TBMI26 – Computer Assignment Reports Reinforcement Learning

Deadline – March 14 2021

Author/-s:
Martin Gustafsson <margu424>
Hoda Fakharzadehjahromy <hodfa840>

1. **Define the V- and Q-function given an optimal policy. Use equations and describe what they represent. (See lectures/classes)**

Let us assume we have an optimal policy π which returns an action a_t when given a state s_t.
The action a_t then moves the agent to state s_{t+1}.
We also have a reward function r(s, a) which given a state and an action returns a reward.
With this we can define the Value function (V) as follows:

$$\mathrm{V}^*(s_t) = \sum_{k=0}^{\infty} r(s_{t+k}, a_{t+k})$$

This means that we follow the optimal policy and collect all the rewards along the way.
We can also define the Q-function as follows:

$$\mathrm{Q}(s_t, \mathrm{a}) = r(s_t, a) + \gamma \mathrm{V}^*(s_{t+1})$$

The Q-function encodes the expected reward for performing an action a in a state s_t and then following the optimal policy. The discount factor γ is used to control the value of short-term rewards compared to long-term rewards.

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

The learning rule is defined as follows:

$$Q(s_t, \mathrm{a}) = (1 - \eta)Q(s_t, \mathrm{a}) + \eta(r(s_t, a) + \gamma \mathrm{V}^*(s_{t+1}))$$

This equation defines how we update the values in the Q-function given a learning rate η.
The equation takes the current estimate and a new estimate and takes a weighted average of the two estimates. The weights are controlled by the learning rate, a learning rate close to 0 means we prioritize previous estimates while a learning rate close to 1 means we prioritize new estimates.

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

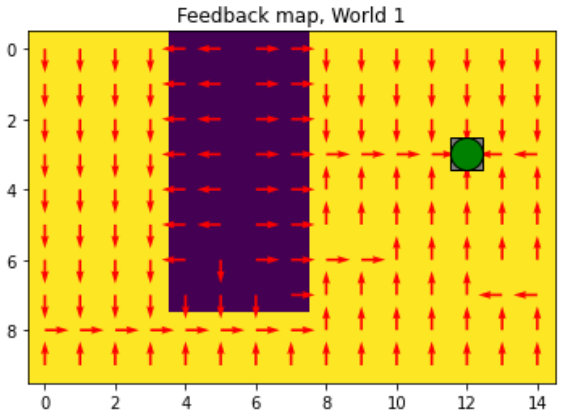
First we initialize the Q-table to small random negative values.
In order to avoid problems with the robot exiting the borders of the world we set the Q-value for the action to cross the border to $-\infty$, this means that the action of going left when at the left border of the world has a Q-value of $-\infty$.
Then we run a specified number of episodes and with each episode we initialize the world and randomize the position of the robot.
Then we loop until we are at the goal state and in each step either explore a random action or take the best action given the current policy based on the exploration factor.

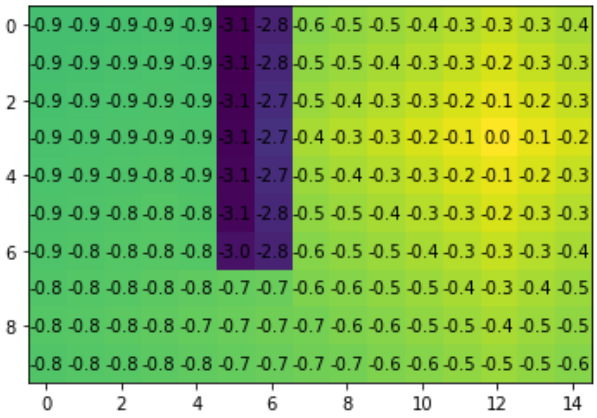
In order to not exit the borders of the world in a step we try to execute the action that we selected and if it is not valid we try again until we find a valid action.
Then we just update the Q-table according to the learning rule.

