of exploration probability and learning rate is calculated based on the type selected before. If this value is bigger than 1, it will be set as 1. Randomly taking a number from 0 to 1, if it is smaller than exploration probability, the current action is determined by one of the exploration cases, otherwise the exploitation case with the maximum reward of current state. For the four actions, the transition of state number is different: if moving up (*a=1*), decrease 1; if moving right (*a=2*), increase 10; if moving down (*a=3*), increase 1; if moving left (*a=4*), decrease 10.

After each action, the Q-function will be updated by the following formula with respect to state (*s*), action (*a*), reward, learning rate and discount factor.

$$Q_{next}(s,a) = Q_{crt}(s,a) + a_k\{reward(s,a) + \gamma \max[Q_{crt}(s',:)] - Q_{crt}(s,a)\}$$

The termination condition of each trial is that the robot reaches the goal state (*s=100*), or learning rate is smaller than 0.005. After finishing all the trials in a run, the execution time and the final Q-function are recorded. The optimal path with greedy policy is extracted by the following formula.

$$\pi^*(s) \in \arg\max_a Q^*(s,a)$$

Based on the policy, we can calculate the total reward (*Rt*) of the optimal path using the following formula with respect to rewards and learning rate.

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

It determines present value of future rewards. Rewards received *k* steps in the future

is discounted by its own factor. A small discounted factor forces the model to focus more on intermediate rewards from next few steps, while a large one forces the model to take into account future reward more strongly, also called farsighted. However, if the robot can not reach the goal state finally, the total reward will be set as -1.

Finally, after all 10 runs, output the number of goal-reached runs, average program execution time of them and trajectory plotting with total rewards according to the optimal policy.

➢ Results:

ε_k , α_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	--	--
$\frac{100}{100+k}$	0	10	--	1.64
$\frac{1+\log(k)}{k}$	0	0	--	--
$\frac{1+5\log(k)}{k}$	0	10	--	1.52

The table above shows the number of goal-reached runs and execution time for the parameter combination of 8 cases separately. We can find that only two of the cases have goal-reached runs: when the exploration probability is the 2nd type and the 4th type with discount factor 0.9. (The learning rate is set to the same value as the exploration probability.)

As for the other cases, the goal state is not reached in any of the 10 runs. Thus, the average execution time of the goal-reached runs is not available.

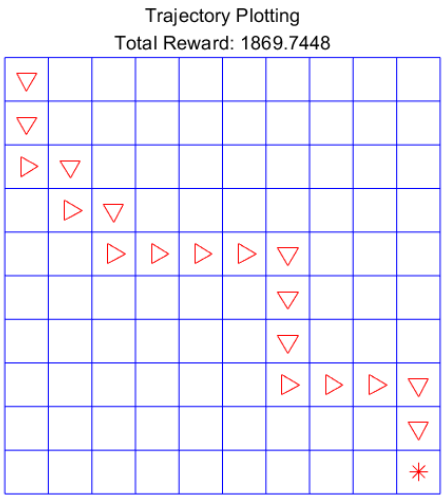
Let's focus on the two successful cases.

(1) $\varepsilon_k = \alpha_k = \frac{100}{100+k}, \gamma = 0.9$

No. of goal-reached runs: 10

Execution time: 1.64 sec

Total reward: 1869.7448

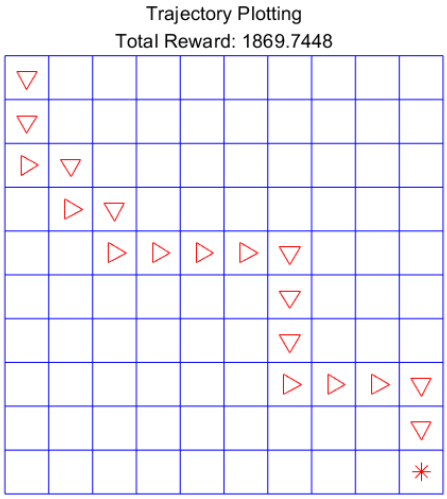


(2) $\varepsilon_k = \alpha_k = \frac{1+5\log(k)}{k}, \gamma = 0.9$

No. of goal-reached runs: 10

Execution time: 1.52 sec

Total reward: 1869.7448



From the results above, we can find that both of these two kinds of parameter combination can let the model obtain the same optimal policy and path with the same total reward. Their trajectory plotting is also the same. Both of them have a short average execution time.

