TBMI26 – COMPUTER ASSIGNMENT REPORTS REINFORCEMENT LEARNING

Deadline – March 15 2020

Nour Elhouda Qweder (nouqw898)

Weng Hang Wong (wenwo535)

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)**

$$V(s_i) = \sum_{k=0}^{\infty} \gamma^k r_{1+k}$$

V-function:
A function V(s) of the state (s) that tells us the value of being in the state (s) under a policy, given that we will continue to follow the policy (agent's strategy). And to obtain the optimal value by maximum value function V(s) for all states.

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

Q-function:
The expected return reward from start attempting action in state s(k) (which is the blue one) and the optimal policy in (red in the previous equation).

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

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

Q learning equation determines the value that taking the state k and action I will be updated by the previous estimate of the Q-value(the blue part) plus reward with next timestamp plus the gamma times maximum Q-value in the following state across all possible actions will take in the following state.
The learning rate η: indicates how much new information will affect the Q-value.
The discount factor γ: determines the present value of future rewards.

$$Q_*(a) \leftarrow E(R_t | A_t = a)\ \forall a \in \{1, \dots, k\}$$

We define the value of selecting an action as the expected reward when taking that action.
We need to remember that is the goal is to maximize the expected reward.

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

We have three main steps:

1. Initialize the world Q-table, and the hyperparameters
   - **a.** Start by initializing the world, Q-table (world. ysize * world.xsize * 4 matrix) "here there are 4 possible actions to choose one of them randomly."
   - **b.** Start by inf as a start Q-value for all actions which cause the robot to fall off.
2. Training loop: by repeating a series of actions number of times (episodes)
   - a. Reset the world
   - b. Exploration factor is set according to how many episodes have been run according to the total
   - c. If the state is not terminal
     - i. The state position is the old one.
     - ii. Select new action according to Q and epsilon
     - iii. Update the Q_value of the new state
   - d. Set all (Q_value =0) for current state if the robot is in the goal.
3. Reach the optimal policy by applying argmax of the Q-matrix along the action dimension.

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.**

In world = 1 (is simple) we have one punishing block close to the center otherwise free area where our robot needs to go through until hit the goal "the blob".
The used parameter:

- Episodes = 2000
- Learning rate = 0.5
- Discount factor = 0.9
- Epsilon = 0.9 to 0.045 (decrease according to episodes number)

- Learning rate can be set to 1 in all static worlds (in this lab that is world 1 and 3)
- Learning rate can be set to 1 in all static worlds (in this lab that is world 1 and 3)

World 1 and world 3 are static worlds in this lab, since nothing change from episode to episode and the robot can move freely without being forced to move in a random direction, so the goal is to converge to the optimal policy as fast as possible. in this case we can use learning rate =1.

-In case of world 2,4: randomness is confusing the learning manner, which makes it is important to use a small value of learning rate plus large value of number of episodes that helps the algorithm to not get confused by the randomness.

