Computer Assignment Reports Reinforcement Learning

Andreas Stasinakis & Mim Kemal Tekin

March 6, 2019

Contents

Contributors: Mim Kemal Tekin(mimte666) & Andreas Stasinakis(andst745)

In order to pass the assignment you will need to answer the following questions and upload the document to LISAM. You will also need to upload all code in .m-file format. We will correct the reports continuously so feel free to send them as soon as possible. If you meet the deadline you will have the lab part of the course reported in LADOK together with the exam. If not, you'll get the lab part reported during the re-exam period.

Question 1

Define the V- and Q-function given an optimal policy. Use equations and describe what they represent. (See lectures/classes)

$$\hat{V}(s_t) = (1-\eta)\hat{V}(s_t) + \eta\gamma^m r_{t+k}$$

$$\hat{Q}(s_k, a_j) = (1-\eta)\hat{Q}(s_k, a_j) + \eta\Big(r + \gamma \max_a \hat{Q}(s_{k+1}, a)\Big)$$

where
$0 \leq \gamma \leq 1$: Discount Factor
$0 \leq \eta \leq 1$: Learning Rate
m : States from Number

V- and Q functions are defines as above. V-function finds expected reward just for a certain state in the future. In example, reward of going up movement and all other directions if we have, so if we have left and right movements from the s_t we create a case condition for these 2 movements to see all possible rewards from s_t. Q-function calculates the expected reward for each action in each state. So when we say $V(s_t)$, we mean the expected reward that we can obtain from the movement form s_t to another states. But we use Q-function with this notation to describe each movement from s_t: $Q(s_t, \text{up})$, $Q(s_t, \text{down})$. In Q-function learning we take the optimal movement from all posibble movements and we have also ϵ (Exploration Rate) to explore the map with a probability. This rate provide us to move to different direction from the only optimal one.

Question 2

Define a learning rule (equation) for the Q-function and describe how it works. (Theory, see lectures/classes)

The learning rule(equation) for the Q- function defined before is:

$$\hat{Q}(s_k, a_j) \leftarrow (1-\eta)\hat{Q}(s_k, a_j) + \eta(r + \gamma\hat{V}(s_{k+1})) = (1-\eta)\hat{Q}(s_k, a_j) + \eta(r + \gamma max_a \hat{Q}(s_{k+1}, \alpha)$$

, where $\hat{Q}(s_k, a_j)$ is the previous estimate, η is the learning rate, r is the reward from moving to state k to state $k+1$ and a_j the action we choose.

In general, the updated $\hat{Q}(s_k, a_j)$ is the sum of two proportions. The first one is the previous estimator multiplied by 1 - learning rate. The second one is again the sum of two quantities. The reward and the product of the discount factor with the estimate of future optimal value. The last summation is multiplied by the learning rate.

Question 3

3. Briefly describe your implementation, especially how you hinder the robot from exiting through the borders of a world.

In this task we implement reinforcement learng in order to train a robot to find a specific target in different worlds. The procedure is simple but powerfull. We initialize the Q function(all the formulas we need are presented in previous tasks) with zero values. We also want to put some borders to the robot. More specific, you do not want the robot to up if it is in the top of the world. In order to achieve that we put -inf all the values, given an action, we do not the robot to choose. For example if the robot is in the las left row of the world, we put the reward for going left -inf. As a result, the robot will never choose to go in this direction. One can say that instead of an initial Q function of zeros can be used random numbers. This is also correct,

but in this case we have to change the reward for the terminal point equal to zero. We can skip that, starting with zeros in the initial Q function.

For each iterations(episode) we initialize the robot in a random position. The robot tries to find the goal for a given number of steps. For each step, the robot chooses a action given the Q function, its current possition and some probabilities. After going to the next state, we check if this state is valid and it is not the robot will find another valid action. Finally, we update the Q function, using the formula in previous task. The procedure stops if we find the terminal or after the total number of steps we use.

Question 4

4. Describe World 1. What is the goal of the reinforcement learning in this world? What parameters did you use to solve this world? Plot the policy and the V-function.

The first world is one static world and probably the easiest one to find the target. The goal of the the reinforcement learning in general is to find the cirle target. We should also train the robot in order to avoid the blue area. For example when the robot start inside this blue area should leave it directly because the feedback there is negative. We can confirm that from the policy plot. We solve this world using the parameters below:

Discount factor : 0.9

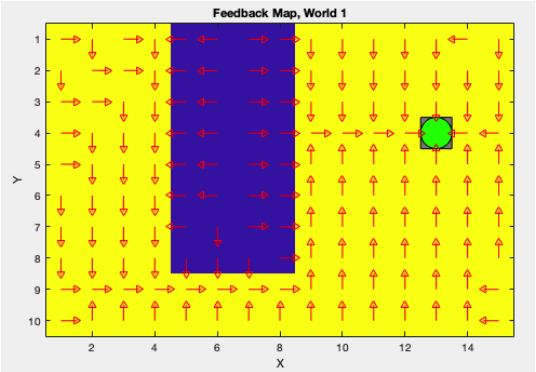
Learning rate : 0.9

Maximum steps : 100

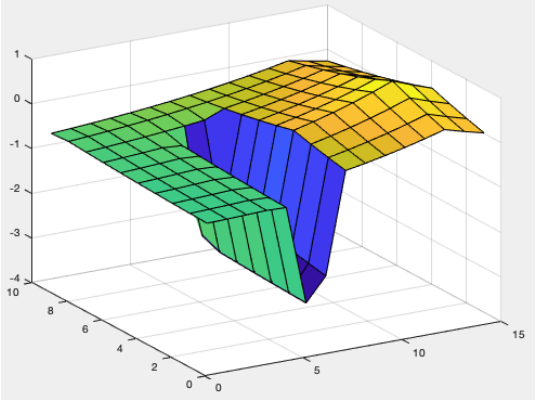
Total episodes : 1000

Exploration factor : 1 - number of current episode / Total episodes.

