TBMI26 – Computer Assignment Reports Reinforcement Learning

Deadline – March 14 2021

Author/-s: Wuhao Wang

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)**

Qfunction:

$$Q_\gamma^\pi(x,a) = E_\pi\left[\sum_{t=0}^{+\infty} \gamma^t r_{t+1} | x_0 = x, a_0 = a\right]$$

V-function :

$$V(s_t) = \sum_{k=0}^{\infty} \gamma^k r_{t+k} \text{ where } 0 \leq \gamma \leq 1$$

Q-function tells us the expected future reward if we take an action **a** in a state **x** and then follow the optimal policy.

V-function tells us about the expected gain in certain state if we always follow the policy.

For instance, V(3)=5 means when we are in state 3 and follow the policy, we will eventually get score 5.

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

We define **η** as learning rate, $\boldsymbol{\gamma}$ as attenuation of future reward, **r** is the immediate reward of taking action **α** in stage **s**.

Q-function : ***Q(s,a)*** = $(1-\eta) * Q(s,a) + \eta * [r + \gamma * \max_{\alpha'}(Q(s',\alpha'))]$

After taking an action, Q-function will immediately update the Q-function. It's an weighted average from current Q-function and the immediate and future reward from taking this action. In this way, once the robot reaches the target, it will be rewarded huge value (r), and the last stage and the action will also have higher Q-funtion value. With the learning iteration increase, the optimal path will be granted higher value than other paths.

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

I set two parameters to control complexity. The first one is to control how many **iterations** of the learning. In each learning iteration, second parameter will control the **maximum step** the robot can try if it cannot find the target.

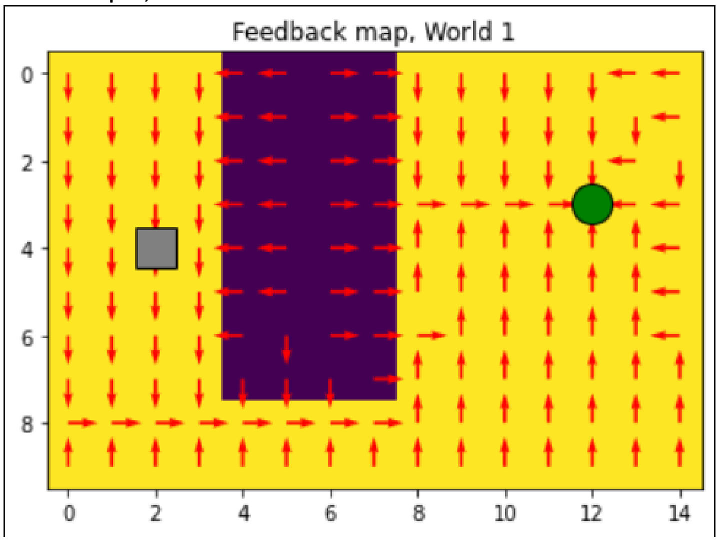
In each iteration, the action will be randomly chosen or follow the current optimal policy (the optimal value from the current Q-funtion), this is controlled by setting a hyper-parameter **epsilon** . When an action is done, the Q-function will be updated immediately according to the equation mentioned above.

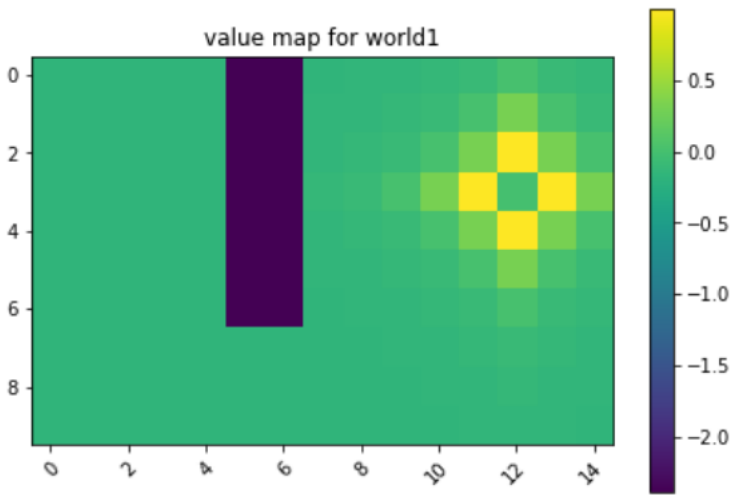
For the borders of a world, I get variable ***'valid'*** from ***action()*** function, if the action is not valid(cross the borders), the action will be withdrawn and choose another action until the action is valid. During the initial stage, the invalid action will be rewarded minus infinity.

