EE5904/ME5404 Neural Networks Part II Project Report

AY2022/2023, Semester 2

Department of Mechanical Engineering

Reinforcement Learning Project Q-Learning for World Grid Navigation

Sun Shaowei

A0263124N

E1010482@u.nus.edu

April 2023

1. Introduction of reinforcement learning

In the area of machine learning, reinforcement learning (RL) is invented to design the policy how intelligent agents should take actions in an environment to maximize the notion of cumulative reward. Markov decision process (MDP) is always employed to describe the information from the environment and the corresponding expected returns calculated by Bellman equations. This assignment only tackles the deterministic problem in the 10*10 maze. To find the optimal policy, Q-Learning with ε-greedy exploration method is utilized to find the optimal policy and the maximum rewards. Different from the classical dynamic programming methods, Q-Learning algorithm doesn't assume the knowledge of an exact mathematical state-transition model, but only uses reward received from observed state transitions. Iterative updates rely on the temporal difference, which is the difference between estimate of $Q^*(s_k, a_k)$ and current estimate $Q_k(s_k, a_k)$. The Q-learning equation is denoted by Eq.1:

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

Where s_k and a_k are the state and actions, α_k is the learning rate, r_{k+1} is the reward and γ is the discount rate. Q-learning update rule converges to Q^* as the iterations become infinite, which is guaranteed by two laws: (1) $\sum_{k=0}^{\infty} \alpha_k^2 < \infty$ and $\sum_{k=0}^{\infty} \alpha_k \rightarrow \infty$. (2) All (s, a) pairs are (asymptotically) visited infinitely often. In this assignment, learning rate α_k and ε-greedy coefficient are set to be the same and decrease with the time. Q-learning algorithm converging relies on the balance between Exploitation and Exploration. Exploitation chooses the best action ($a_{k+1} = \max_{a'} Q_k(s_{k+1}, a')$) at the current state, whereas exploration chooses one of remaining actions ($a_{k+1} \neq \max_{a'} Q_k(s_{k+1}, a')$), which guarantees the probability of visiting all states in the infinite time.

The main purpose of this report is to investigate how the Q-learning algorithm performs with different parameters including the learning rate α_k, the discount rate γ, and the ε-greedy coefficient.

2. Task 1

2.1 Configuration of Q-learning algorithm

In this assignment, the robot starts from the upper left corner and ends at the lower right corner in the 10*10 grid. It has four actions: moving up, right, down, and left with no diagonal movement involved. The reward of each action in every state is listed in the file Task1.mat. Each row represents each state with four columns recording rewards from the corresponding actions. In Task 1, there are 8 sets of the learning rate α_k, the discount rate γ, and the ε-greedy coefficient for testing. The program is required to run 10 times, and 3,000 trials are to be conducted in each run to iteratively update Q-values. In each running time, the stopping sign is raised when the Q-learning algorithm converges, or the trials have reached the maximum number. For each trial, termination condition is robot reaching goal state ($s_k = 100$), or the learning rate smaller than the pre-defined threshold or reaching the maximum number of time step k_{max}. When the whole iteration process is completed, a column vector representing the action at each state selected by the optimal policy (named "*action*" in the script) is saved as well as the average execution time of the "goal-reaching" runs. It is possible that some of the runs may not yield an optimal