Policy:



V-function:



Question 5

Describe World 2. What is the goal of the reinforcement learning in this world? What parameters did you use to solve this world? Plot the policy and the V-function.

The second world is similar to the first one, but a randomness also occurs. That is also clear from the plots, because despite the fact that in the final policy there is no obstacle, the robot does not go directly to the target but sometimes it follows different path. In the initialization of this world, with 20% probability, it generates the state big negative feedbacks in the same blue area of world 1. This makes the robot learn to not go close to that area like in the first world and the robot always goes around to reach the goal state.

We solve this world using the parameters below:

Discount factor : 0.9

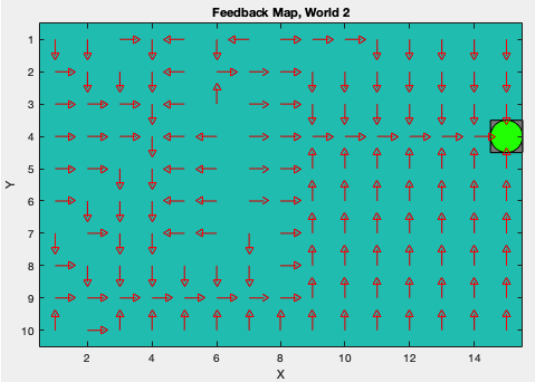
Learning rate : 0.2

Maximum steps : 100

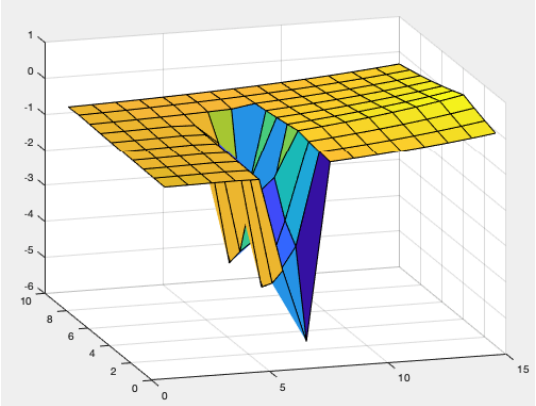
Total episodes : 1500

Exploration factor : 1 - number of current episode / Total episodes.

Policy:



V-function:



Question 6

Describe World 3. What is the goal of the reinforcement learning in this world? What parameters did you use to solve this world? Plot the policy and the V-function.

This world's logic is the same as world 1. It is staticly created, always same feedbacks and same starting point. We tried same parameters as world 1 and it worked well. Once the robot captures the shortcut between start and the goal, it starts to increase the reward of that path. Finally we got a solution that always uses that path, we can see the huge reward difference on the V-function plot.

Training parameters:

Discount factor : 0.9

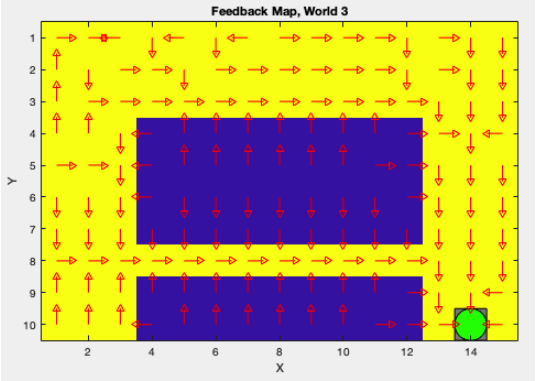
Learning rate : 0.9

Maximum steps : 100

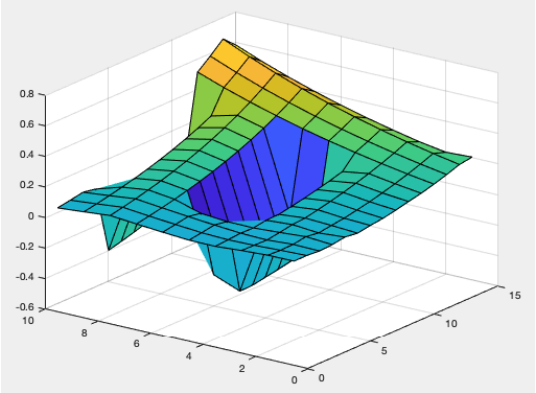
Total episodes : 1000

Exploration factor : 1 - number of current episode / Total episodes.

Policy:



V-function:



Question 7

Describe World 4. What is the goal of the reinforcement learning in this world? How is this world different from world 3, and why can this be solved using reinforcement learning? What parameters did you use to solve this world? Plot the policy and the V-function.

In this world we can see that the robot moves carefully against to blue area. It learned to move away from it and if it is next to the edge of it, it moves the opposite way. We can see this movement on the V-function plot. So we can conclude this to the feedbacks of those states. In the end the robot does not take the shortcut as in the world 3.

Training parameters:

Discount factor : 0.9

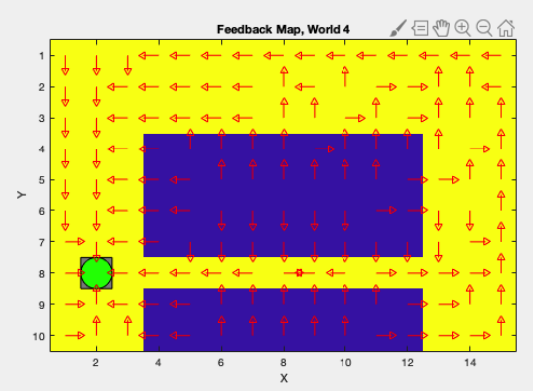
Learning rate : 0.2

Maximum steps : 100

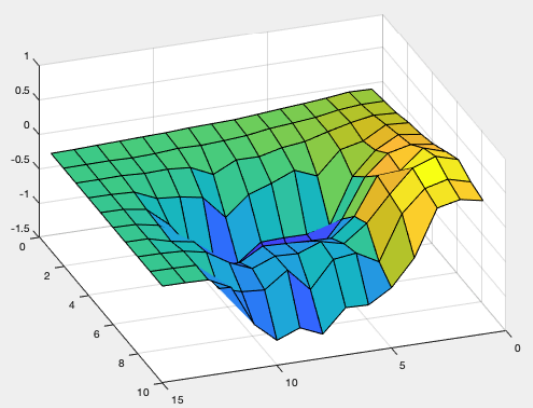
Total episodes : 2500

Exploration factor : 1 - number of current episode / Total episodes.

