Arunava Manna

A0232541R

Arunava.manna@u.nus.edu

EE5904

Project 2: Q-Learning for World Grid Navigation

Contents

Project Description

Scenario:

The goal of this project is to find out an optimal path for a robot, which is now located at the position (1,1) of a 10x10 grid, to reach the location (10,10) of that grid.

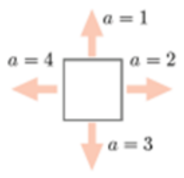
Robot current location

1	11	21	31	41	51	61	71	81	91
2	12	22	32	42	52	62	72	82	92
3	13	23	33	43	53	63	73	83	93
4	14	24	34	44	54	64	74	84	94
5	15	25	35	45	55	65	75	85	95
6	16	26	36	46	56	66	76	86	96
7	17	27	37	47	57	67	77	87	97
8	18	28	38	48	58	68	78	88	98
9	19	29	39	49	59	69	79	89	99
10	20	30	40	50	60	70	80	90	100

goal

Agent: A robot

Actions:



Sates: each block of the 10 x 10 matrix is a state for our experiment.

Rewards: A 100x4 matrix which correspond a reward for each state and each action.

Task1

➢ **Introduction**

The goal of task1 is to find an optimal policy with the given set of epsilons, learning rate functions and two gamma values. There is total 10 runs and each run have maximum of 3000 trials.

The termination conditions of one trial are when

1. The states will reach (10,10) position that is when state =100
2. When alpha value from the epsilon function have less than 0.005 value.
3. When the trial reaches 400 steps.

The termination conditions of one run are when

1. When maximum number of trials is 3000
2. When the Q state converges

The Q state convergence condition is norm(N-Q) <0.005.

(N=old Q values and Q=updated Q value after taking the new actions)

The task focuses on two gamma values 0.5 and 0.9.

In a trial, when the robot starts from state 1, Q state values starts to update using the below formula:

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

In the above equation, s_k is the current k^{th} step state number, a_k is the k^{th} step action chosen by exploration or exploit, α_k is the k^{th} step learning rate generated from the epsilon function, r_{k+1} is the reward from the reward function correspond to the k^{th} state and action, γ is the learning rate, which is either 0.5 or 0.9, $\max_{a'} Q(s_{k+1}, a')$ is the maximum Q value correspond to an action of $(k+1)^{th}$ state and $Q_{k+1}(s_k, a_k)$ is the updated Q value corresponding to k^{th} state and action value.

- **Robot Movement Restrictions:**

To deal with the boundary conditions in the robot movement, I have put below logics.

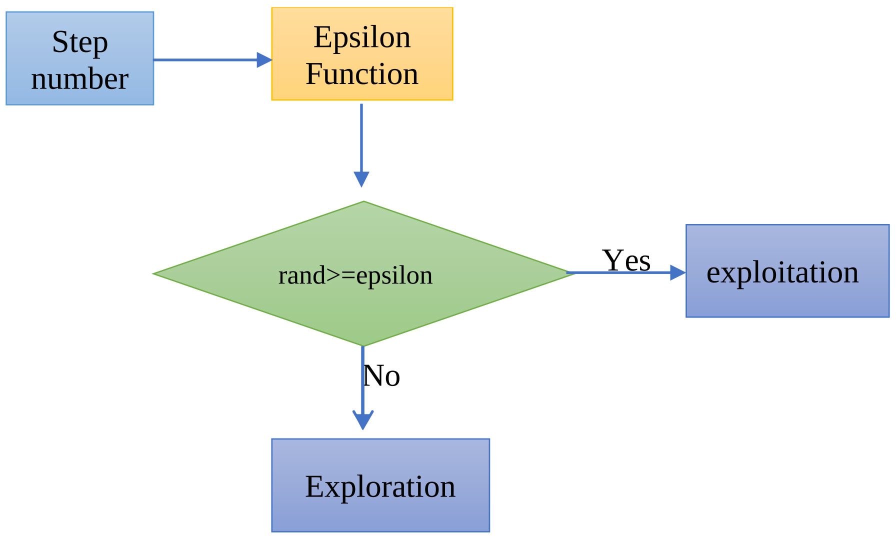
For left and right boundary, I have two predefined list which states the state numbers which cant allow left or right movement.

```
                    lb=[1,2,3,4,5,6,7,8,9,10];
                 rb=[91,92,93,94,95,96,97,98,99,100];
for the upper and lower boundary states I have used below two logics:
                 if mod(s,10)==1 (for lower boundary)
                 if mod(s,10)==0 (for upper boundary)
```

