NATIONAL UNIVERSITY OF SINGAPORE

ME5404 Neural Networks

Reinforcement Learning Project
Q-Learning for World Grid Navigation

Assignment – AY21/22 Semester 2

Name	Student Number	E-mail Address
Jin Hoontae	A0243155L	E0816449@u.nus.edu

EE5904/ME5404
Neural Networks
Project 2 AY2021/2022

Q-Learning for World Grid Navigation

Jin Hoontae A0243155L

E0816449@u.nus.edu +65 8620 5429

Abstract

Q-Learning algorithm was constructed to perform the given task in which an agent (the robot) needs to learn to move to a goal state in an optimal manner. Multiple parameter types were attempted, and the results suggested that when the exploration value (ϵ_k) decreases too rapidly, the robot tends to find the optimal path with a lower total reward as it only focuses on taking the best action. A high discount rate value was also important in that it makes the robot take into account future rewards more than immediate rewards.

Keywords: Q-Learning, exploration value (ϵ_k), discount rate (γ)

1. Introduction of Reinforcement Learning

In Reinforcement Learning (RL), there exist intelligent agents that continuously learn to achieve the given task. The key mechanism is to maximize the total reward through numerous trials under the assumption that the agents are equipped with no knowledge of the desired outputs in advance. In this assignment, a single agent is used; that is, the robot learns to reach the goal state (10,10) from the initial state (1,1) in the 10×10 grid world. Of multiple RL methods, a deterministic algorithm of *Q-Learning* with ϵ-greedy exploration method is utilized for this assignment, which means that it only considers rewards received from state transitions. The algorithm is to be updated iteratively thereby constantly measuring the worth of taking an action (a) at a state (s) under the policy (π). The equation can be 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 action, α_k is the learning rate, r_{k+1} is the reward and γ is the discount rate. In order to effectively maximize the algorithm, it is necessary to balance two strategies appropriately: Exploitation and exploration. The former chooses currently known best action ($a_{k+1} = \max_{a'} Q_k(s_{k+1}, a')$), whereas the latter attempts one of the remaining actions ($a_{k+1} \neq \max_{a'} Q_k(s_{k+1}, a')$), which enables the robot figure out other best possible plan to complete the task given. In order to balance the trade-off, ϵ-greedy exploration value is designed to decrease as the learning continues.

The main purpose of the report is to investigate how the Q-learning performs with different parameter types/values (namely, learning rate (α), discount rate(γ), and ϵ-greedy exploration).

2. Task 1

2.1. Implementation of Q-Learning algorithm

In this report, the agent (robot) is strictly designed to choose four actions; that is, it can only move up, right, down or left with no diagonal movement allowed, which are defined as a_1, a_2, a_3 and a_4, respectively. As mentioned previously, at every beginning, the robot is positioned at the top-left corner (1,1) in the 10×10 grid world, and the goal state (10,10) is located at the bottom-right corner. In Task 1, ϵ_k and α_k are set to the same value, γ can either be 0.5 or 0.75, and 8 different sets were conducted in total. The program is required to be run 10 times, and 3,000 trials are to be conducted in each run to iteratively update Q-function. During the trial process, it is programmed to stop when it reaches the goal state and when the learning rate value becomes below a pre-defined threshold ($5e^{-3}$). When the whole iteration process is completed, a column vector representing the action selected by the optimal policy (named "*action_numeric*" in the script) is to be saved as well as the average execution time of the *"Goal-reached"* runs. 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 the experiment

Table 1 summarizes the number of the goal-reached runs and average execution time for each set. It was discovered that not all the sets were able to produce an optimal policy that guides the robot to the goal state. As can be seen in the table, all the parameter types of ϵ_k and α_k ended up arriving in the goal state only when the discount rate (γ) was 0.9. Hence, 4 of 8 sets accomplished the desired task. For the unsuccessful models when γ is 0.5, it was observed that the robot repeatedly moves up and down for the optimal path ($\pi^*(1) = 3 \; and \; \pi^*(2) = 1$), which is a sign of failing the performance as shown in **Fig. 1(a)**. On the other hand, as can be seen in **Fig.1(b and c)** when γ is 0.9, the robot reaches the goal state. Hence, it can be said from this result that it is necessary to select a proper value for the discount rate, as it significantly determines the performance of the RL model. **Fig 1(b)** is the optimal path created by $\frac{100}{100+k}$, $\frac{1+5\log(k)}{k}$ and $\frac{1+\log(k)}{k}$, which shows that the same reward value (=1858.8443) was obtained. Moreover, the optimal policies for every state were also identical. For $\frac{1}{k}$, even though the robot reached the goal state, the total reward value was rather lower (=1797.4509). As for the execution time of the successful models, $\frac{1}{k}$ was observed to complete the task in the shortest time (=35.39s), whereas the other types were rather slow($\approx 98.17s, 85.65s$ and $91.64s$ for $\frac{100}{100+k}$, $\frac{1+\log(k)}{k}$ and $\frac{1+5\log(k)}{k}$ respectively). The difference in this execution time, perhaps, is because the form of $\frac{1}{k}$ tends to be the simplest, which enables the computation process to be more easily done.