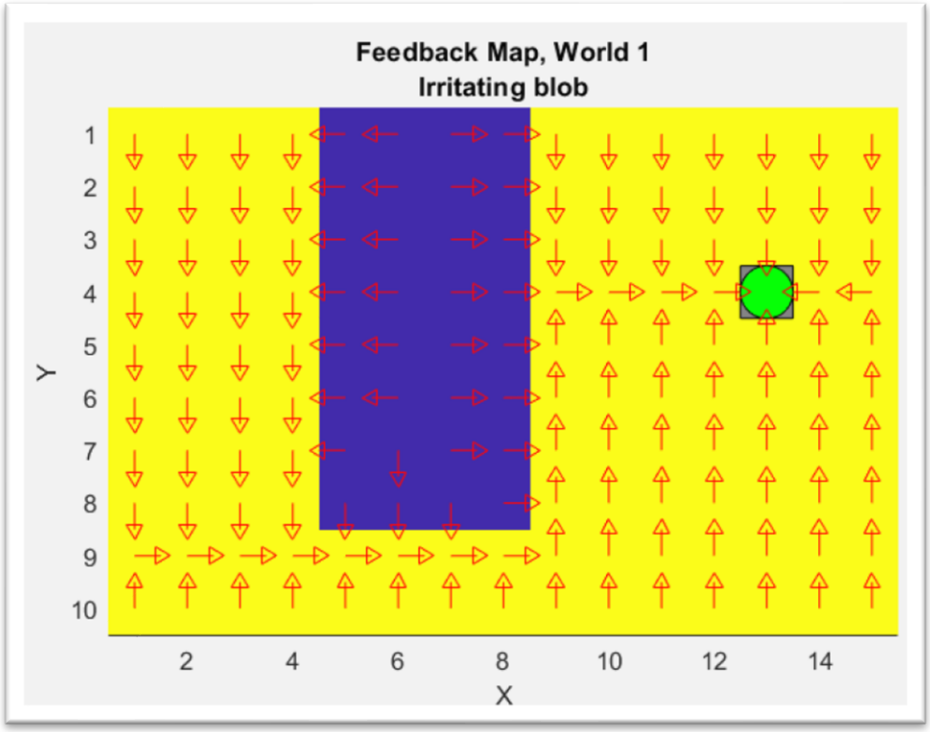


Figure 1 Policy map

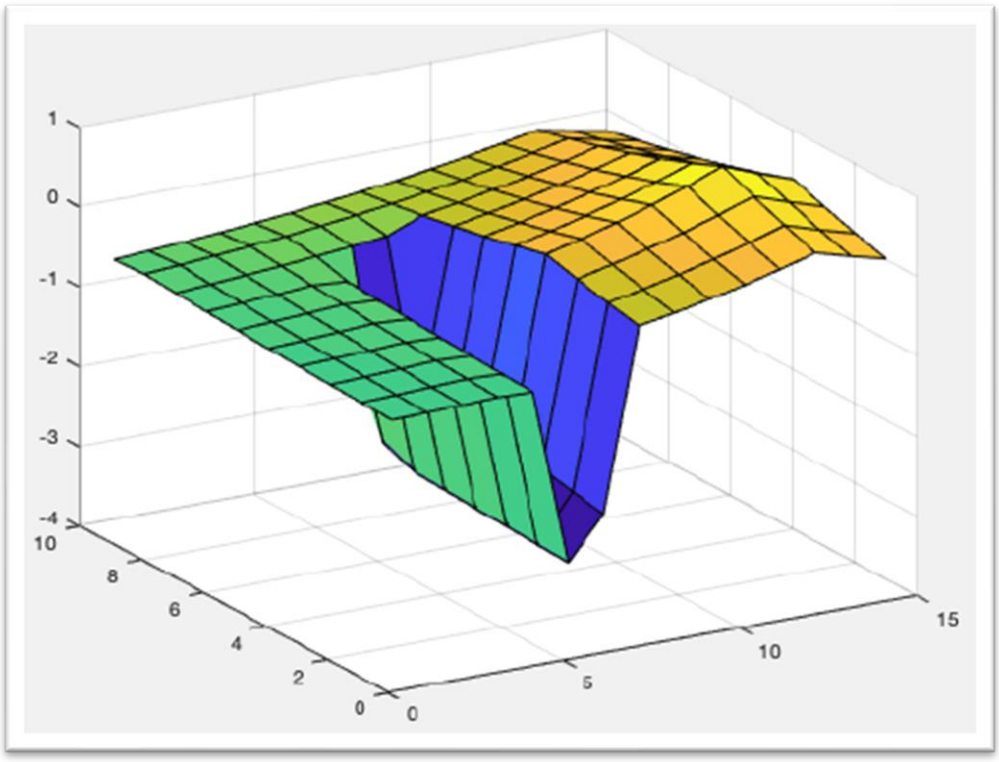


Figure 2 V-function

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.**

- Discount factor: 0.9
- Learning rate: 0.2
- Maximum steps: 100
- Total episodes: 1500
- Exploration factor: 1 - no of current episode / Total episodes no.

