TBMI26 – Computer Assignment Reports

Reinforcement Learning

Deadline – March 15 2020

Authors: Karolin Stork and Viktor Gustafsson

Q1. Define the V- and Q-function given an optimal policy. Use equations and describe what they represent. (See lectures/classes)

$$V^*(s) = \max_a Q(s,a) \qquad Q(s_k, a) = r(s_k, a) + \gamma V^*(s_{k+1})$$

The optimal policy describes the best action to take in each state. The V-function returns the reward for the action that gives the highest reward. The Q function for a certain state and action is the reward in the current state taking an action (ie the feedback from the state the robot travels to) + discount factor*the best V-function for the next state.

Q2. Define a learning rule (equation) for the Q-function and describe how it works. (Theory, see lectures/classes)

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

Fig 1. Updated Q for a specific state and action= (1-learning rate)*Q(old_x,old_y,action) + learning rate*(feedback + discount factor*V(new_x, new_y)

The new Q-value for a given position and action is calculated using learning rate, Q-value for the given position and action, feedback for the new state, discount factor and the V-value for the next state given a specific action.

Q3. Briefly describe your implementation, especially how you hinder the robot from exiting through the borders of a world.

We check if the new state is in an invalid position and then we update Q(state, action) as -inf for the previous state for the action that resulted in the invalid state. However in world 4, due to the randomness factor between chosen action and actual action we have chosen to not punish an invalid action to avoid learning errors.

Q4. 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.

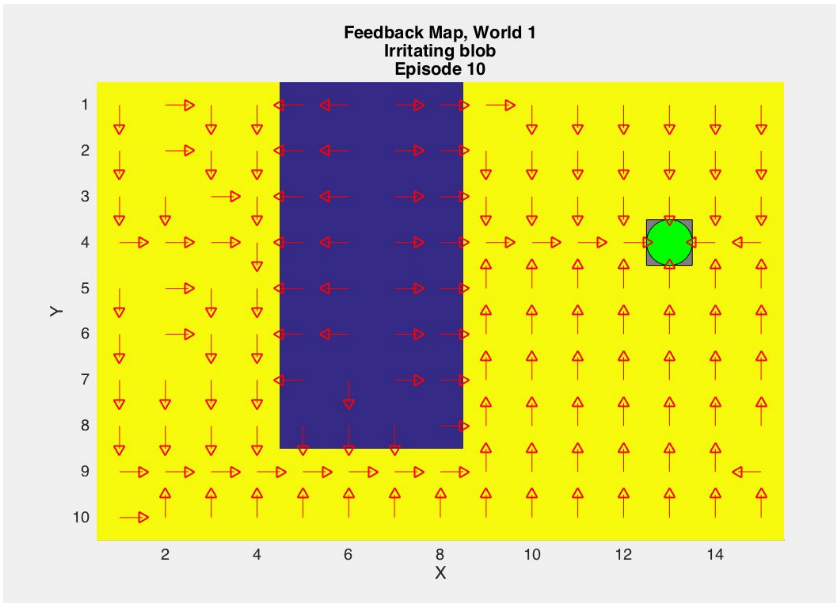


Fig 2. The policy of world 1.

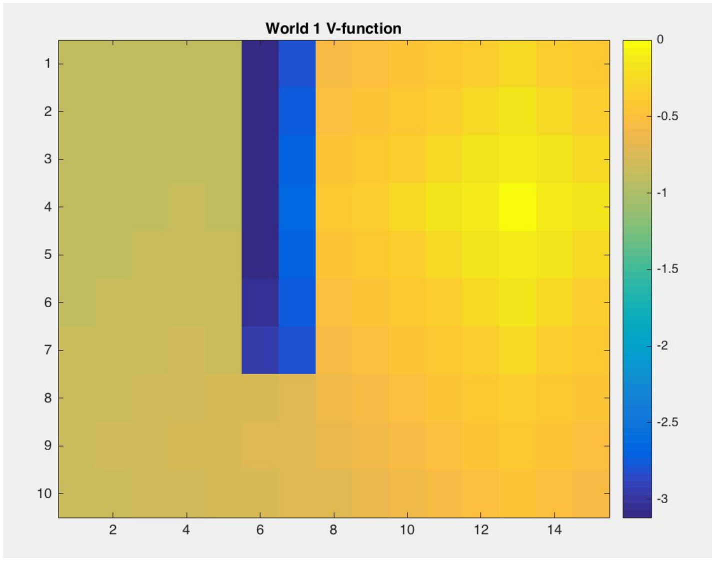


Fig 2. The V-function of world 1.

