TBMI26 – Computer Assignment Reports Reinforcement Learning

Deadline – March 14 2021

Author/-s: Mohammed Ali & Simon Alsén

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)**
   $V^*(s)$ denotes the value function, which for a given policy returns the expected, future reward for state s. The function represents how good it is for the agent to be in a given state s given a policy. The function is defined as the Q value of the action a that results in the maximal Q value for state s:
   $$V^*(s) = \max_a Q(s,a)$$
   $Q(s_k, a)$ denotes the Q function, and returns the expected, future reward of action a in state s for a given policy. The Q function is defined as the sum of the reward of action a in state s_k and the product of the discount factor γ and the value function for the next state. The discount factor γ determines the importance of future rewards. If γ is close to 0, the algorithm will try to maximize short-term rewards, whilst if γ is close to 1 the algorithm will instead try to maximize long-term rewards.
   $$Q(s_k, a) = r(s_k, a) + \gamma V^*(s_{k+1})$$
   Given an optimal policy, the Q function gives the expected, future reward for all available actions a, while the value function gives the expected, future reward for the best action a.

2. **Define a learning rule (equation) for the Q-function and describe how it works. (Theory, see lectures/classes)**
   $$\hat{Q}(s_k, a_j) \leftarrow (1-\eta)\hat{Q}(s_k, a_j) + \eta\left(r + \gamma \max_a Q(s_{k+1}, a)\right)$$
   s_k is the current state, a_j is an available action, η is the learning rate, r is the reward for taking action a_j in state s_k and γ is the discount factor (described in question 1) which is multiplied with the value from the value function for the next state s_{k+1} (also described in question 1).

   The learning rate η determines the importance of new information. If η is close to 0, the algorithms will put more emphasis on previously learned information (i.e., $\hat{Q}(s_k, a_j)$), while a value close to 1 will put more emphasis on newly acquired information (i.e., $r + \gamma \max_a Q(s_{k+1}, a)$) and overwrite previous experience.

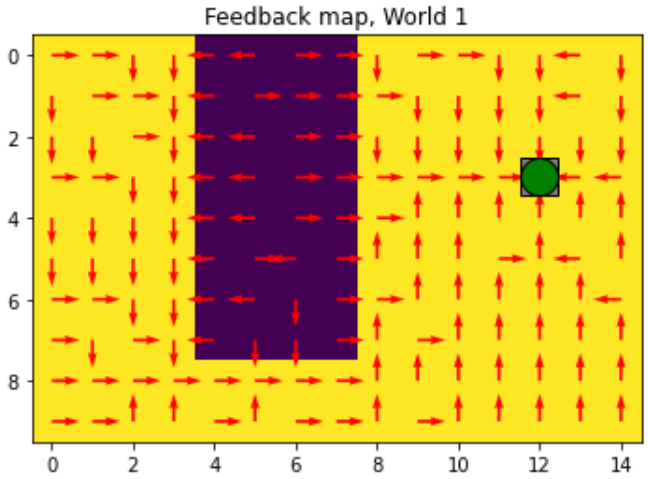
3. **Briefly describe your implementation, especially how you hinder the robot from exiting through the borders of a world.**

The Q table is initialized with random values. For each episode, the algorithm is run, and the Q table is updated using the equation described in question 2. At each step, the algorithm either makes a random action with probability ε, or a greedy action with probability $1-\varepsilon$. To hinder the robot from exiting through the borders of a world, we set the Q values for such actions (i.e., actions where self.valid is not 1) to negative infinity.

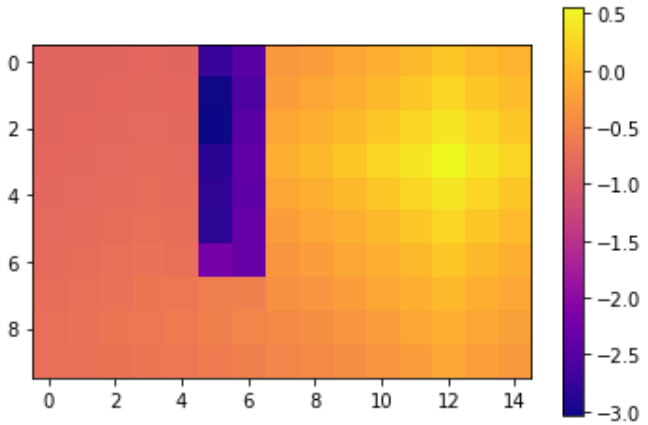
4. **Describe World 1. What is the goal of the reinforcement learning in this world? What parameters did you use to solve this world? Plot the policy and the V-function.**
The agent is initialized in a random starting position, and the endpoint is located at $(12,3)$. The reward for all positions is -0.1 except for the rectangle in the middle where the reward is -2.32. The agent's goal is to find the cheapest path to the endpoint.

