EE5904 Neural Networks

Project 2: Q-Learning for World Grid Navigation

Name: Zhang Fei

Matriculation No.: A0117981x

Email: E0506561@u.nus.edu

Task 1: Implement of Q-learning algorithm

Implementation:

We can apply ∈-greedy exploration to retrieve Q function matrix and use greedy search to find out the optimal policy.

The Q-Learning algorithm is implemented in such steps:

```
1. Initialize all the input parameters
2. Apply ∈-greedy exploration for every trail
   2.1. For every transition
        2.1.1. Pick action
        2.1.2. Apply action
        2.1.3. Update Q-value
3. Retrieve optimal policy
```

1. Initialize all the parameters
   Set input parameters according to project requirement. Exploration probability ϵ_k and learning rate α_k are set according to time step k. Initialize Q-value matrix with shape (100,4) to zeros. Set discount factor gamma γ = 0.5 or 0.9, initial state s = 1.

2. Implement ∈-greedy exploration
   Generate a random number r between 0 and 1 using rand function. If r >= exploration probability ϵ_k, then we apply exploitation method to select the action produce the largest Q value. If r < ϵ_k, then we can apply exploration method to randomly select other action except the action gives the best Q-value. And avoid any action that will result negative reward as the robot is only allowed to travel within the grid. When the action is selected, we can determine the next state and the reward when the robot transits to next state. The we can update Q-value using the following equation:
   $$Q_{k+1}(s_k, a_k) = Q_k(s_k, a_k) + \alpha_k(r_{k+1} + \gamma \max_{a'} Q_k(s_{k+1}, a') - Q_k(s_k, a_k))$$
   Update Q value and move to next state. Check the loop conditions inside every trail. When goal state reach s = 100 or learning rate $\alpha_k < 0.005$, then the Q function program starts next trail. The trail loop stops when trail reaches maximum iteration number, or it stops when Q value reaches optimal values which means the updated Q values are not changing anymore.

3. Retrieve optimal policy
   From the final Q matrix, select the action yields the largest reward in each state with greedy policy.

Result:

Execute the program 10 times with different set of parameters. The number of times the goal state is reached, and the average execution time are recorded down in the table below.

Table 1 goal-reached runs and execution time for different parameters

ϵ_k, α_k	No. of goal-reached runs		Execution time (sec.)	
	$\gamma = 0.5$	$\gamma = 0.9$	$\gamma = 0.5$	$\gamma = 0.9$
1/k	0	0	-	-
100/(100+k)	0	10	-	6.5506
(1+log(k))/k	0	0	-	-
1+5log(k))/k	0	8	-	8.3758

The robot managed to reach goal state when γ is 0.9 and ϵ_k, α_k is 100/(100+k), or γ is 0.9 and ϵ_k, α_k is (1+5log(k))/k. The total reward is 1861.9 when robot reached goal state under optimal policy. The figure 1 below shows the trajectory of robot's transition from initial state to final state.

