TBMI26 – Computer Assignment Reports Reinforcement Learning

Deadline – March 14 2021

Author/-s: Yuki Washio (timwa902) and Nicolas Taba (nicta839)

In order to pass the assignment you will need to answer the following questions and upload the document to LISAM. Please upload the document in PDF format. **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.

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

$$V(s_t) = \sum_{k=0}^{\infty} r_{t+k}$$

$$Q(s_k, a) = r(s_k, a) + \gamma V^*(s_{k+1})$$

Where γ is the discount factor and takes values between 0 and 1.
r is the reward and $V^*(s)$ is the maximum value of the Q function in a state s with respect to the action taken.

V is a function that calculates the expected accumulated reward (r) given a state s_k by following a certain policy. By choosing the maximum value of V for every state, we choose the optimal policy. The values of V are unknown at the start of the exploration. The Q-function evaluates the expected reward for each action in each state. In order to find the optimal policy, we return the maximum value for this. For a given state, the Q-function calculates the reward for the different actions available and will return the action that yields the best reward when we look for the optimal policy.

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

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

The updated estimate of Q for a state s_k is calculated as expressed above with the previous estimate of Q in state s_k being informed by the estimate of the next state.

3. **Briefly describe your implementation, especially how you hinder the robot from exiting through the borders of a world. (RESUBMIT EXPLORATION STRATEGY HERE)**

We initialize our Q matrix with random numbers ranging from 0 to 1 and set the borders as -infinite in order to dissuade our agent from exiting the limits of the board. Setting those values to the borders prevents our agent from choosing the action that would lead them out as the penalty for that movement will be – infinite. We then implement a loop over several episodes (times the agent tries to reach the goal). While the agent isn't in the goal and that the next step is valid, we implement the Q function update and keep on feeding the next state to our agent as well as its reward.

We set our discount and learning rates heuristically, whereas the value of the exploration factor decreases as the we go further into the number of episodes. We set up epsilon such that:

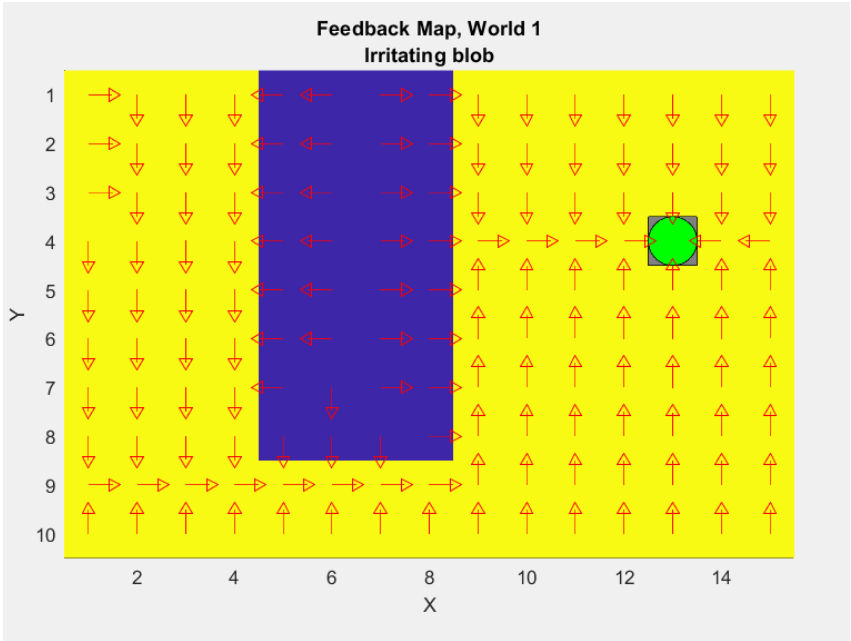
$$\varepsilon = 1 - \frac{episode\_number}{N_{episode}}$$

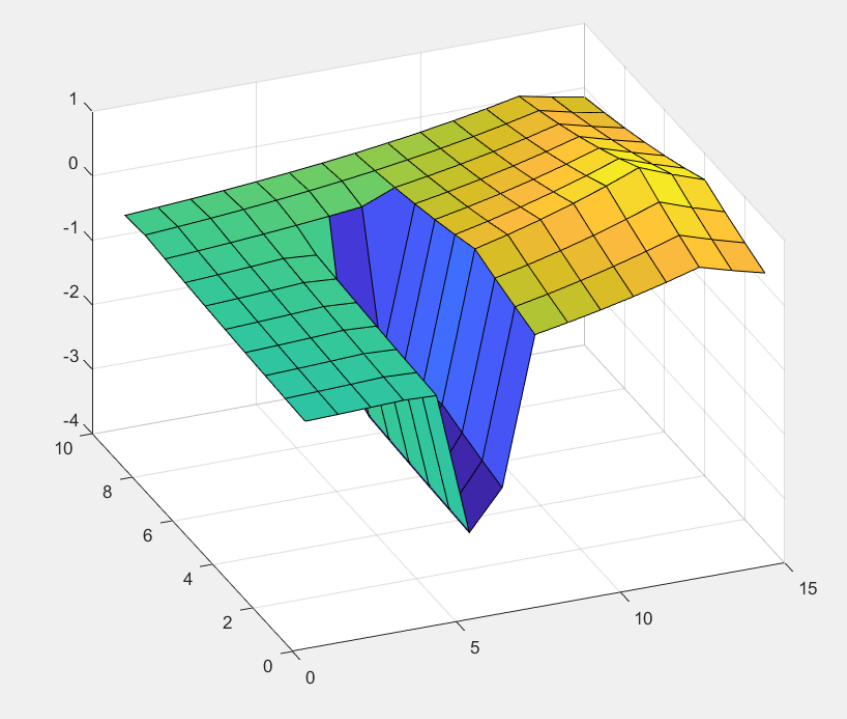
This value decreases as we perform reinforcement learning. In the beginning, the agent explores more and takes random actions more often whereas at the end of the training, the agent exploits more.

