TBMI26 – Computer Assignment Reports
Reinforcement Learning

Deadline – March 14 2021

Author/-s:

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)**

$$V^*(s_k) = \sum_k \gamma r_k$$

The V-function describes the expected reward when being in an uncertain state following (in this case) the optimal policy. Gamma is a discount factor $0 < \gamma < 1$ and r is the reward for state k.

The Q-function describes the value for each action in each state, given that the optimal policy is followed when an action has been taken.

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

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

Q is updated by a factor for doing action j in state k plus a factor for the future reward of doing the action and then following the optimal policy.
η is the learning rate $0 \le \eta \le 1$. A large η overwrites previous values in the Q-matrix while a smaller η forces the Q-matrix to rely on older values.

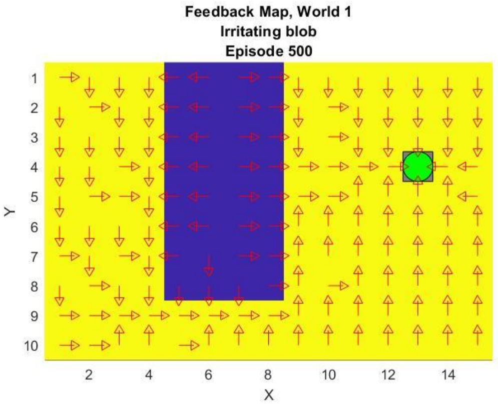
$$\mathrm{Q}(\mathrm{s_k}, \mathrm{a_j}) \leftarrow (1-\eta)\mathrm{Q}(\mathrm{s_k}, \mathrm{a_j}) + \eta\left(r + \gamma V^*(s_{k+1})\right) =$$
$$(1-\eta)\mathrm{Q}(\mathrm{s_k}, \mathrm{a_j}) + \eta\left(r + \gamma \max_a Q(s_{k+1}, a)\right)$$

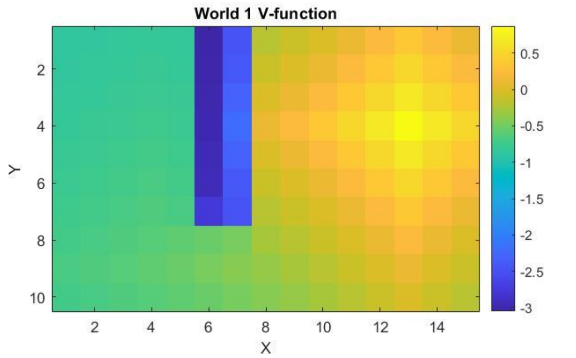
3. **Briefly describe your implementation, especially how you hinder the robot from exiting through the borders of a world.**
The Q-matrix is a (10,15,4)-matrix, (y,x,actions). It is initialized with uniformly distributed random numbers and updated using the update function above.
The (y,x,action_j)-matrix (j = 1,2,3,4) is limited in the direction of action_j by an infinitely negative reward. For the up-action the Q-matrix top row is negatively infinite, meaning there is no possibility to learn to choose the up-action at these positions.

After training the robot uses the optimal actions from the Q-matrix, $\max_a Q(s_k, a_j)$ for a given state s_k to find its way towards the goal.

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.**

World one has a rectangular space the robot needs to avoid. The states containing the space have a greater negative reward than the other states.
To solve this world, it was enough to linearly decrease epsilon (the exploration coefficient) for each episode. Adding a linearly increasing η reduced the number of episodes needed.



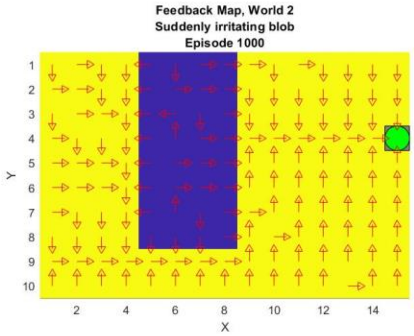


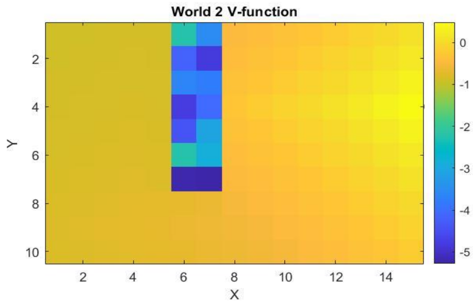
Episodes	η	ε	γ	B
500	0.2	1.0	0.9	1/3

B is a factor that is multiplied with the number of episodes. After this B*Episodes ε and η will start to linearly decrease and increase, respectively.