Policy:



V-function:



Question 8

8. Explain how the learning rate α influences the policy and V-function in each world. Use figures to make your point.

The learning parameter is really important factor for the reinforcement learning. It controls of how much of the new information will overlap the old one. More specific, a value close to 0 uses mostly the previous information, while a value close to 1 makes the model replace the oldest information with the new one. Higher learning rate might train faster, but on more complex problems it should be learned by many trials and slow, then we need smaller learning rate and longer trainin episodes.

For instance, world 1 and 3 were similar by their initialization and were easier to solve than 2 and 4. By this, we could use higher learning rates. As we can see there is no much difference between world 1 and 3. But world 2 had randomness at its implementation, so this made us to set lower learning rates to be able to train the robot to be precise in the both cases of initialization. We can see the difference between low and high learning rates below. World 4 seems like world 3 visually. But in world 4, there was different feedbacks in the states to train the robot to stay away from the restricted blue areas. In this world, it was the clearest one to understand the importance of learning rate. Because with higher learning rates it could not find the policy to reach target.

We change only learning rates of the experiments. First rows of each world are lower learning rate and second rows are higher learning rate

Training parameters:

Discount factor : 0.9

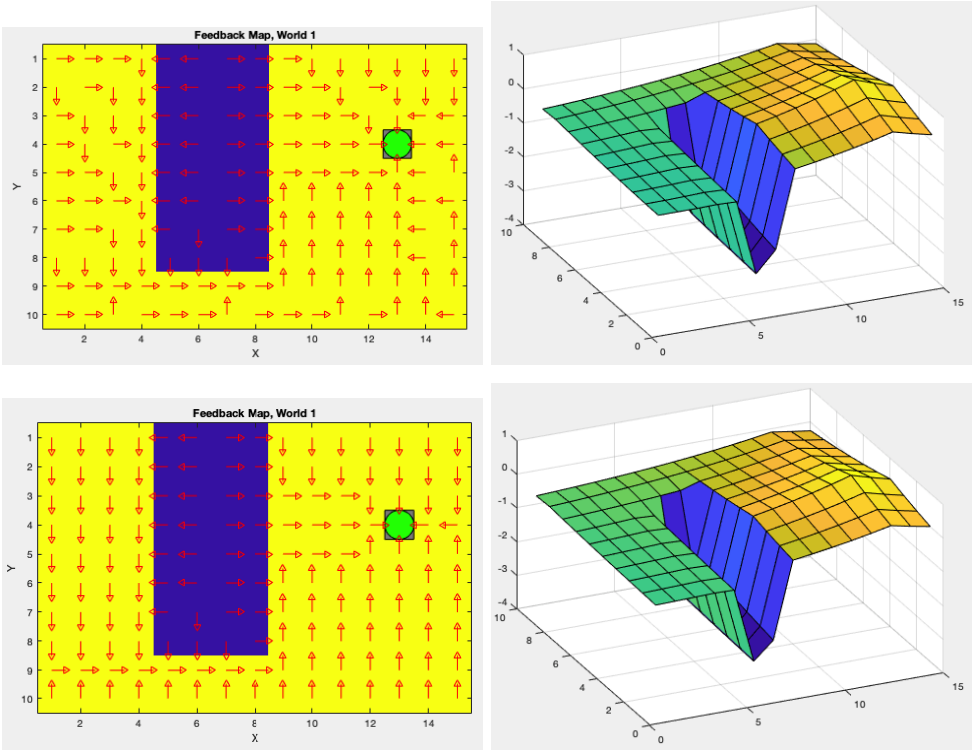
Learning rate : 0.2 (First rows) vs 0.8 (Second rows)

