TBMI26 – Computer Assignment Reports Reinforcement Learning

Deadline – Mars 12 2018

Author/-s:
Villiam Rydfalk
Jonathan Sjölund

To pass the assignment, you will need to answer the following questions and upload the document to LISAM. If you meet the deadline we correct the report within one week after the deadline. Otherwise we give no guarantees when we have time.

1. **Define the V- and Q-function given an arbitrary policy as well as a given optimal policy (See lectures/classes).**
   V is the expected reward when being in a future state, when following a certain policy. Q is the expected future reward for doing action *a* in state *s* and then follow the optimal policy.
   $$Q(s_k, a) = r(s_k, a) + \gamma V^*(s_{k+1})$$
   $$V^*(s) = \max_a Q(s, a)$$

2. **Define a learning rule for the Q-function (Theory, see lectures/classes).**
   Updating the Q-function value by looking at the current state when executing an action *a,* and the future state when following the optimal policy.

3. **Describe your implementation, especially how you hinder the robot from exiting through the borders of a world.**
   Q-learning is implemented as this:
   1. Initializing all values, like the parameters and the Q-matrix (as a 3D zero matrix).
   2. Create a for-loop for running every iteration (episode) for updating the Q-function.
   3. In the loop a starting state is initialized every time for a selected world and the learning process for the Q-function starts.
   4. A while true loop runs till the state for the Q-function reaches the goal.
   5. While looping the action is selected by either doing greedy or random search with a percentage ε.
   6. The Q-function is updated, and we do this for N number of iterations (episodes).

   If an action is not valid a negative reward is given. For worlds 1-3, minus infinity is the reward, and for world 4, -3 is the reward. If minus infinity is used as a negative

reward for world 4 then everywhere we go is bad and we only rely on random movement.

4. **Describe the differences between the worlds explored by the robot. Any surprises?**
   Surprised that the learner has a much harder time trying to learn how to walk from HG to home than vice versa since the world is identical and only the starting position and goal have changed. This is a realistic scenario though since it is harder in real life to walk home from HG. :P

5. **For each world: Plot the V-function, i.e. how do you get to the goal from each position.**