Hence, when robot state reaches to those states it will append the state to a flag list and then remove those from the actual action list. Exploration and exploitation will happen on the updated actual list.

- **Exploration and exploitation procedure:**

In greedy policy the agent has to either explore new states or have to rely on the old state actions. Neither of them alone can’t assist agent to reach to the final state. Since if the robot always chooses exploration, then it will choose an arbitrary path all the time. Whereas during exploitation it will always choose the best action chosen for that state. In that the agent will not be able to find the optimal path. To overcome this, we have different functions which will produce different values for different functions. The selection of exploration and exploitation will be done via below condition.



Where rand is generating a random float value in between 0 and 1. There are four types of epsilon function which is provided to us.

The function is designed in such a way that the epsilon will have higher value for very few steps and that in turn will reduce the chances of having exploration.

- **Optimal policy generation:**

I have used an optimal policy function which will be called after each successful trial. Here successful defines the robot has reached the state 100 and that can be optimal or not. Hence, the Q state of these trials will be taken into consideration and then the policies will be selected against the maximum Q values.

$$\Pi = \underset{a}{arg\,max} Q(s,a)$$

And these policies corresponding to a state have some rewards and when the state is being selected as optimal or if the robot can reach to a final state using some of those polices then the final return will be calculated depending on a discount factor variable and the rewards.

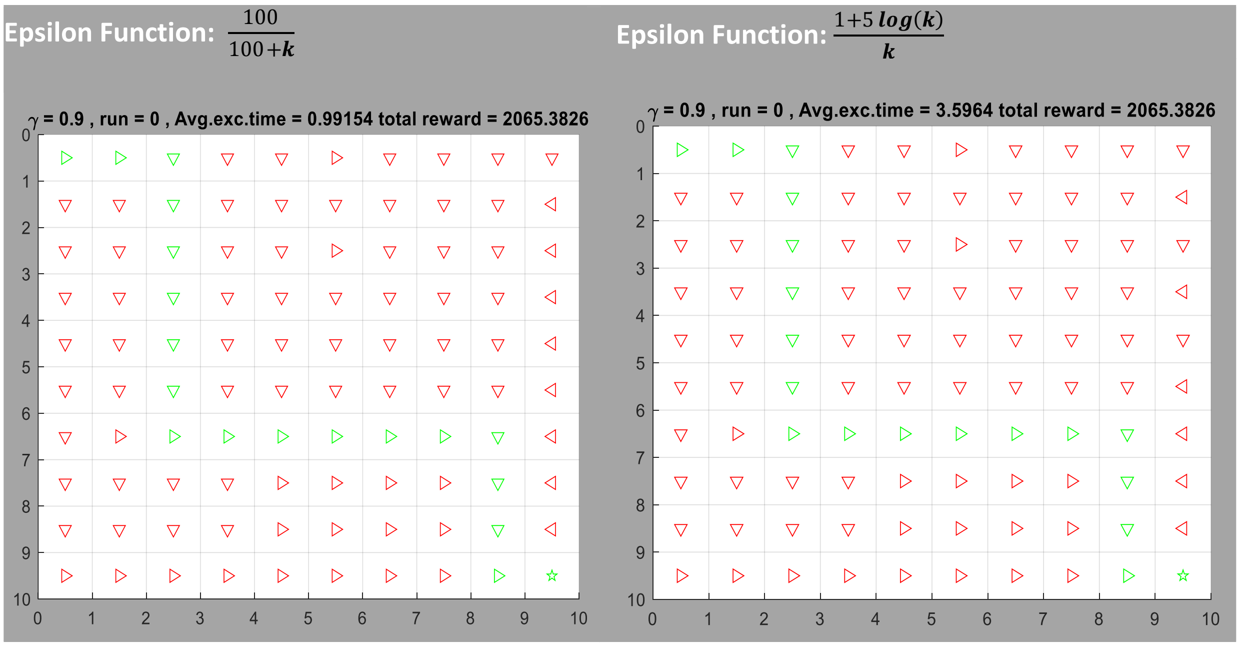
Hence, $R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$, where γ = discount factor and r = reward of a certain state.

- **Result:**

This final return corresponds to the optimal path for that run. When all the runs are completed, the maximum optimal policy reward is the reward of the optimal path.

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	-	0.99154
$\frac{1+\log k}{k}$	0	0	-	-
$\frac{1+5\log(k)}{k}$	0	10	-	3.5964

Optimal Path:



Reward for the optimal path:

Epsilon, alpha Function: $\frac{100}{100+k}$

All Actions:
[2,3,2,3,3,2,2,2,2,2,2,3,3,3,3,3,2,2,2,1,3,3,3,3,3,3,2,3,2,1,3,3,3,3,3,3,2,3,2,2,3,3,3,3,3,3,2,3,2,2,2,3,2,3,
3,3,2,2,2,2,3,3,3,3,3,3,2,2,2,2,3,3,3,3,3,3,2,2,2,2,3,3,3,3,3,3,3,3,2,3,3,4,4,4,4,4,4,4,1]

Epsilon, alpha Function: $\frac{1+5\,log(k)}{k}$

All Actions:
[2,3,2,3,3,2,2,2,1,2,2,3,3,3,3,3,2,2,2,2,3,3,3,3,3,3,2,2,2,2,3,3,3,3,3,3,2,3,2,2,3,3,3,3,3,3,2,3,2,2,2,3,3,3,
3,3,2,2,2,2,3,3,3,3,3,3,2,2,2,2,3,3,3,3,3,3,2,2,2,2,3,3,3,3,3,3,3,3,2,3,4,4,4,4,3,4,4,4,1]

For both the functions:

Optimal Paths: [1,11,21,22,23,24,25,26,27,37,47,57,67,77,87,88,89,90,100]

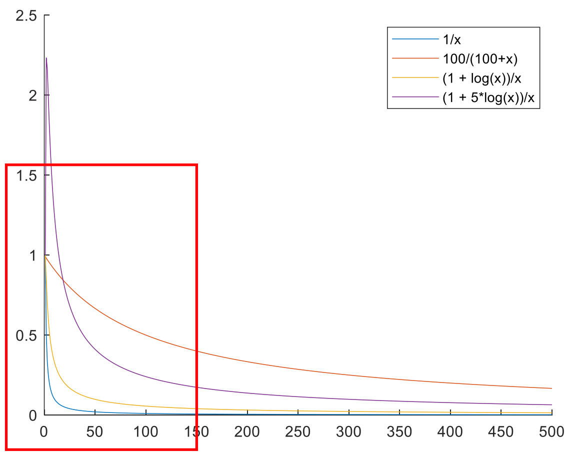
Optimal Rewards of the above paths:

26	**74.7**	**60.75**	0	0	0	0	0	0	0
0	0	**1.45**	0	0	0	0	0	0	0
0	0	**13.77**	0	0	0	0	0	0	0
0	0	**4.72**	0	0	0	0	0	0	0
0	0	**1.59**	0	0	0	0	0	0	0
0	0	**0.47**	0	0	0	0	0	0	0
0	0	**18.94**	**25.18**	**26.15**	**35.14**	**36.99**	**23.38**	**13.95**	0
0	0	0	0	0	0	0	0	**6.79**	0
0	0	0	0	0	0	0	0	**27.79**	0
0	0	0	0	0	0	0	0	**1667.55**	0

The above table shows how the reward values are changing in the direction of optimal path.

➢ Comments:

From the above results, it can be observed that the first and third epsilon function is failing to produce any optimal path for both the gamma or discount factor scenario. The plot of different functions is shown below:



Above plot shows how the function output changes with the k values which varies from 1 to 100. The plot illustrates that the first and third function always has low values. For very short period of time, it has values more than 0. Whereas the value pattern is different for the second and fourth function. Since the fourth function has values from 1 to around 0.3. It gives the robot enough chance to explore as well as exploit. Similarly, the fourth function does the same although it has values mostly below 0.5. Hence it takes more time to reach the final state since most of the time robot is exploiting. The red area of the figure shows the higher chance area of having exploration.

Although for both functions the optimal paths were achieved when the discount factor is 0.9. According to me, it can be because when the discount rate was 0.5, difference between the optimal policy and the current state is low, and this led to a less update in the current state.

$$r_{k+1} + \gamma \max_{a'} Q(S_{k+1}, a') \Rightarrow Estimate\ of\ Q^*(s_k, a_k)$$

Temporal difference = $Estimate\ of\ Q^*(s_k, a_k)$ - $Estimate\ of\ Q(s_k, a_k)$

Since this temporal difference for discount rate of 0.5 is lesser than the difference for 0.9. and the update will slower. That's why the agent is not able to reach to the final state before that the alpha is becoming greater, or the step is crossing the max step value.

Also, we know that the Q value will only converge when,

(1) $\sum_{k=0}^{\infty} \alpha_k^2 < \infty$ and $\sum_{k=0}^{\infty} \alpha_k \to \infty$
(2) $all\ (s, a)$ pairs are (asymptotically) visited infinitely

The given functions are fulfilling the first conditions which can explained from the plot itself.

For the first and third function, due to their nature, they are not able to fulfil the second condition. They are always limited to either exploitation which will prevent them to visit all the states or the actions.

Since both the learning rate and the epsilon have same value, so for the first and third function the update of Q function will be very slow and after sometimes, there will be almost zero update since the function value is dropping with the increase in k value. Hence the Q value will be converged sooner without visiting all the states, Hence the run will be failed to give optimal policy.

Task2

For task2, the first thing which I tried was to increase the epsilon value of both first and third function by 0.6. to have good chance of both exploration and exploitation. The reason of choosing 0.6 is trial and error process. I tried from 0.1 to 0.9 which didn't give me good result in term of execution time or the total number of goal state reaching runs. So, I had below resulted with the given reward function.

❖ Trial 1: epsilon function +0.6, With step max = 350

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	-	13.68
$\frac{100}{100+k}$	0	0	-	-
$\frac{1+\log k}{k}$	0	10	-	6.3912
$\frac{1+5\log(k)}{k}$	0	3	-	10.25

- Trial 2: With trial = 1000, step max= 350 and keeping all the other things same

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	-	0
$\frac{100}{100+k}$	0	10	-	0.7588
$\frac{1+\log k}{k}$	0	0	-	0
$\frac{1+5\log(k)}{k}$	0	9	-	2.818

I also tried with adding first and third function since adding them will having some increase in the values.

New Function = $(\frac{1}{k} + \frac{1+\log k}{k} + 0.5)$ whose lowest average execution time is 0.65 secs from different attempts.

The test was done with trial = 2000, step max = 350, gamma = 0.8. The function helped the agent reach to the goal for 7 out of 10 runs.

Even, I tried the function $exp(-0.001 \times k)$, but the execution time was around 4 to 5 secs in average.

Hence, as the task2 experiment I would like to keep the second given function and this new function to evaluate the speed of the learning with gamma = 0.9.