World 2 which is like world 4 (with an approximately learning rate in this case).
It can be noticed (from the plot) the randomness in world 2.
The low value of learning rate indicate that the learning process occurs very slowly since a tiny update to the weights happened. In other hand it requires to increase the number of episodes.
It can be mentioned here that the learning rate controls how the model is adapted to this problem.
We could see that learning rate depends on the environment, if the training happened in a static environment, it recommended to use alpha = 1 wherewith more noise, it recommended to use a lower value of alpha which indicates the learning process occurs in slow terms.

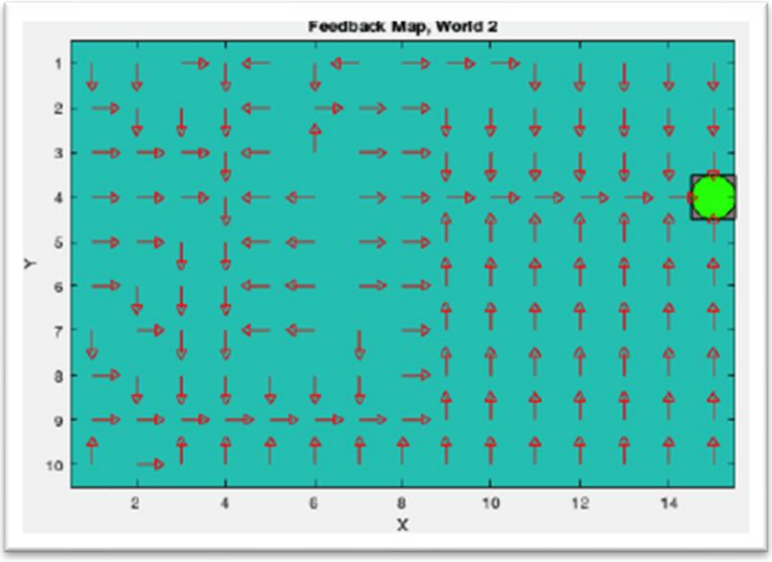


Figure 3 Policy world 2

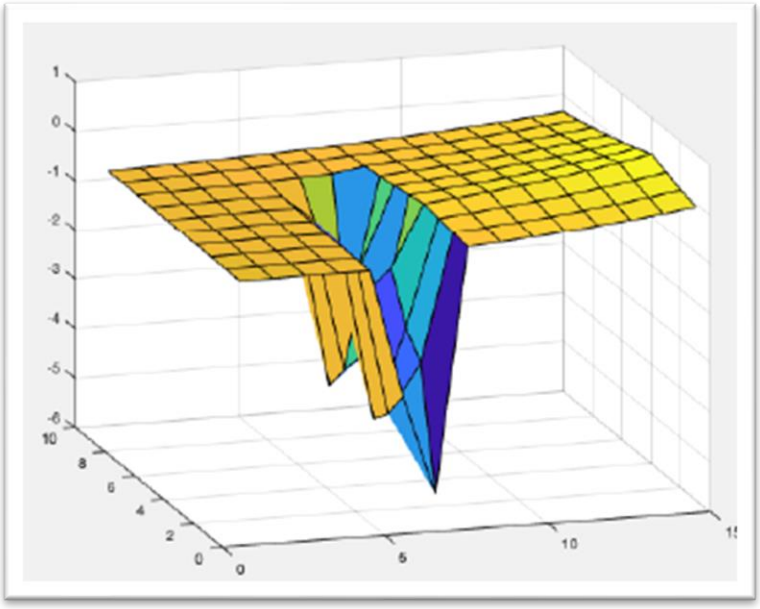


Figure 4 V-function world2

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.**

   This is a static world as world 1 (as mentioned in question 4)

- Discount factor: 0.9
- Learning rate: 0.9
- Maximum steps: 100
- Total episodes: 1200
- Exploration factor: 1 - no of current episode / Total episodes no.