Maximum steps : 100

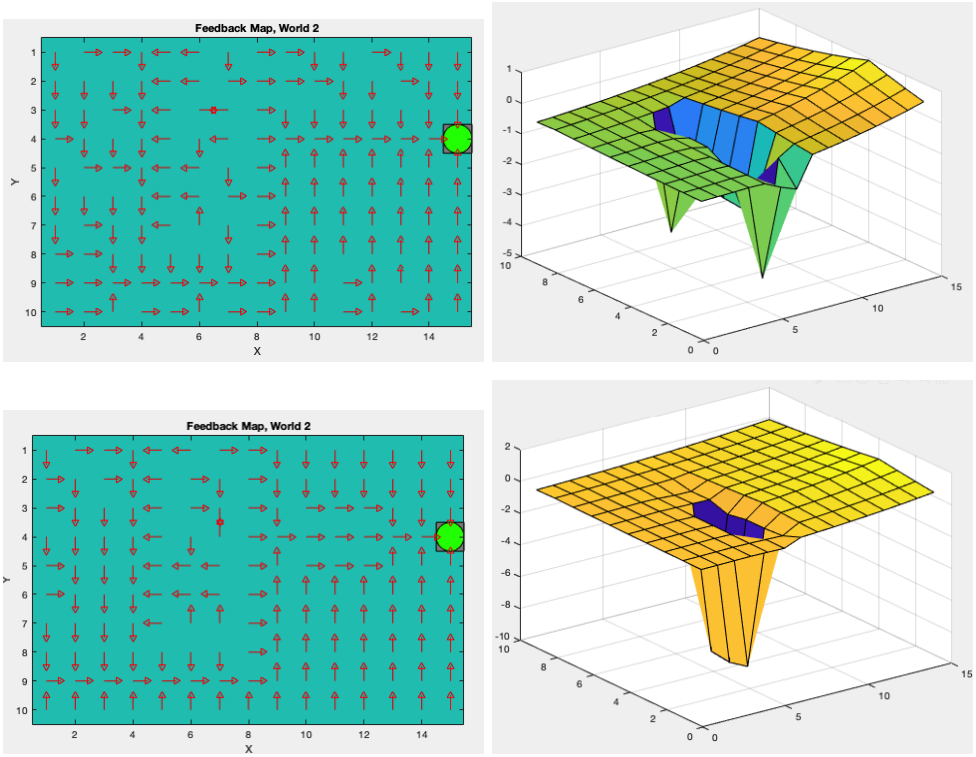
Total episodes : 2500

Exploration factor : 1 - number of current episode / Total episodes.