We used the following hyperparameters:
$\eta = 0.5$
$\gamma = 0.9$
$\varepsilon = 1$
ε decay = 0.999
Episodes = 1000

Policy:



Value function:



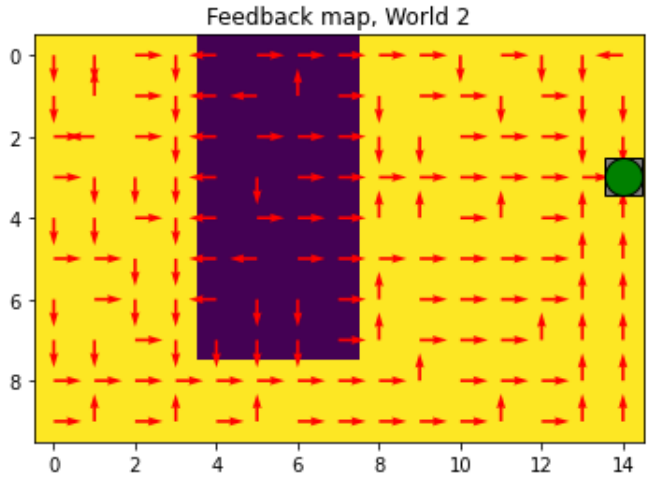
5. **Describe World 2. What is the goal of the reinforcement learning in this world? This world has a hidden trick. Describe the trick and why this can be solved with reinforcement learning. What parameters did you use to solve this world? Plot the policy and the V-**

function.
The agent is initialized at a random starting point, and it's goal is to find the cheapest path to the end point at $(14,3)$. There is a 20% chance that the world will contain a rectangle in which the reward is -11 (in the remaining parts of the map the reward is -0.1).

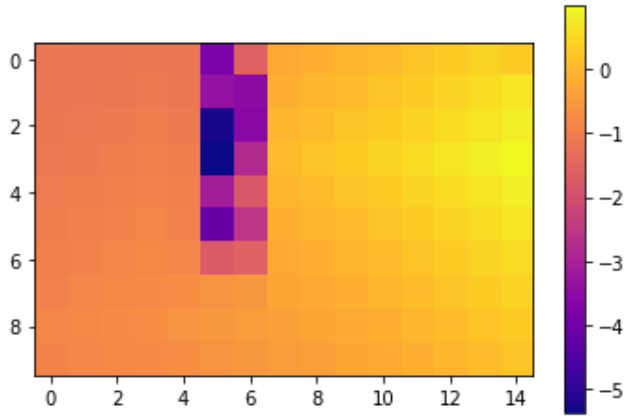
In order to solve the trick by reinforcement learning, the learning rate most be set to a low value. This will make the algorithm put more emphasis on previously learned experience. If the learning rate is set to a large value, the algorithm will "forget" how to properly handle the rectangle since it only appears in 20% of the runs.

We used the following hyperparameters:
$\eta = 0.1$
$\gamma = 0.95$
$\varepsilon = 1$
ε decay = 0.99999
Episodes = 5000

Policy:



Value function:

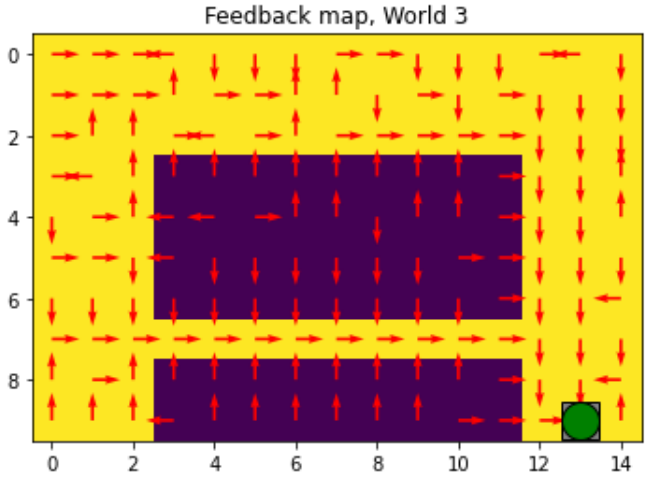


6. **Describe World 3. What is the goal of the reinforcement learning in this world? Is it possible to get a good policy from every state in this world, and if so how? What parameters did you use to solve this world? Plot the policy and the V-function.**
The agent is initialized at starting position $(1,7)$, and the goal is to find the cheapest path to the endpoint at $(13,9)$. The world contains to rectangles in which the reward is -0.5. In the

