TBMI26 – Computer Assignment Reports Reinforcement Learning

Deadline – March 15 2020

Author/-s: Gustav Wahlquist & William Stokke
Guswa204 & Wilst664

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)**

   $Q^*(s,a) = r(s,a) + \gamma V^*(g(s, \mu^*(s)))$

   Where r = reward for action a in state s
   γ *= discounting reward factor*
   $V^*(g(s, \mu^*(s)))$ *= max Q-value of new s*
   $g(s, \mu^*(s))$ *= sum of future rewards given a policy*

   $V^*(s) = \max Q(s,a)$ given a policy

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

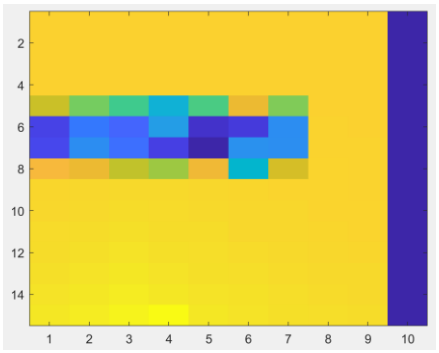
   New $Q(s,a) = (1-\alpha)Q(s,a) + \alpha Q^*(s',a')$
   Where α = learning rate
   s' = new state
   a' = new action

   The learning rule works in the way that you update the q-value for a specific state and action each time you choose that action. The q-value in the state you moved from is updated based on the optimal value of the current state(which in turn is affected by all possible actions and their rewards from the given state).

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

We have a table called "q-table" where we store the values for each action in every state. The table is updated for each action according to the learning-formula described above. We have a factor epsilon which controls the probability of taking an exploring action and discarding the action with the highest value (exploitation). The factor epsilon is decreasing for each iteration by a given factor until a minimum value of epsilon is reached. We iterate the algorithm for a number of episodes, and an episode is run until either the goal is reached, or the maximum amount of actions is reached. The reason for terminating if the maximum amount of actions is reached is so that we don't waste time if the algorithm gets stuck in a loop. When the algorithm has run for the given number of episodes the best policy is calculated by choosing the action with the highest value in each state.

To hinder the robot from exiting through the borders we initiate the actions that would make the robot exit the world to "-inf" so that these actions will never have a better value than valid actions. The robot can still choose a action which exits the world if this action is chosen randomly, but by the design of the world the robot will not actually take this action.



In the plot above you can see this area as the blue edge on the right-hand side.

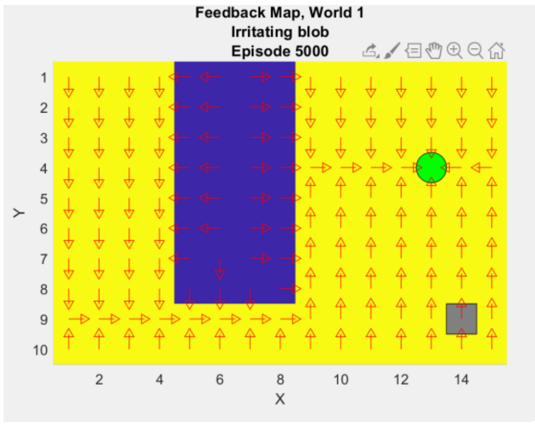
4. **Describe World 1. What is the goal of the reinforcement learning in this world? What parameters did you use to solve this world? Plot the policy and the V-function.**

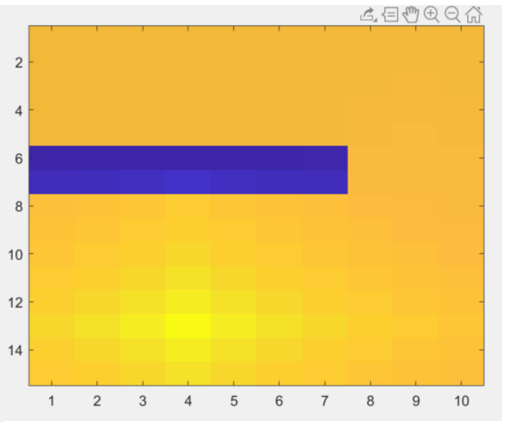
World 1 is a constant world with an area with large negative reward. The starting position of the robot is random. The position of the goal is fixed.

The goal of the reinforcement learning in this world is for the robot to reach the goal with as big total reward as possible.

Starting epsilon: 0.9
Epsilon decay rate: 0.01
Minimum epsilon: 0.1
Learning rate(α): 1
Discount factor(γ): 0.8
Number of episodes: 5000
Max actions: 100

Percentage of times the robot reaches the goal within 100 actions with the final policy (1000 tries): 100%





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.**

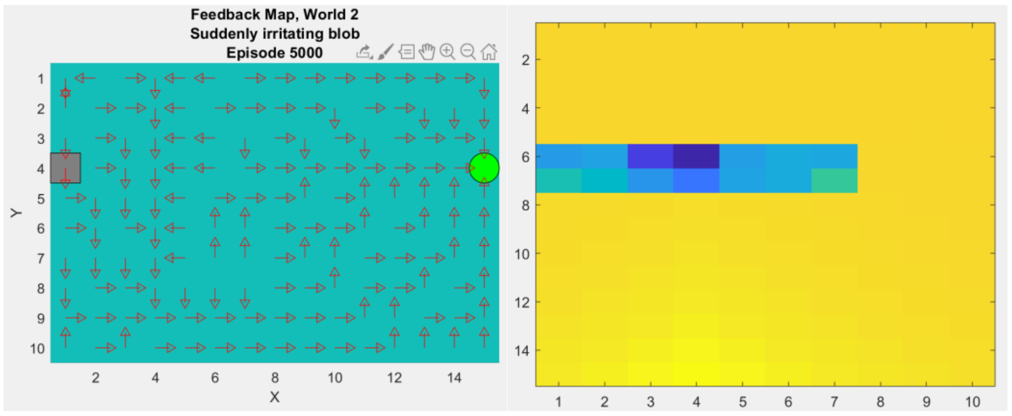
   World 2 is a world which alternates randomly between being initiated to either the same as world 1 or a world without the area with large negative reward. The starting position of the robot is random. The position of the goal is fixed.

   The goal of the reinforcement learning in this world is for the robot to reach the goal with as big total reward as possible. Doing this by finding one and only one optimal policy for the robot. (In this case the solution is not optimal for the two different versions of the world because of the limitation mentioned above).

   This is possible to do with reinforcement learning since the algorithm will take into account that the world is not deterministic. As you can see in the policy plotted below, the policy is still to go around the area with large negative feedback. This is due to the risk is to great that this area would give large negative feedback, even though the area is only there 20% of the time.

   Starting epsilon: 0.9
   Epsilon decay rate: 0.01
   Minimum epsilon: 0.1
   Learning rate(α): 0.1
   Discount factor(γ): 0.8
   Number of episodes: 5000
   Max actions: 100

   Percentage of times the robot reaches the goal within 100 actions with the final policy (1000 tries): 97,9%



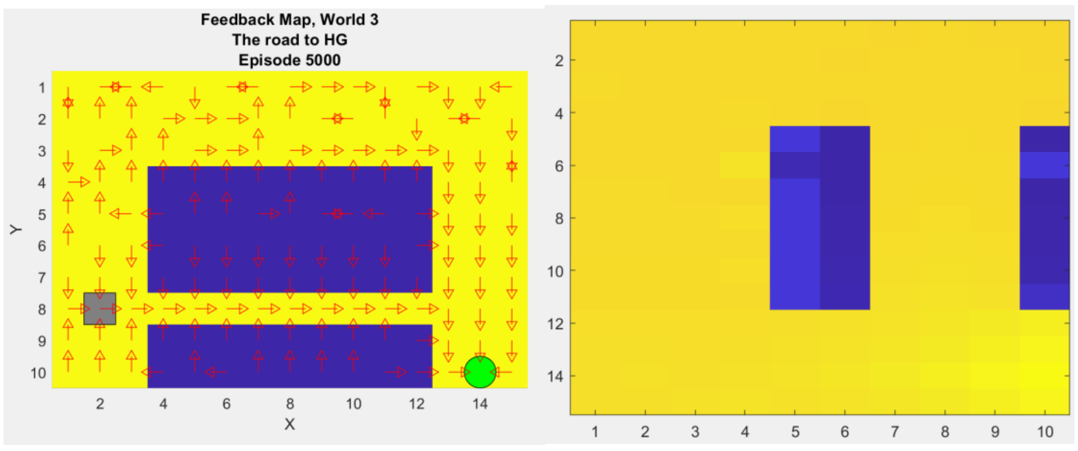
6. **Describe World 3. What is the goal of the reinforcement learning in this world? Is it possible to get a good policy from every state in this world, and if so how? What parameters did you use to solve this world? Plot the policy and the V-function.**

World 3 is a constant world with an area with large negative reward. The starting position of the robot is fixed. The position of the goal is fixed.

The goal of the reinforcement learning in this world is for the robot to reach the goal with as big total reward as possible.

To get a good policy from every state in the world you would need to force the robot to go there since when it finds the narrow path between the areas with large negative values, the robot will not go and update the values on the states higher up in the world (you can see this in the plotted policy). To get a good policy in these states you could force the robot to take a number of random steps in the beginning of an episode without updating the weights, and this is to mimic an random initiation. This would make it so that the algorithm is forced to find good policies from more states then the static initiation state.

Starting epsilon: 0.5
Epsilon decay rate: 0.001
Minimum epsilon: 0.1
Learning rate(α): 1
Discount factor(γ): 0.8
Number of episodes: 5000
Max actions: 100



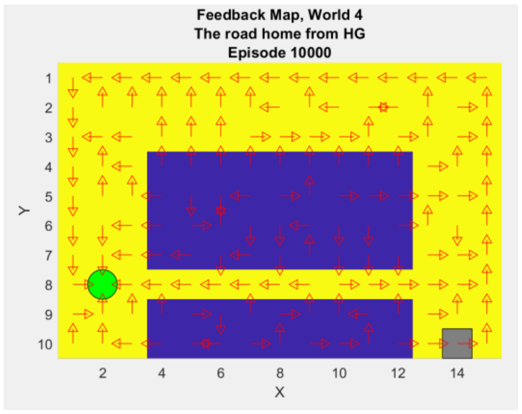
Percentage of times the robot reaches the goal within 100 actions with the final policy (1000 tries): 100%

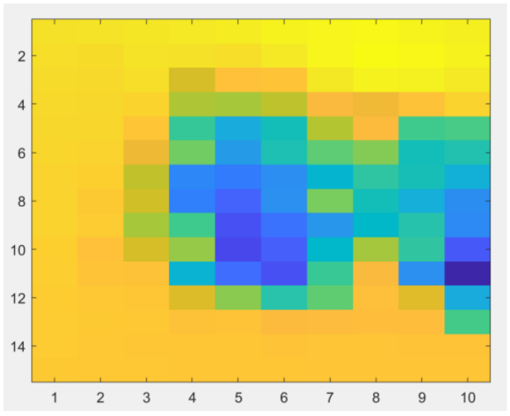
7. **Describe World 4. What is the goal of the reinforcement learning in this world? This world has a hidden trick. How is it different from world 3, and why can this be solved using reinforcement learning? What parameters did you use to solve this world? Plot the policy and the V-function.**

   The goal of the reinforcement learning in this world is for the robot to reach the goal with as big total reward as possible.

   World 4 is a constant world with an area with large negative reward. The starting position of the robot is fixed. The position of the goal is fixed. The "hidden trick" in this world is that there is a fixed chance (0.3) that the robot takes a random action. Since the risk of taking a random step is accounted for in the calculation of optimal V-function, the algorithm will take it into account when choosing an optimal policy.

   Starting epsilon: 0.5
   Epsilon decay rate: 0.001
   Minimum epsilon: 0.01
   Learning rate(α): 0.3
   Discount factor(γ): 0.95
   Number of episodes: 10000
   Max actions: 100

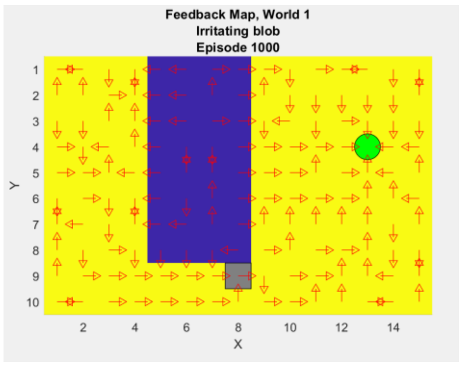


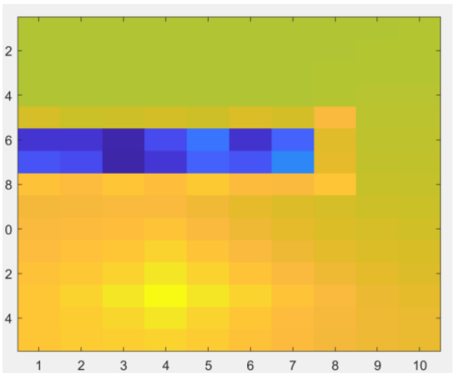


   Percentage of times the robot reaches the goal within 100 actions with the final policy (1000 tries): 99,8%