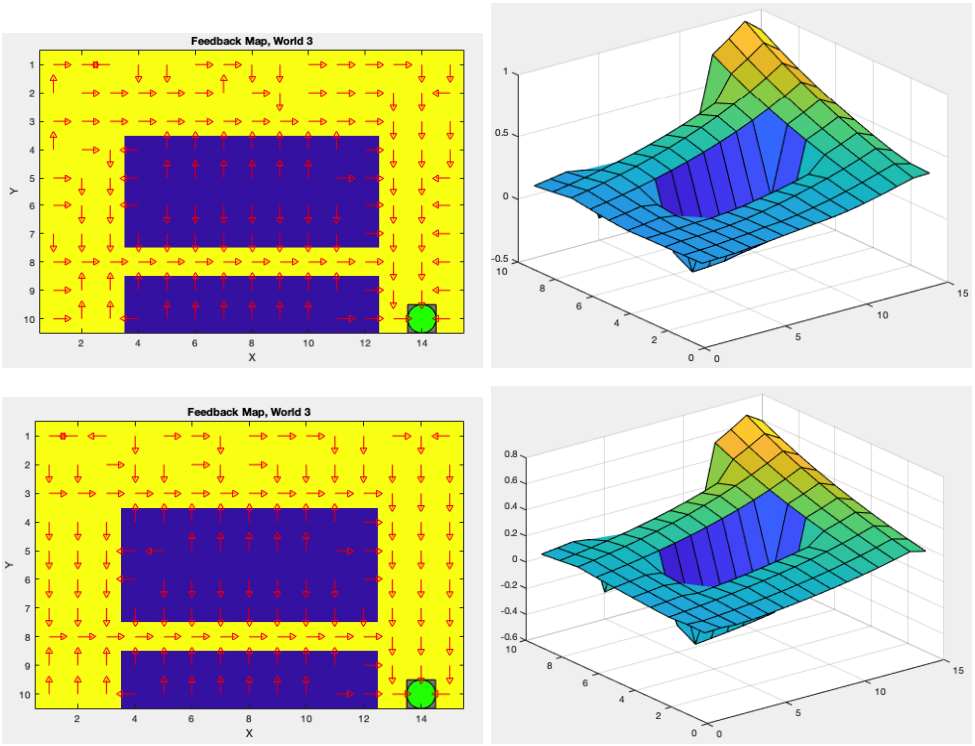
World 1



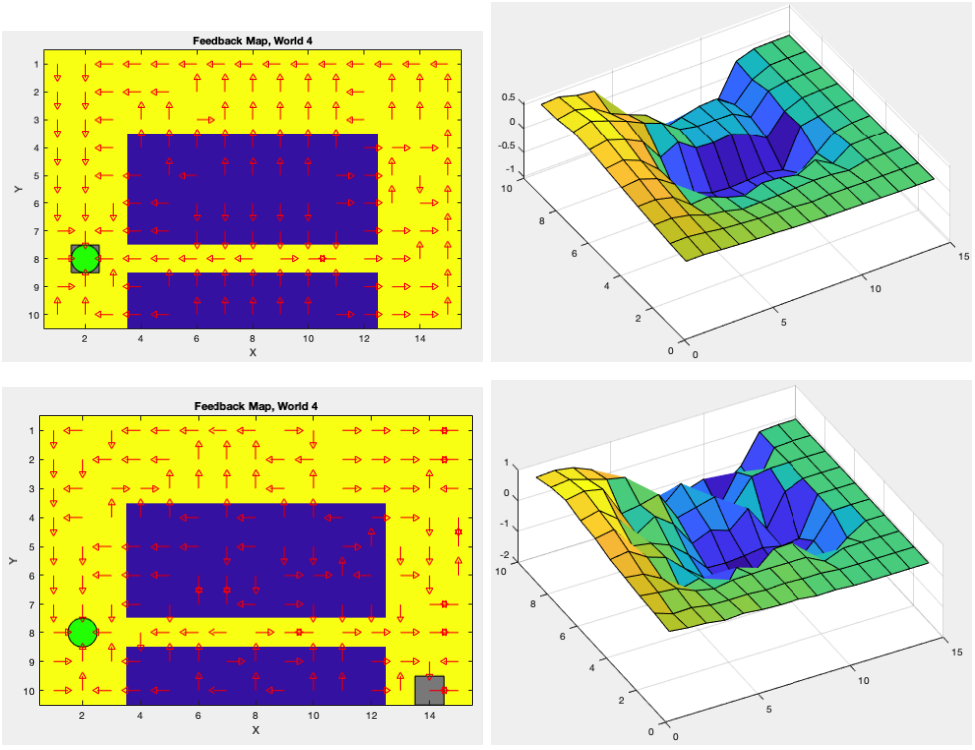
World 2



World 3



World 4



Question 9

9. Explain how the discount factor γ influences the policy and V-function in each world. Use figures to make your point.

The discount factor γ determines the importance of the future rewards. More specific, lower discount factor will give importance to short term rewards and higher discount factor will give the importance to long term rewards. This effect again possible to see V-function plots below.
In general, from V-function plots, we can see when we use low discount factors we get really sharp map. On the contrary, when we set higher discount factor, we get smoother maps. Most of the case, both settings work, but it will be easier to get failed when we have really close reward differences between states as in the case of low discount factor. So the policy can be changed by thinking of only short term rewards. When we use higher we train the robot with more confident because of higher difference between the adjacent states. Even in the world 4, the discount factor changes the resulting/expecting policy. Because the weights were prepared for avoiding the shortcut, but with short term rewards it gains importance and we can say that robot risks itself to earn as much as rewards in the short term to reach the goal.

Training parameters:

Discount factor : 0.1 (First rows) vs 0.9 (Second rows)

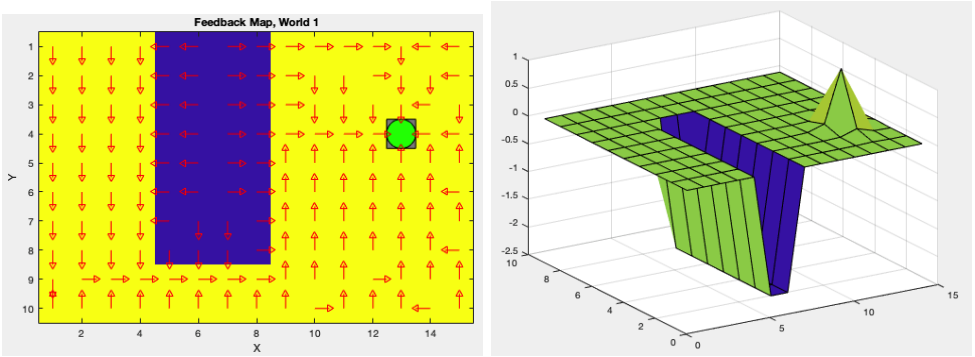
Learning rate : 0.2

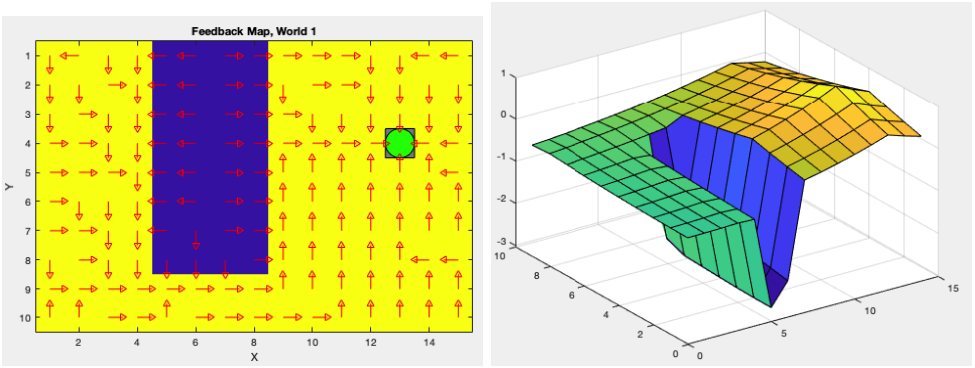
Maximum steps : 100

Total episodes : 2500

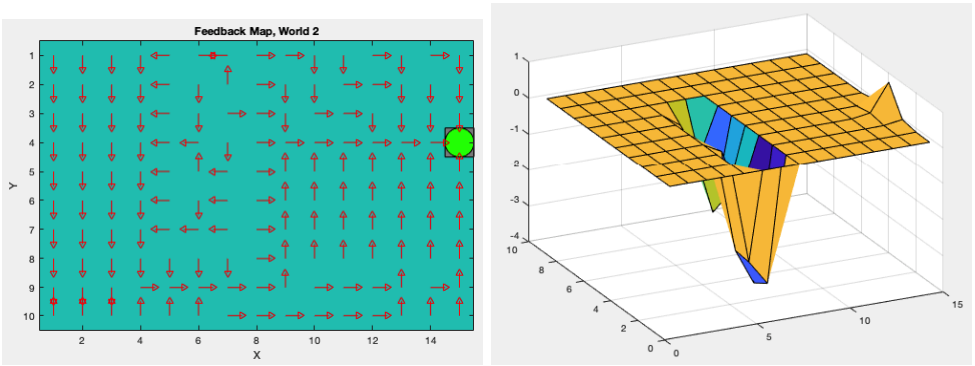
Exploration factor : 1 - number of current episode / Total episodes.