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 goal in this world is to find the shortest path to get the term and avoid walking in purple regions.
hyper-parameters:
learning rate(eta) = 0.2;   attenuation of future reward(gamma) = 0.8.
random threshold(eps) linearly increases from 0.6 to 0.9 according to learning iterations.
Total learning iteration: 2000

For example, a situation is like this.

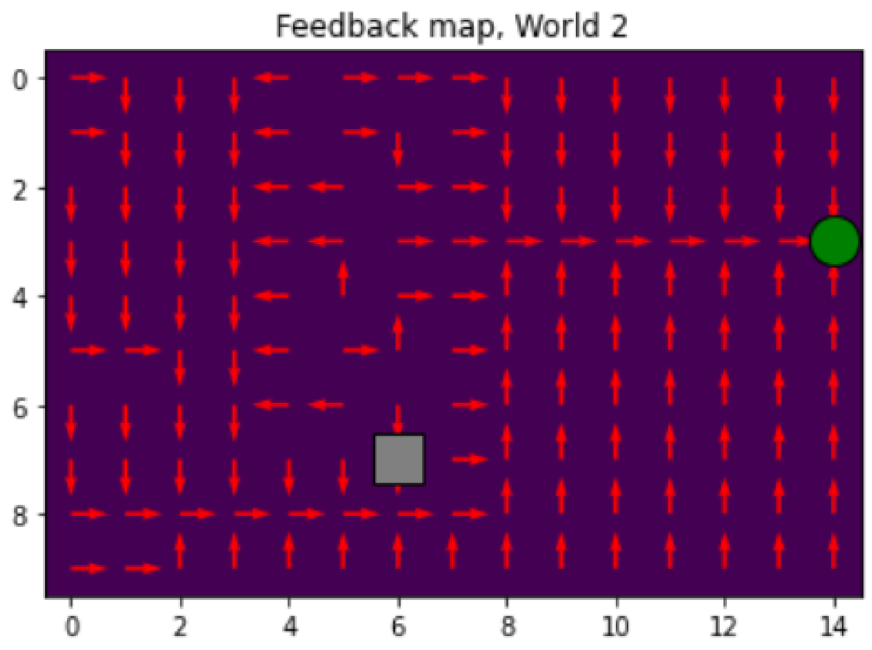


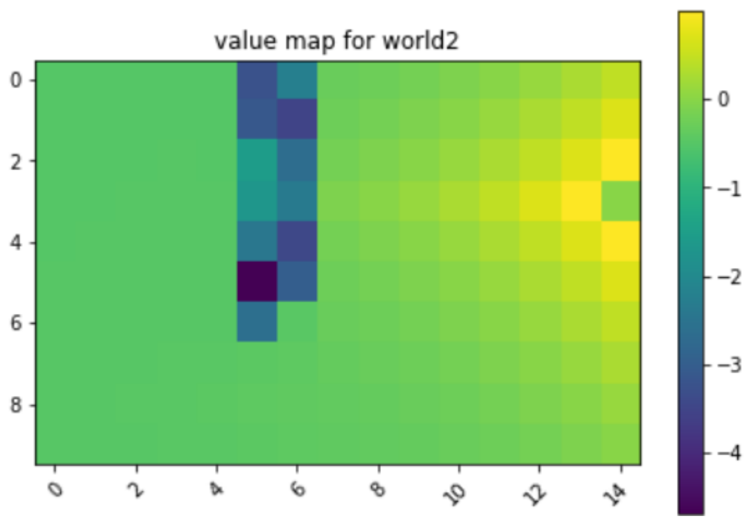


5. **Describe World 2. What is the goal of the reinforcement learning in this world? This world has a hidden trick. Describe the trick and why this can be solved with reinforcement learning. What parameters did you use to solve this world? Plot the policy and the V-function.**

   In the world 2, the blob will show up sometimes. When the blob shows up, algorithms will force agent to learn how to avoid touching blob. Because walking into blob will result in lower reward. When there is no blob, the algorithm will try to train the agent to find closest path, because it will result in higher reward. In this way, reinforcement can handle this world.

   hyper-parameters:
   learning rate(eta) = 0.2;   attenuation of future reward(gamma) = 0.8.
   random threshold(eps) linearly increases from 0.2 to 0.8 according to learning iterations.
   Total learning iteration: 2000