Episodes: 1000
Learning rate = 0.6
Discount = 0.9

The goal of this world to find the 'cheapest' way to HG (green circle) wherever the robot spawns. All states had a small negative feedback except for the dark blue area which had a large negative feedback. With the parameter settings described above 10 test runs were solved with an average of 11.7 steps.

Q5. 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.

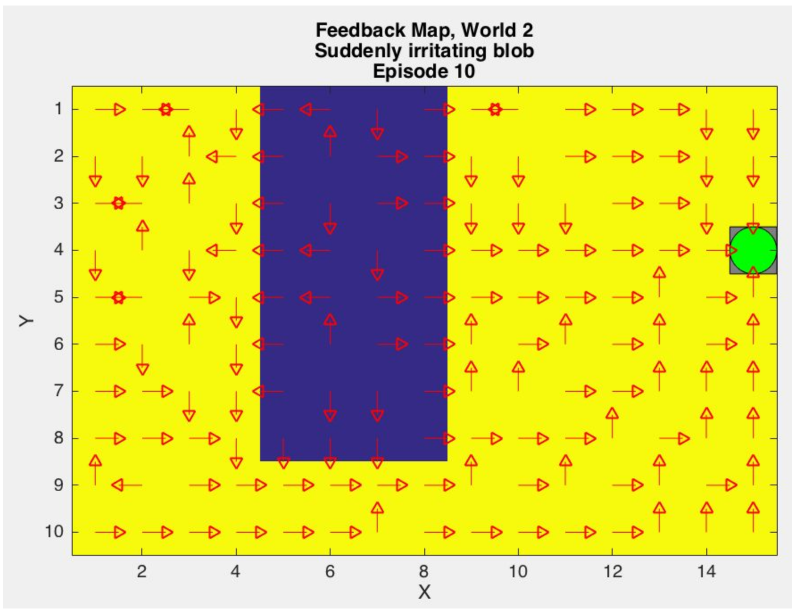


Fig 3. The policy of world 2.

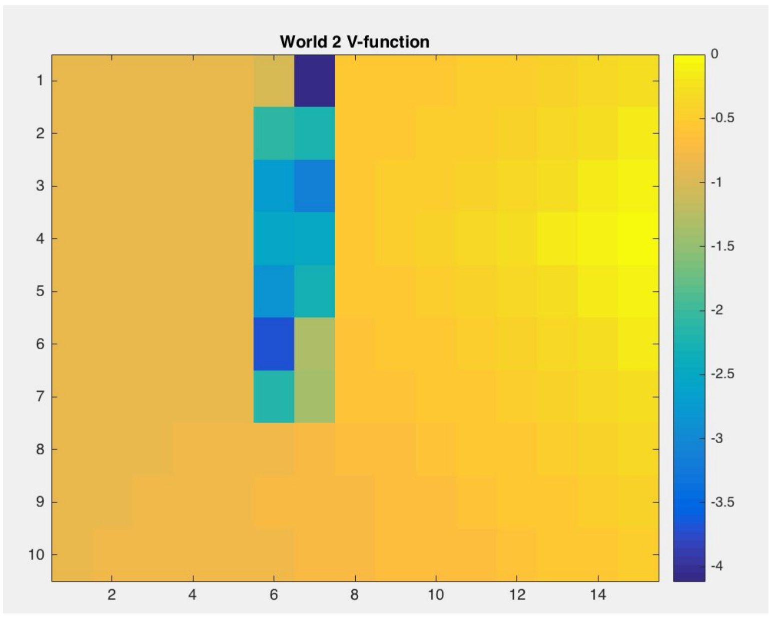


Fig 4. The V-function of world 2.

World 2 is similar to world 1 in the sense that the robot spawns randomly and the goal is to find the 'cheapest' way to HG. However, when traveling through the states the blue area can suddenly appear/disappear at the upper half of the world. This can be seen in the differences between fig 2 and fig 4 where the V-function is much more negative and prominent around that area in fig 2. To be able to handle this it is important that we have a 'longer memory' to avoid this potentially large negative feedback. A longer memory is represented by a lower learning rate.

Episodes = 1000
Learning rate = 0.1
Discount factor = 0.9

With these parameters the problem was solved with an average of 12.1 steps for 10 test runs.