Table 1. Parameter values and performance of Q-Learning

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	10	N/A	35.39
$\frac{100}{100+k}$	0	10	N/A	98.1651
$\frac{1+\log(k)}{k}$	0	10	N/A	85.6548
$\frac{1+5\log(k)}{k}$	0	10	N/A	91.6417

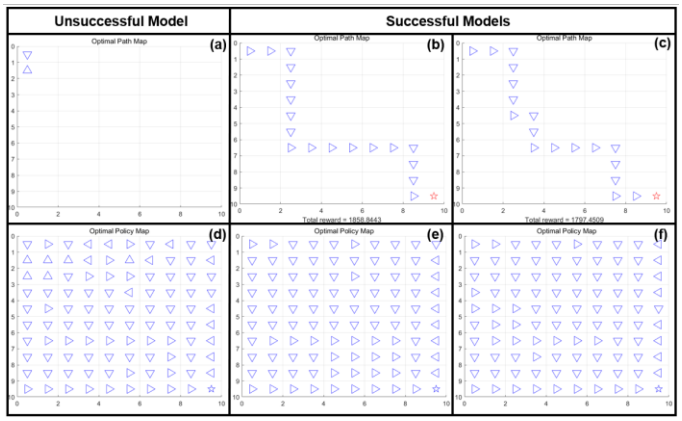

Figure 1. Examples of the Q-function (a: when the robot fails to reach the goal state, b & c: when it successfully reaches the goal state, d: optimal policy of the unsuccessful model for each state, and e & f: optimal policy of the successful model for each state)

The performance difference between successful and unsuccessful models can further be examined in **Fig. 1(d,e and f)**. For **Fig.1(d)**, the optimal policy for each state is computed in a way that it does not effectively guide the robot to the goal state, whereas the other figures show that it flawlessly lets it arrive in the goal state as the policies do not make it go backward but only move towards the destination.

2.3. Analysis on different parameter types and values

As only 4 of 8 sets, in total, were able to achieve the task given, an analysis on the parameter types was conducted in order to deduce such different results. **Fig. 2** illustrates the trajectories of ϵ_k and α_k with increasing k steps [1, 200]. As can be observed in the figure, the value of the parameter type ($\frac{1}{k}$) decreases exponentially at the very start from 0 to 20 and converge very close to zero thereafter, whereas no sharp decrease can be found for the trajectory of $\frac{100}{100+k}$. This probably proves that the reason of the robot obtaining a lower total reward could be due to the steeply decreasing exploration value. As for $\frac{1+5\log(k)}{k}$, its initial value is greater than 1 and remains above it approximately until k reaches 14. However, as ϵ_k and α_k are defined as probabilities, this phenomenon is not desirable. Hence, an assumption was made: when the parameter value is greater than 1, the program forces it to be one instead, which allows us to obtain optimal policies appropriately. One thing to note is that $\frac{1+5\log(k)}{k}$ was able to complete the task successfully with the discount rate (γ)=0.9, nevertheless. This perhaps implies that, in Q-Learning, it is sometimes acceptable for the exploration value to be consistently equal to 1 in the early iterations, meaning that letting the robot explore/select minor actions instead of the optimal action ($\epsilon_k = 1$) can offer the flexibility to find possible optimal solutions more easily and effectively.

As for the discount rate, the higher value (=0.9) was observed to produce better results than the lower value. The mechanism of it is that a small value lets the robot behave in a short-sighted manner, meaning that it only tries to focus on the next immediate steps. On the other hand, a big value significantly influences it to consider future steps. In the reward matrix given, it was found that it is more advantageous to choose the higher value to complete the task as it helped the robot to reach the goal state in all runs.

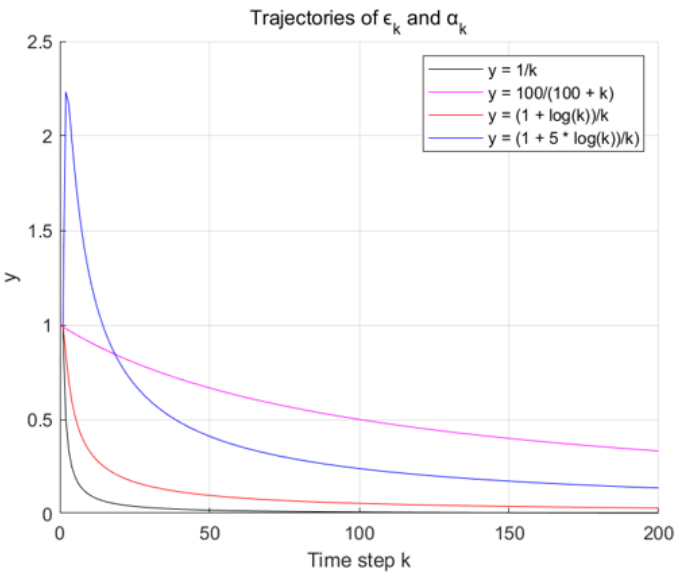

Figure 2. Trajectories of ϵ_k and α_k with increasing k values [1, 200]