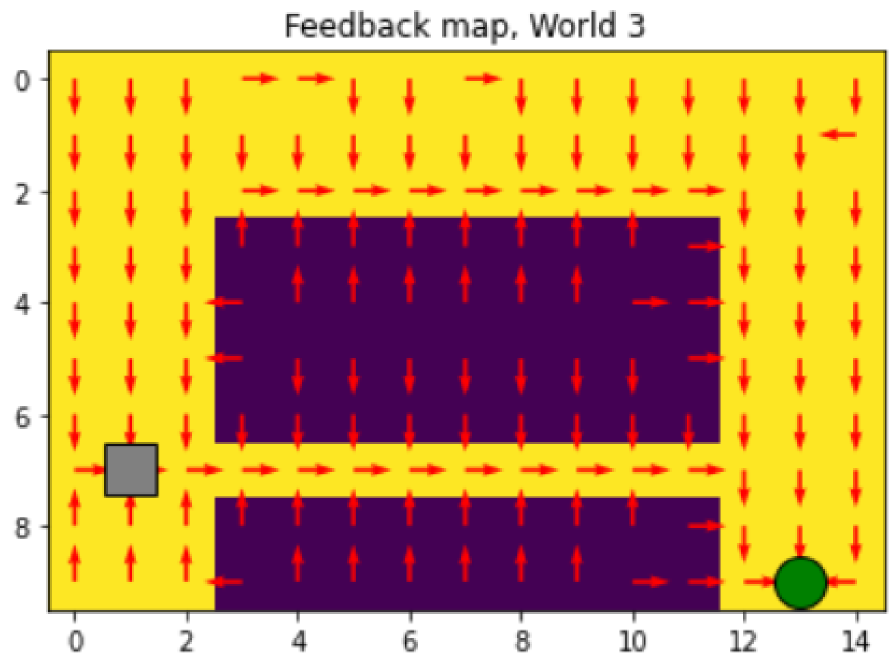


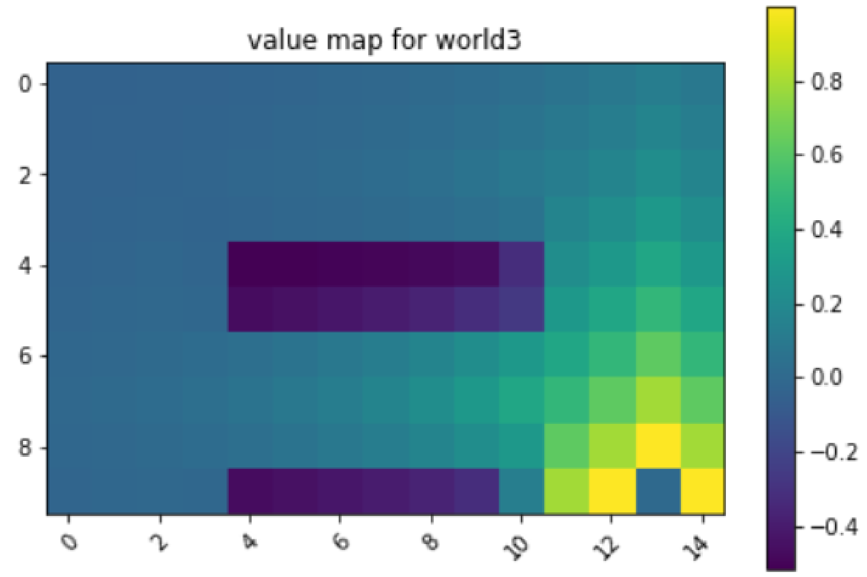


6. **Describe World 3. What is the goal of the reinforcement learning in this world? Is it possible to get a good policy from every state in this world, and if so how? What parameters did you use to solve this world? Plot the policy and the V-function.**

The target of world 3 is to find the a specific road, index is [7,:] ,to the destination. It's possible to get a good policy from every state in this world. One can set epsilon become larger so that there will be more chance for exploration.

In the previous task, the reward of finding destination is **30**, now I increase it to **40**. And I find 40 also works well in World 1 and 2, so I just keep this hyper-parameter for World1 and 2.
hyper-parameters:
learning rate(eta) = 0.2; attenuation of future reward(gamma) = 0.8.
random threshold(eps) linearly increases from 0.2 to 0.8 according to learning iterations.
Total learning iteration: 2000

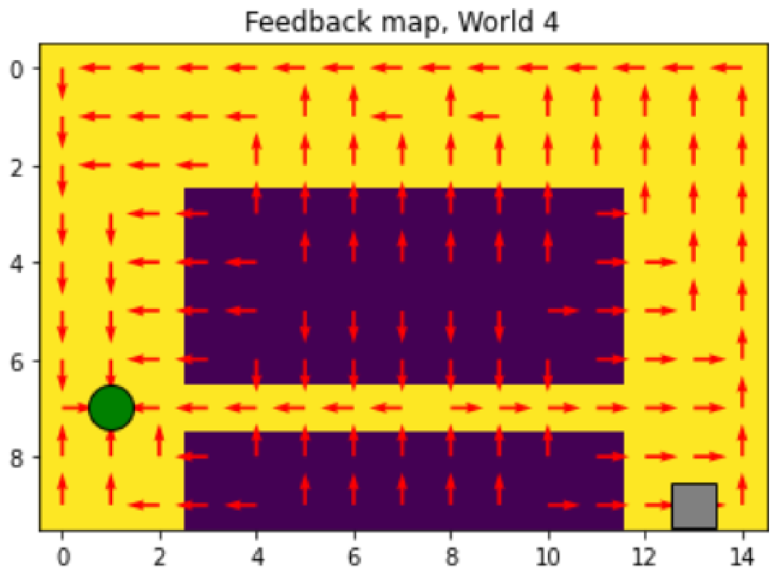


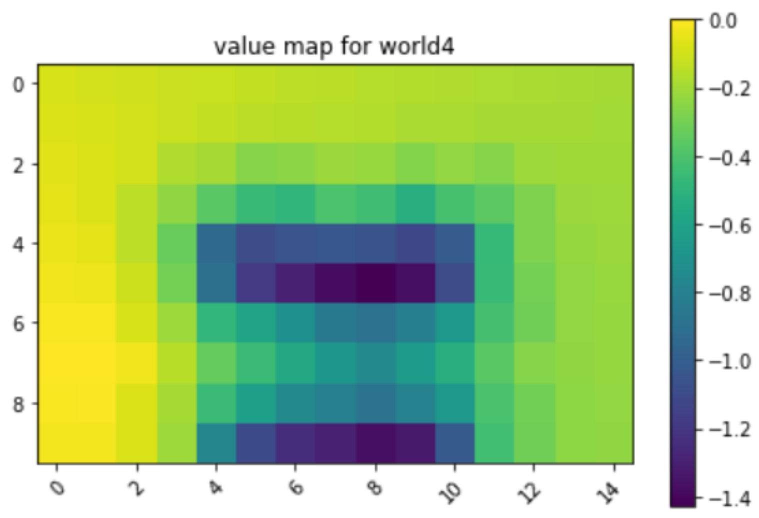


7. **Describe World 4. What is the goal of the reinforcement learning in this world? This world has a hidden trick. How is it different from world 3, and why can this be solved using reinforcement learning? What parameters did you use to solve this world? Plot the policy and the V-function.**

   In the world4, the goal is to find the path with maximum expected reward. In this world, every action has 30% chance to be randomly re-selected without informing the system. So, during the training, the Q-function will also take this 'randomly move' into consideration. From the Q-function map, we can see agent avoid walking into the narrow yellow path because there is higher risk to walk into blob area which will lead to great minus reward. Alternatively, agent will be trained to pick the path which is most far away from blob area because following that path takes lowest risk to walk into blob.

   hyper-parameters:
   learning rate(eta) = 0.01; attenuation of future reward(gamma) = 0.97.
   random threshold(eps) linearly increases from 0.2 to 0.8 according to learning iterations.
   Total learning iteration: 8000

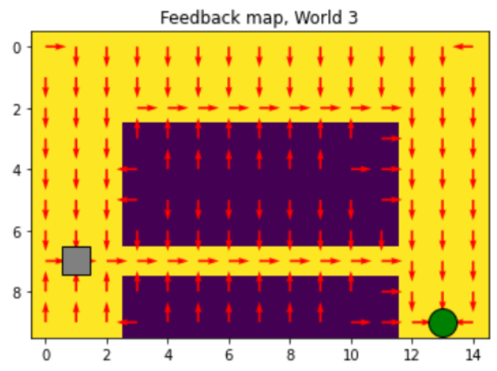




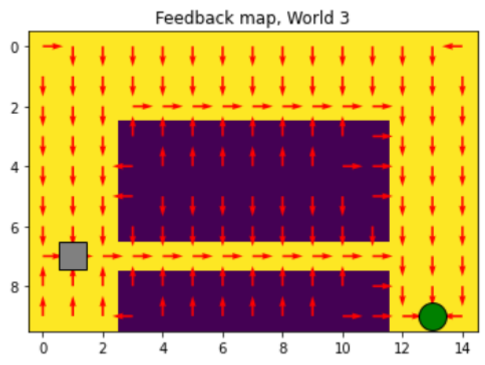
8. **Explain how the learning rate α influences the policy and V-function. Use figures to make your point.**

   World 3 is more stable to make compare between different parameters, so I will focus world 3 for discussion.

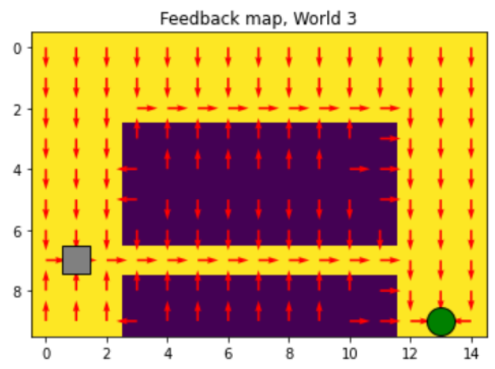
lr = 0.1



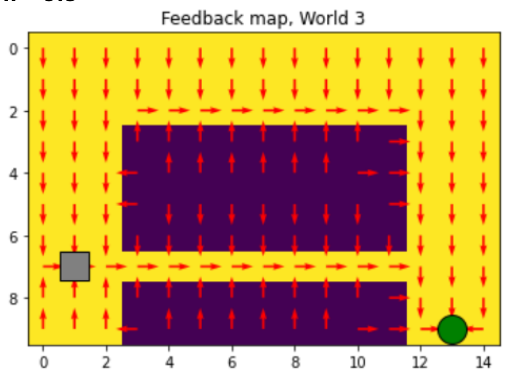
lr = 0.4



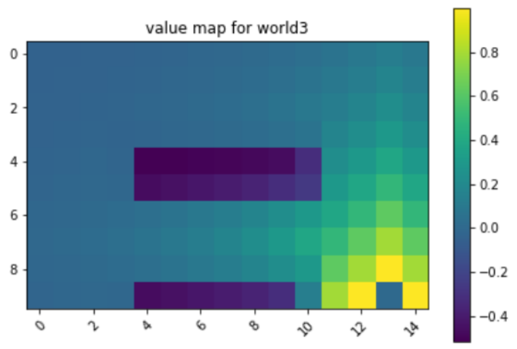
lr = 0.7



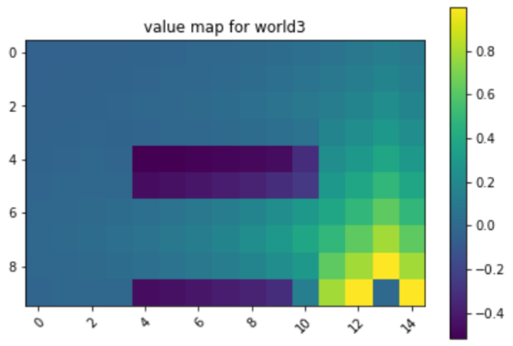
lr =0.9



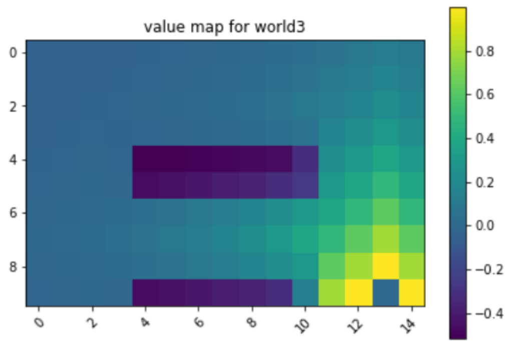
lr = 0.1



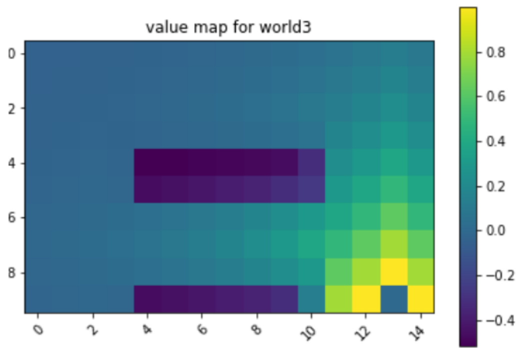
lr = 0.4



lr=0.7



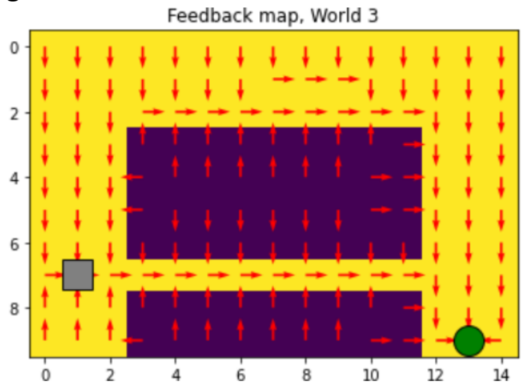
lr = 0.9



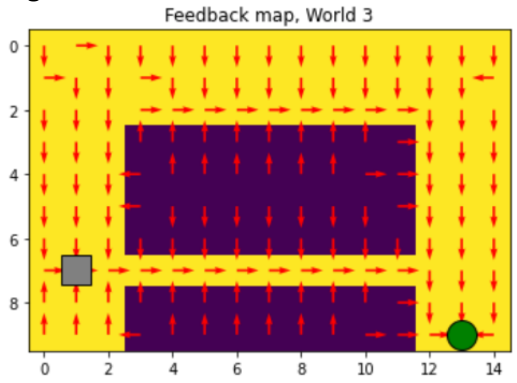
Although all the results look same good, but when the learning rate is too high, it will take more iterations to converge. From the updating function of Q-function, we can tell that higher learning rate will make 'memory' less important and 'learning' more important.

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

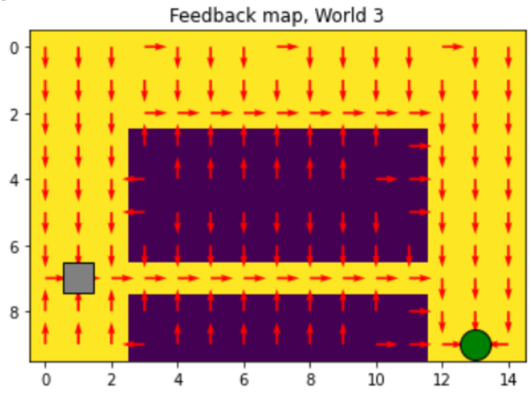
gamma = 0.2



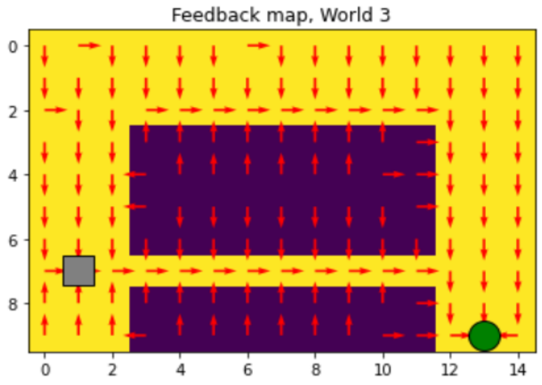
gamma = 0.5



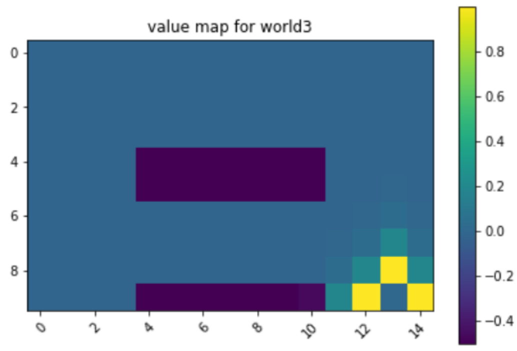
gamma = 0.8



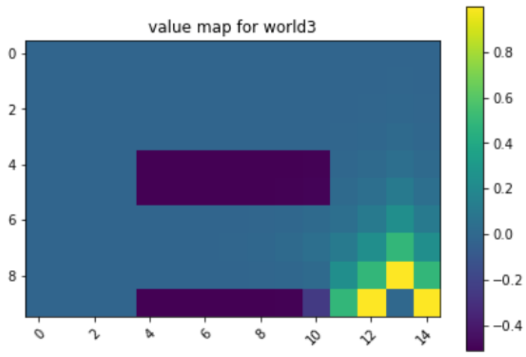
gamma=1



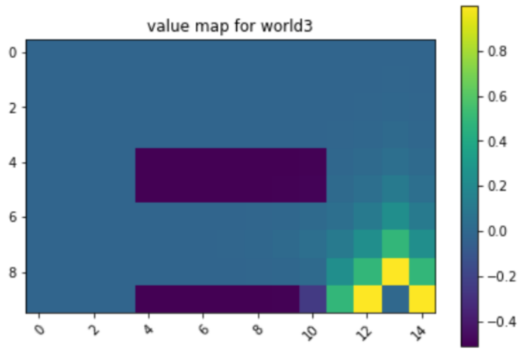
gamma = 0.2



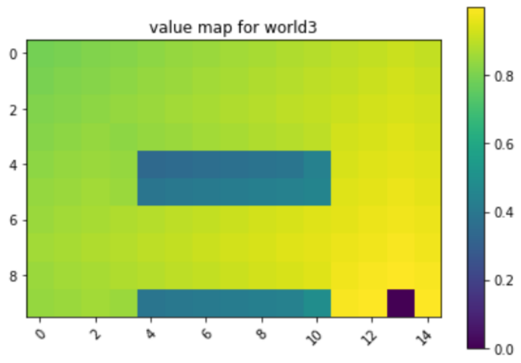
gamma = 0.5



gamma = 0.8



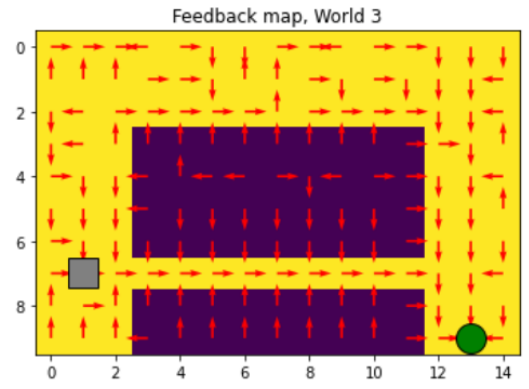
gamma = 1



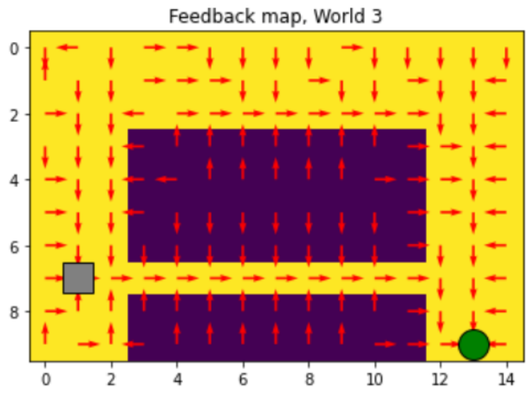
Gamma will influence the effect of future reward. In world 3, we set reward of reaching destination to be 1. So, from the V-function, we can see the reward from getting destination can have more influence on other states with the increase of gamma.

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?

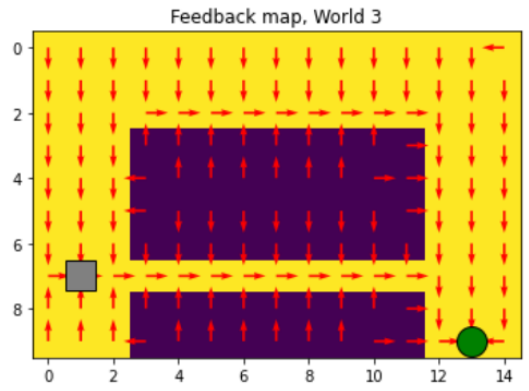
ε = 0.2



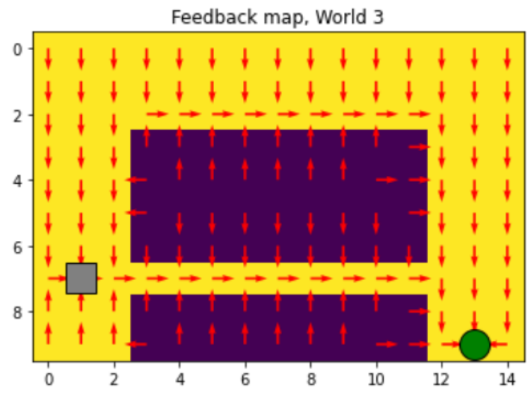
ε = 0.5



ε = 0.8



ε = 1



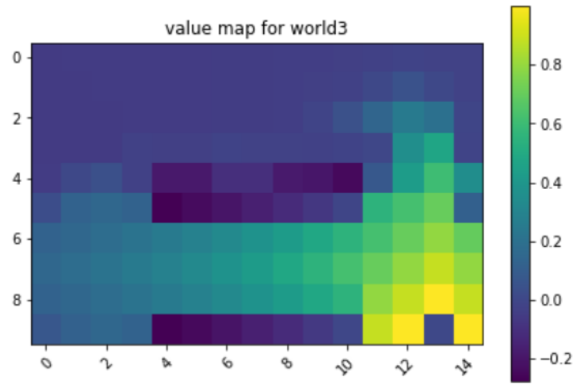
When the epsilon is too small, it shows lack of exploration. We can see in the picture of epsilon = 0.2 and epsilon = 0.5, there are many states seems not so reasonable from the arrow plot as well as v-function. But when epsilon become 1, it becomes random search, which takes much longer time to converge.

In the training, I apply 'staircase-strategy' to choose the epsilon:

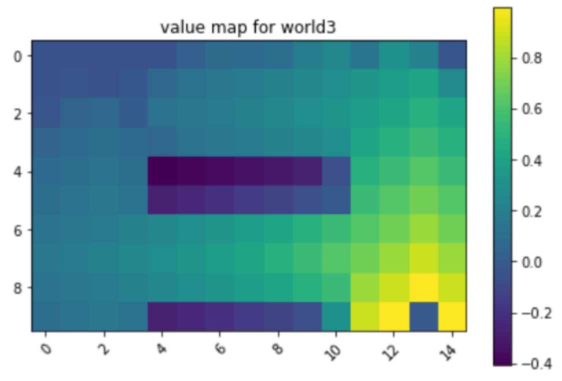
```
$if\ i/itertion < 0.5:$
    $eps = 0.8$
$elif\ i/itertion < 0.8:$
    $eps = 0.6$
$elif\ i/itertion < 1:$
    $eps = 0.2$
```

where i is the current iteration and iteration is the total iteration.

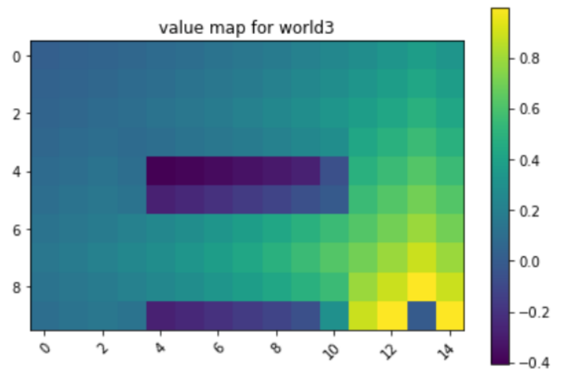
ε = 0.2



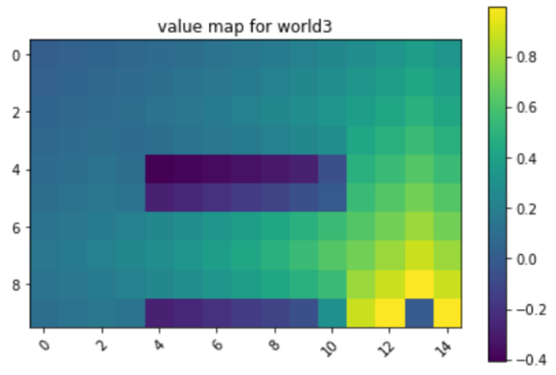
ε=0.5



ε=0.8



ε=1



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?**

    The original Dijkstra's cheapest path finding algorithm is a brute force algorithm, it can work in **statistic irritating blob** world since purple area can help us reduce the possibility that we need to test. However, if we apply this algorithm in **"Suddenly irritating blob" world,** it may need much more time to find the path since sometimes all the grids are having the same value.

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

    1 Most game have an optimal strategy, the most famous one could be Alpha Go. We can use reinforcement learning in this area to train the players.
    2 Training robots to accomplish some complex tasks like auto driving.

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?**