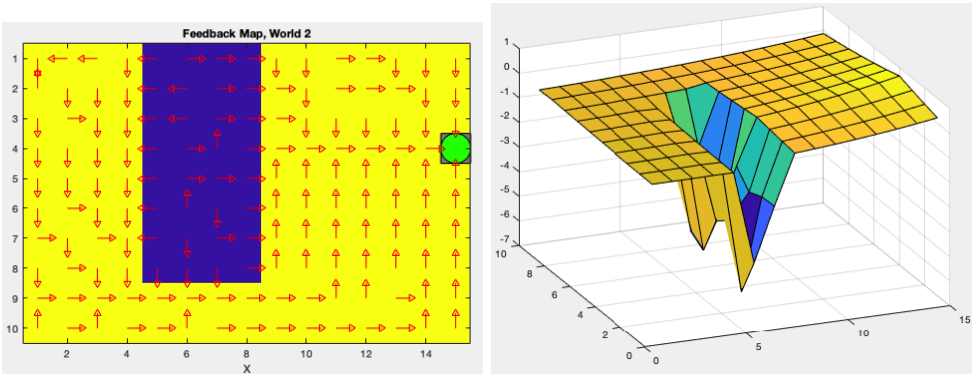
World 1



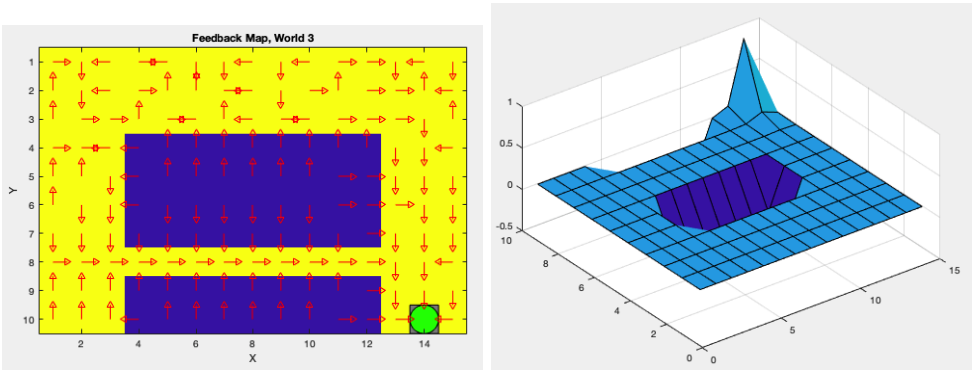


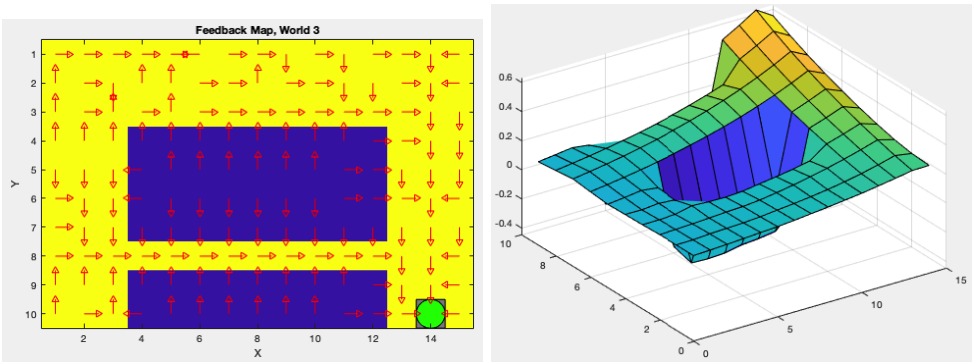
World 2



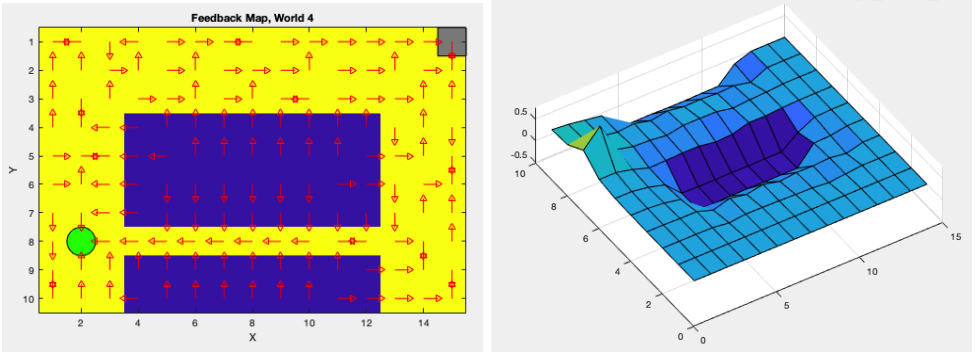


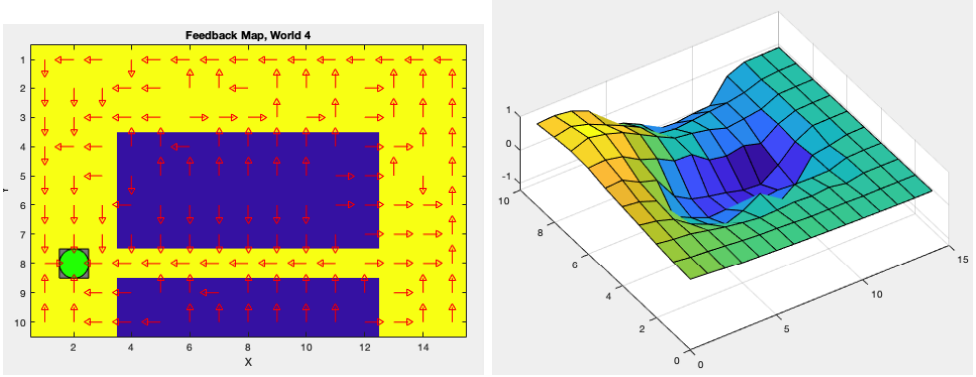
World 3





World 4





Question 10

Explain how the exploration rate ϵ influences the policy and V-function in each world. Use figures to make your point. Did you use any strategy for changing ϵ during training?

ϵ parameter provide us to explore the map during training. Q-function as usual chooses the optimal direction of the current state, but with exploration rate we have a probability to not choose always optimal direction. When we use *epsilon* we should modify the function that returns the optimum direction. It should find optimum direction and pick one direction randomly. After this, the function should return the random one by checking the exploration rate; if probability that is drawn more than ϵ the function should return the optimal action. In our training we decrease the *epsilon* by time in order to after one point stop to explore and make the current policy better.