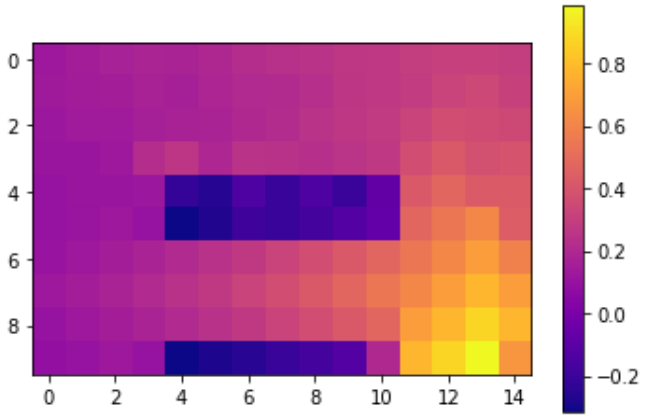
remaining parts of the world the reward is -0.01. The shortest path from the starting position to the end point goes through a narrow passage between the rectangles. To get a good policy from every state, ε must be set to a high value and the decay of ε can not be too fast. By doing this, the agent will put more emphasis in exploring the whole map.

We used the following hyperparameters:
$\eta = 0.8$
$\gamma = 0.9$
$\varepsilon = 1$
ε decay = 0.9999
Episodes = 1000

Policy:



Value function:



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 agent is initialized at starting position $(13,9)$ and the goal is to find the cheapest path to the endpoint at $(1,7)$. The map looks similar to world 3 with two rectangles where the reward is set to -0.5, and a narrow passage between them. However, unlike the previous worlds, there is a 30% chance that a random action will be performed. Because of this, the cheapest and safest path will be the long way around both the rectangles, rather than the

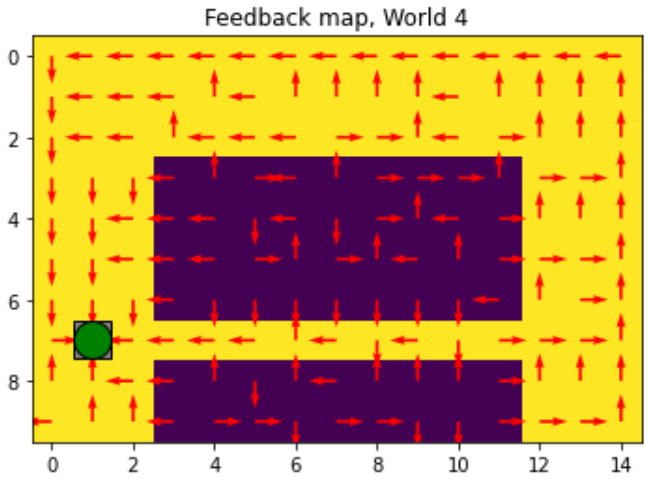
path between the rectangles where one of these random actions can cause the agent to reach some of the -0.5 spots.

In order to solve this problem, we initialized the Q-matrix with zeros instead of random values. We also had to skip the code that sets the Q-value to -inf if an action is not valid. This has to be done because a valid action can become non-valid due to a random action being performed. In such case, the Q-value for that state and valid action will be permanently set to -inf. When too many Q-values are set to -inf, the agent will eventually not be able to move anywhere.