I have created one function to generate a new reward set which is mentioned on the instruction set as qevalreward which includes some gaussian noise with the actual given reward set. **After I tried with the new function on this new reward set the average execution time is around 0.7 secs. According to my analysis this is the fastest way of getting the optimal policy.**

Reward function: rewardgeneration()

Gamma: 0.9, Trials: 1800

Alpha>0.009

Step max:350

Function $(\frac{1}{k} + \frac{1+\log k}{k} + 0.5)$

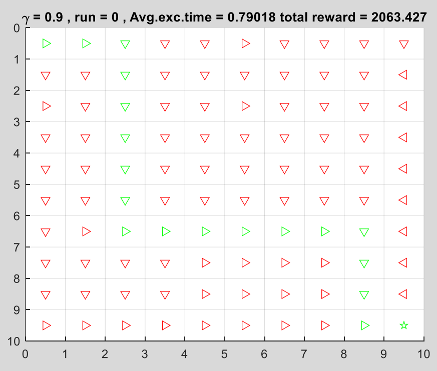
Reward function: rewardgeneration()

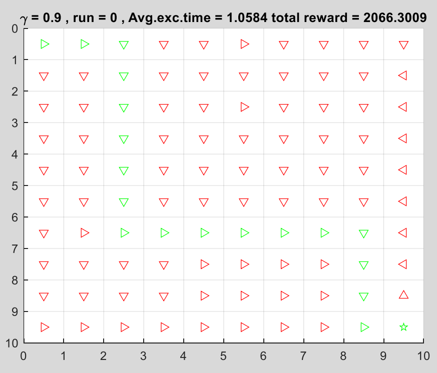
Gamma:0.9, Trials: 1800

Alpha>0.009

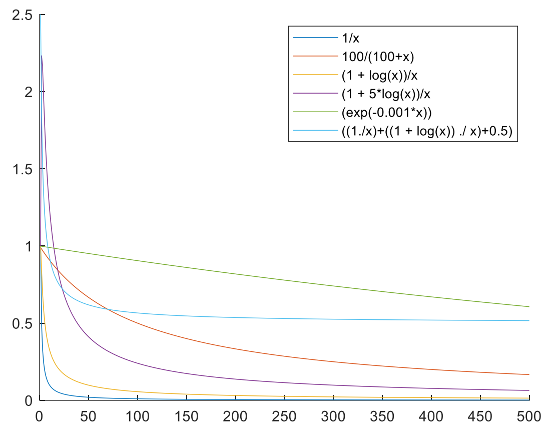
Step max:350

Function $\frac{100}{100+k}$





The total reward cannot be said if that is the optimal or not since the reward function is getting generated randomly every time the program runs.



The graph shows that the new function is settling to a value more than 0.5 after some times.