It can be noticed that it has a bigger area that covered by a punishing terrain a "blob" in comparison with the world 1, 2.
However, there is also a slightly thin path which is sort of a quick alternative path to the goal instead of walking around the "blob" and this quick path divided the "blob" into two areas.
we need to find a balance between decreasing exploration factor (epsilon) and a high learning rate, at the end it will discover that the thin path is the best alternative to hit the goal.

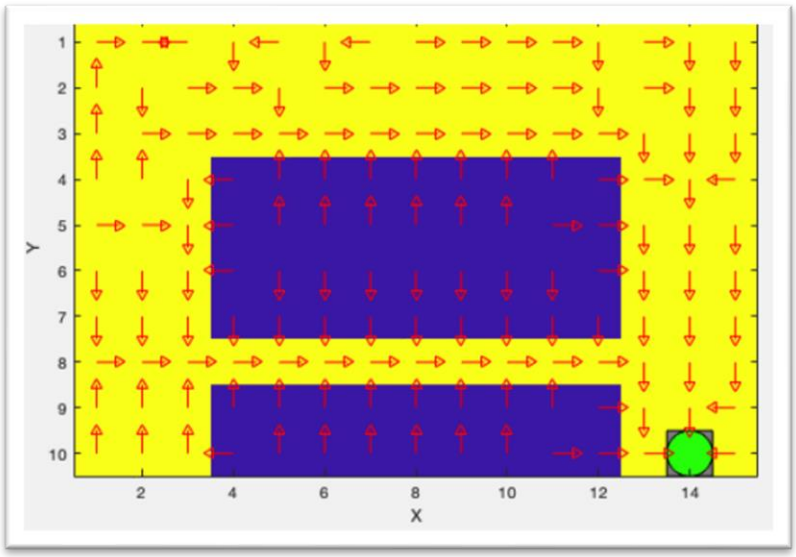


Figure 5 world 3, policy

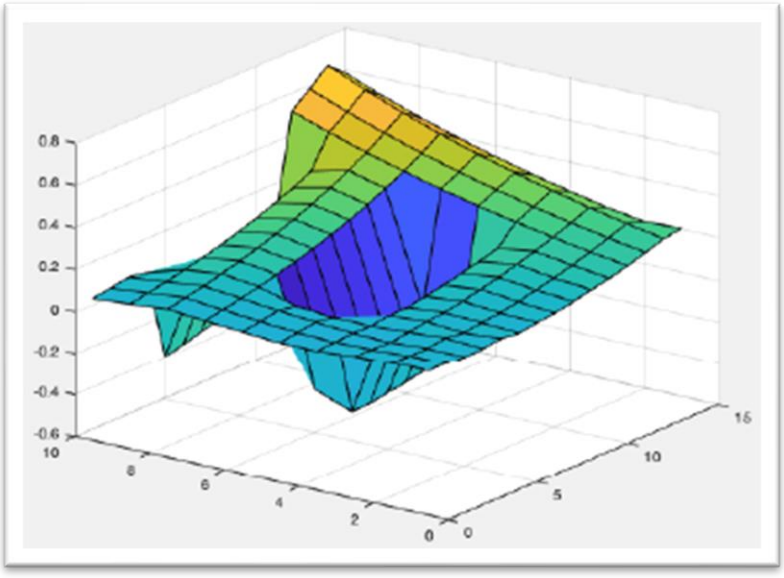


Figure 6 world 3, V-function

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.**

- Discount factor: 0.9
- Learning rate: 0.2
- Maximum steps: 100
- Total episodes: 2200
- Exploration factor: 1 - no of current episode / Total episodes no.