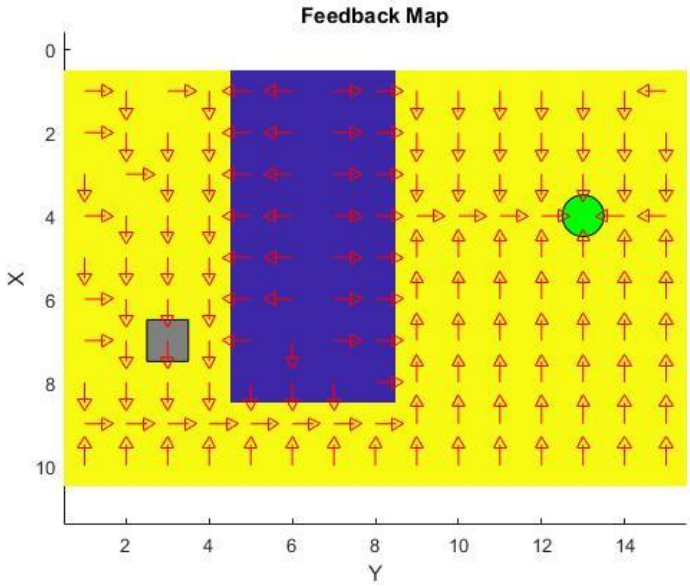

Figure 1: Path from Start to Goal for World 1

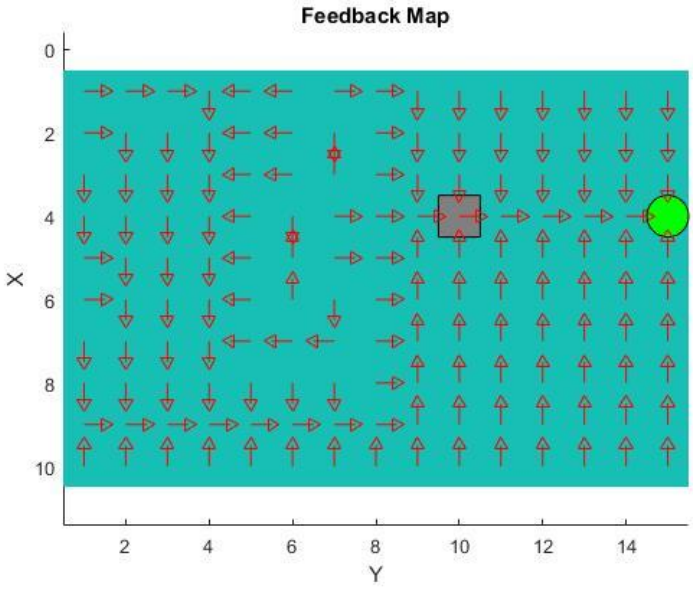

Figure 2: Path from Start to Goal for World 2

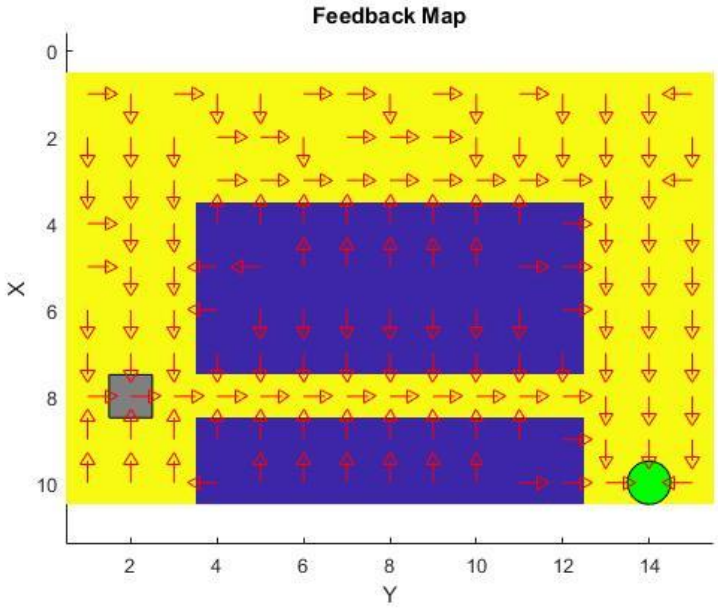

Figure 3: Path from Start to Goal for World 3

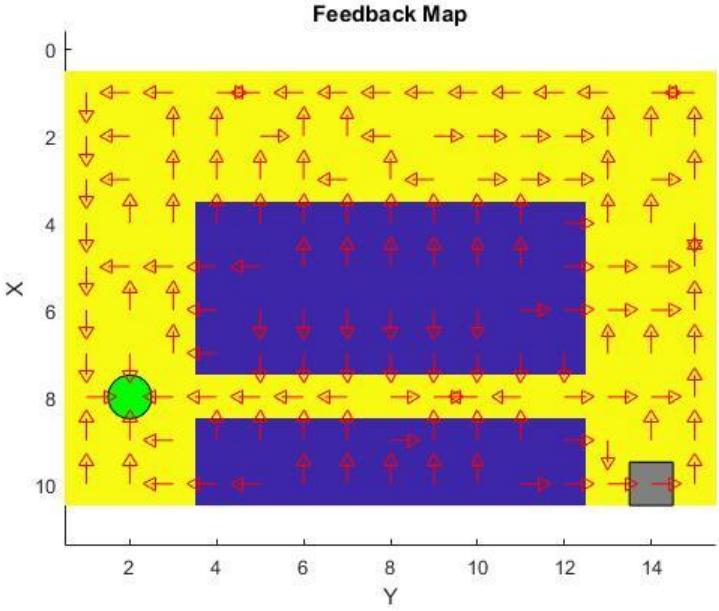

Figure 4: Path from Start to Goal for World 4

6. **For each world: describe the key observations you have made with respect to parameter choices. Provide documentation of the parameters you have used for each figure! A good rule is to provide each figure with a caption. Plot policies and the V-function for appropriate worlds to the extent you find appropriate to explain what you have done and learned during the assignment.**

