EE5904/ME5404 Neural Networks Part II Project Report

AY2021/2022, Semester 2

Department of Electrical and Computer Engineering

Q-Learning for World Grid Navigation

Jin Lexuan A0232696W

e0724495@u.nus.edu

A report submitted in fulfilment of the requirements of
the National University of Singapore for
the EE5904/ME5404 Neural Networks module

April 9, 2022

1 Task description

Using Q-learning with ε-greedy exploration. The robot is to move from the initial state (s = 1) to the goal state (s = 100) with the maximum total reward of the trip. The map is shown in Figure 1. Different choice of action at each state will bring rewards accordingly.

In this task, the model is a deterministic one. Dynamic programming methods are applied to find the optimal policy. For each state, the robot can take one of the four actions. However, in some certain states, some of actions are forbidden due to the boundary of map. Actions are illustrated in Figure 2(up, right, down, left).

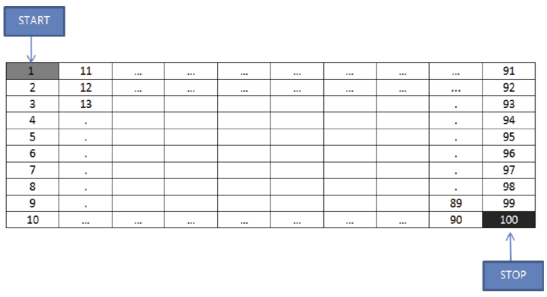

Figure 1: Task map

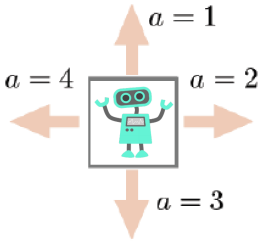

Figure 2: Actions of robot

Learning process:

- The robot learns in 1 run
- One run consists of the N trials
- Each run starts with a set of initial values of the Q function (100 x 4 matrix)
- Each trial starts when the robot moves from state 1
- Each trial ends when the robot reaches state 100
- The Q values are passed to the next trial
- Each run ends when the Q values converge to the optimal values

2 Formulas and Functions

Total reward for a state transition:

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

R_t determines present value pf future rewards. Rewards received k steps in the future is discounted by factor γ^{k-1}. Small γ focuses more on intermediate rewards for next the few steps. Large γ takes into account future rewards more strongly.

'Worth' of actions at different states is given by:

$$Q^{\pi} : S \times A \to R$$

$$Q^{\pi}(s,a) = E^{\pi}[R_t | s_t = s] \to R_t | s_t = s$$

This is deterministic transition. Expectation returns from taking action a as state s at time step t by following action π.

Optimal policy is the state transitions that maximize the Q-values.

$$Q^{\pi}(s,a) = E^{\pi}[r - t + 1] + E^{\pi}[\gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} | s_t = s]$$

Values of Q-function are optimal if they are greater or equal to that of all other policies for all (s,a) pairs, i.e.,

$$Q^*(s,a) = maxQ^{\pi}(s,a)$$

Greedy policy: At each s, select a that yields the largest value for the Q-function. When multiple choices are available, such a can be picked randomly. Optimal policy:

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

Model-Free Value Iteration: When state transition model is unknown, the Q-function can be estimated via iterative update rule by using the reward received from observed state transitions.

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

Exploitation:Use greedy policy to select currently known best action (with probability 1 - ε_k):

$$a_{k+1} = \max_{a'} Q_k(s_{k+1}, a')$$

Exploration:Try action other than current known best action(with probability ε_k):

$$a_{k+1} \neq \max_{a'} Q_k(s_{k+1}, a')$$

3 Performance of task1

According to the requirements and formulas, the MATLAB program is designed and implemented. The following results can also be obtained by running the file $RL\_task1.m$.

Figure 3 shows the optimal policy for every state. Figure 4 presents the execution of optimal policy and it can finally reach the target state(next page). The execution of optimal policy represents the corresponding action column vector in a straightforward manner.

Figure 3: Optimal policy

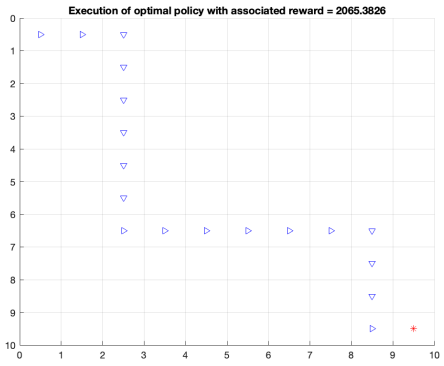

Figure 4: Execution of optimal policy

The results of different γ value and ε expression are shown in Table 1. There are two situations that the robot can reach the goal. The number of runs and the execution time are recorded. Accordingly, the maximum reward is 2065.3826 when the discount value is 0.9 and takes the optimal policy. It is also demonstrated in the title of Figure 3 and Figure 4.

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	N.A.	N.A.
$\frac{100}{100+k}$	0	10	N.A.	0.7500
$\frac{1+log(k)}{k}$	0	0	N.A.	N.A.
$\frac{1+5log(k)}{k}$	0	9	N.A.	1.8134

Table 1: Parameter values and performance of Q-learning

From the table and the figure, it can be concluded that:

- Only when $\gamma = 0.9$, the robot can find the solution to the goal. The reason is that a higher γ value enable the robot to explore more future rewards, while smaller γ contributes to the short-sightedness of the robot.

- The decreasing speed of different ε_k expression is compared in Figure 5 (next page). $\frac{100}{100+k}$ and $\frac{1+5log(k)}{k}$ have a relatively slow decreasing speed. It indicates that the exploration action contributes to the reaching of target. A slow decreasing ε_k can enable more exploration behaviors. A suitable ε_k should balance exploration-exploitation trade-off.

- The time for $\frac{100}{100+k}$ to finish 10 runs is less than half of $\frac{1+5log(k)}{k}$ situation. According to the curve of two expressions, $\frac{100}{100+k}$ allows more exploration which may be beneficial to find the path to the goal, making robot learn in a faster speed.

- Exploration is significant for finding new information, while exploitation focuses on optimizing what is already found. The ε-greedy algorithm is the combination of them. Consequently, ε_k expression ought to be selected with carefulness.

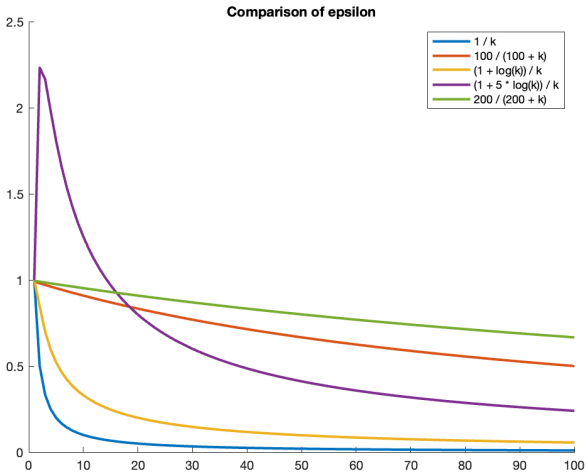

Figure 5: Comparison of different ε_k expression

4 Design for task2

It can be concluded in task1 that for a high γ value and ε_k expression with a slow decreasing speed, the possibility and speed of reaching the goal. In task2, γ is also designed as 0.9 and

$$\varepsilon_k = \frac{200}{200 + k}$$

After being tested by a fake *evalreward*, these two choices have an acceptable performance. The program running time for 10 runs is no more than 2 sec based on the situation of the map. The times of reaching the target is 10 out of 10. The result is acquired by running file *RL_main.m*.