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 simple world with a big blob of bad in the middle. The goal of reinforcement learning in this world is to get the robot to avoid the blob.
The parameters we used for this were:

Learning rate	Discount factor	Exploration factor	Number of episodes
0.3	**0.9**	**0.95 – 0.2 (linearly)**	**10000**

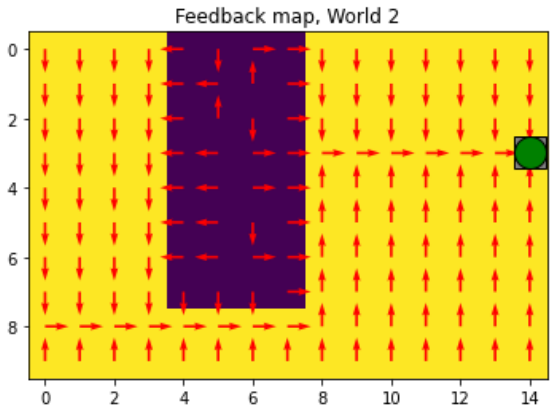


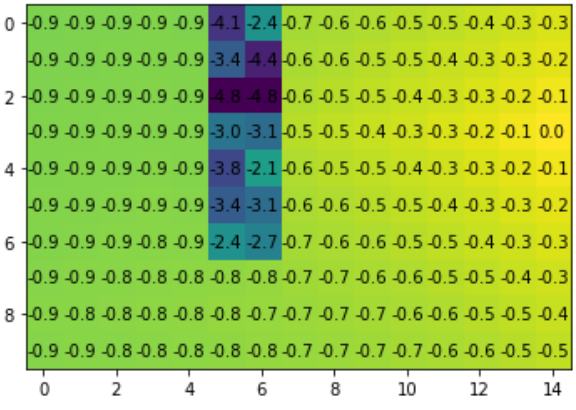


5. **Describe World 2. What is the goal of the reinforcement learning in this world? This world has a hidden trick. Describe the trick and why this can be solved with reinforcement learning. What parameters did you use to solve this world? Plot the policy and the V-function.**

World 2 is very similar to world 1 but with the twist that the blob is only there sometimes.
So we want a robot that still avoids the blob but we do not know if the blob is there so the robot should avoid the place where the blob could be all the time.
This can be solved with reinforcement learning because given enough training it will learn that the blob might be there and to avoid that spot.
The parameters we used for this were:

Learning rate	Discount factor	Exploration factor	Number of episodes
0.1	**0.9**	**0.95 – 0.4 (linearly)**	**50000**

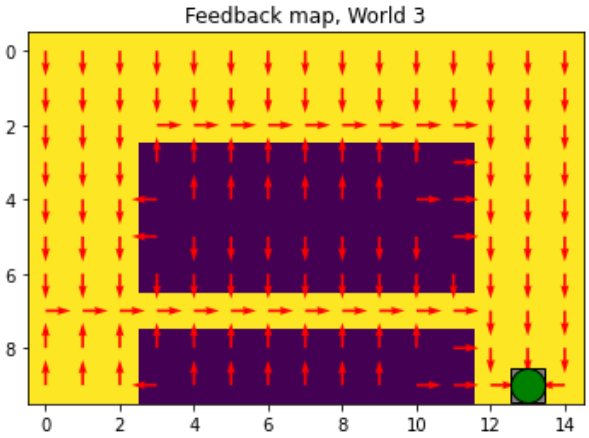


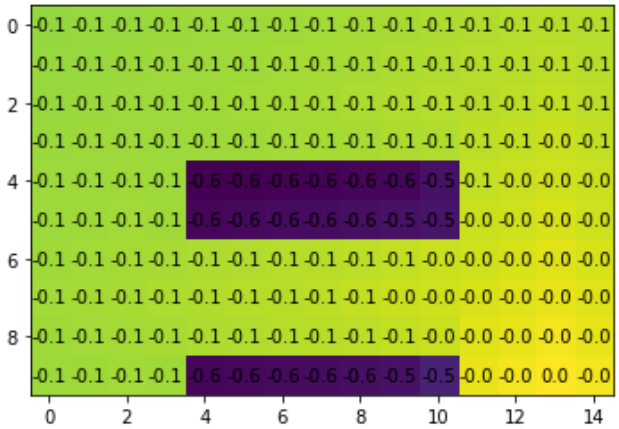


6. **Describe World 3. What is the goal of the reinforcement learning in this world? Is it possible to get a good policy from every state in this world, and if so how? What parameters did you use to solve this world? Plot the policy and the V-function.**