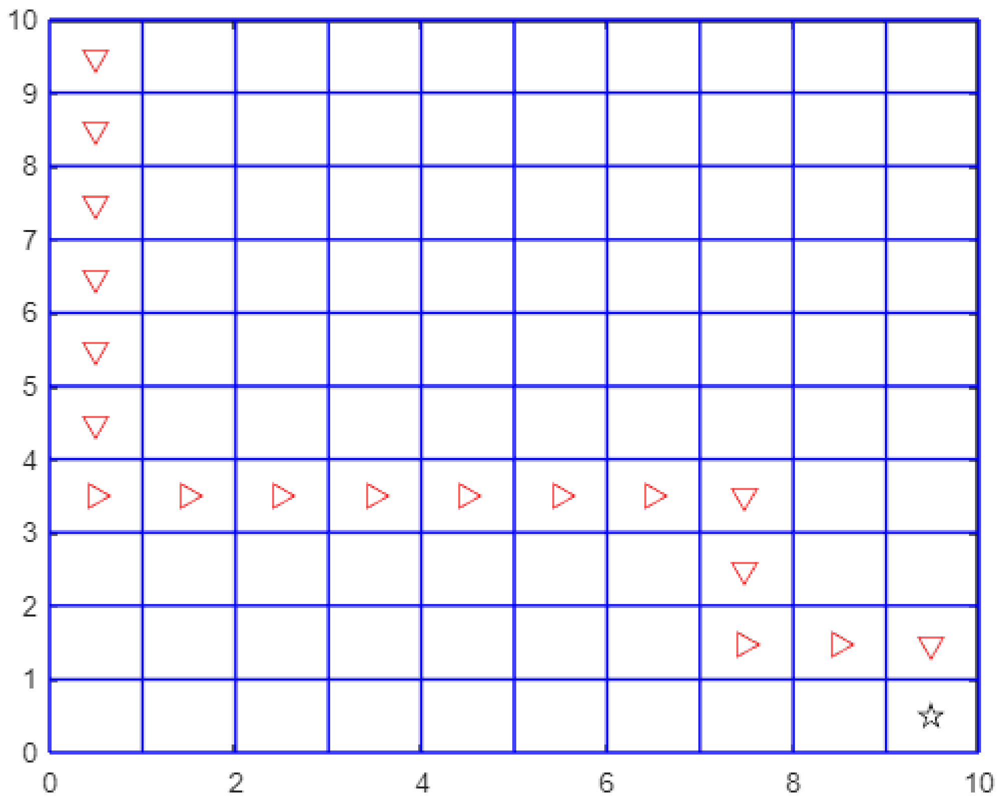

Figure 1 optimal path using Q-learning

Comments:

The robot was able to reach goal state only when γ = 0.9, and ϵ_k, α_k is 100/(100+k) or (1+5log(k))/k. For small discount factor γ = 0.5, it will result in state/action values representing the immediate reward, while a higher discount factor γ=0.9 will result in the values representing the cumulative discounted future reward that the robot expects to receive. Therefore, a smaller discount rate forces robot focuses more on immediate rewards from next few steps, and a higher discount rate makes robot farsighted, resulting the robot takes into account future rewards more strongly and reach goal state.

Larger discount rate will result longer execution time since the robot takes into account future rewards.

With the same k value, larger ϵ_k, α_k values make the robot reach goal state more likely as the robot has high tendency to explore new actions. Therefore, balancing the trade-off between exploration and exploitation is crucial to train the robot to reach the goal state.

Task 2: Design my own Q-learning

Since the execution time and final state are determined by ϵ_k, α_k and γ from experiment in task 1. The objective is to find the optimal parameters yield the optimal policy for robot to reach final state with best rewards and least execution time. Since a small discount factor will reduce the total rewards when the robot reaches final state. Then fix γ = 0.9, adjust ϵ_k, α_k to observe the execution time and No. of goal-reached runs.

Table 1 Results of trying different ϵ_k, α_k

ϵ_k, α_k	$\frac{100}{100+k}$	$\frac{500}{500+k}$	$\frac{1000}{1000+k}$	$\frac{1500}{1500+k}$	$\frac{2000}{2000+k}$	$\frac{2500}{2500+k}$	$\frac{3000}{3000+k}$
No. of goal-reached runs	10	10	10	10	10	10	10
Execution time (sec.)	4.522	1.5657	0.4466	0.4216	0.5507	0.6370	0.8041

Since ϵ_k, $\alpha_k = \frac{1500}{1500+k}$ produce least execution time, then fix ϵ_k, α_k, tune gamma to observe the results.

Table 3 Results of trying different γ when $\epsilon_k = \alpha_k = \frac{1500}{1500+k}$

γ	0.8	0.85	0.9	0.95	0.99
No. of goal-reached runs	10	10	10	10	0
Execution time (sec.)	0.4150	0.4557	0.4216	0.4271	-

Table 4 Results of trying different ϵ_k when $\gamma = 0.95$

ϵ_k	0.4	0.5	0.6	0.7
No. of goal-reached runs	10	10	10	10
Execution time (sec.)	0.5411	0.2933	0.2178	0.2218

Result:

Based on my experiments, I set parameters discount factor $\gamma = 0.95$, learning rate alpha $\alpha_k = \frac{1500}{1500+k}$, exploration probability $\epsilon_k = 0.6$.