Now, in (world = 4)

It can notice the similarity between world 2 and world 4, but after following the robot steps (I noticed the robot spend more time in a random direction), However, this problem can be solved by running much time (reinforcement technique) because we cannot give the solution or decide, and the robot is learning by trying different policies.

We need also to mention that the robot was carefully moving against punishment area since we have a low value of learning rate and high value of discount factor.

I this case, the discount factor is close to 1, the agent will care more about each single step, thus, it will evaluate each of its actions based on the sum of total of its future reward, while. The robot does not use the advantages of shortcut as he did in the world 3.

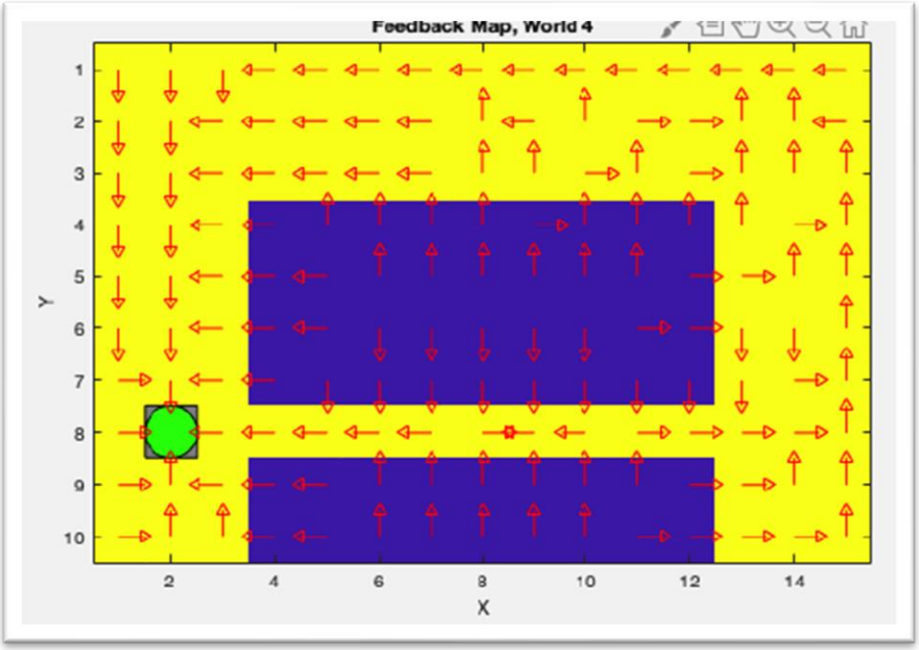


Figure 7, policy

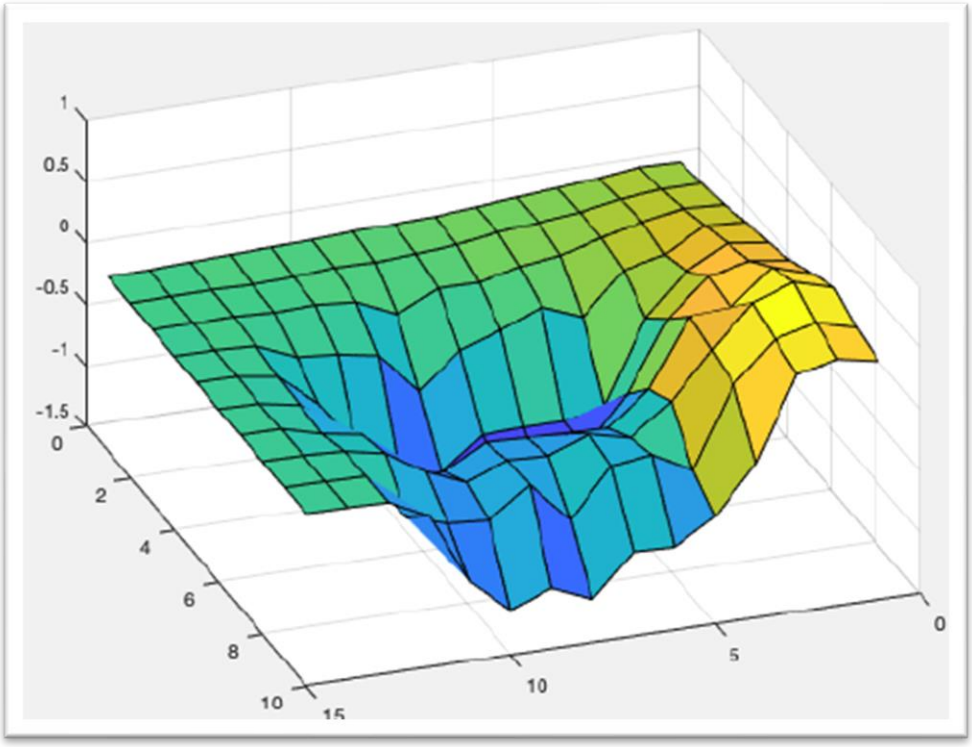
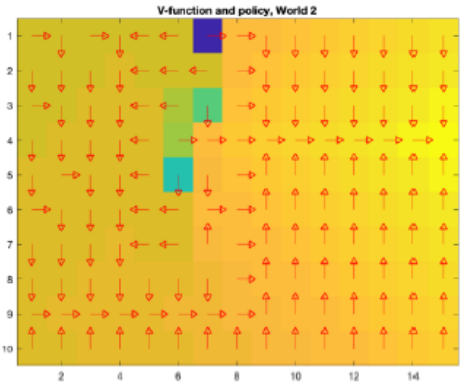
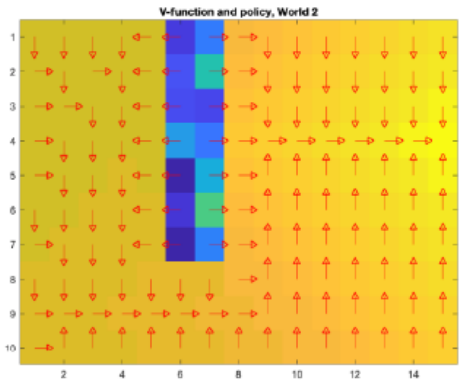


Figure 8, V-function

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

   For world (1 and 3) it can be noticed that learning rate with a sensible value does not influence (especially when there is no element of chance), whereas when the chance has a role (such as in world 2 and 4) the learning rate plays a vital role, the method is likely to not memorize what it has learned before when we have a high learning rate (in world 4, it comes into play often random movements)





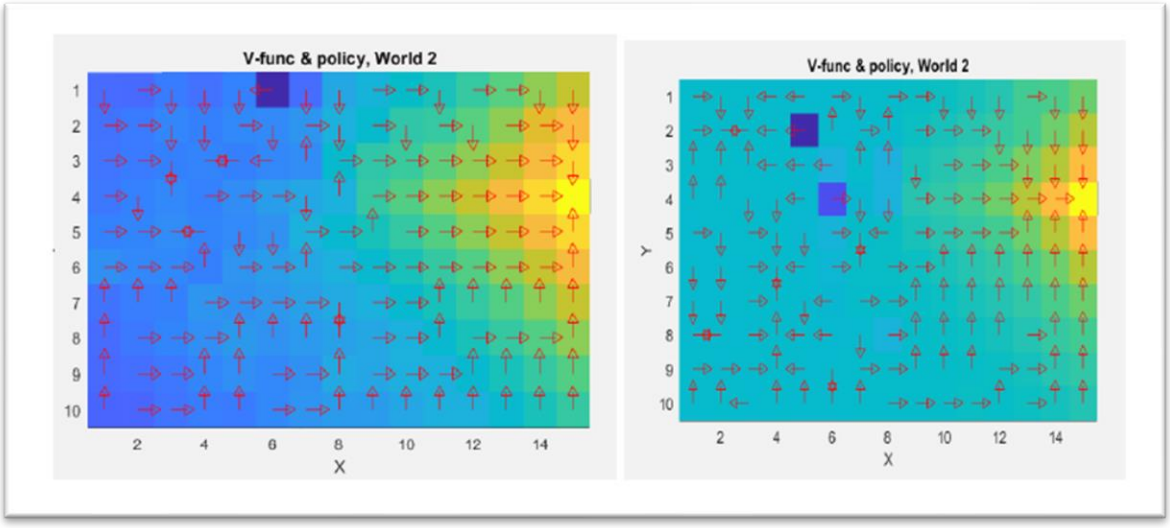
Left we have learning rate = 0.7

Right, we have learning rate = 0.5

In later episodes, the right there is number of states did not see the “blob” which leads to a poor decision making, it might go through the dangerous area.

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

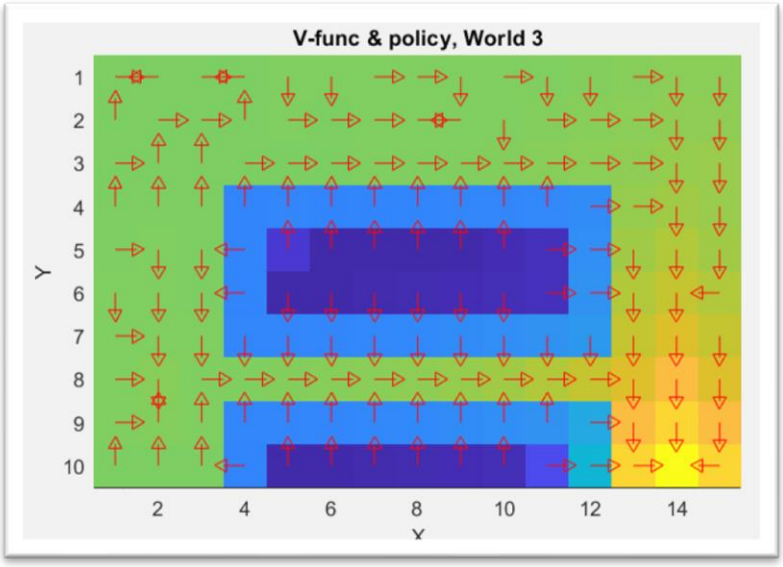
   I notice when we have a higher discount factor the robot take more risk and go through the higher-penalty zones (it seems the robot got encouragement to reach the goal quickly, regardless the size of the risk of penalty zone) while when we have a lower discount factor the robot will try to avoid the penalty zones looking for a lower penalty but will also slow down training, as making the goal no valuable, which make a risk of stuck in a non-penalty area and get no progress in learning policy.



Left side,
Discount factor = 0.9 (suitable)
Learning rate = 0.9
Right side
Discount factor = 0.7
Learning rate = 0.9
in right side case, it can be noticed that the “blob” is punished but not as heavily as in the suitable case of discount factor (left side).

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?**

Discount factor = 0.9 (suitable)
Learning rate = 0.9
Epsilon = 0.3 (bad case because of too low) with world = 3
If the exploration rate is too low, which cause a low search area (as we can see in the example below, which will cause ignoring many parts of searching area).
While in case of high epsilon, resulting in training will slow down



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?**

    Dijkstra's would not work on the "suddenly irritating blob". We need to add extra stuff to adjust and manage stochastic phenomena plus extra knowledge of the world and how it works. It might work pretty good in (world = 1).

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

    - An example of reinforcement "doctor running the medical trial to find best medicine for each patient".
    - An automatic grass cutter and teach it to the best way to cut all grass in a short time.

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?**