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 static. We have an obstacle that is placed in the way of our agent. The agent must learn to avoid this obstacle or if it finds itself inside the obstacle, it exits it the fastest possible to reach the goal.

Discount factor: 0.9
Learning rate: 0.9
Episodes: 1000

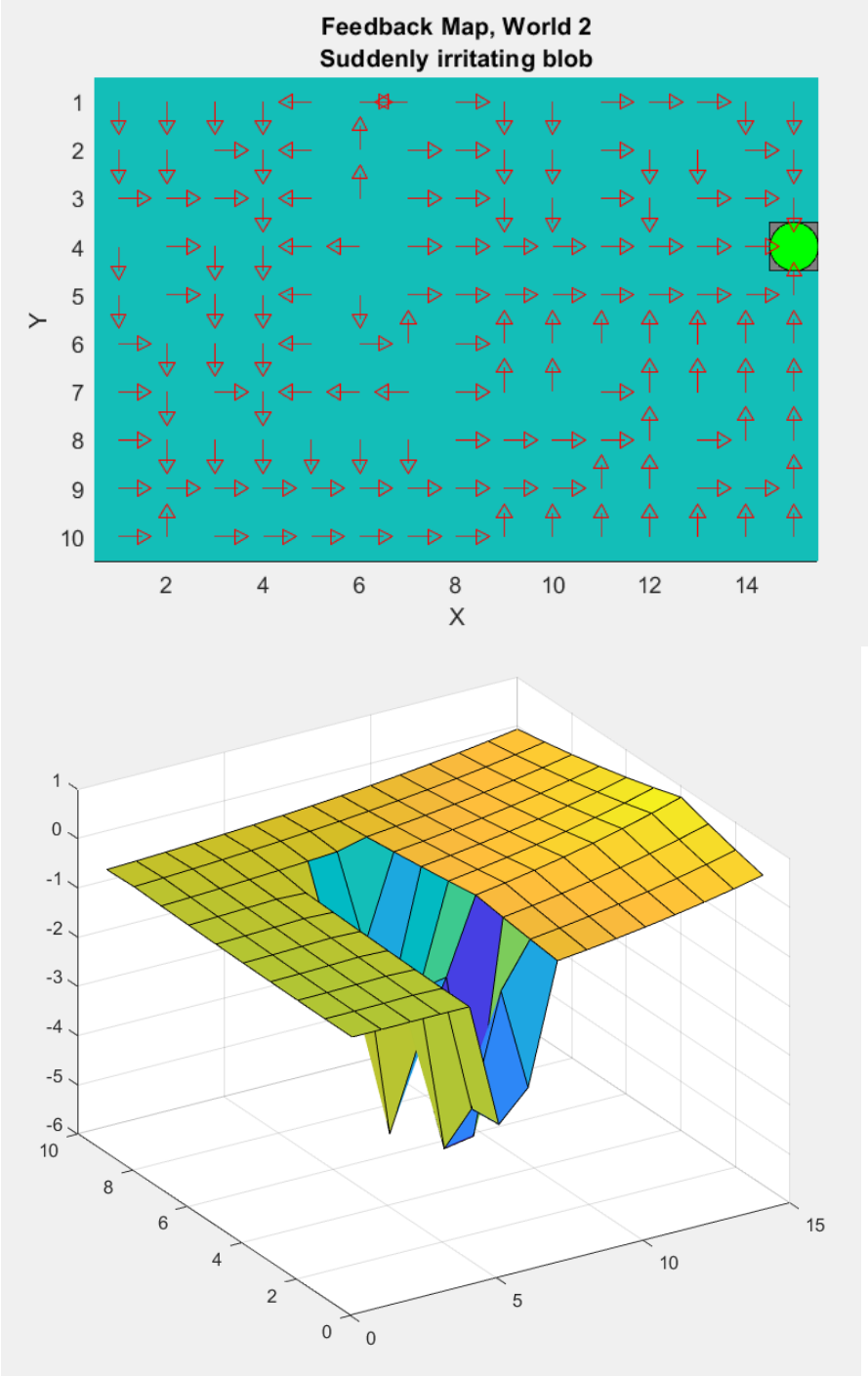




5. **Describe World 2. What is the goal of the reinforcement learning in this world? This world has a hidden trick. Describe the trick and why this can be solved with reinforcement learning. What parameters did you use to solve this world? Plot the policy and the V-function. (RESUBMITTED END OF ANSWER)**

   The second world presents no obstacle but there is a randomness in the policy chosen for each state. There is a 20% chance that the states "obstacle" from the previous world generate negative feedback. The agent does not go directly to the goal and avoids generally the location of the previous obstacle. We lower the learning rate so that when the agent encounters the random area, it keeps that information for longer and searches for other possible paths that yields more consistently higher reward to converge to a solution for the problem.