Q6. 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.

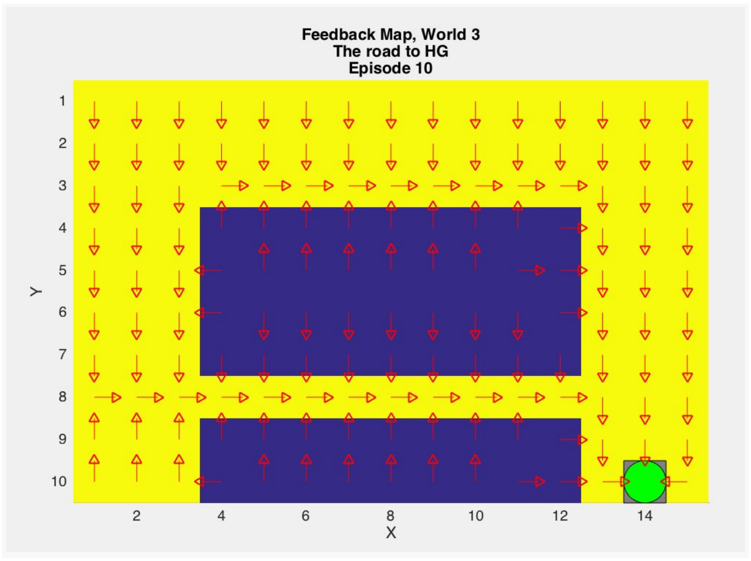


Fig 5. The policy of world 3.

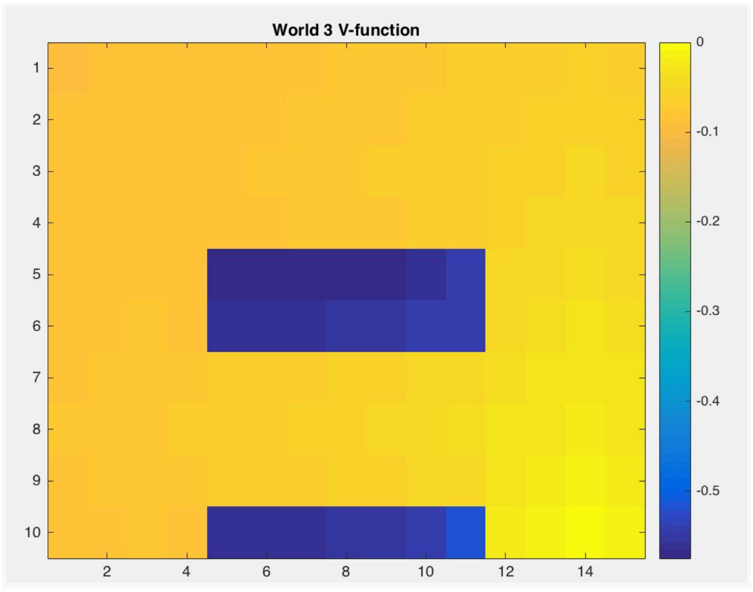


Fig 6. The V-function of world 3.

Episodes= 1000
Learning rate = 0.9
Discount = 0.9

The goal of the reinforcement in this world was to take the shortest way possible to HG but avoid the large blue field as this punished the robot with more negative feedback. The agent always spawns at the left hand bottom corner and it's objective is to traverse the world towards the right hand bottom corner without moving through the blue rectangles. As there is no randomness in spawning location, suddenly irritating blob etc. we could solve this with a short memory ie high learning rate. With the parameter settings above the problem was solved with an average of 14 steps for 10 test runs. This is the lowest number of steps possible as both start and end position are fixed. We could probably lower the number of training episodes to achieve the same result. To achieve a good policy for every state in this world we would have to adjust the epsilon to allow for more exploration than exploitation.

Q7. 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.

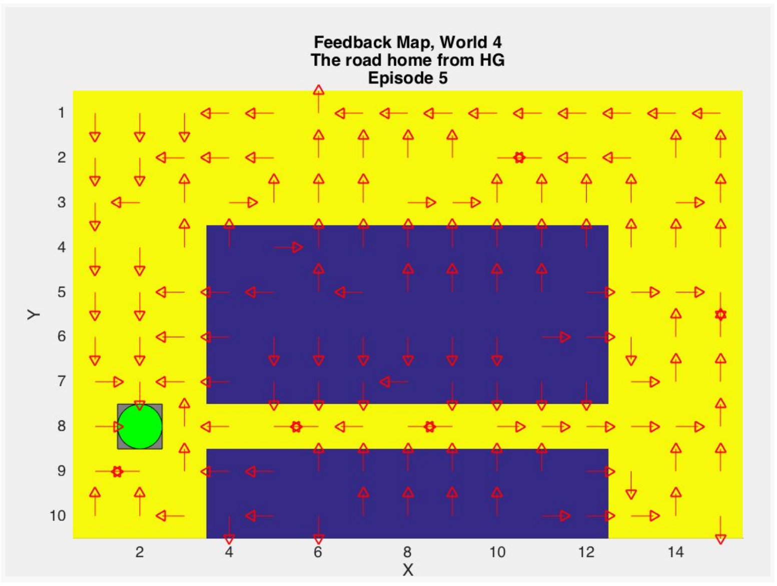


Fig 7. The policy of world 4.

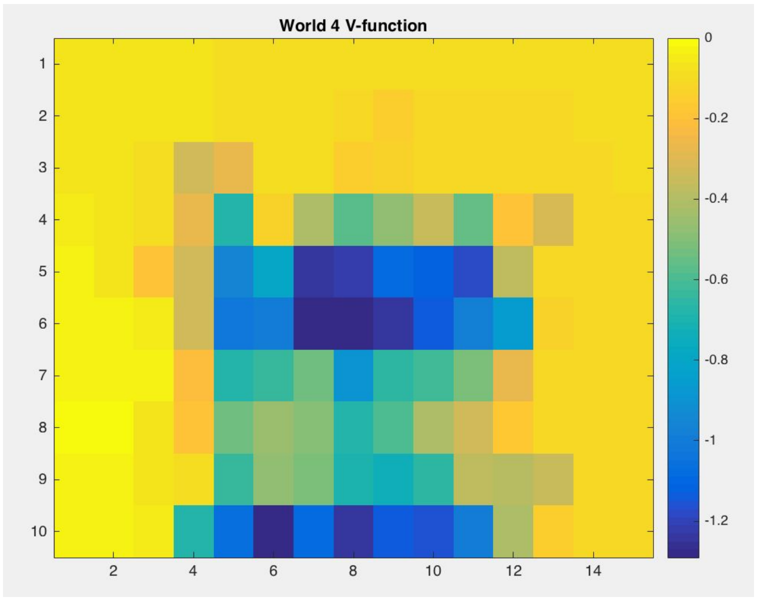


Fig 8. The V-function of world 4.

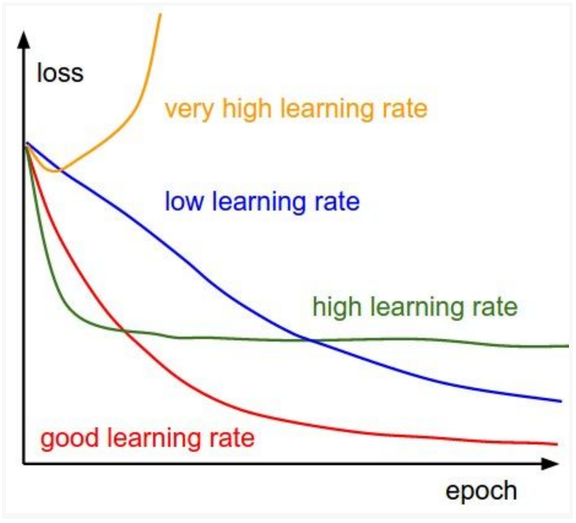
Episodes = 4000
Learning rate= 0.3
Discount factor = 0.9

Actions in world 4 seems to have a stochastic element to them, in that new states seems to be randomized with some likelihood p, and the correct new state from action with the likelihood 1-p.

In this world we had to remove the punishment for taking an action that results in an invalid state. This is because, if randomness picked the next state the action taken will be permanently removed from the optimal policy even though the action could be optimal. For instance, say that the agent stands next to the bottom wall, the agent picks action 'up', but randomness kicks in and the agent moves down resulting in an invalid state. Here the action 'up' would previously get punished by -inf and removed from the policy permanently. For 5 test runs the problem was solved with an average of 86.6 steps with the parameter settings described above. As can be seen in both fig 7 and fig 8, the road between the two blue areas is now avoided due to the risk of the randomness causing the robot to step to the sides.

