TBMI26 – Computer Assignment Reports Reinforcement Learning

Deadline – March 15 2019

Authors: Anubhav Dikshit <anudi287>
Hector Plata <hecpl268>

In order to pass the assignment, you will need to answer the following questions and upload the document to LISAM. **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)**
   - The Q-learning algorithm is a way to train a system to find the best action to do in a specific situation. For each action taken the system will be rewarded or punished dependent on how good the action was.
   - Q-function: is the expected future reward of doing a specific action 'a' in a given state s_k and is defined as equation(1):
   $$Q(s_k, a) = r(s_k, a) + \gamma V^*(s_{k+1})$$
   - Where r is the reward of doing the specific action in that state, V*(s_k+1) is the reward in the following state if the optimal policy is followed. This value function for the optimal policy is defined as V*(s)=max$_a$ Q(s,a) and describes the reward of doing the best action in a given state. In our example the best action is a step in the direction which brings us closer to the target on the way with the least cost.

2. **Define a learning rule (equation) for the Q-function and describe how it works. (Theory, see lectures/classes)**
   - Learning rule: The Q(s,a) function, is the base of the V(s) function. It is used during training to experiment around the best policies. It is updated according to the following equation(2):
   $$Q(s_k, a_j) \leftarrow (1-\alpha)Q(s_k, a_j) + \alpha(r + \gamma \max_a Q(s_{k+1}, a))$$
   - Where r is the reward, α is the learning rate and gamma is the weight of how much reward it gets from doing an action that will lead to a state that is known to be good. Gamma can be interpreted as a term reward adjustment parameter.

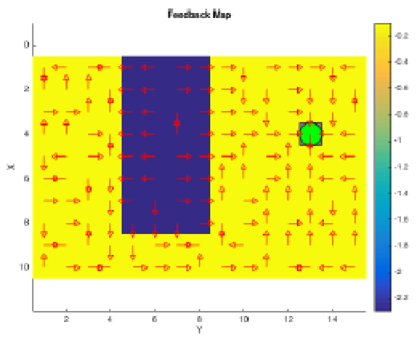
3. **Briefly describe your implementation, especially how you hinder the robot from exiting through the borders of a world.**
   - In our case the Q function is realized as 3-dimensional table with the size of the map (aka world) on the x and y axis, and the corresponding rewards of doing a specified action in the 3rd dimensions. The valid movements are up, down, left and right. The world has size 10*15 which gives the Q-table a size of 10 * 15 * 4. In the study negative feedback where used. A bad move gives high negative feedback and a good move gives low negative feedback. A motivation to use exclusively negative feedback values is to punish ways which don't lead directly to the target but have extra loops. Each time the robot steps onto a field without there being a known high positive rewarded that field is made less attractive.

   - The table was initialized with zeros i.e. all actions are considered as equally good in the beginning. The reward of taking a step outside the world along the boarders was set to -inf. This is to make sure the robot doesn't get stuck at the edges of the map (aka world). Without this measure there is a chance that walking outside is considered as the optimal action. This is the case if during training all the other actions have proven to be poor, but this action has not. If the exploring factor at the same time is low the robot will not take random actions which could get it back into the world after reaching the boarder and it will therefore get stuck trying to walk in a forbidden direction at each iteration. Getting stuck is caused by ignoring illegal actions, which is the principal measure to keep the robot in the world. In this study a feedback of -0.1 was given if walking on the free ground and if walking on an obstacle. An obstacle should not be imagined as a wall but rather a territory in which it is unattractive to move, e.g. sandy ground versus a paved road in the example of a car.

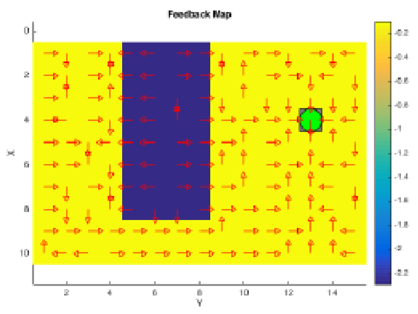
   - In order to make the robot explore unknown territory it does not always follow the optimal policy but does sometimes take a step into a random direction, just to see if this might lead to an even better solution. The probability of the robot taking a random step can be adjusted using the epsilon parameter. The adjustment is done by the user but in some strategies the algorithm does adjust the parameter automatically to have a positive impact on the learning progress and the final performance.

   - The best action is based on the V*(s) function. In our implementation the it is the action with the highest reward value. The probability to take this optimal action is (1 - epsilon) and the probability to take random action is epsilon. The random parameter epsilon should be high in the beginning in order to explore new ways to walk instead of trusting the known data. For each step taken the Q-function is updated according to 2nd equation. Since the action of walking outside the borders are set to infinity this option is never considered as the optimal. There is a chance though, that the random action takes place and the robot therefore wants to move in that direction. In this case the Q-function is simply not updated, and the algorithm goes on. When the robot reaches the goal, the algorithm start from the beginning and the robot is placed at the starting position. The next iteration is running on the updated Q-function.

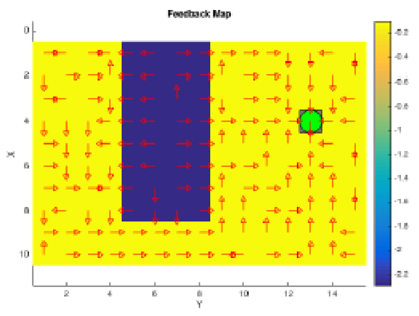
4. **Describe World 1. What is the goal of the reinforcement learning in this world? What parameters did you use to solve this world? Plot the policy and the V-function.**
   - Varying iterations (gamma = 0.9, epsilon initial =0.9, decay = 0.98 and alpha = 0.5)



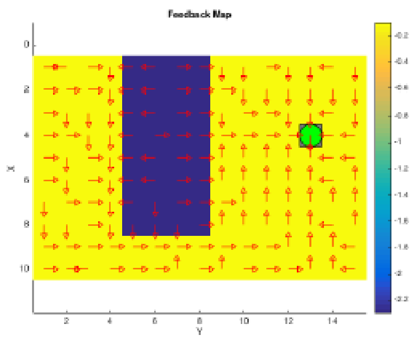
(a) 100 iterations



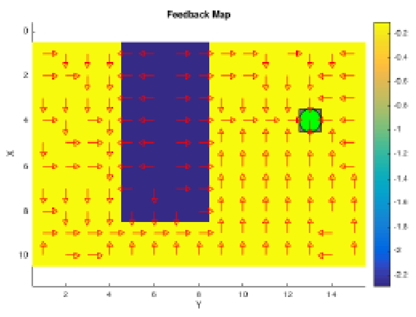
(b) 200 iterations



(c) 300 iterations

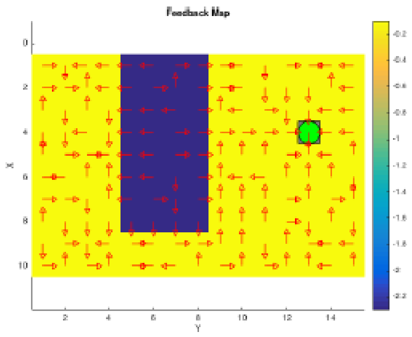


(d) 1000 iterations

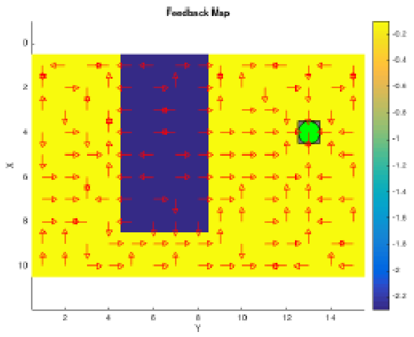


(e) 1000 iterations - exception $\epsilon_{Decay} = 0.999$ used

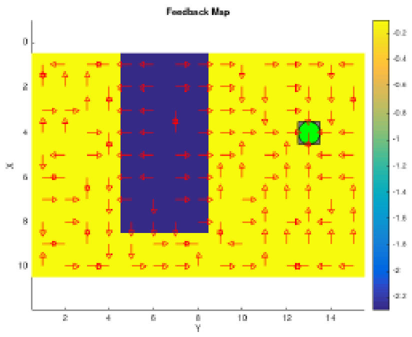
- Varying learning rate (gamma = 0.9, epsilon initial =0.9, decay = 0.98)



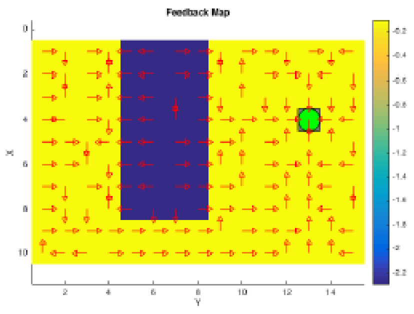
(a) 100 iterations $\alpha = 0.3$



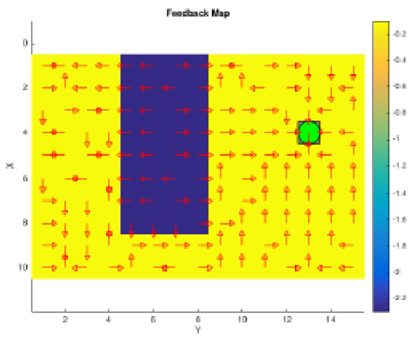
(b) 200 iterations $\alpha = 0.3$



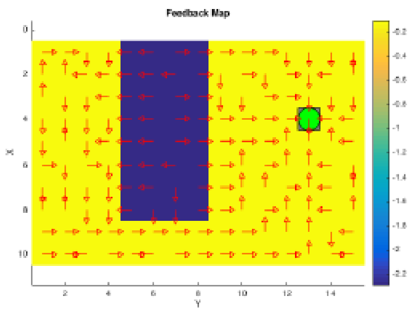
(c) 100 iterations $\alpha = 0.5$



(d) 200 iterations $\alpha = 0.5$



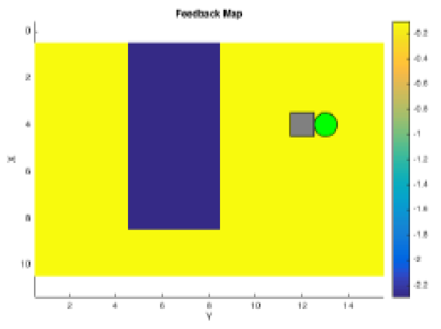
(e) 100 iterations $\alpha = 0.7$



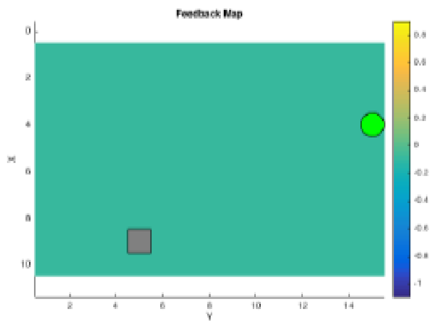
(f) 200 iterations $\alpha = 0.7$

-
- In this world we used random initialization, using the smaller decay factor-faster decay of the random step probability epsilon we get: After 100 iterations the probability of taking a random step is at 0.11, after 200 iterations at 0.015, after 300 iterations at 0.002 and after 1000 iterations at 1.5e-9, so practically at zero. Using the slow decay, the probability after 1000 iterations is still at 0.33. Comparing figure 3f with 2e we see that the slower decay helps to find a better solution (V*(s)). More exploration is performed. This comes at the cost of longer paths to the target executed by the robot since often random actions are taken. The solution becomes better, but the robot does make less use of it. This called the exploration exploitation Dilemma.

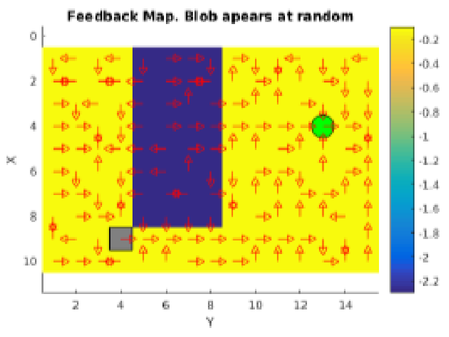
5. **Describe World 2. What is the goal of the reinforcement learning in this world? What parameters did you use to solve this world? Plot the policy and the V-function.**
    - Parameters gamma=0.9, elision initial = 0.9, decay = 0.98, alpha = 0.5.



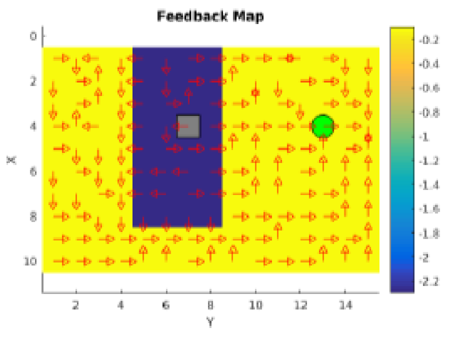
(a) world 2 - With blob



(b) world 2 - Without blob



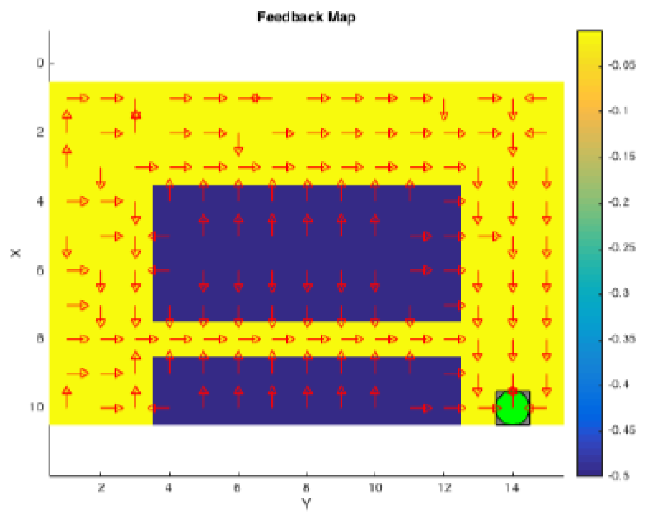
(a) $V(s)$ world 2 when using only 100 iterations to train the system



(b) $V(s)$ world 2 when using 500 iterations to train the system

- The challenge in the world 2 is the change of the world during training. The system must adapt to the change in the environment. The interesting aspect is if and how fast the system can adapt. Above figure shows the worlds that were used for this experiment, where the "flat" world is the most common and the blob appears on random at some iterations.

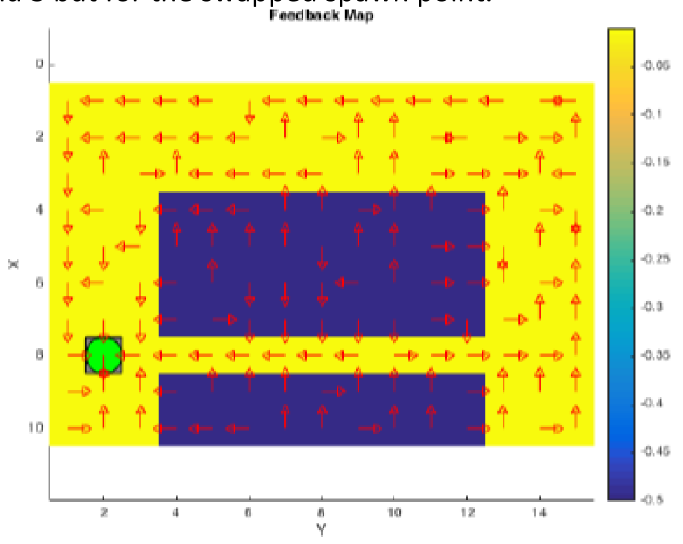
6. **Describe World 3. What is the goal of the reinforcement learning in this world? What parameters did you use to solve this world? Plot the policy and the V-function.**
   - World 3 and 4 share the same topology only the start and end positions are swapped.



world 3 - 100 iterations

   - Parameters gamma=0.9, elision initial = 0.9, decay = 0.999, alpha = 0.5.

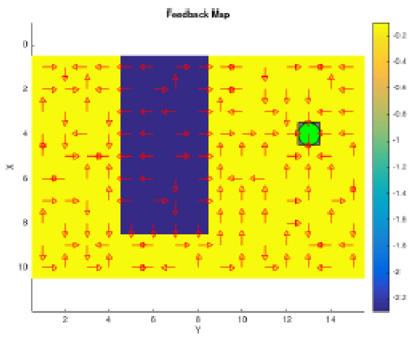
7. **Describe World 4. What is the goal of the reinforcement learning in this world? How is this world different from world 3, and why can this be solved using reinforcement learning? What parameters did you use to solve this world? Plot the policy and the V-function.**
   - Same as world 3 but for the swapped spawn point.



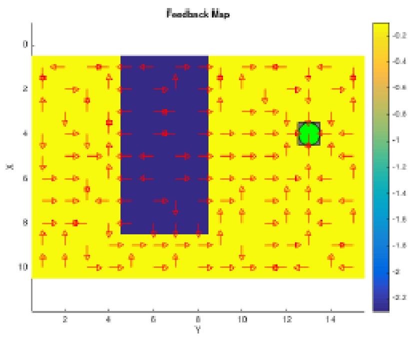
world 4 - 2000 iterations

- In world 4 the robot takes uncontrolled random steps at random intervals which simulate the alcohol intoxication after a HG visit, the consequence of this is that it hard to find a solution which makes the robot go the obviously shortest way. Even after 2000 iterations the shortest way is not found. The reason for this is that the robot will learn that it is dangerous to walk inside the tunnel because of the risk to slip out to the blue area. Hence the safest way is to walk around the tunnel instead.

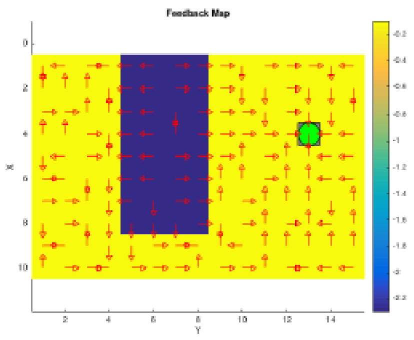
8. **Explain how the learning rate α influences the policy and V-function in each world. Use figures to make your point.**



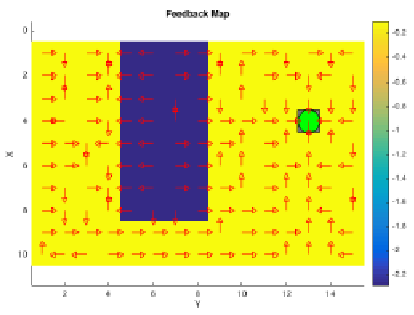
(a) 100 iterations $\alpha = 0.3$



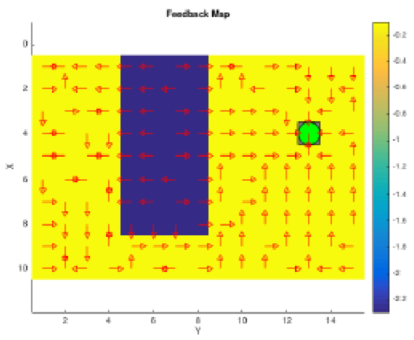
(b) 200 iterations $\alpha = 0.3$



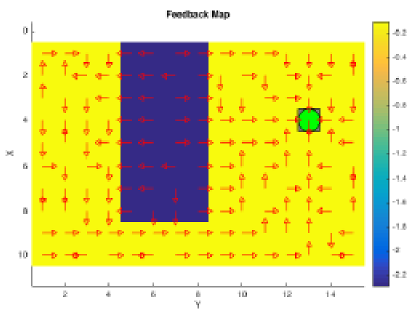
(c) 100 iterations $\alpha = 0.5$



(d) 200 iterations $\alpha = 0.5$



(e) 100 iterations $\alpha = 0.7$

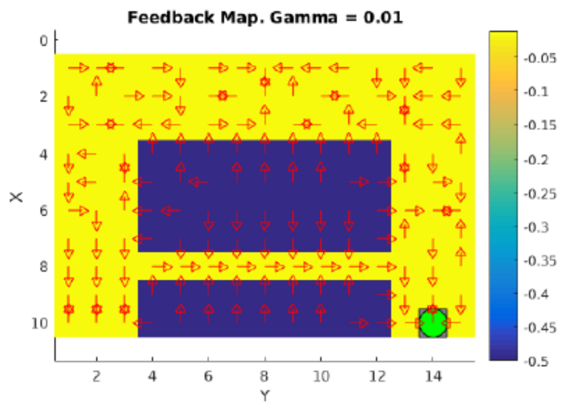


(f) 200 iterations $\alpha = 0.7$

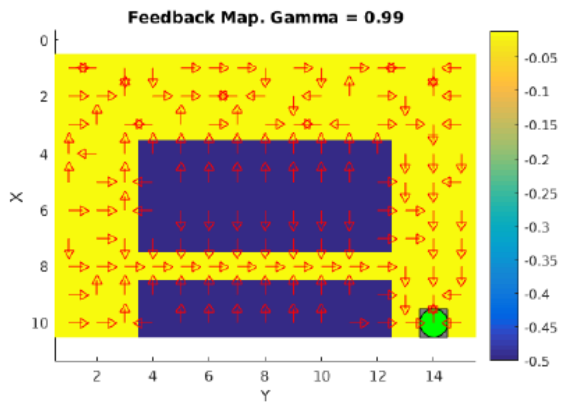
-
- The influence of the learning rate can have been studied on the example of world. The robot finds a shortest way in all cases. The patterns are a bit different though. A high alpha will lead to a lot of parallel patterns since the previous good move are overwritten by a new one if the exploring pattern at the same time is high this phenomenon will be increased. On the other hand,

if alpha is kept low the robot will rely on earlier calculated costs and will rather walk a good old pattern than trying a new one.

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



(a) $V(s)$ for world 3 $\gamma = 0.01$



(b) $V(s)$ for world 3 $\gamma = 0.99$

- From the above we can see that it's hard for the robot to enter the tunnel from the starting position, when gamma = 0.01, that is because it doesn't get much reward from doing that now. Once it is in the tunnel it walks straight through because that seems to be good. Since the robot is long-term rewarded, it will easily walk in to the tunnel as it believes it is the reasonable in the long term although even though it might not be the best option now.

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

- We know that it is beneficial to not shift too early from exploration to exploitation. Although doing so can in the short-term result in the executed path of the robot improving more quickly, but the solution does improve much over time. Rather than the initial values the decay is an important parameter. For the given problem a decay in the range of 0.98-0.999 has been used whereas the lower value seams too low and the higher values rather high. The higher the complexity of the problem is the higher the decay value should be chosen to allow a long phase of exploration.

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 case of using Dijkstra's cheapest path finding algorithm instead of reinforcement learning, the look up table should be initializied by the developer of the model. Because of this, the shortest path is going to be encoded on the values of the look up table, this means that the robot is not learning anything by itself, rather, it’s “learning” the optimal path given by the initialized Q.

- As for the sudden irritation blob world, there are two possibilities. The first one is creating a look up table for each world and using each one depending on the current environment. The other one, is creating only one look up table for the most recurrent world, this would yield a sub-optimal route, since it’s going to use the same path, independently of the world.
- Like before, the agent wouldn’t learn anything, since the best path is already encoded on the table.

12. **Can you think of any application where reinforcement learning could be of practical use? A hint is to use the Internet.**
    - Loan application processing is an ideal place for reinforcement learning. While traditional machine learning is applied but the models are trained and predict at very time specific intervals eg: when applicant first applies even before we collect more data about the person. The whole journey of loan processing tends to take time and further information might be a determinantal to the approval. This is an ideal place where reinforcement learning can shine, tracking applicants even after the loan disbursable and tuning itself to identify the applicant who tend to default much later in the lending cycle.

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?**