Q-LEARNING FOR WORLD GRID NAVIGATION

MODULE: EE5904 NEURAL NETWORKS

NAME: VENKATARAMAN MEENAKSHI

MATRIC ID: A0247125J

EMAIL ID: e0920432@u.nus.edu

Task 1:

The implementation of Q-Learning algorithm involves the following steps:

i) Loading the reward function that has been provided.
ii) Selection of the actions based on the probability factor.

$$a_k = \begin{cases} a \in \arg\max_{\hat{a}} Q_k(s_k, \hat{a}) & \text{with probability } 1 - \epsilon_k \\ \text{an action uniformly randomly selected from all other actions available at state } s_k & \text{with probability } \epsilon_k \end{cases}$$

iii) Finding the next state based on the action taken in the current state.
iv) Updation of Q-value.

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

Once the learning has been completed, the total reward is obtained as

$$R_{t+1} = \sum_{k=0}^{\infty} \left[\gamma^k r_{t+k+2} \middle| s_{t+1} = s' \right]$$

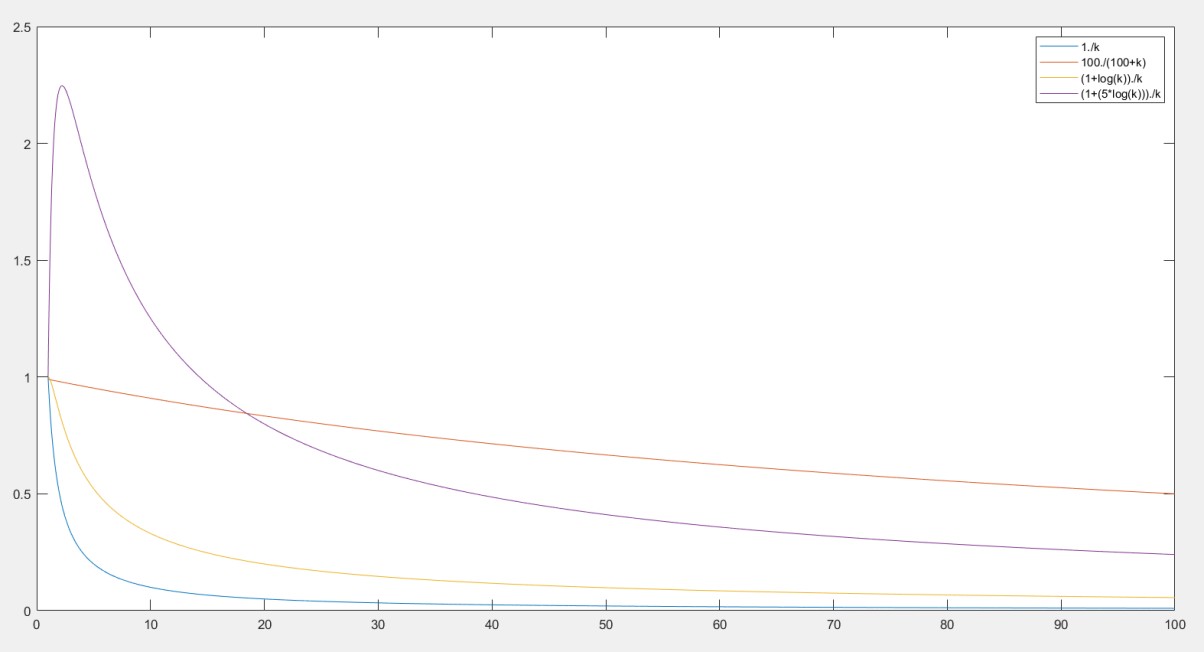
The following table provides the results that have been obtained from implementing the Q-Learning algorithm using the reward function that has been provided.

ε,α	Number of Goal Reached Runs		Execution Time	
	γ=0.5	γ=0.9	γ=0.5	γ=0.9
1/k	0	0	-	-
100/(100+k)	0	10	-	2.353932
(1+log(k))/k	0	0	-	-
(1+5*log(k))/k	0	10	-	3.329074

From the above table, it can be seen that for smaller discount factor of 0.5, the Q-Learning Algorithm seeks to achieve intermediatory steps rather than each the end state. Thus the system does not reach the end for any value of probability factor or learning rate when discount factor of 0.5 is used. The discount factor 0.9 on the other hand, which is on the higher end, aims to

achieve future rewards and thus yields better performance in the case of implementation using probability factors 100/(100+k) and (1+5*log(k))/k respectively.