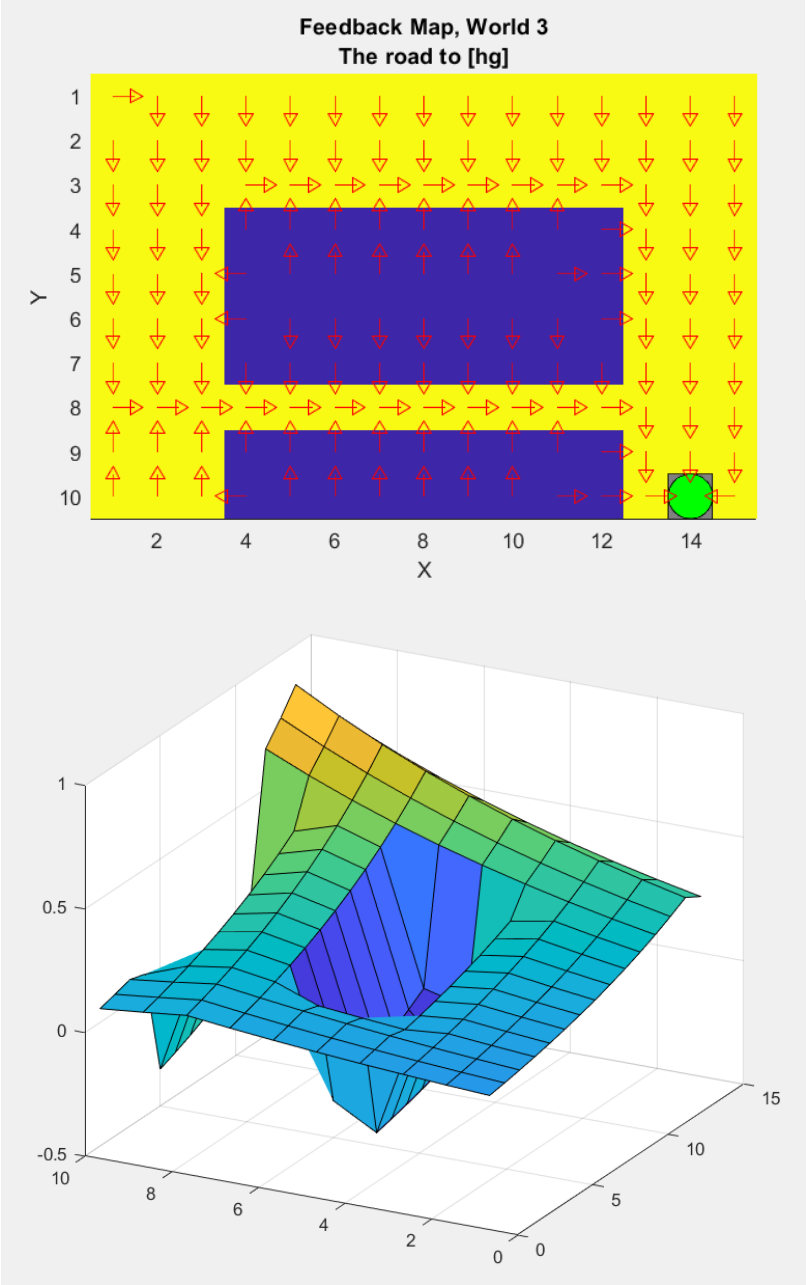
   Discount factor: 0.9
   Learning rate: 0.2
   Episodes: 1500



6. **Describe World 3. What is the goal of the reinforcement learning in this world? Is it possible to get a good policy from every state in this world, and if so how? What parameters did you use to solve this world? Plot the policy and the V-function. (RESUBMITTED POLICY DISCUSSION)**

   This world is similar to world 1, but presents a shortcut that goes through the previous obstacle. The agent must learn to go through this shortcut in order to maximize its reward.

   Discount factor: 0.9
   Learning rate: 0.9
   Episodes: 1000



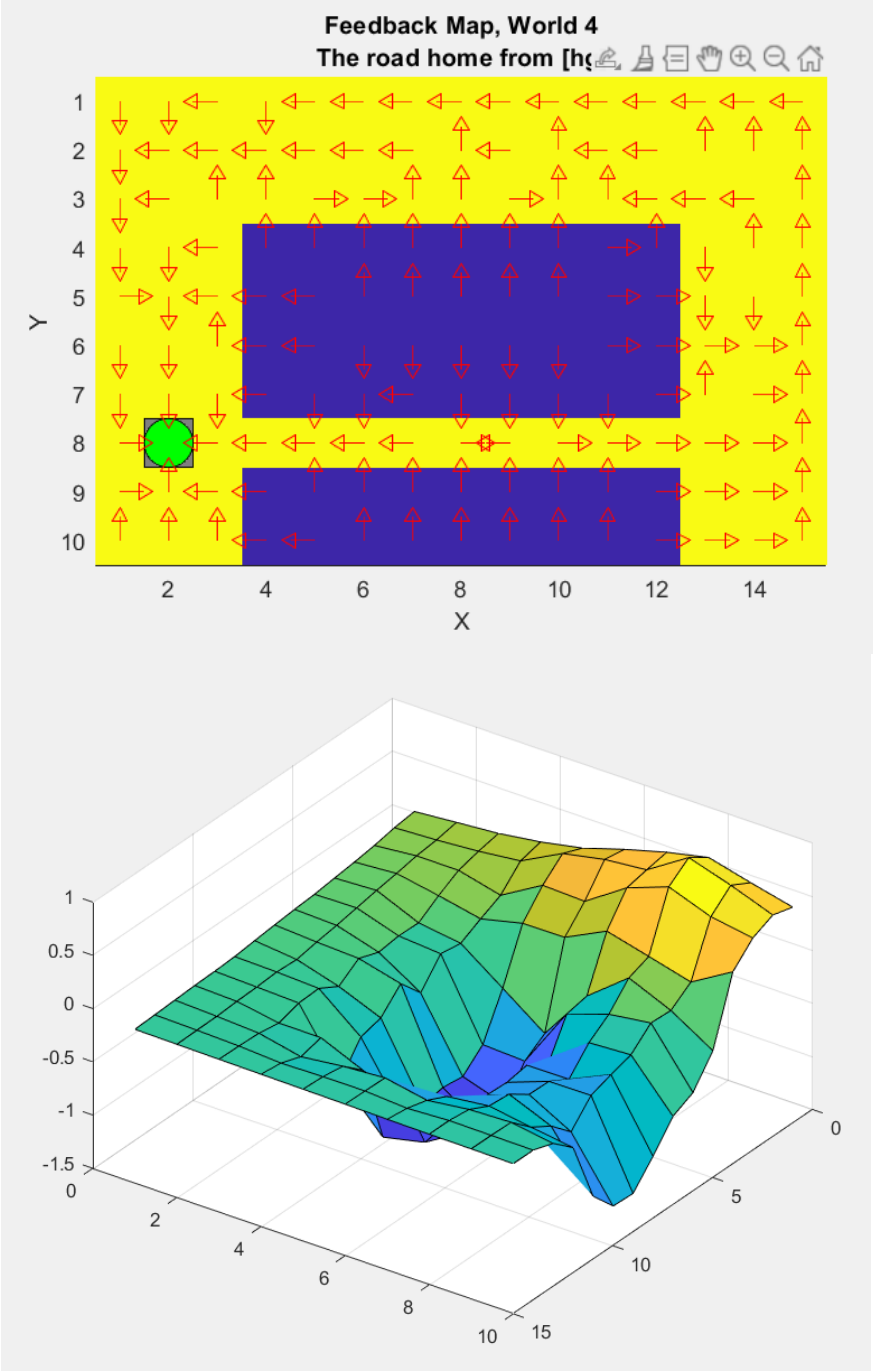
We obtain a good policy for every state and the shortcut yields the best result for this static world. The difference between this world and world 1 stems from the shape of the obstacle. It is more favorable (i.e it takes less steps) to move through the shortcut than to go around so the expected reward is higher by taking this shortcut in world 3. Given that our exploration policy is greedy at the start and exploits further in, it explores more of the parameter space and can yield different paths to the goal that exploit the shortcut, but also provide a solution to go around the obstacle.

7. **Describe World 4. What is the goal of the reinforcement learning in this world? This world has a hidden trick. How is it 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. (REINFORCEMENT LEARNING DISCUSSION RESUBMITTED)**

The world looks similar to that of the previous world, however, the policy map and the V-function tell us that there is a negative cost associated with taking the shortcut and policies that force the agent back on its tracks. From the name of the world “The road home from HG”, we infer that the problem is taken the other way around. The agent is supposed to return to the starting position of world 3 using the information of the path forward.

The V-function landscape informs us that the expected rewards in the shortcut area is low, thus the agent goes back to try and find a better route with a higher reward. The agent chooses paths that stay away from the obstacle and maximize V for all positions. Even though there are contradictory optimal policies that force the agent out of the shortcut that we would intuit as optimal, the agent can still find a way around because that path exists. It also seems that the agent tries to stay the furthest away from the blue area while trying to reach the target. We can imagine that there is some strong negative cost associated with being close to it that influence the landscape of V strongly. This could also explain the back-pedalling policies in the shortcut and the policies that point away from the obstacle even when the agent is not in the blue area. This can still be solved by reinforcement learning because the V-function for the optimal policy is just the maximum value it takes to get from the starting point to the final point.

Discount factor: 0.9
Learning rate: 0.2
Episodes: 3500



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

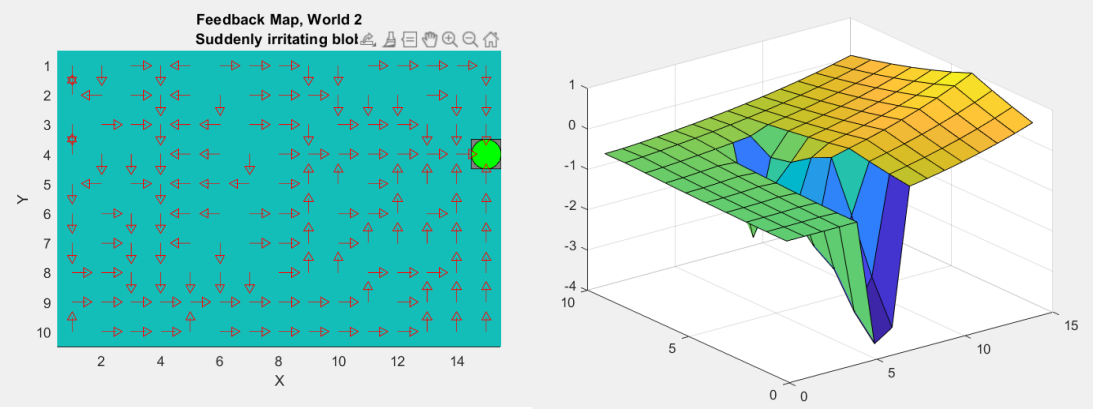
The learning rate determines how much new information is prevalent in the calculation of our Q- and V-functions. The smaller it is, the more the previous learned information is important. From our equations, the first term becomes important for both V- and Q-functions). Higher learning rates are used for simple problems as we can more easily discard older results whereas for more complex problem, we need to pay more attention to new information and weight more toward it.

In order to illustrate the effect of the learning rate, we will take an “easy problem” and a harder one. We use world 2 and 3 to illustrate the effect using learning rates of 0.2 and 0.9

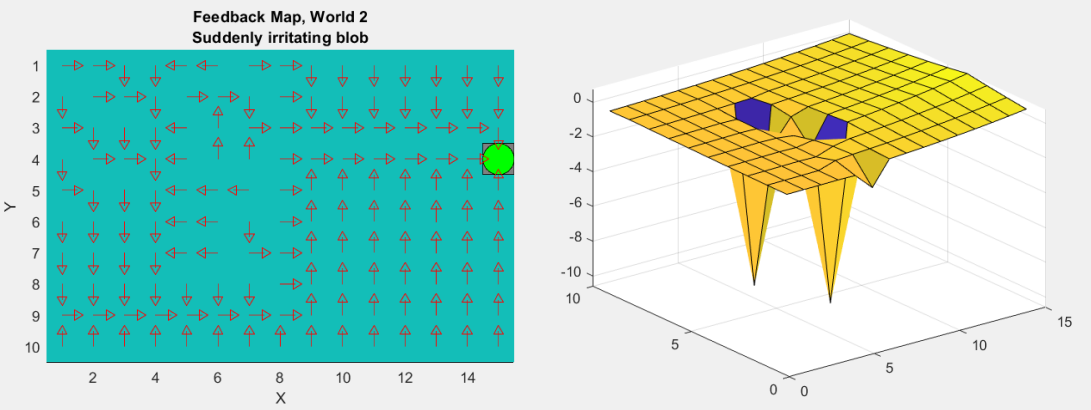
for each. We keep all other parameters the same (exploration strategy, episodes and discount factor).

World 2

Learning rate: 0.2



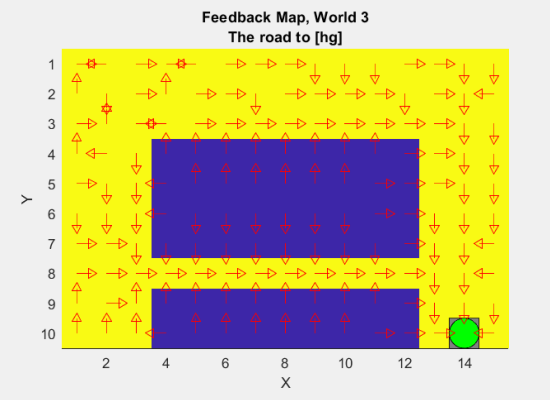
Learning rate: 0.9

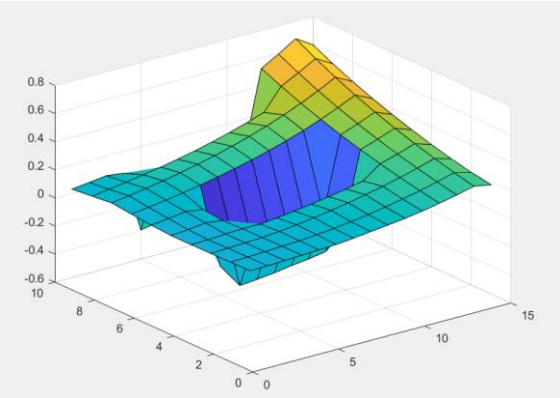


In this instance, we can see that the agent is less careful about avoiding the area that has a probability of suddenly being forbidden when the learning rate is high because previous information about those states is not retained. This flattens the V-function landscape. We also note that with a higher learning rate, we converge to a local optimum faster for the Q-function. The policy results obtained for a high learning rate are the same than for the static world. The agent optimized more strongly towards the safer path with the higher learning rate.

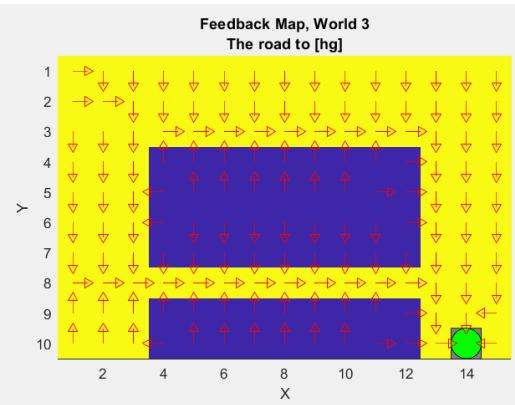
World 3:

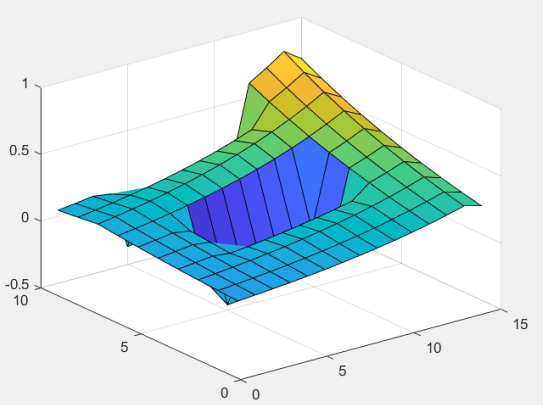
Learning rate: 0.2





Learning rate: 0.9





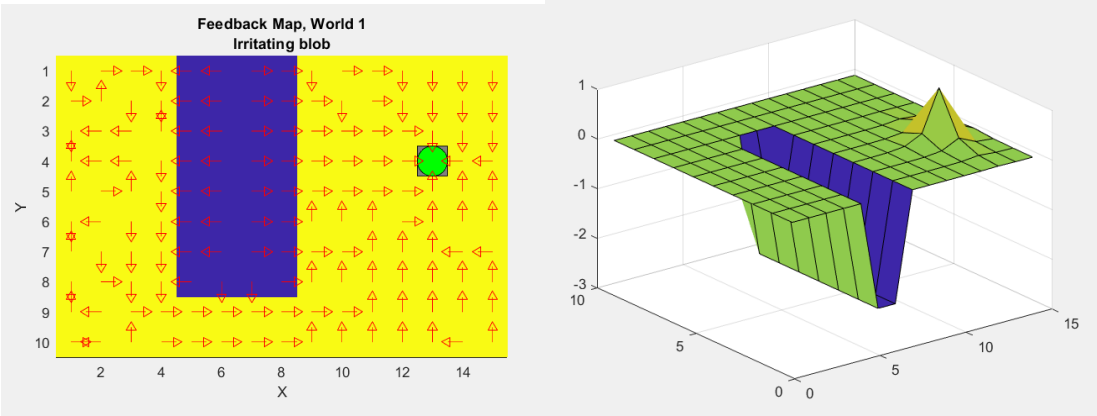
In this case, we see that for a simple problem, the optimal policy (taking the shortcut) is found in both cases. However in the case of the lower learning rate, the agent finds “good enough” policies as the previous information learned takes more importance even if it does not find the best path.

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

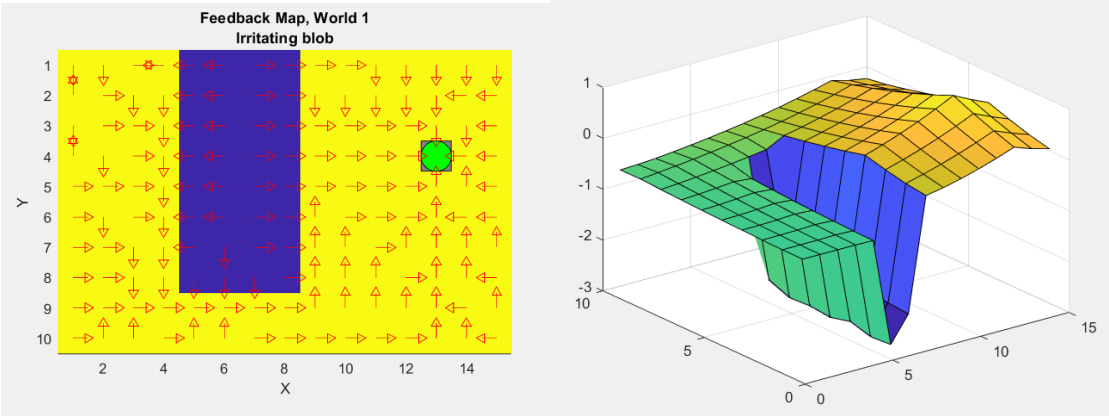
The discount factor determines the importance of future reward. The smaller the discount factor, the more the agent values short term reward and the opposite happens when the discount factor is close to 1. The trade-off here is that for longer timescale, the total reward is susceptible to variance whereas focus on short-term gains, the total reward is more biased.

To illustrate our point we are going to use an easy and hard problem (world 1 and 4 respectively) while changing the discount factor from 0.2 to 0.9 for each.

<u>World 1:</u>
Discount factor: 0.2



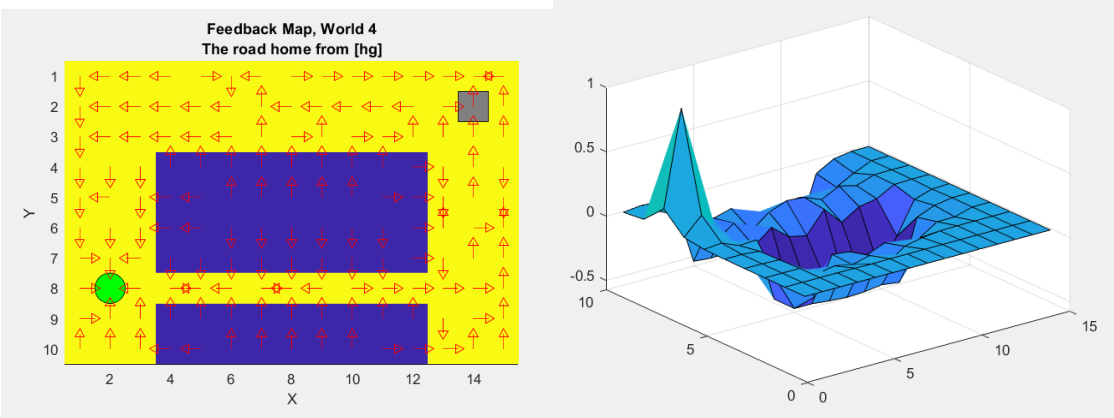
Discount factor: 0.9



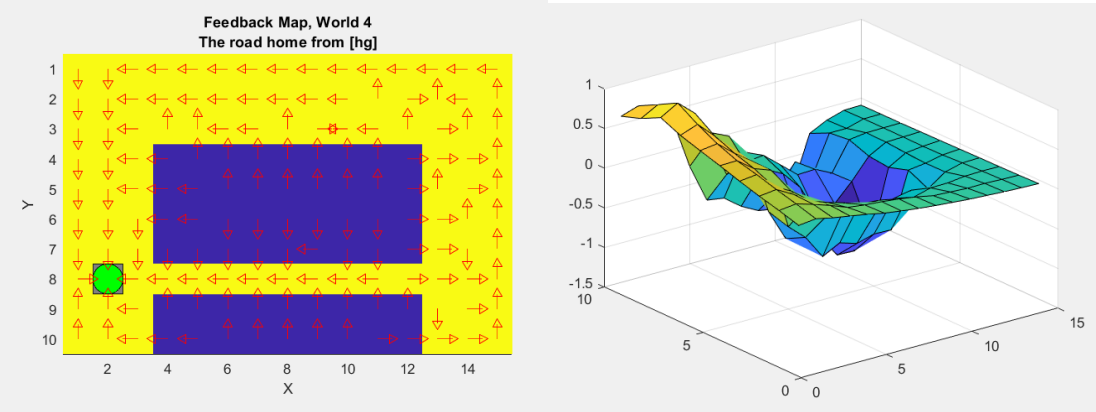
In this case, we can see that when the discount factor is low, the agent values "getting closer to the goal" equally for almost all positions except when just a little away from the goal. The agent can actually get stuck in certain positions if with this policy whereas the higher discount factor allows a more smooth and varying landscape for the V-function and illustrates long term rewards as the agent gets closer to the goal.

<u>World 4:</u>

Discount factor: 0.2



Discount factor: 0.9



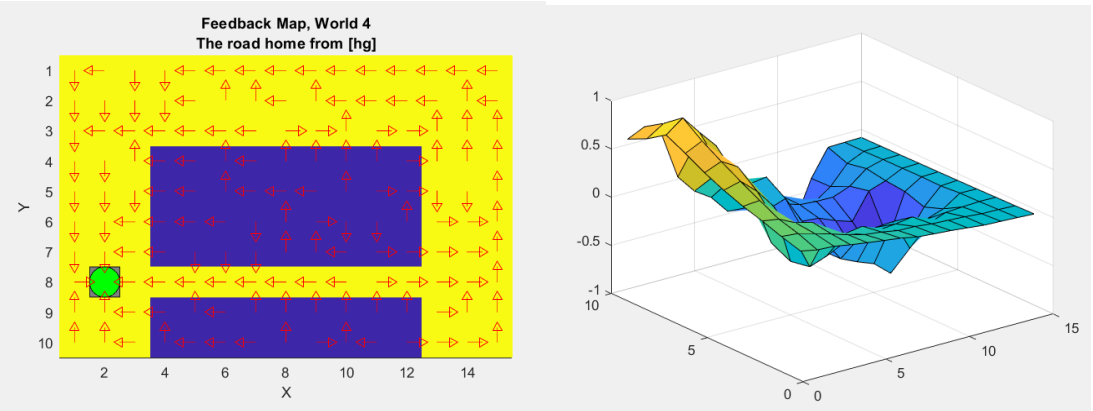
Here, the lower discount factor did not even yield a solution. The agent was stuck. The short term reward of getting closer to the goal did not yield good results whereas higher discount factor allowed once again a more smooth V-function and a possible path to the goal.

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

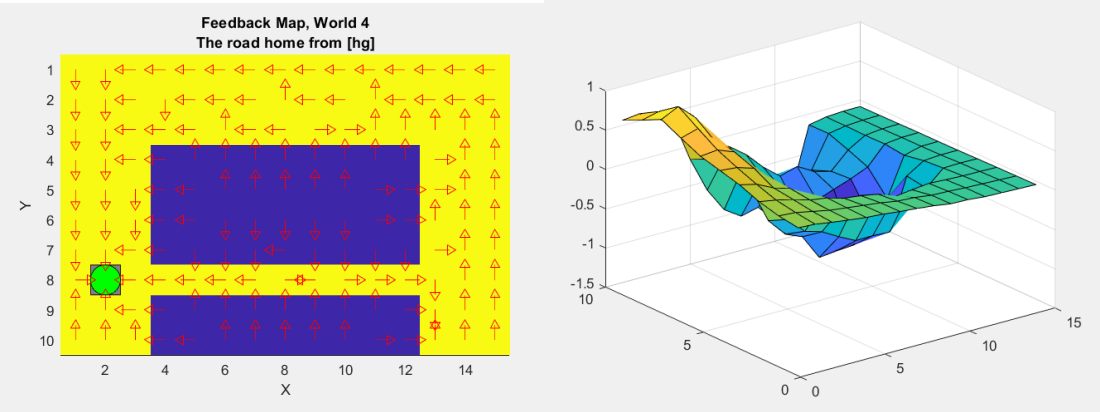
The epsilon value determines how high the probability of exploring non-optimal policies is. This allows our agent to explore the V-function landscape more thoroughly. In our implementation, we decrease the value of epsilon with each episode in order to gradually stop the exploration and make our agent focus on finding the optimal policies to reach the goal. A lower value of epsilon means that our agent tries to exploit more the V-function rather than explore the possible states available.