Hence, several conclusions could be drawn from the analysis:
1. Parameter types that show a steep decrease at the beginning and become close to zero too early are not desirable options as they either let the robot behave without considering the exploration or end up obtaining a lower total reward.
2. Parameter types that smoothly decrease tend to generate a better performance as it takes the exploration into consideration.
3. A high discount rate is a desirable option as it equips the robot with the far-sighted "thought".

3. Task 2

A new 100×4 reward matrix, named "qeval.mat" containing the variable (qevalreward), was created to investigate how the constructed Q-learning would behave. The original reward matrix contained a reward value (9,999) near the goal state. Hence, to ensure that the robot completes the task, a reward value (9,999) remained at the same state while other reward values were shuffled row-wise. 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.

Moreover, it was discovered that the original reward matrix was designed in a way that the robot does not move erratically such as moving out of the grid map, jumping from a bottom state to a top state and vice versa. Thus, it was compulsory to add extra codes to ensure that the robot moves stickly in four directions (up, down, left and right) and does not leave the grip map for the new reward function.

3.1. Results and Analysis of the task 2

0.9 was chosen for the discount rate because, as previously proven in **Section 2.3**, a higher value of it tends to enable the robot to learn more efficiently due to its *far-sighted* ability. Three different parameter types ($e^{-0.001k}$, $e^{-0.0015k}$ and $e^{-0.002k}$) were attempted. **Fig. 4** shows the trajectories of ϵ_k and α_k with increasing k steps [1, 3000]. Unlike $\frac{1}{k}$ and $\frac{1+\log(k)}{k}$, there is no sharp decrease within the range and they decrease more smoothly, not getting close to zero too early. Hence, a prediction could be made that they would be effective enough to complete the new task. **Table 2** shows the results of the new parameter types. All of them successfully managed to find an optimal path in all runs and the execution time was not too long.

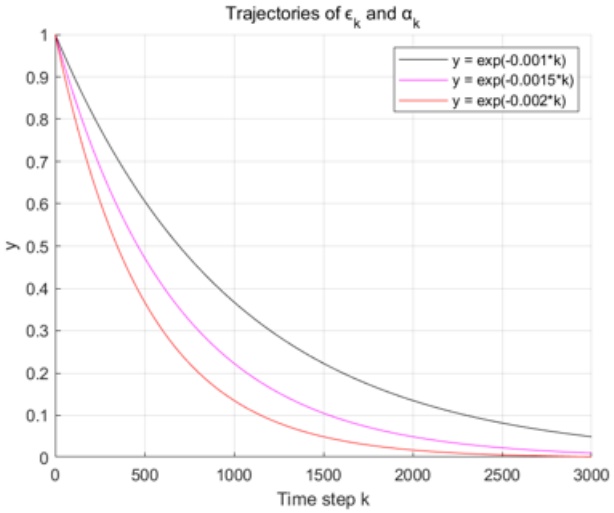

Figure 4. Trajectories of ϵ_k and α_k with increasing k values [1, 3000] for exponential functions

Table 2. Parameter values and performance of Q-Learning for new parameter types

ε_k, α_k	No. of goal-reached runs	Execution time (sec.)
	$\gamma = 0.9$	$\gamma = 0.9$
$e^{-0.001k}$	10	69.3618
$e^{-0.0015k}$	10	74.313
$e^{-0.002k}$	10	67.6764

Fig 5. shows the results of the new parameter types. They all produced the same optimal policies for every state as well as the same optimal path.

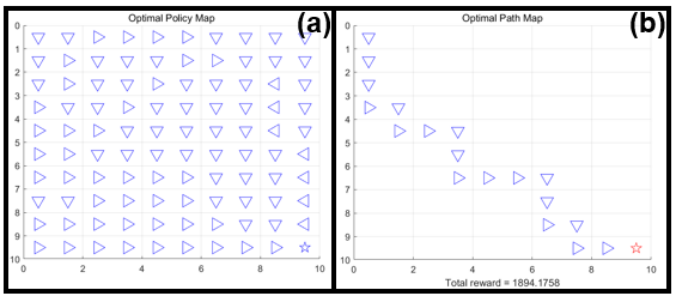

Figure 5. Results of the Q-function for new parameter types (a: optimal policies for every state, and b: an optimal path)