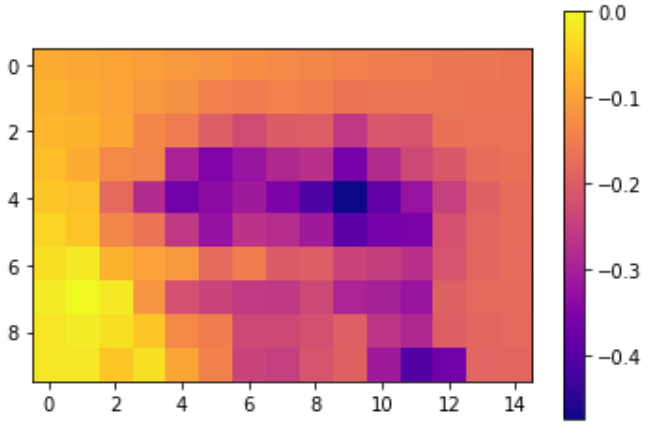
For this world we used the following hyperparameters:
$\eta = 0.1$
$\gamma = 0.95$
$\varepsilon = 0.5$
ε decay = 0.9999
Episodes = 4000

By using the obtained policy, the agent is able to reach the endpoint in about 40-50 steps.

Policy:



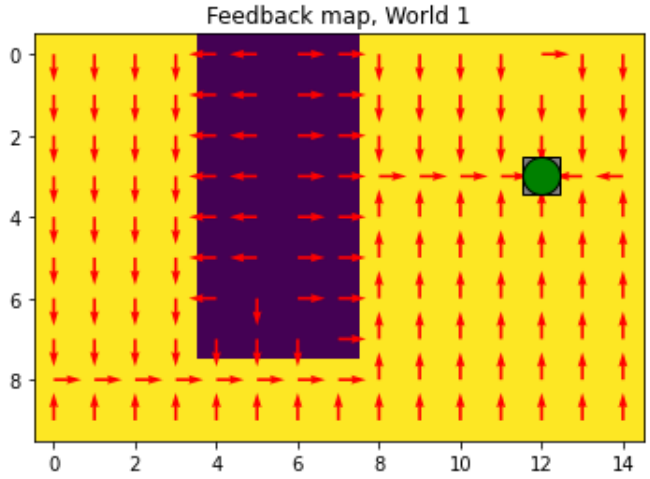
Value function:



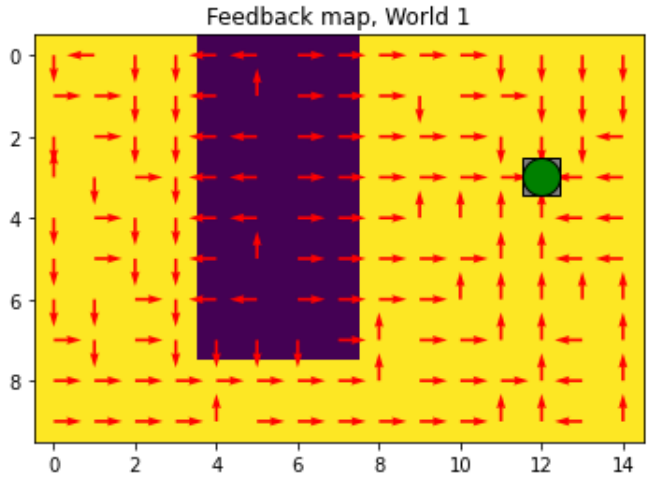
8. **Explain how the learning rate α influences the policy and V-function. Use figures to make your point.**

The learning rate determines the importance of new information. A low learning rate can result in slow convergence of the V values, while a high learning rate may cause oscillation of the V values *since the algorithm will overwrite previous experience with newly acquired information. In deterministic worlds such as world 1 a high learning rate is preferable (it will generate a better policy in a shorter time), while the algorithm will require a lower learning rate in order to find a good policy in non-deterministic worlds such as world 2 and 4 (since the worlds are changing, the algorithm must rely on previous experience to a greater extent).* The following images illustrates how the algorithm converges faster after 1000 episodes when the learning rate is set to 1 than when it is set to 0.4.

Learning rate 1:



Learning rate 0.4:



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

The discount factor determines the importance of future rewards. A low value will make the algorithm prioritize short-term rewards, while a high value will make the algorithm maximize the long-term rewards.

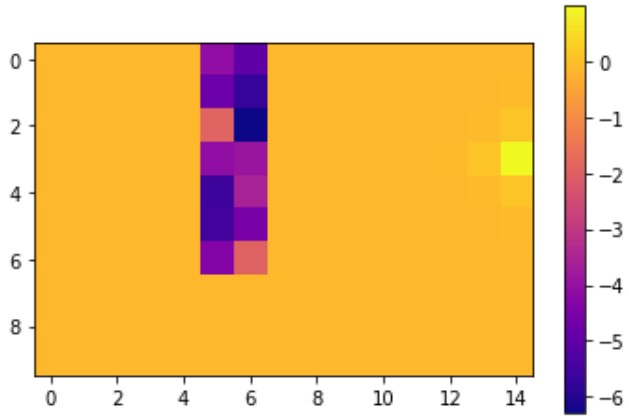
If the discount factor γ is set to a high value, $\max_a \hat{Q}(s_{k+1}, a)$ (i.e., the maximum reward in the next state) will have a bigger impact on the updates of the Q-values and the choices of actions. If instead γ is close to 0, the reward r will determine the new estimate of Q.

$$\eta\left(r + \gamma \max_a \hat{Q}(s_{k+1}, a)\right)$$

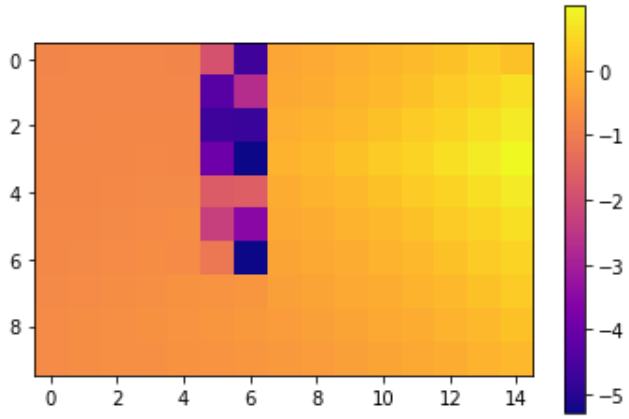
The following two plots illustrates how a higher discount factor makes the algorithm focus more on maximizing long-term rewards.

In the first figure (where $\gamma = 0.2$), all Q-values except for the rectangle, the goal, and the positions just next to the goal are about the same. In the second figure (where $\gamma = 0.9$), the Q-values are more varied, and they gets higher closer to the goal. This is a result of the discount factor making future rewards (i.e., $\max_a \hat{Q}(s_{k+1}, a)$) more important when estimating Q.

Discount factor set to 0.2:



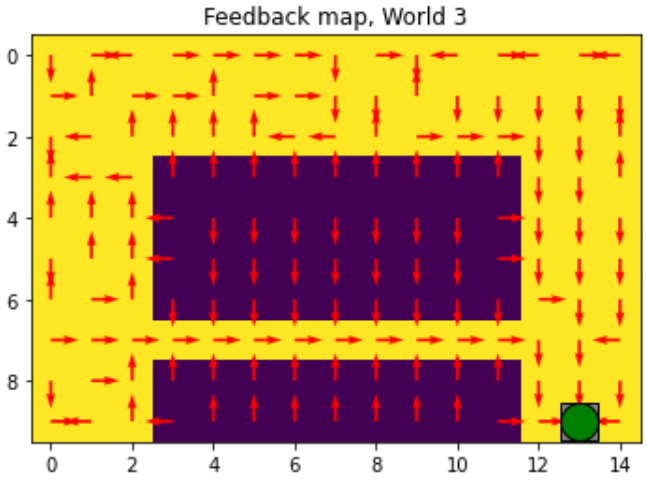
Discount factor set to 0.9:



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?**
A high exploration rate increases the probability of the algorithm performing a random action, and thus a bigger part of the world will be explored. This will increase the run time, but also allow the algorithm to find a good policy for every possible state.

We used a strategy where we decreased the exploration rate at every episode by multiplying the exploration rate by a decay parameter that is less than 1.

The plot below shows the policy when the exploration rate is set to 0. With a low exploration rate, the algorithm will not explore the whole world well, hence the poor policy for the upper part of the world.



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 a static world like “Irritating blob” world Dijkstra’s cheapest path finding algorithm will be able to find the optimal path. Dijkstra’s cheapest path finding algorithm will however not work in a non-static world such as the “Suddenly irritating blob” world, because which the shortest path depends on whether or not the irritating blob is present.

12. **Can you think of any application where reinforcement learning could be of practical use? A hint is to use the Internet.**
    Reinforcement learning can be applied in many different fields. Reinforcement learning is for example used in self-driving cars. Another common application is to assist in trading of stocks and other financial instruments. Reinforcement learning is also used in the program AlphaZero which can learn to play many different games.

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?**