World 3 is different from the previous worlds in that there is a blob we want to avoid but there is a small path through the blob that we can use if we want to.
The start state is always the same in this world but with a high enough exploration factor we can get a good policy from every state.
The parameters we used for this were:

Learning rate	Discount factor	Exploration factor	Number of episodes
0.2	**0.9**	**0.95 – 0.4 (linearly)**	**10000**

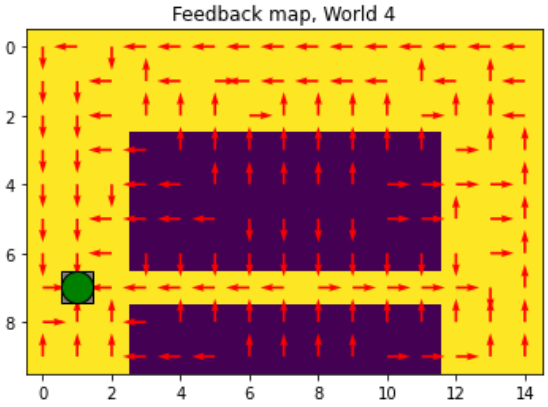


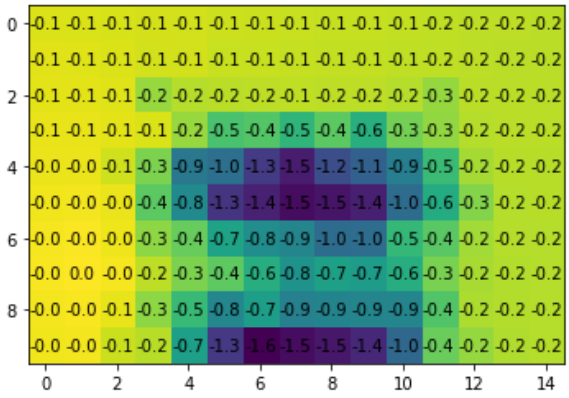


7. **Describe World 4. What is the goal of the reinforcement learning in this world? This world has a hidden trick. How is it different from world 3, and why can this be solved using reinforcement learning? What parameters did you use to solve this world? Plot the policy and the V-function.**

World 4 is the same as world 3 but with a trick when it comes to the action being taken.
There is a chance that the robot malfunctions and performs a random action instead of the specified one.
The goal of reinforcement learning in this world is to avoid the blob with this malfunction in mind, so keep as far away from the blob as possible in order to not malfunction into the blob, therefore the shortcut that exists in the world might not be that good to take as a malfunction means you enter the blob.
This can be learned by reinforcement learning by repeatedly trying and then seeing that shortcut is not worth it as a malfunction means a penalty.
he parameters we used for this were:

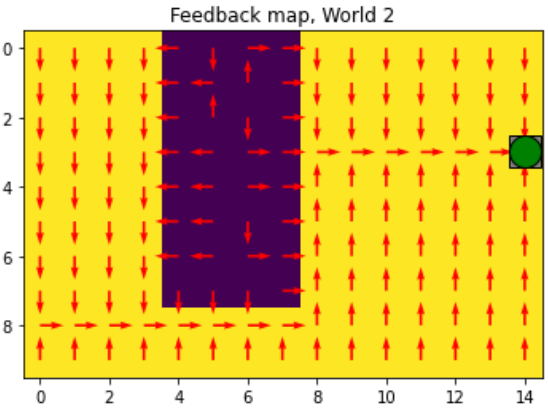
Learning rate	Discount factor	Exploration factor	Number of episodes
0.2	**0.95**	**0.95 – 0.5 (linearly)**	**50000**