The following plot figure provides the representation of the functions that have been provided:



From the above plots of the learning rates(probability factors), it is observed that functions 1/k and (1+log(k))/k decrease extremely fast, within 10 time steps and thus learning becomes very slow at earlier time steps as a result of which the system does not reach the end state when these functions are used. On the other hand, 100/100+k as well as (1+5log(k))/k undergo more gradual decrease as the time steps progress and thus learning slows down only at later time steps, thereby allowing the step to learn faster in order to reach the desired end state. Moreover, these functions also facilitate more exploration in the initial states with more exploitation towards the end which is more helpful since states attain high values of rewards towards the end of the learning process while other functions undergo exploitation from very early stages where states are still in the learning process as can be see above. From the table results as well as the plots shown above, it can be seen that when the value of discount factor chosen is on the lower end and when the learning rate undergoes decay to very low values at a faster rate, the update in Q value will be negligible at earlier states thereby resulting in local convergence at intermediate states as a result of which the system will not be able to attain its goal at the end state. Thus a combination of higher value of discount factor as well as the choice of learning rate having slower decay will help in better learning.

The following image shows the optimal actions that have been obtained for each state using the maximum of Q values obtained in each state.

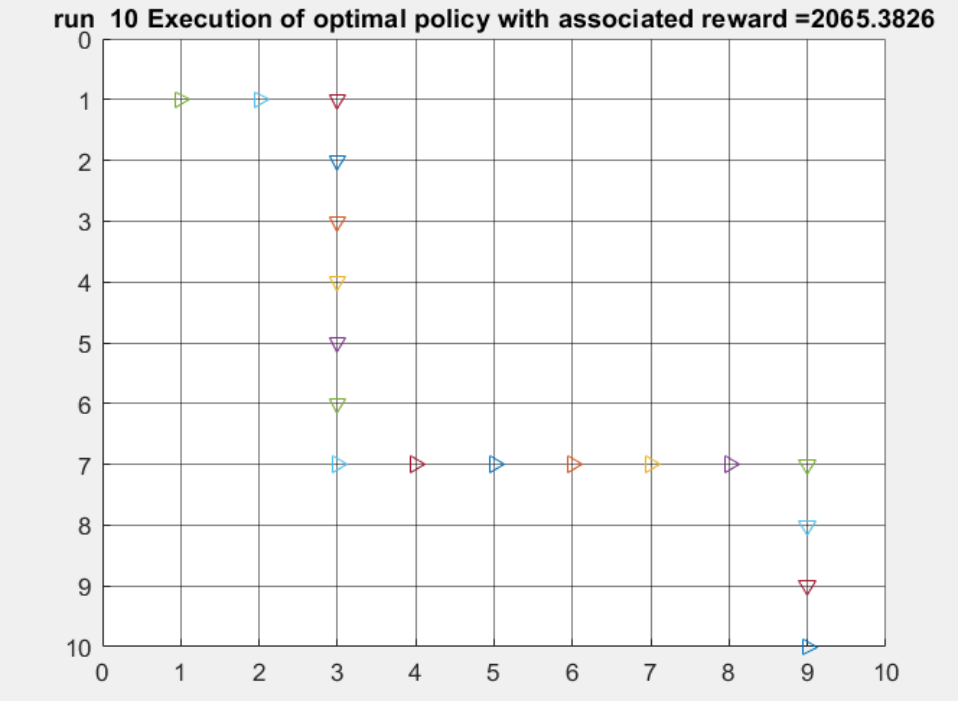
State	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Action	2	3	2	3	3	2	2	2	2	2	2	3	3	3	3	3	2	2	2	1	3