We can see the differences between the low rate and high rate below on the plots. We again can talk about sharpness and smoothness. Higher exploration rate results a smoother map, because this rate will let the robot to explore different paths than the optimal one. Otherwise the robot cannot explore and once it found a way it will stuck on that path only, cannot learn another path. In the world 3, it is the clearest to realize what we just mentioned. The robot finds the highest reward on the one path and after that it uses that path during the whole training (it is 0.1 probability). But in the same world, higher exploration rate, we can see it is smoother, although the optimal result is the exactly same path. This means the robot visited other alternative paths too, but finally it found the optimal one.

Training parameters:

Discount factor : 0.9

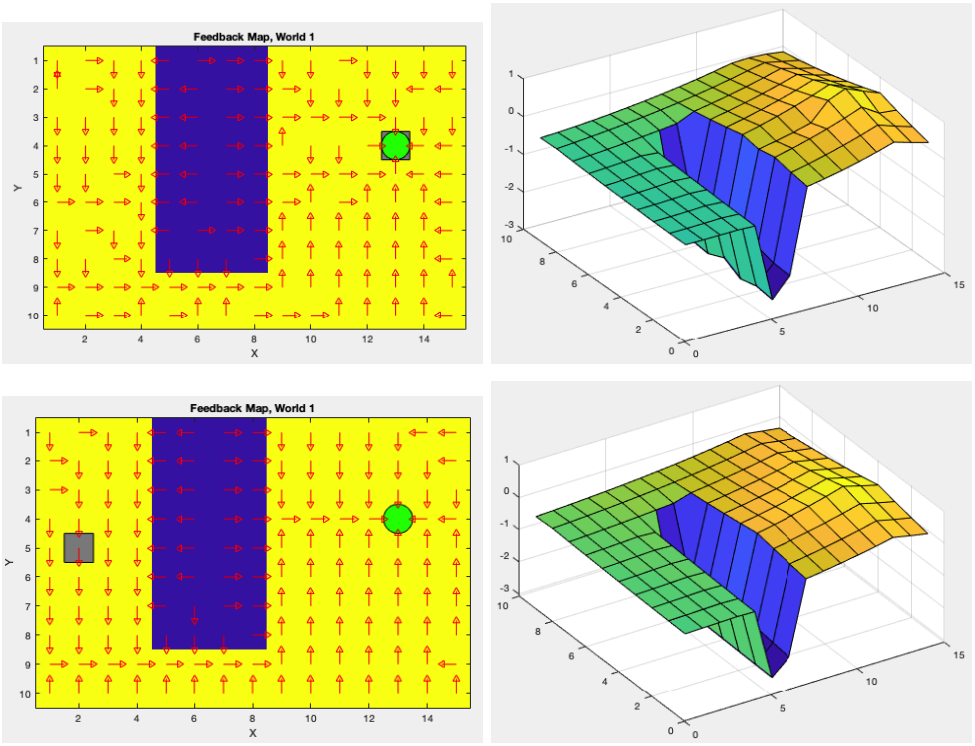
Learning rate : 0.2

Maximum steps : 100

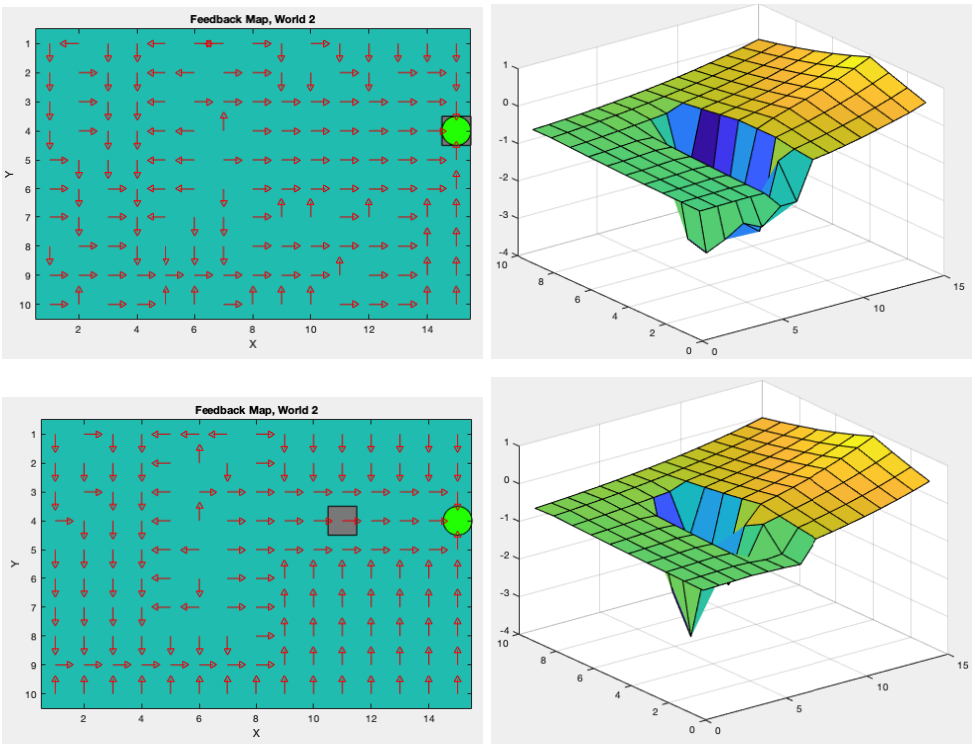
Total episodes : 2500

Exploration factor : 0.1 (First rows) vs 0.9 (Second rows)