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.**

World 2 has about 1 in 6 chance of generating world one but with a much greater negative reward for the rectangular space and the rest of the time the world is generated without any obstacles. The reinforcement learning makes sure the robot stays on a trajectory to always avoid the rectangular space. By increasing the exploration time before starting to linearly increase ε the number of episodes could be reduced. Since the world was changing it was important that the old path was not overwritten by a temporarily better path, hence a very low, constant, η was used so the Q-function relied heavily on previous experience. The path seemed to become stable around 1000 episodes.



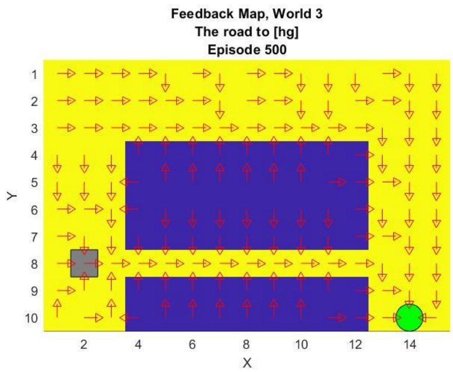


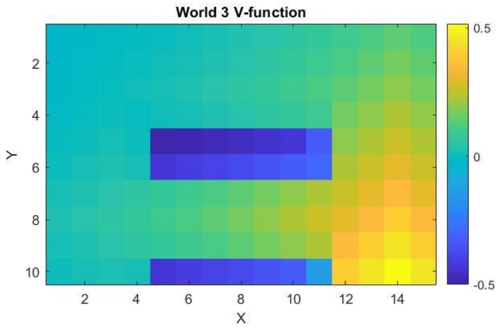
Episodes	η-init	ε-init	γ	B
1000	0.1	1.0	0.9	333

B is a factor that is multiplied with the number of episodes. After this B*Episodes ε will start to linearly decrease.

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.**

   World 3 consists of two paths too the goal, one longer and wider and one shorter and narrower. It was possible to get a good policy from every state using the same trick as in world 1 with a linearly increasing/decreasing η/ε after a certain exploration time.





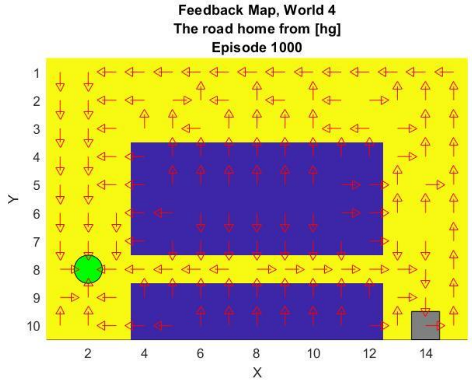
Episodes	η	ε	γ	B
500	0.2	1.0	0.9	1/3

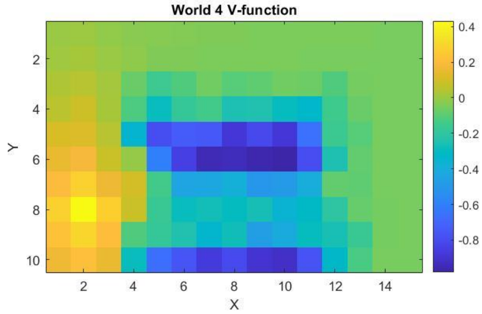
B is a factor that is multiplied with the number of episodes. After this B*Episodes ε and η will start to linearly decrease and increase, respectively.

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.**

   World 4 has the same layout as world three, but the robot and goal have switched positions. World 4 also have a 30% that the robot will take a random action at any state. Since the tunnel is only one space wide the robot will not go in the tunnel if it searches to maximize longtime rewards because there is a high probability for a very big negative reward.

This time a similar strategy as world 2 was implemented where the robot relied heavily on previous experience to navigate the world. Since there is a big risk for racking up negative rewards because of the randomness of the movement the Q-function will seek to stay along the perimeters to maximize the reward.



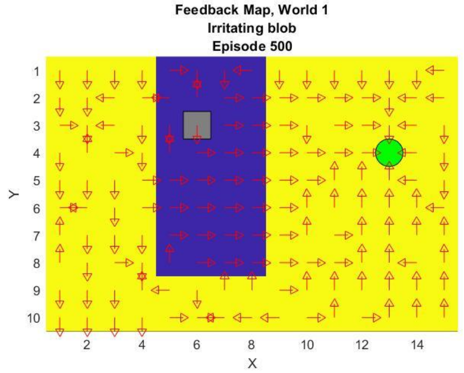


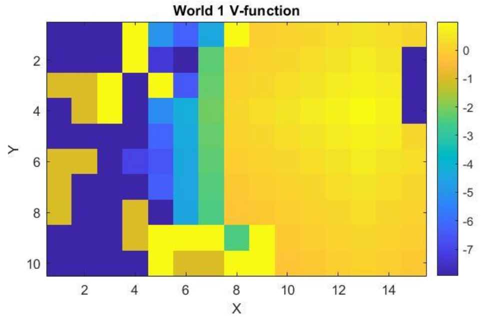
Episodes	η	ε	γ	B
1000	0.1	1.0	0.8	2/3

B is a factor that is multiplied with the number of episodes. After this B*Episodes ε will start to linearly decrease.