Q8. Explain how the learning rate influences the policy and V-function. Use figures to make your point.
Learning rate will influence how much the Q values are updated for a new observed Q-value. This will influence how fast changes in the optimal policy is spread throughout the Q-matrix.

A low learning rate is good for historical robustness which can be important when noise/randomness is included in the agent's world. A low learning rate decreases the convergence rate, however. A high learning rate will increase the convergence rate initially, but a too high rate will make the performance oscillate around local minimas.



Q9. Explain how the discount factor influences the policy and V-function. Use figures to make your point.
The discount factor influences how much the values of future states will influence the policy. A large discount factor will result in a policy that priorities rewards further ahead, while a smaller discount will favour rewards in a shorter spatial area. Therefore a discount factor close to 1 will make the behaviour of the model more 'long-term' and vice-versa for 0.

Q10. Explain how the exploration rate influences the policy and V-function. Use figures to make your point. Did you use any strategy for changing during training?
Exploration rate is the chance that a random action is picked instead of the optimal one. This is beneficial when the agent has not yet explored the world fully. Therefor, a high exploration rate is important initially. To get a good performance, an eventual decrease and 0 value of the exploration rate is also important.

We therefore set the exploration rate as (maxEpisodes-episode)/maxEpisodes to achieve this behaviour. A polynomial or square root in the denominator could be used to influence the behaviour of the decrease in exploration rate further.

Q11. 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?
Dijkstra's will work well and will find the shortest distance to the goal in a deterministic world such as the irritating blob world. If we however use dijkstra's in a world where each the states are not static, dijkstra's will most probably not converge to an optimal policy. Consider that Dijkstra's has a found an optimal path when the irritating blob is not present. This means that dijkstra's policy will traverse the potential blue area at all times, as Dijkstra's scans for optimal policy once. This might not be optimal. Therefor, the q-learning approach is better in the suddenly irritating blob world, as the q-learning learns the optimal policy over time and thus approximates the expected value of traversing the randomized blue area.

In short, reinforcement learning is better at adjusting for randomness in state and action as the Q function in a way approximates the expected reward, as historical rewards are discounted with the learning rate.

Q12. Can you think of any application where reinforcement learning could be of practical use? A hint is to use the Internet.
An application is to train a model that is able to beat a game. Such systems have been made both for simpler Atari games, as well as more complex modern games such as starcraft. The more complex the game, the more complex the model and it's training, but the underlying methodology is still usually based in reinforcement learning. A common method for this is deep-q learning. The most famous and one of the best performing models is google's deepmind which applies deep-q learning.

Another application is in finance, where trading bots can be built using reinforcement learning. This fits the problem well, as different states result in different actions in trading. The action depends on many different factors that makes up the state, such as current price, p/e, derivative, positive/negative news and other curves that correlate. The combination of all these states should result in different actions (buy/sell/hold) that in the end maximise the reward-function (difference price bought and sold).

Q13. (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?

World 5: No dark blue area, random spawning.
World 6: No dark blue area, random spawning, two goals.
World 7: Bigger world, multiple dark blue areas, random spawning.
World 8: Yellow square that is a portal to a state close to goal.
World 9: Dark blue grid, orange square, random spawning.
World 10: White solid areas, random spawning.
World 11: Almost everything is dark blue except around spawning area.
World 12: Combination of white, dark blue and orange areas.

World	# test runs	Average # steps to solve problem	# episodes	Learning rate	Discount factor
5	10	8.5	4000	0.3	0.9

6	10	4.4	4000	0.3	0.9
7	10	24.9	4000	0.3	0.9
8	10	7.7	4000	0.3	0.9
9	10	13.9	4000	0.3	0.9
10	10	11.5	4000	0.3	0.9
11	10	-	-	-	-
12	10	-	-	-	-

The algorithm worked well for all worlds except 11 and 12. For world 11, where almost all states were punishing for the robot, the robot stayed close to its spawning area. For both world 11 and 12 we were unable to find a stable convergence, initially the number of steps used to solve the problem declined but after a few hundred episodes it diverged. We tried to raise the discount factor to 1 because the robot in world 11 had to walk through a lot of negative feedback to arrive at the goal. However, this didn't seem sufficient for solving the problem. Due to the fact that no randomness was present in the worlds the learning rate shouldn't affect the result significantly.