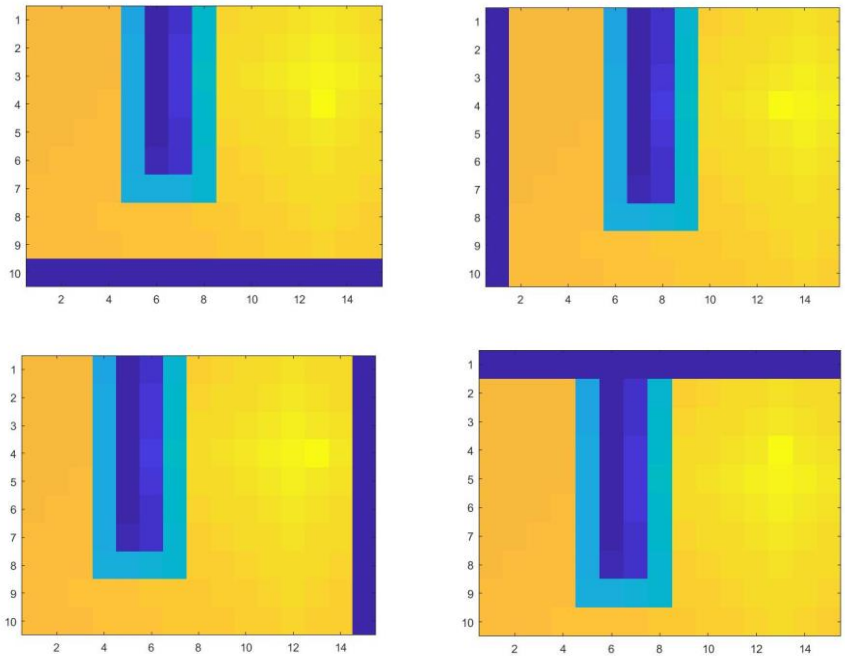

Figure 5: *Policy for World 1*

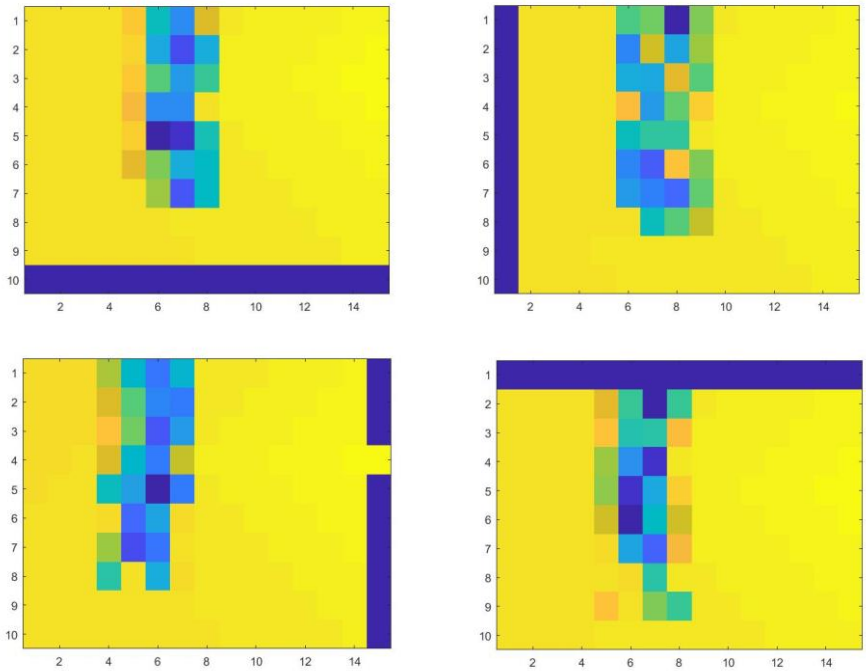

Figure 6: *Policy for World 2*

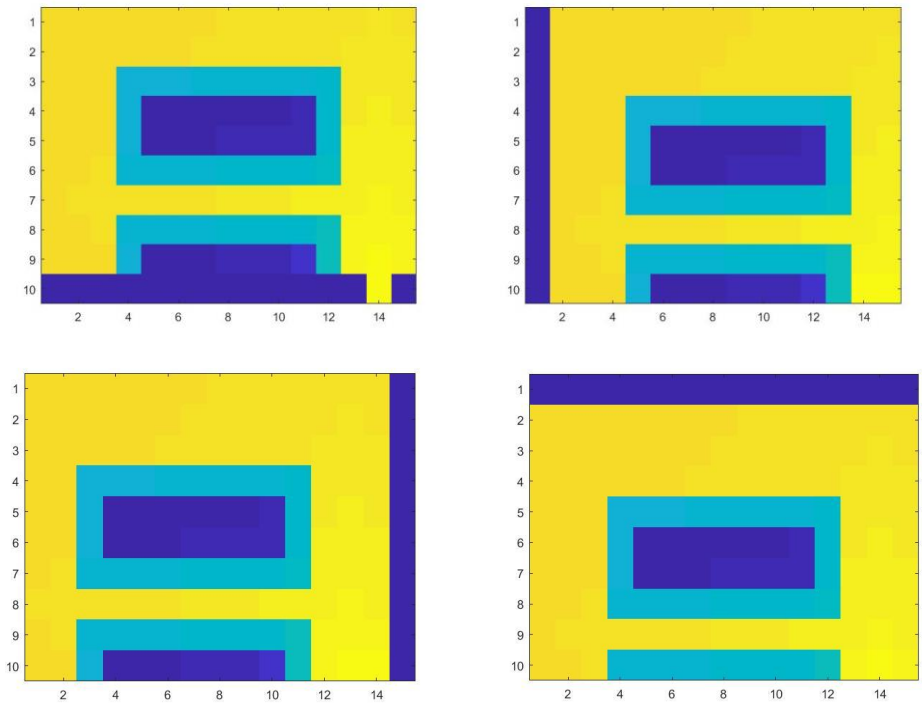

Figure 7: *Policy for World 3*

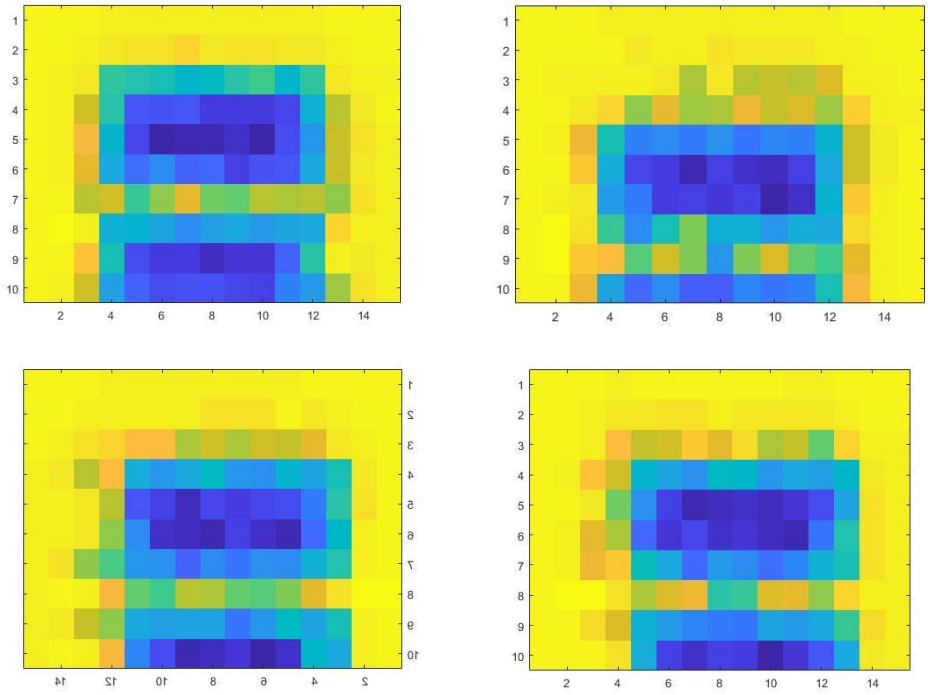

Figure 8: *Policy for World 4*

Table 1: *Parameter choice for each world*

Worlds (1,2,3,4)	Learning rate (α)	Discount factor (γ)	Exploration factor (ε)	Number of iterations	Reward for invalid action
1	0.4	0.9	0.8	4000	-infinity
2	0.4	0.9	0.8	4000	-infinity
3	0.4	0.9	0.8	4000	-infinity
4	0.2	0.7	0.8	6000	No reward

We chose the parameters according to the suggestions in the lecture slides. Good start value for the learning rate is between 0.1-0.5, good start value for the discount factor is 0.9, and the exploration factor should start big and decrease. For the final world we had to change the negative reward since we noticed that the robot did not want to move anywhere and only relied on random movement. Since world 4 was harder it was natural that more iterations were needed and since there were more iterations it seemed necessary to change the learning rate and discount factor (lower it) because we train longer and do not want to be bias toward some paths that might not be "optimal". For the first three world we tested for some values that seemed reasonable and it worked "perfectly" on the first try and did not change much except for the number of iterations. We noticed that if we lowered the number of iterations we could get stuck in some places when the exploration factor got closer to 0.0.