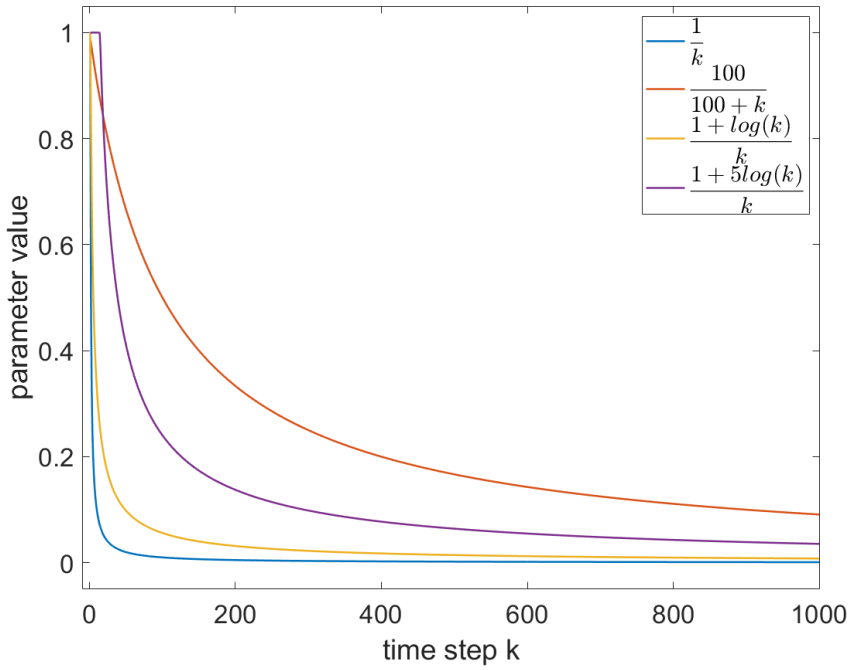
In addition, the output results are also expressed in the following variables. The variable of "policy" records the optimal policy about these 100 states in 10 runs. The variable of "s_record" is a record of optimal path.

Task 2:

Based on the experiments in task 1, in order to find an optimal policy accurately and efficiently, we need to do some analysis on how to choose the value of the key parameters: discount factor, exploration probability and learning rate.

First, the discount factor γ can influence the update of Q-function and the calculation of total reward as mentioned before. If it is set too small, the model will focus more on intermediate rewards from next few steps, while if it is a large value, the model will be forced to take into account future reward more strongly, which can explain why 0.9 can work but 0.5 cannot in task 1. Therefore, I choose 0.9 as the discount factor in task 2, which can make the model more farsighted.

As for the value of exploration probability ε_k and learning rate α_k, it is necessary to observe the trend of the four types with respect to time skep k



From the figure above, we can find that these 4 types have different decay rates with the increase of time step. Comparatively speaking, the slowest decaying type is $\frac{100}{100+k}$ and the second slowest decaying type is $\frac{1+5\log(k)}{k}$. These two types are exactly used in the two cases that the goal state is reached successfully in task 1.

As mentioned before, a higher value of exploration probability means the action has a higher probability to carry out exploration of more possible choice, instead of exploitation of currently known best choice. For learning rate, a higher value can help Q-function update quickly and better avoid stuck in the local extremum.

Therefore, according to the analysis above and the results in task 1, I set the parameters $\varepsilon_k = \alpha_k = \frac{100}{100+k}$ and $\gamma = 0.9$ as my choice in "RL_main.m".