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

A smaller η will emphasize already learnt experience. A large initial η will continuously overwrite old values, so when there is no clear path to the goal the robot will have difficulty to find an optimized path because there is no previous data to rely on.

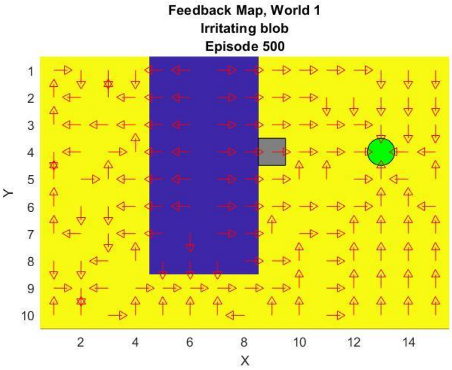


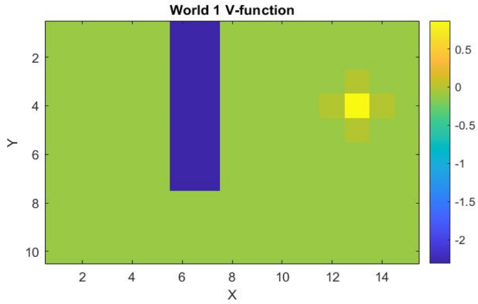


Episodes	η	ε	γ	B
500	0.9	1.0	0.9	2/3

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

A small gamma will maximize short term rewards while a big gamma will maximize the long-term rewards.



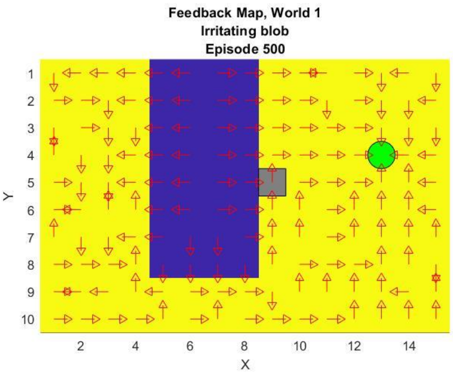


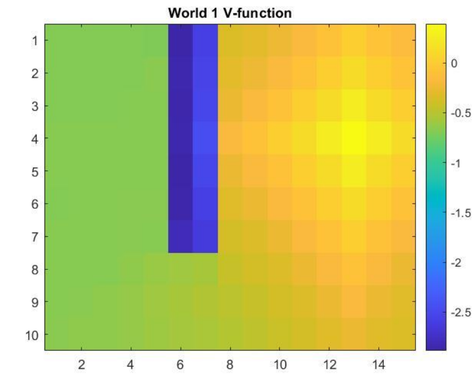
Episodes	η	ε	γ	B
500	0.1	1.0	0.1	2/3

In the figures above the value function is maximized for short term rewards, meaning that the best policy for only a few states is chosen. Going in the obstacle is immediately very negative while everything almost every other step is valued the same, except the steps immediately next to the goal.
It is hard to find an optimal policy on the left side of the obstacle because it doesn't learn the long-term rewards of moving away from the goal toward the bottom.

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 exploration factor is the probability that the robot will take a random action. During early training, the exploration factor needs to be big to make the robot evaluate the entire course. Towards the end the exploration factor should be smaller to make the robot evaluate a more optimized path towards the goal.





Episodes	η	ε	γ	B
500	0.1	0.9	0.9	2/3

Here ε is large and constant, the movement of the robot is largely random. Given the number of episodes it will find paths tending to the goal, but they will not be optimized. The results for the policy seem to have some randomness to the direction of the arrows. Since the robot often moves randomly and have a small η the v-function will be the same as with a decreasing ε (Q4).

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 irritating blob world, the optimal policy would be the same since the reinforcement learning learns the optimal path and world does not change.

    In the suddenly irritating blob world, the policies would differ. The reinforcement learning would learn to avoid the walls while Djikstras would probably move the robot through the wall since the cost would be lower to do so most of the times. Depending on implementation and rules of the world one would probably be preferred over the other.

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

    Reinforcement learning could be used in computer games to create harder AI adapted to the players playstyle.
    It can be used for trajectory optimization in self driving cars.
    It can be used as investment AI for the stock market.

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?**