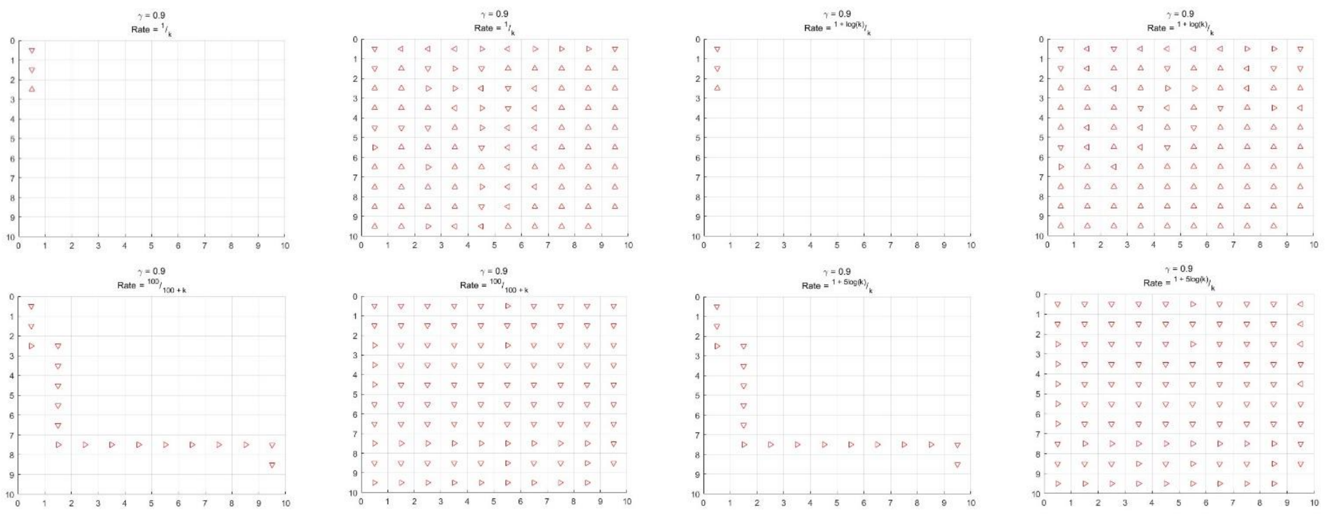

Figure 1: the optimal policies generated from the Q-learning using different parameters.

policy that results in the robot reaching the goal state; these runs are not to be counted. To solve this problem, the shaping reward function is inserted to modify the Q-learning algorithm. After conducting 8 different sets, the number of the goal reached runs and the average time taken are to be compared in order to examine the performance of each parameter of the Q function.

2.2 Results of reinforcement learning experiments

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	NaN	NaN
$\frac{100}{100+k}$	0	10	NaN	4.62
$\frac{1+\log(k)}{k}$	0	0	NaN	NaN
$\frac{1+5\log(k)}{k}$	0	10	NaN	14.58

Table 1: Parameter values and performances of the default Q-Learning algorithm.

Table 1 records the number of goal-reached runs and corresponding average execution time of successful runs. It is discovered that some of parameters could not generate an effective policy to guide the robot to reach the goal. All failed cases are featured by an endless loop, which trap the robot without breaking the circulations. The reasons are composed by the ineffective discounting rate and the unreasonable learning rate α_k or ε-greedy coefficient. In the provided rewarding system, exploration could not impose Q values at a large extent. The sharply decreasing learning rate reduces the benefits brought by the exploration, while the sparse reward or the unreasonable reward enhances the exploitation. If the discounting rate is set to be a large number, the information from further states would help to relieve this terrible situation, because the later state

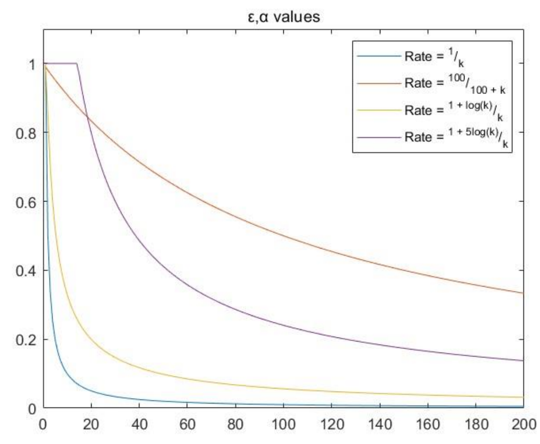

Figure 2: Trajectories of ϵk and αk with increasing k values [1, 200]

would transmit its high reward through the network and guide the stray robot.

Also, discounting rate, learning rate, and ε-greedy coefficient decide the iteration time in these successful cases. The average iteration time for $\varepsilon_k, \alpha_k = \frac{100}{100+k}, \gamma = 0.9$ is 4.62 seconds, whereas that for $\varepsilon_k, \alpha_k = \frac{1+5\log(k)}{k}, \gamma = 0.9$ is 14.58 seconds. The difference is generated by the decreasing slop of the learning rate and ε-greedy coefficient. Steep slopes would constrain the probability of exploration at the early stage and even more seriously dampen the attempts of finding the optimal policy. Two successful functions share the same characteristic of maintaining gentle slopes at the early state.

Hence, there are several conclusions drawn from the experiments:

1. Small value of discounting rate would make the robot short-sighted and impede it from finding the optimal path to the goal. An extremely small value even makes robot roam in the endless loop with unreasonable rewards.
2. Steep decreases of learning rate and ε-greedy coefficient would dampen the exploration at the early stage, which would cost a long time for the robot to reach the goal. Similarly, the worst situation is that the probability of exploration is almost zero, while the benefit from exploration is multiplied by an extremely small learning rate, which would trap the robot in the endless loop.

2.3 Reward shaping functions

Reward shaping functions are designed to guide the agent when the sparse rewards or the unreasonable rewards could not guide the agent to the right directions. This assignment defines potential-based shaping function using the normalized Manhattan distance for grid environments. After applying the reward shaping functions, robot finds the optimal path even with the previously failed parameters. Execution time shortens quite a lot, which is listed in the Table 2.