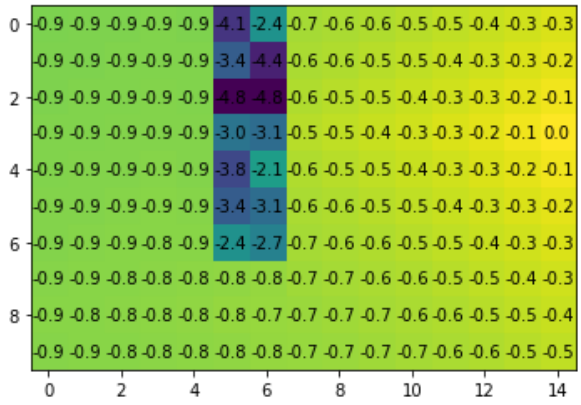




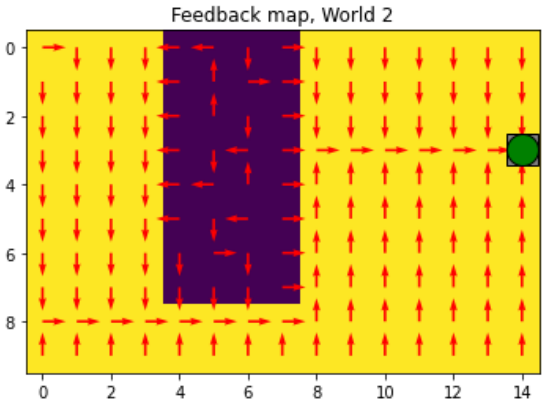
8. Explain how the learning rate α influences the policy and V-function. Use figures to make your point.

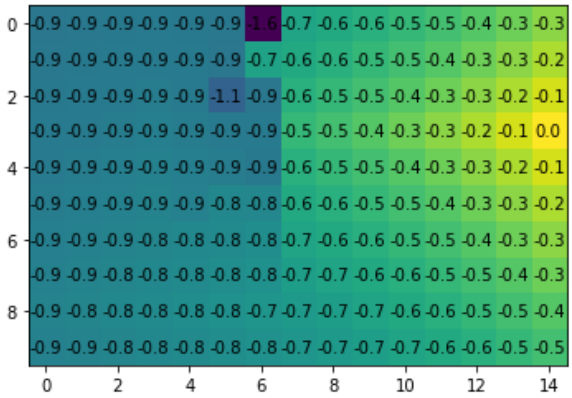
Learning rate	Discount factor	Exploration factor	Number of episodes
0.1	0.9	0.95 – 0.4 (linearly)	50000





Learning rate	Discount factor	Exploration factor	Number of episodes
0.9	0.9	0.95 – 0.4 (linearly)	50000

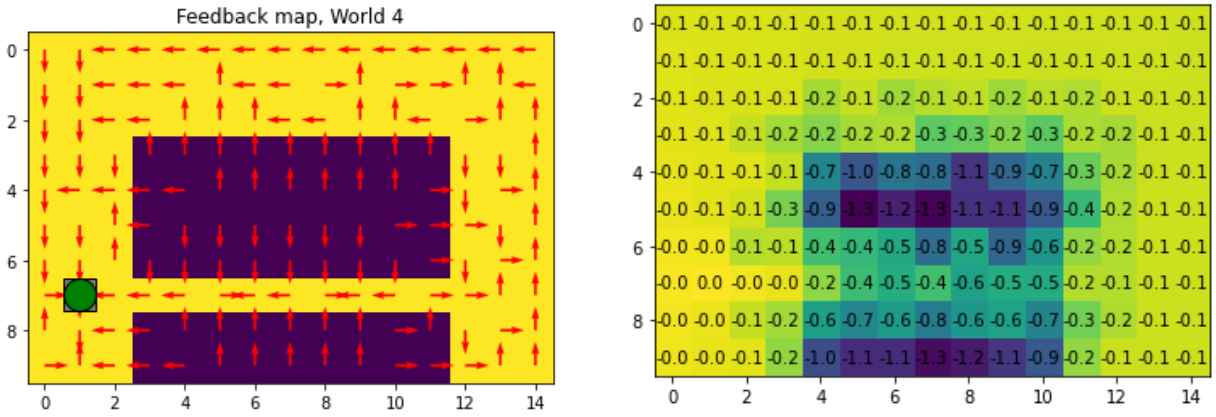




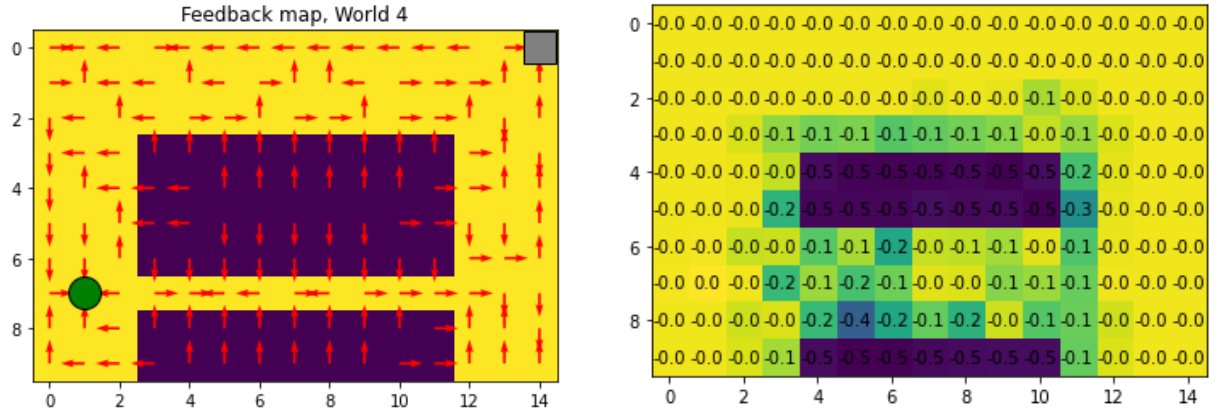
Here we see the policy and the V-function for two different learning rates.
We see that in the one with a high learning rate the blob is not very well visible in the V-function. This is because with a high learning rate we replace previous experiences with new experiences so the times we find a blob are replaced with the times we don't find a blob resulting in a world-view that does not see the blob.

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

Learning rate	Discount factor	Exploration factor	Number of episodes
0.2	0.9	0.95 – 0.4 (linearly)	5000



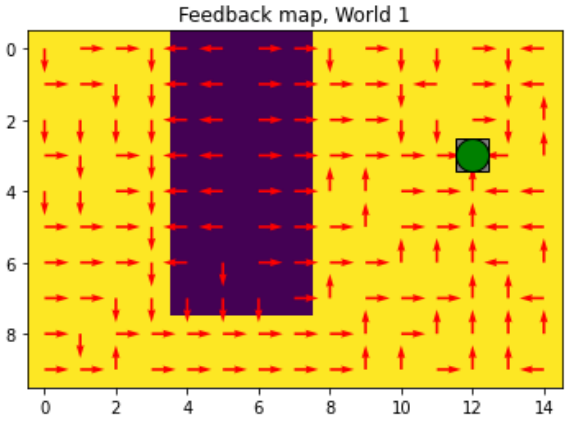
Learning rate	Discount factor	Exploration factor	Number of episodes
0.2	0.1	0.95 – 0.4 (linearly)	5000

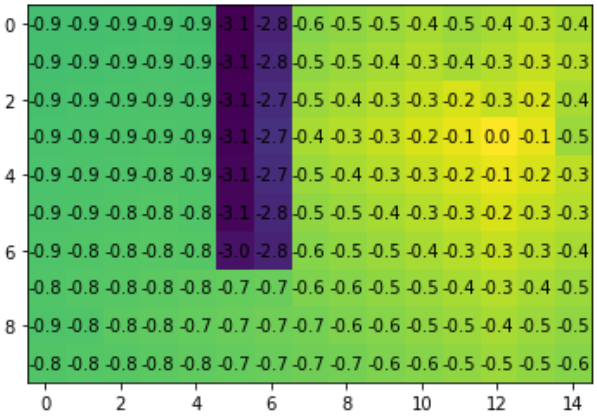


What we see in the images above is that with a low discount factor short-term rewards and punishments are shown clearer in the V-function. So the blob is clearly visible in the low discount factor case as we have found it and then concluded that that was bad, this compared to the case with a higher discount factor where we have found the blob but still realize that we are now closer to the goal so it is not all bad. We also see that in the case with a low discount factor we try to avoid the blob so much that we end up in the top right corner, we are so afraid of the blob that is better to just stay in the corner for as long as possible to avoid the punishment of entering the blob.

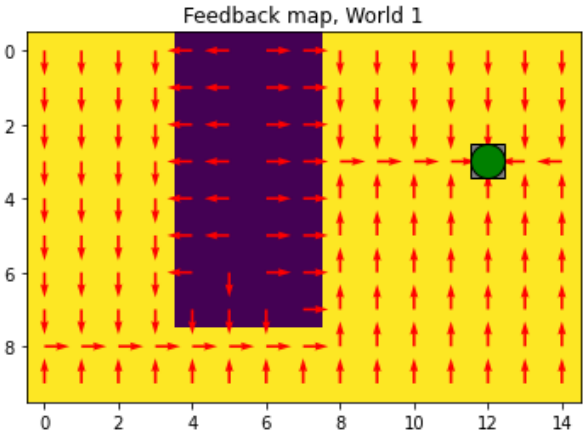
10. **Explain how the exploration rate ε influences the policy and V-function. Use figures to make your point. Did you use any strategy for changing ε during training?**

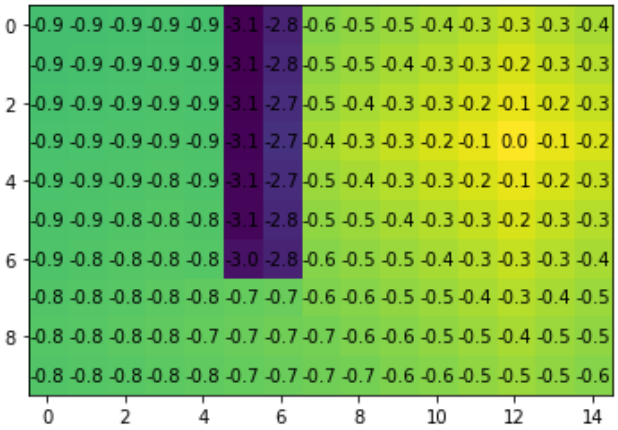
Learning rate	Discount factor	Exploration factor	Number of episodes
0.3	0.9	0.05	10000





Learning rate	Discount factor	Exploration factor	Number of episodes
0.3	0.9	0.95	10000





Here we can see the effect of having a low exploration factor. We keep some of the original path that we get from the initialization of the Q-table. This is because with a low exploration factor we might never explore a specific action and thus never update it's value in the Q-table.
This is what we think has happened with the action just above the goal in the policy for the low exploration factor. We never explored the option of going down (directly to the goal state) so we never updated the Q-table with the result of that action and thus never considered that action.

In our implementation we changed the learning rate linearly with the episode number. So if we defined a range of 0.8 – 0.4 exploration factor over 1000 episodes we would at episode 1 have an exploration factor of 0.8, at episode 1000 have an exploration factor of 0.4 and at episode 500 have an exploration factor of 0.6.

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?**

For this question we assume that the Dijkstra's algorithm is implemented so that we run it to find the shortest path and then just store this path and follow it every time we test. To do this we would need to run it once for each starting position and save the path from there. So training would be run Dijkstra's from each starting position and save the shortest path to the goal state.
We can then conclude that for the irritating blob world this agent would perform the same as the reinforcement learning agent as the cheapest path is also the one avoiding the blob.
But for the suddenly irritating blob this training process would not work as well as for some starting positions the blob might be there and for some the blob might not. That would mean that for some starting positions we would pass through the blob and for some we would avoid it.
So the Dijkstra's algorithm implementation would not do what we want it to do in this world, it would not reliably avoid the blob.

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

One practical use of a reinforcement learning could be when implementing an algorithm for advertisement recommendations. One could easily see a reward-based system where a click on the advertisement is rewarded and a close is punished. The only problem might be training as you would need real humans to train on but given enough reach it might be possible**.**