Higher exploration rates will lead to smoother V-function landscapes. To illustrate this, we will use world 4 and 3 as previously and use the values 0.2 and 0.9 for epsilon.

<u>World 4:</u>
Epsilon: 0.2



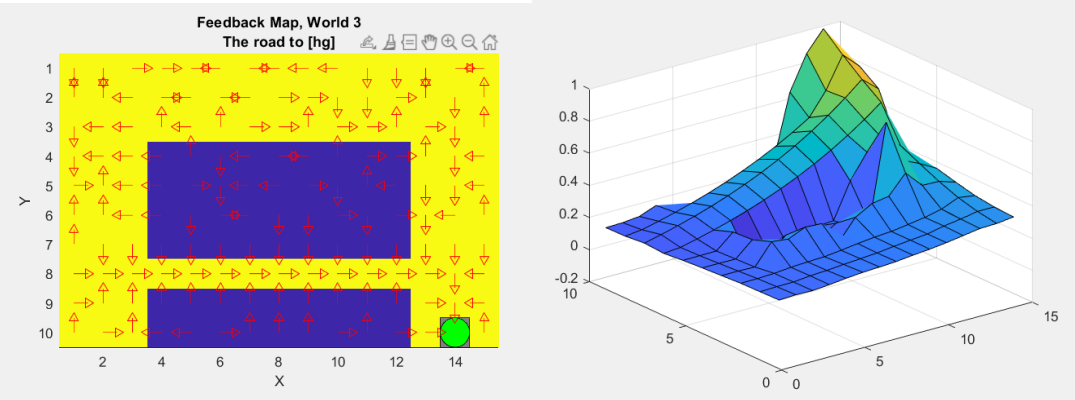
Epsilon: 0.9



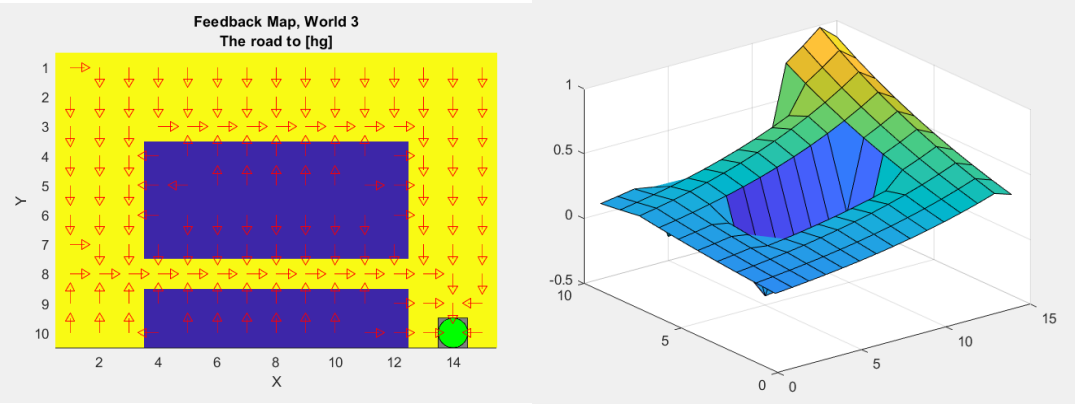
For a higher value of epsilon, the landscape of the V-function is smoother as most of the states have been explored.

<u>World 3:</u>

Epsilon: 0.2



Epsilon: 0.9



Here we see that lower values of epsilon lead the agent to exploit the shortcut much more than for higher values of epsilon that lead to a smooth V-function where our agent has explored all the states.

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?**

Djiktra's algorithm is implemented with the assumption that the adjacency matrix it works with is static and does not change as we traverse it and iterate towards the goal. Reinforcement learning in such a static world would yield a result that would be close to the one from Djikstra. However, in the suddenly irritating blob world, that is not static, Djisktra's algorithm would have trouble with the randomness, the reinforcement learning method might be more efficient in finding a path that yields results that are more consistent and maybe better than for Djikstra's algorithm.
Reinforcement learning does not guarantee us to find the optimal path, but we can find solutions that approximate it with limited computational resources.

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

    Reinforcement learning has been used to produce robots that compete in games. For example the AlphaGo allows its agent to compete with humans in the game of Go that is famously known to have too many configurations to be solvable by an algorithm. This same reinforcement learning scheme was then applied to a game with even more possible configuration: AlphaStar (https://deepmind.com/blog/article/AlphaStar-Grandmaster-level-in-StarCraft-II-using-multi-agent-reinforcement-learning) which is able to play at a competitive level the game Starcraft 2 against humans using restrictions that make it appear to be human. This particular scheme mixes deep learning and reinforcement learning with a particular attention given to how long and what exactly is memorized in order for the agents not to get exploited after optimizing themselves in a particular solution.
    Reinforcement learning is also used for self-driving cars, where agents learn to recognize the different elements on the road to avoid collisions and accidents.

13. **(Optional) Try your implementation in the other available worlds 5-12. Does it work in all of them, or did you encounter any problems, and in that case how would you solve them?**