3. Task 2

ε_k, α_k	No. of goal-reached runs		Execution time(sec.)	
	$\gamma = 0.5$	$\gamma = 0.9$	$\gamma = 0.5$	$\gamma = 0.9$
$\frac{1}{k}$	10	8	0.57	0.78
$\frac{100}{100+k}$	10	10	1.00	1.38
$\frac{1+\log(k)}{k}$	10	10	0.60	0.89
$\frac{1+5\log(k)}{k}$	10	10	1.19	1.74

Table2: Parameter values and performances of the Q-Learning algorithm with reward shaping functions.

This work designed a new 100*4 reward grid in the file "qeval.mat". The designing principle is that the distance to goal and the distance to the shortest line connecting starting point and goal point synchronously decide the value of reward. New reward grid would induce the robot to find the shortest way. Moreover, a new parameter type (e^{nk}) was attempted in this part as it was found that it decreases considerably gently and its maximum value is not greater than 1, which is appropriate for probability values.

3.1 Analysis of new reward grid

The new designed reward grid has high values in the diagonal lattices in the 10*10 grid. Theoretically, the robot would follow the shortest direction and easily reach the goal. The comparison is selected from previous experiments, where $\varepsilon_k, \alpha_k = \frac{100}{100+k}, \gamma = 0.9$. The optimal path learned from the new reward grid is different from the previous one.

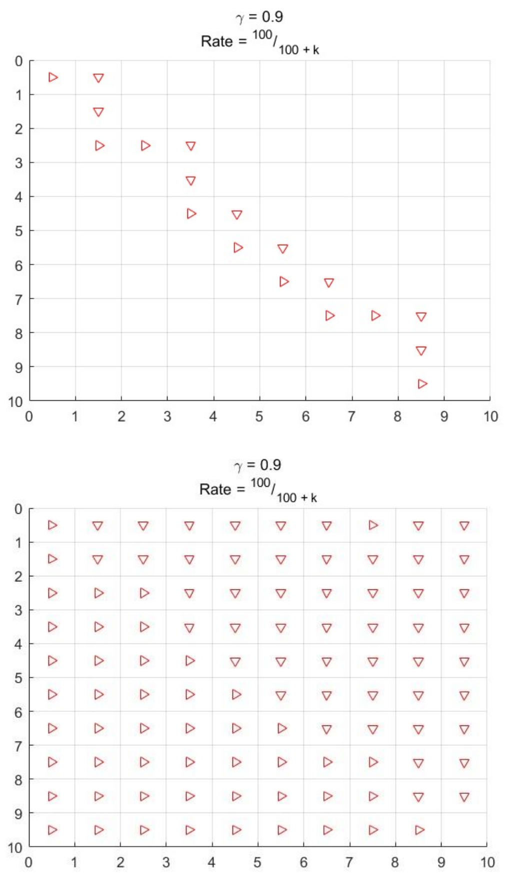

Figure 3: the optimal policies generated from the Q-learning using new reward grid.

3.2 Analysis of new parameters

New parameters decrease in the pattern of gentle slopes, which maintain the high probability for the agent to explore and keep the benefits without being erased. To test the influence from new learning rate and ε-greedy coefficient, the discount rate is set to be a reasonable value. Q-learning algorithm is accelerated by new parameters because the average time for 10 goal-reached runs is just 2.15 seconds. By comparing the results from previous experiments, it could be concluded that the freedom of exploration could effectively accelerate the iterations.

ε_k, α_k	No. of goal-reached runs	Execution time(sec.)
	$\gamma = 0.9$	$\gamma = 0.9$
$e^{-0.001k}$	10	2.15

Table 3: Parameter values and performances of the Q-Learning algorithm with new configurations.

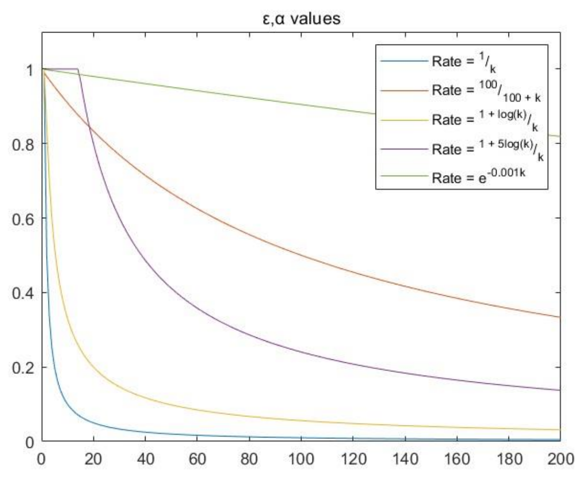

Figure 4: gently decreasing parameters.