TBMI26 – Computer Assignment Reports  
Reinforcement Learning

Deadline – Mars 12 2018

Author/-s:

Albin Bergström albbe673

Lovisa Hassler lovha997

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. 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 optimal policy (See lectures/classes).**

V: Value function of being in a state, if we follow the optimal policy.

Q: The expected future reward of doing an action in a state, and then following the optimal policy.

1. **Define a learning rule for the Q-function (Theory, see lectures/classes).**

Q(state, action) =

(1 – alpha) \* Q(state, action)

+ alpha \* (reward + gamma \* Qmax)

alpha = learning rate. A value close to zero puts more emphasis on the learned experience, while a value close to 1 will prioritize new information.

gamma = discount factor, a value close to zero will maximize the short term rewards while a value close to 1 will prioritize long term rewards.

reward = the reward given for going to a specific state. Can be both positive and negative.

Q = the expected future reward of doing an action in a state, and then following the optimal policy.

Qmax = the max reward for the next state (that can be reached from current state, taking one action following optimal policy).

1. **Describe your implementation, especially how you hinder the robot from exiting through the borders of a world.**

The robot is given a positive reward for reaching the goal and a weighted negative reward for each step on yellow/blue ground. The action of trying to exit through a border is given an infinite negative reward.

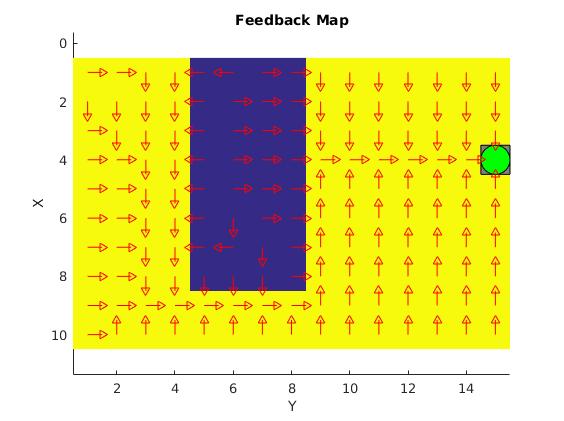
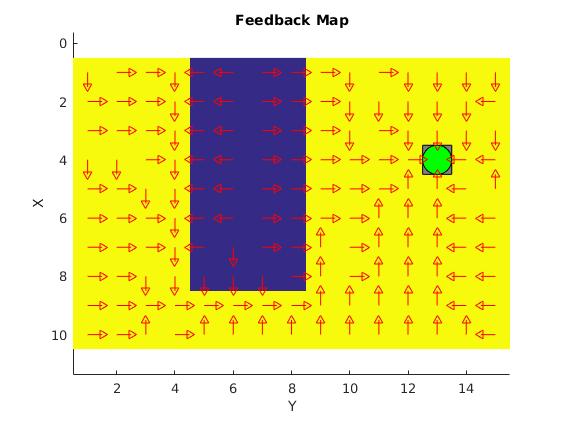
During the training the robot will go from using random exploration with a very high probability to using almost only greedy exploitation at the end of the training.

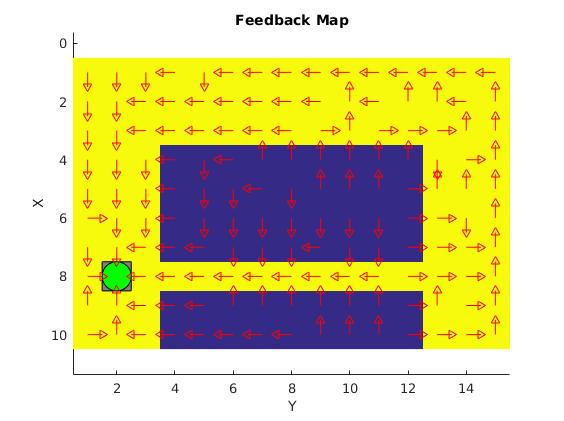
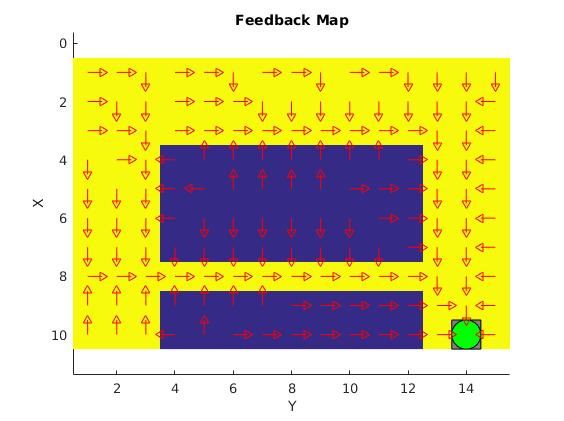
1. **Describe the differences between the worlds explored by the robot. Any surprises?**

The “Suddenly irritating blob” was a little bit confusing at first, but made us change our code for the better. The same regarding “the road home from HG”. At first we created a matrix with the rewards in the beginning of the program, but this was not a feasible solution when the conditions of the world changed during training.

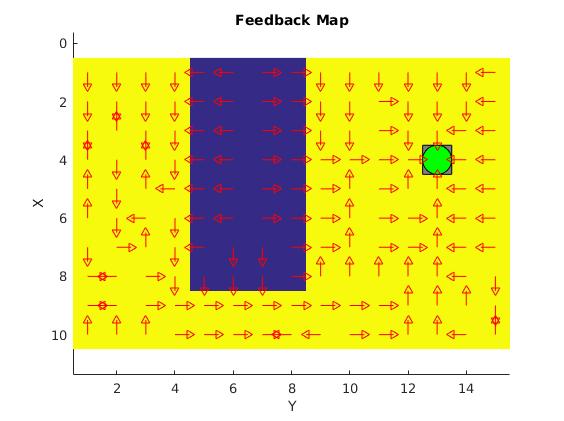
1. **For each world: Plot the V-function, i.e. how do you get to the goal from each position.**

World 1 & 2 (1 000 & 10 000 epochs respectively)

World 3 & 4 (1 000 & 10 000 epochs respectively)

All plots have gamma = 0.7 and alpha = 0.1.

1. **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 in order to explain what you have done and learned during the assignment.**



We tried setting gamma to 0.1 on world 1 instead of 0.7, and an observation that can be done is that there is a lot more “conflicts” in the final policy where two states point to each other (see figure to the left).

World 2 and 4 required more epochs (10 000 instead of 1 000) than world 1 and 3 to generate a good policy, due to their unpredictability.

Another observation that was made was that the amount of random steps taken during the training did not affect the end policy to any higher degree, at least not compared to changing the number of epochs, which more or less determines the quality of the end policy.

The amount of random actions introduced during the training was highly correlated with the execution time of the training. A probability 0.1 of making a random move in each state resulted in a execution time of 9.052322 seconds for 1 000 epochs. Probability 0.9 gave 28.704674 seconds and a probability that started at 1 and decreased over time gave an execution time of 15.075605 seconds. It was hard to notice any quality difference between the generated policies, which was strange. One would think that a very random learning algorithm would yield a worse result.

1. **What would happen if we where to only use Dijkstra's shortest path finding algorithm in the ''Suddenly Irritating blob'' world? What about in the static ''Irritating blob'' world?**

If we were to run the Dijkstra's algorithm only once to create an optimal policy we would run into problems in the “Suddenly Irritating blob” world, since it changes from time to time. If we on the other hand were to run Dijkstra's algorithm every time the environment changes it would give an optimal path every time.

1. **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?**

The robot is a little drunk (or maybe just tired) and has problems walking straight. The narrow passage between the two blue areas is therefore a risky path and is not used. The robot instead chooses the wider path at the top of the map.

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

Alpha Go is an example where reinforced learning has been used to train a program to be better than the best human Go players, without supervision from someone who can play the game. A more practical example is also the industrial robot manufacturer Fanuc that uses reinforcement learning to train their manufacturing robots, with no human interaction. They will improve themselves until they perform warehouse tasks with great precision and speed.

1. **How does the different parameters () influence learning and appearance of the Q- and V-functions?**

η = alpha (in our code) = learning rate. A value close to zero puts more emphasis on the learned experience, while a value close to 1 will prioritize new information.

γ = discount factor, a value close to zero will maximize the short term rewards while a value close to 1 will prioritize long term rewards.