7. **What would happen if we were to only use Dijkstra's shortest path finding algorithm in the ''Suddenly Irritating blob'' world? What about in the static ''Irritating blob'' world?**
   It would cause problem in the "Suddenly irritating blob" world since sometimes the world have equal cost everywhere and for the case when there is wall in the way it would ignore it. For the irritating blob we would have a "wall" that we must (or would like to) go around so we would find a path to the goal without any problem.

8. **Include an in-depth description of the to/from HG worlds (world 3 and 4). What happens on the way from HG? How and why can this problem be solved with Q-learning? Which path does the robot prefer, and why?**
   When going to HG the robot generally prefers the shortest path, and when going home it prefers the longer path. The problem seems to be that in world 4 there seems to be a 30% chance that the agent takes a random action while it thinks it does another action which results in a weird update for the Q-function. Prioritizing moving left more than moving right makes the agent more prone to take the shorter path than the longer. Another solution is to check the agents current and previous position and look at the action it took, if the action does not result the agent moving from the before current position then we know something is not right and do not update the Q-function. This is, however, dangerous for the agent which is why it wants to take the longer path since it is wider and safer.

9. **Can you think of any application where reinforcement learning could be of practical use? A hint is to use the Internet.**
   Have been used for predicting handovers in a cellular network. Training a system so it knows where it should do a handover depending on where the user is, and which cell towers are closest.

10. **How does the different parameters (α, γ …) influence learning and appearance of the Q- and V-functions?**
    If alpha is too high, then small changes in later stages would have high effect and vice versa. The discount factor affects how much to prioritize a specific policy and affects the cost of each policy from start to goal. The exploration factor influences how much to explore compared to choosing the optimal policy. It is good to have a big exploration factor to explore more in the beginning and when we get closer to the end we want to do greedy search more (chose the optimal policy found). However, for some worlds we do not want to completely rely on greedy search because if we have not trained well enough the system might never reach its goal towards the end and therefore the exploration factor should not reach 0.0 but some minimum value. The negative reward affects the appearance of the world since if we set it to negative infinity we will never go to some places and for world 4 it did not work since then the robot did not want to go anywhere.