Appendix: MATLAB folders

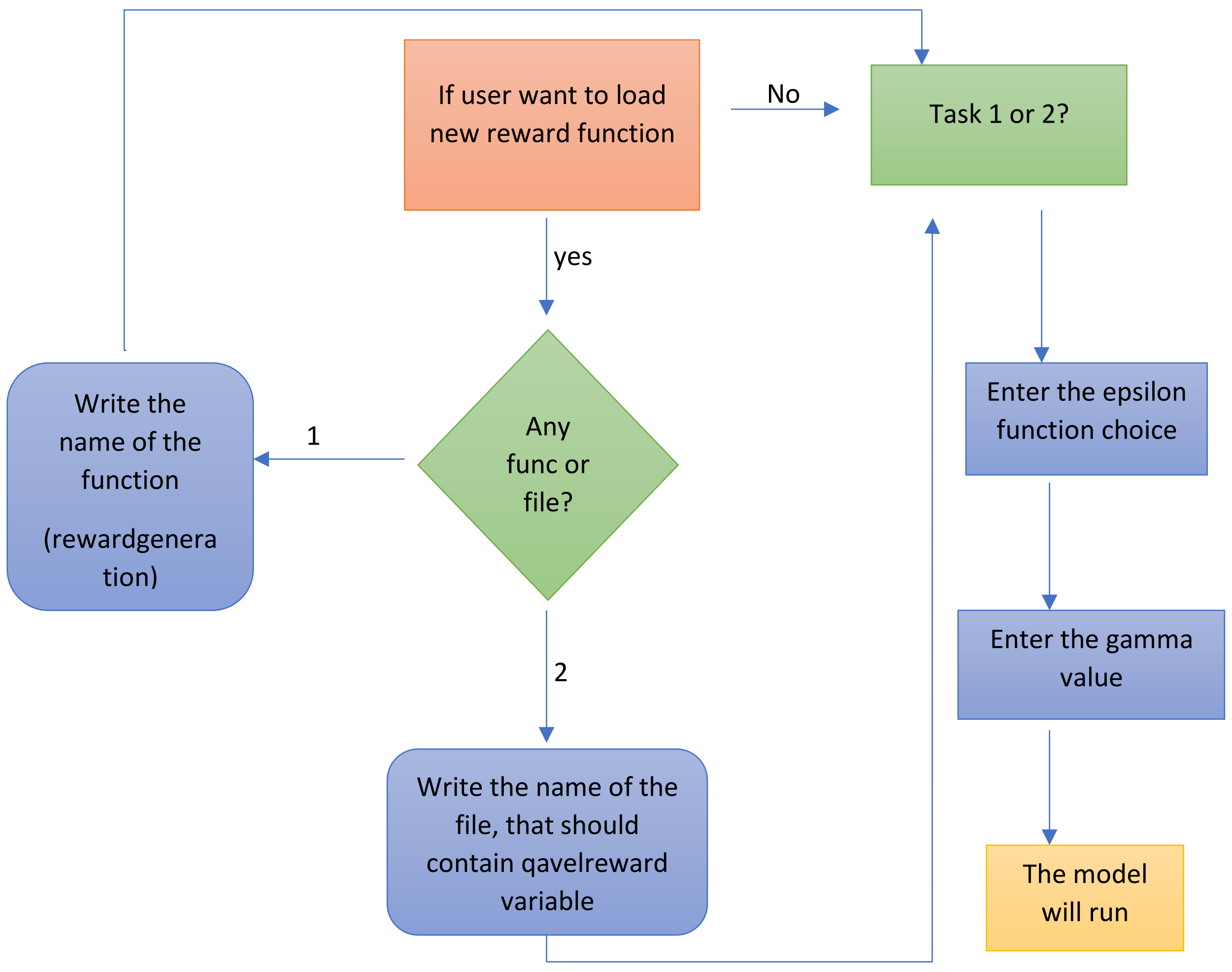
In the A0232541R folder, there are two sub folders

- m files
- demo paths
- PDF version of the report

The m files folder contains below files

- RL_main.m: this is the main file which will call task1 and task2 functions using different user provided inputs.
- RL_taks1.m corresponds to the task1 problem. This file contains a function which will provide below outputs:
    - max_info: A celltype output. Max_info contains all the x and y information of the graph, the labels, optimal states, optimal actions and policy action of state 1 to state 100.
    - Reach_times: will show the number of times the robot found the optimal path out of 10 runs
    - Run_times: shows the running time of the goal state reached states.
- RL_task2 correspond to the task2 problem and have the similar outputs. But it contains different epsilon functions.
- optimalPolicy.m: At the end of each run if the robot has reached state 100, then this file will check if the robot has found an optimal path or not. Give the outputs like the optimal states, the x,y axis values, optimal policy rewards etc.
- Rewardgeneration.m: generate a reward function using the given reward matrix by adding some gaussian noise.
- Task1.m: Given reward matrix.

Running Instructions of RL_main.m


If user want to load new reward function
No
Task 1 or 2?
yes
Any func or file?
1
Write the name of the function (rewardgeneration)
2
Write the name of the file, that should contain qavelreward variable
Enter the epsilon function choice
Enter the gamma value
The model will run