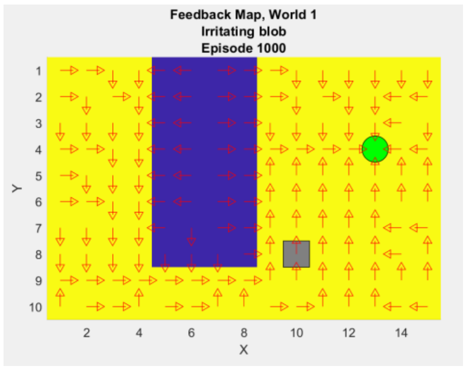
8. **Explain how the learning rate α influences the policy and V-function. Use figures to make your point.**

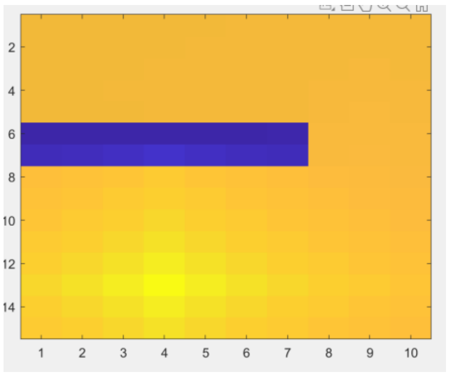
   Learning rate = 0.1





Learning rate = 0.9



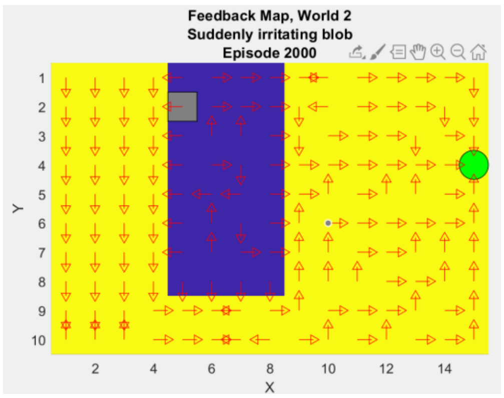


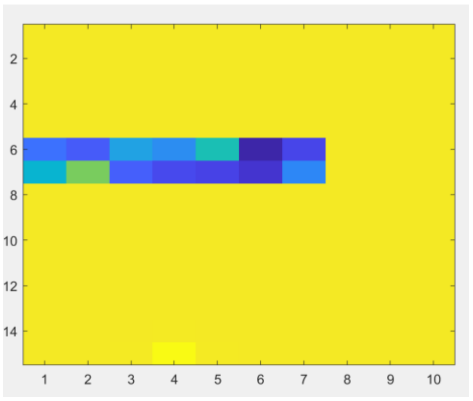
If the world is 100% deterministic, a high learning rate is optimal (preferably 1) since you can be sure that a action will always be the same. This is shown above in the plots.

On the other hand, if there is a random factor at play in the given world, a high learning rate would lead to the random actions being to impactful. A good practice would be for the learning rate to be proportional to the degree of randomness (i.e. that the probability for the random action is similar to the learning rate).

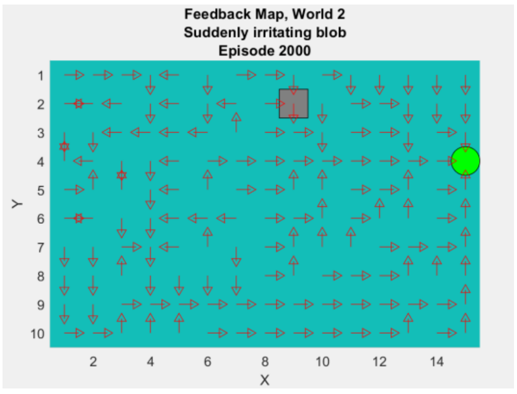
9. **Explain how the discount factor γ influences the policy and V-function. Use figures to make your point.**

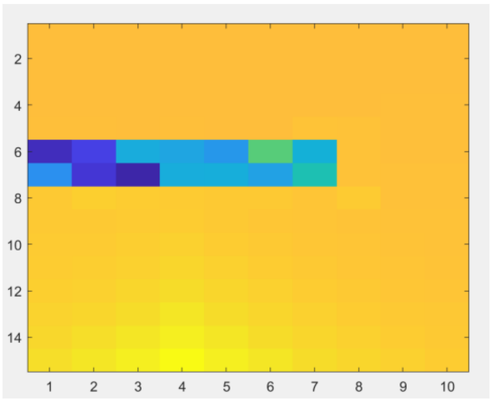
Discount factor = 0.1





Discount factor = 0.8

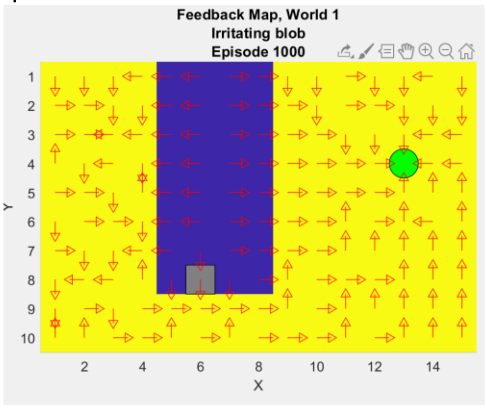


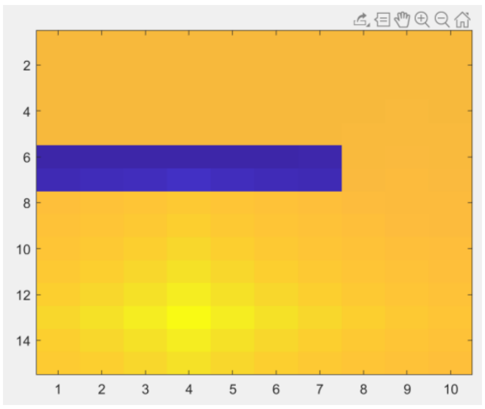


As we can see in the plots above, a higher discount factor will lead to values with large magnitude having a greater impact on the value in the surrounding states. As you can see, the second plot is darker since the large negative values in the middle area as influenced the rest of the map.

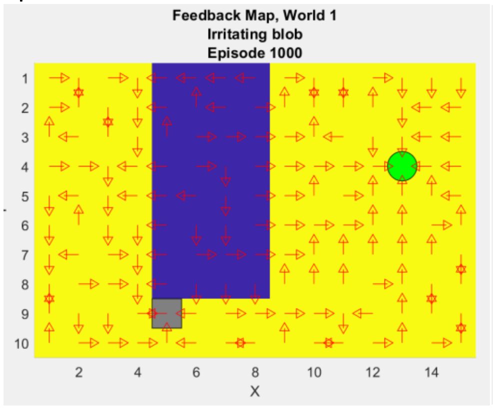
10. **Explain how the exploration rate ε influences the policy and V-function. Use figures to make your point. Did you use any strategy for changing ε during training?**

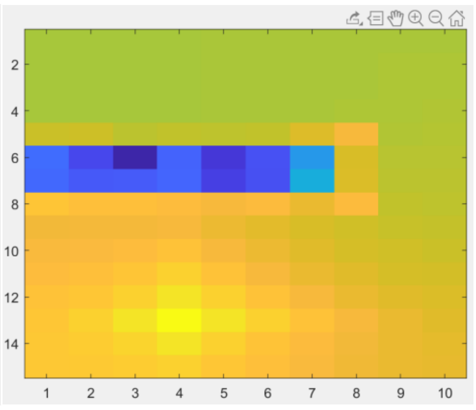
Eps = 0.9





Eps = 0.1





Here we can see that when we use a very low value for epsilon the solution seems to converge slower (we ran both these examples with 1000 episodes). But a to high value of epsilon should make it harder for the algorithm to fine tune the solution

In the lab we have added a decay rate for epsilon and this is so that we will explore more inte earlier episodes when we have little knowledge of the world and exploit more in later episodes.

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?**

    First of all, Dijkstra's algorithm needs to be able to know where the goal is. In neither of these cases the robot knows where the goal is which causes a big problem for the algorithm.

    Secondly, the "Suddenly irritating blob" is not deterministic. This causes a problem since Dijkstra's only runs one time and that could be on either of the two versions of the world.

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

    - It can be used to play games, with examples such as AlphaGo and AlphaGo Zero.
    - It can be used to learn robots tasks in manufacturing, with examples such as a robot learning to pick things and putting them in containers.

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?**