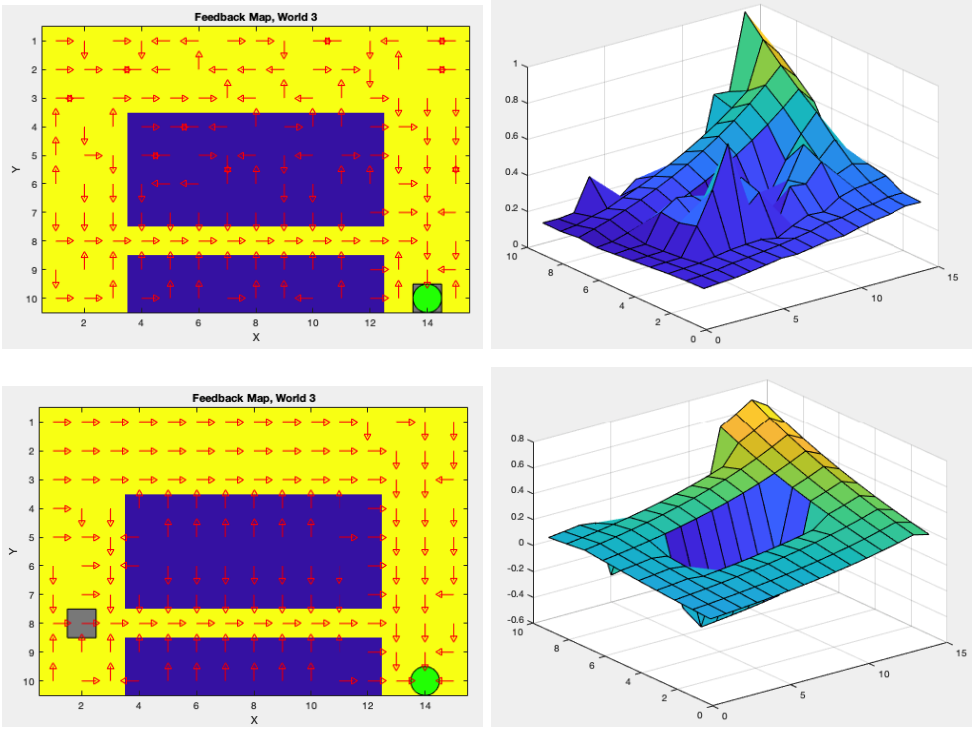
World 1



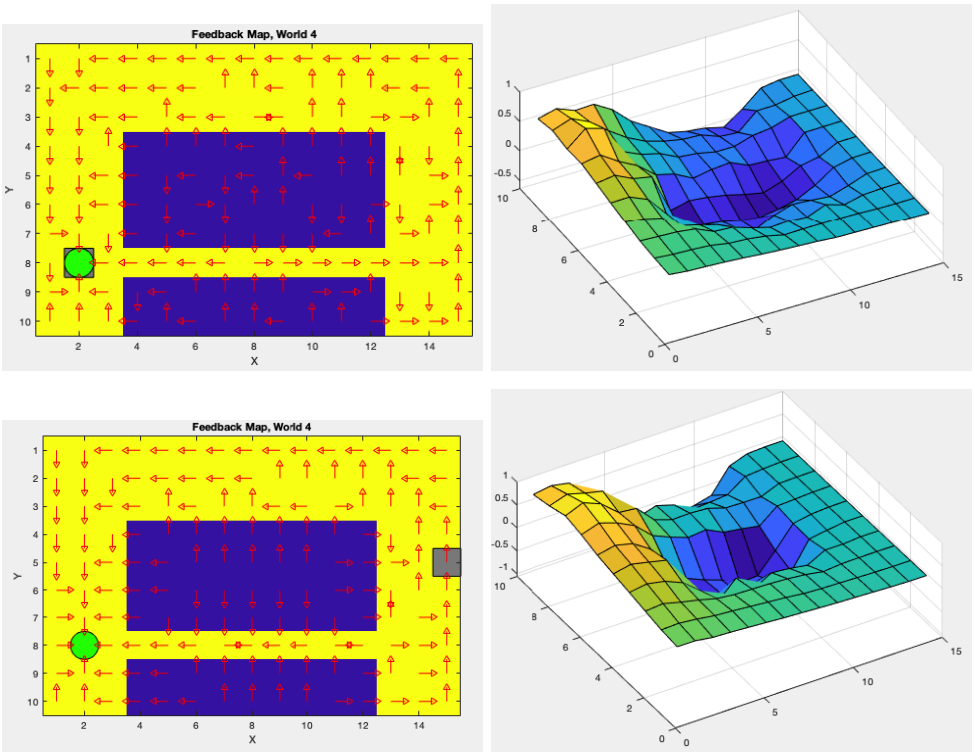
World 2



World 3



World 4



Question 11

11. What would happen if we instead of reinforcement learning were to use Dijkstra's cheapest path finding algorithm in the ''Suddenly Irritating blob''world? What about in the static''Irritating blob" world?

In the first world, static irritating blob, we do not have any randomness, so with reinforcement we can get close results with dijkstra. Dijkstra always finds the optimal path between one point and all the other points, but reinforcement learning can find optimal but it is not guaranteed. So for a static world using dijkstra will give us always the optimal solution which can be better than training reinforcement lerning. The second world which include randomness, dijkstra may have problems to find path in every cases. But with many repetition of reinforcement training we can achieve a solution which is not optimal guaranteed again.

Question 12

12. Can you think of any application where reinforcement learning could be of practical use? A hint is to use the Internet.

Reinforcement learning can have many applications in real world. For example, as we see during the lecture, we can use reinforcement learning in Robotics. One of the most famous achievements until now in this field is a robot which can learn policies to map raw video images to robot's actions. It is also used in popular games, such as Go. AlphaGo is one of the application that can show the power of reinforcement learning. After pretraining the model with previous games, they switched it to reinforcement learning to pass beyond this intellegence. So that it can play better than the state of its only pretrained version and it still trains/develops itself as it plays with a human. One more application of reinforcement learning is the traffic light control. Another application that I found interesting is creating some simulations like natural selection of animals in a environment in a video game.