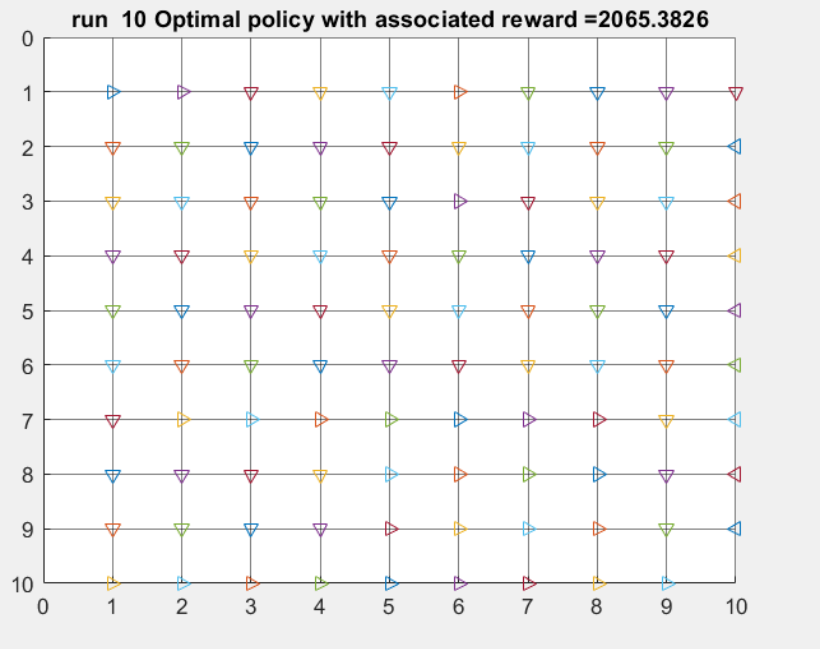
22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46
3	3	3	3	3	2	3	2	1	3	3	3	3	3	3	2	3	2	2	3	3	3	3	3	3

47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71
2	2	2	2	2	3	2	3	3	3	2	2	2	2	3	3	3	3	3	3	2	2	2	2	3

72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
3	3	3	3	3	2	2	2	2	3	3	3	3	3	3	3	3	3	2	3	4	3	4	4	4	4	4	4	1

The given figures below show the path that is followed by the optimal policy obtained as well as the optimal policy for each state. It is seen that the total reward obtained is found to be 2065.3826.





Task 2:

In order to implement Q-Learning using own probability factor and discount factor, the results obtained from Task 1 are taken into consideration. It can be seen that when the rate of decay for learning is slower and discount factor is higher as seen in the case of implementation using functions 100/(100+k) and (1+5*log(k))/k and discount factor of 0.9, the system reaches the end state for all the runs within 100 time steps. Thus, the learning rate which is assumed to be equal to the probability factor is designed in a similar way such that the decrease in learning is slower. The discount factor on the other hand is chosen such that the system aims to achieve maximum future reward rather than maximize its intermediate rewards.

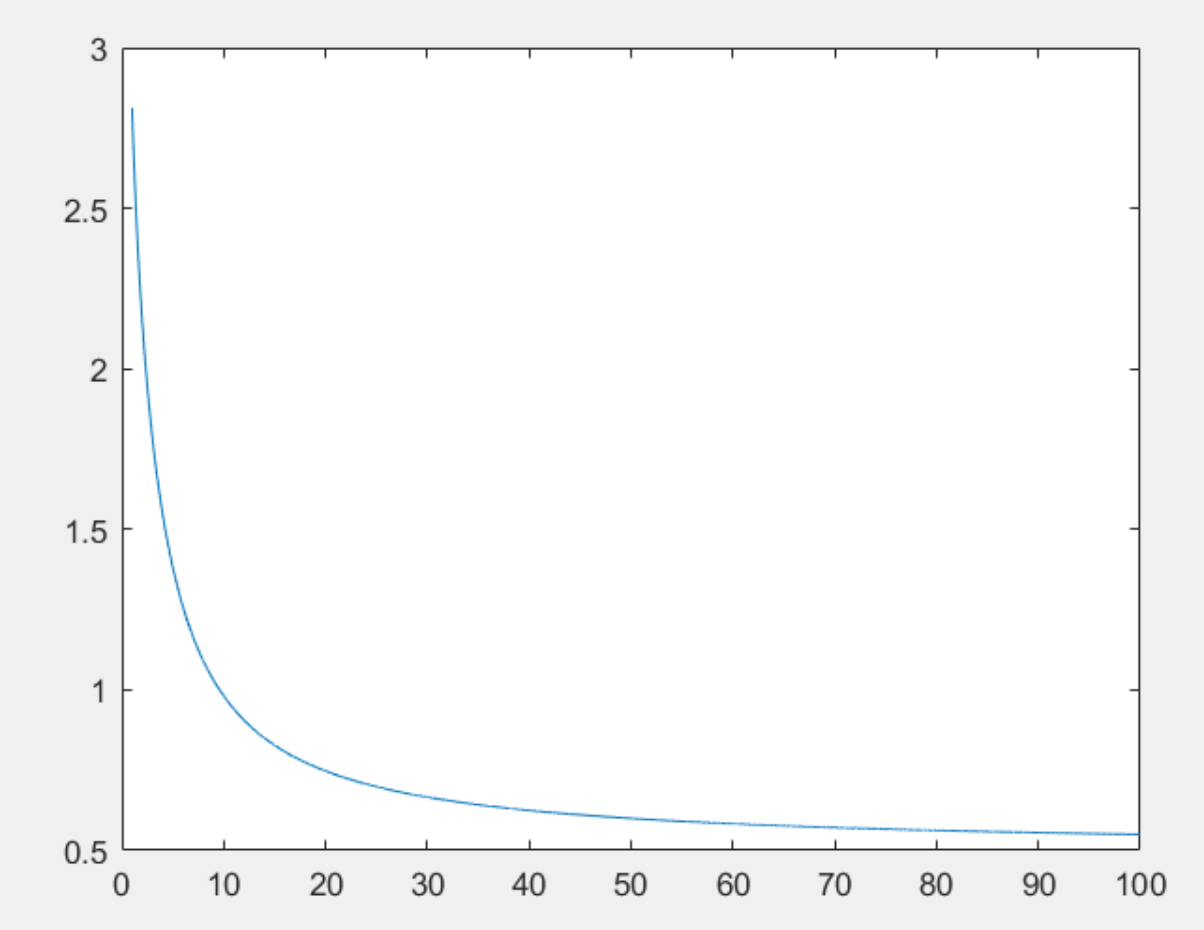
The probability factor function used is:

$$\varepsilon_k = \operatorname{acsch}\left(\frac{k}{5}\right) + 0.5$$

Where acsch(x) refers to the arc cosecant function which is in turn defined as:

$$\ln\left(\frac{1}{x} + \sqrt{\frac{1}{x^2} + 1}\right)$$

The following figure shows the plot of the function:



The following table provides the performance of the system using different discount factors:

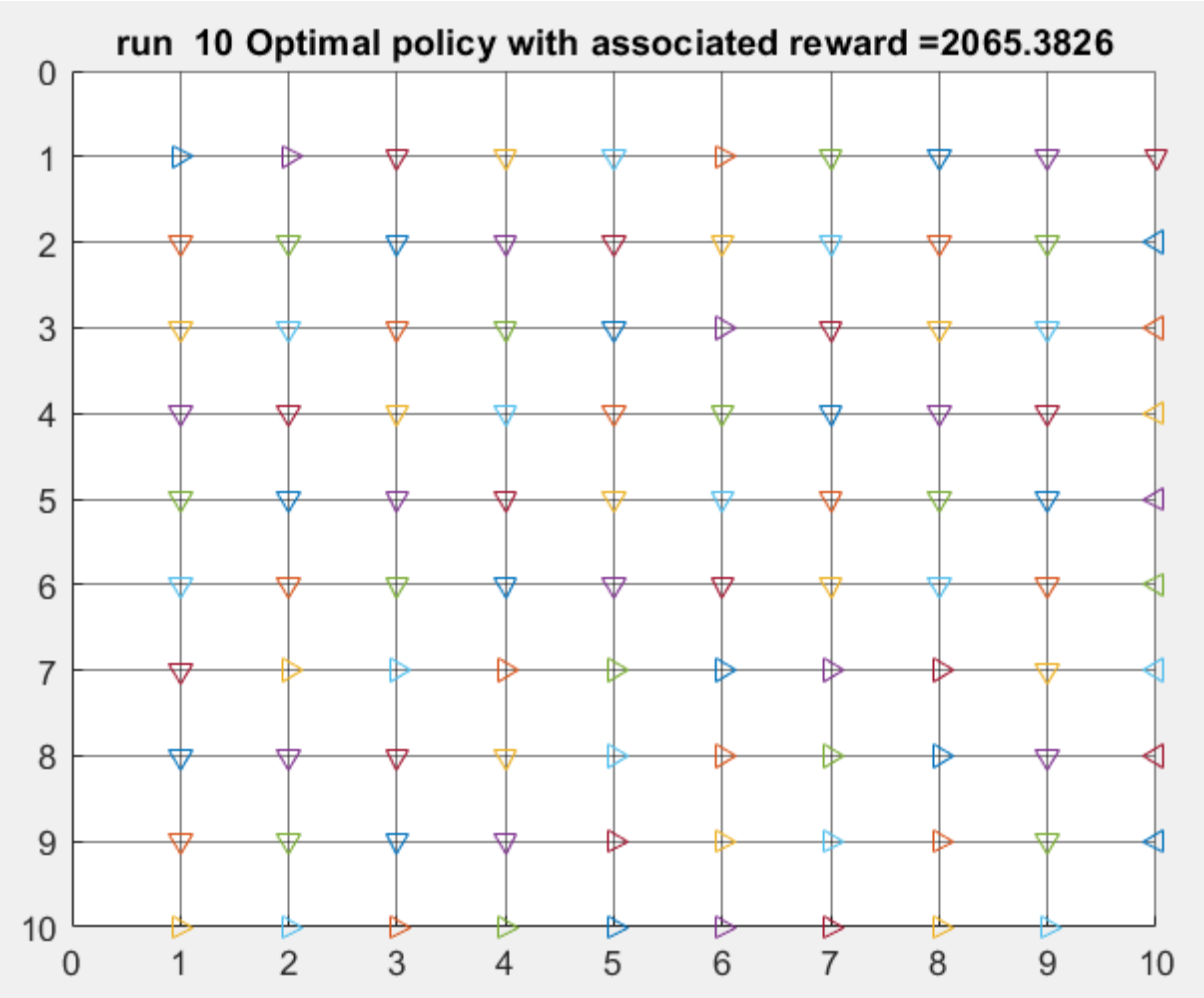
Discount Factor	Task 1 Reward Function		Sample Reward Function	
	Number of Times Goal Reached	Average Execution Time	Number of Times Goal Reached	Average Execution Time
0.5	0	0.31202773	0	0.42996369
0.6	0	0.43715366	0	0.7138602
0.7	10	0.30096838	0	0.45545544
0.8	10	0.25555976	0	0.4494849
0.9	10	0.29545835	10	0.42465288
0.95	10	0.39382428	10	0.63472256

From the table above, it can be seen that implementation using Task 1 Reward Function is able to reach the end state for all 10 runs using discount factors 0.7,0.8.0.9 as well as 0.95 while implementation of Q-Learning using discount factor of 0.9 and 0.95 is found to provide optimal result in terms of the number of runs for which the system reaches the end state. Since goal achieved for all 10 runs using task 1 reward function as well sample reward function is observed only for discount factors and 0.9 and 0.95, the value is selected from these 2 options. Since average execution time for implementation using 0.9 discount factor is found to be lesser, 0.9 is chosen to be the discount factor for implementation.

This function is then evaluated using a sample reward function and tested:

```
Columns 1 through 18

   2     3     3     3     3     3     3     3     3     2     2     3     3     3     3     3     2     3

Columns 19 through 36

   3     2     3     3     3     3     3     3     2     3     3     2     3     3     3     3     3     3

Columns 37 through 54

   2     3     3     2     3     3     3     3     3     3     2     2     2     2     2     3     2     3

Columns 55 through 72

   3     3     2     2     2     2     3     3     3     3     3     3     2     2     2     2     3     3

Columns 73 through 90

   3     3     3     3     2     2     2     2     3     3     3     3     3     3     3     3     3     2

Columns 91 through 100

   3     4     4     4     4     4     4     4     4     1
```

The given figure below shows the optimal policy chosen for each state:



The given figure below shows the optimal path that is taken using the optimal policy that has been obtained in order to reach the end state: