Reinforcement Learning
ACML Assigment 3

Lavinia Pulcinella (i6233926) Enrique Barrueco (i6242336)

November 2020

1 Mountain Car Problem Learning

Since the primary objective of the present assignment was to draw a heat-map of the state-value function we decided to rely on the Q-learning algorithm. The choice was mainly driven by its simplicity and by the fact that it is based on the construction of a Q table. That is, a table containing the Q values (state-action values) for all state action pairs.

Q-learning is a Model-free off policy RL algorithms. That is, it learns the optimal policy values as long as there is enough exploration. It estimates the state-action value function from a target policy that deterministically selects the action of highest value. The algorithm also converges independently of what order the interactions are processed. Thus, the interactions can be collected and remembered so that they can be processed again.

Since Q learning is a model free method, it is based on the interactions with the environment to find the best strategy and thus we need to explore the entire state space while figuring out the best actions. RL algorithms consist in a gradual change between training and making predictions thus they need enough exploration (exploring many states and actions in the environment in order to find more information) to have good performance guarantees.

Thus the naive approach would be to ensure the optimal action at any given time by choosing the action which the agent expects to provide the optimal rewards. By randomizing this approach i.e. by letting the agent choose what it believes to be the optimal action "most of the time" but occasionally acting randomly leads the agent to gather potentially new information about the environment.

Thus, we also introduced the ϵ parameter to perform an ϵ-greedy algorithm. The high level idea is that by letting the algorithm run without any randomization, once the car reaches its target position (the flag) it will likely "remember" the steps taken to reach the target and will stick to those set of actions. By introducing randomization, we let the car try out different combinations of actions and in this way learn a little more about the surrounding environment as well as learn "new" ways to reach the target.

2 Dealing with continuous space

The state state space is continuous, the highest for position is 0.6, the lowest is -1.2, for velocity the highest is 0.07, and the lowest is -0.07.

We decided to discretize the continuous state space by dividing it in bins. The default number in our code is 30 bins, since it led to a rather smooth visualization (but not too smooth). The number of bins is actually a "parameter" that could be tweaked. Of course, the higher number of bins, the more the number of possible combinations increases.

3 Visualizations

Q Learning Analysis In order to try understand how our model was performing and try to find the fastest and most consistent way to solve the challenge we tweaked the two hyper-parameters present in our implementation of Q learning, the discount factor γ and the learning rate α. We measure how good an initialization of the model is, by how fast it completes the challenge. In particular, we use the total reward of each episode.

The min and the max rewards variables correspond to the best and the worst performing model in batches of 100 episodes, that way we keep track of runs where the objective was not reached, this is reflected in the minimum reward value being equal to -200. Table 1 includes the results of running the algorithm with different hyper-parameters, it is sorted by the mean of the average rewards.

	Alpha	Gamma	Min	Max	Means
10	0.15	0.95	-191.67	-152.50	-172.3310
9	0.10	0.95	-194.51	-150.33	-173.2037
11	0.20	0.95	-195.31	-154.08	-173.4822
2	0.15	0.85	-197.85	-158.10	-177.3874
4	0.05	0.90	-196.09	-152.27	-177.6343
5	0.10	0.90	-198.05	-154.32	-179.1530
7	0.20	0.90	-198.49	-159.42	-179.4176
8	0.05	0.95	-199.71	-156.57	-179.6560
0	0.05	0.85	-199.22	-160.83	-181.4017
1	0.10	0.85	-198.32	-164.11	-183.1325
6	0.15	0.90	-199.63	-159.45	-183.6557
3	0.20	0.85	-199.62	-162.78	-186.7522

Table 1: Reward statistics when running the model using different hyperparameters

Given that the best performing value of the discount factor gamma is 0.95 we use it to visualize the scores of 4 runs using different learning rates.

In Figure 1 we can see how the algorithm learns to complete the challenge fast as it goes through the 10000 episodes, however, it does not stick to these learned Q values that lead to faster, more optimal solutions. Instead it often unlearns these successful patterns. Lower values of α seem to be less susceptible to this, as the average reward seems to follow a trend it fluctuates over without being as volatile as the average reward values of the higher α. By looking at this graphs it is possible to save and load the Q values that perform the best and using those in the final model.

Heatmap The goal of this assignment was to plot the value function corresponding to the chosen algorithm.

By performing Q learning we are - as it is part of the algorithm itself - building a 2-dimensional Q table including all the state-action values and, the highest Q value correspond to the value function.

Thus we plotted a heatmap of our Q table according to the corresponding rewards. It is evident how for extreme values of the states (i.e. position and velocity) the rewards are pretty low and, once the car reaches a certain position and velocity the rewards start to increase.

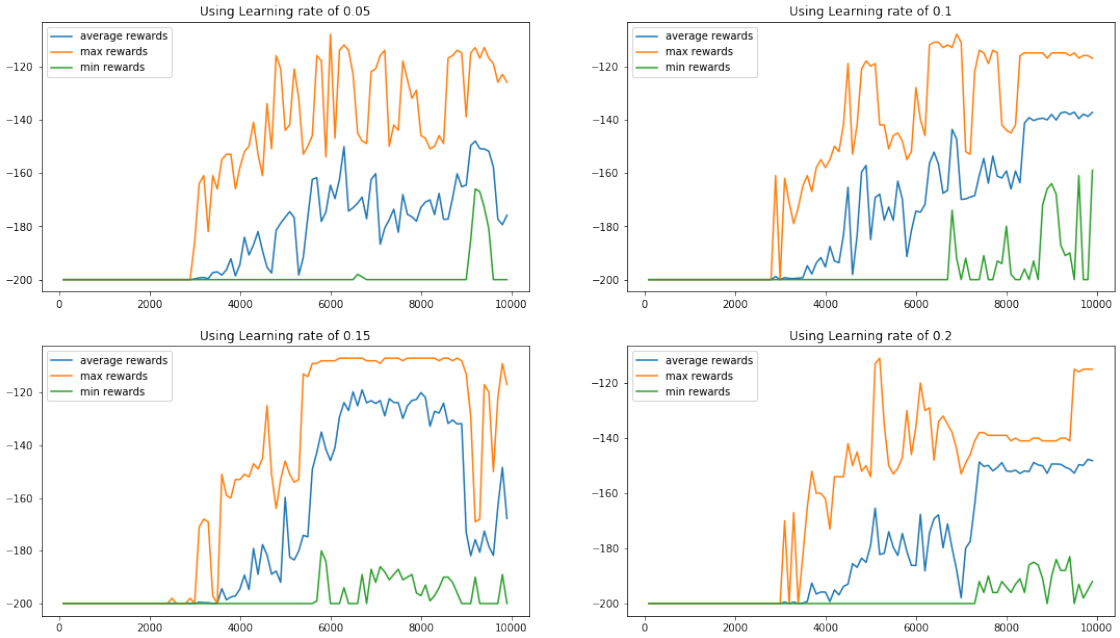

Figure 1: Rewards through 10000 episodes with different learning rates.

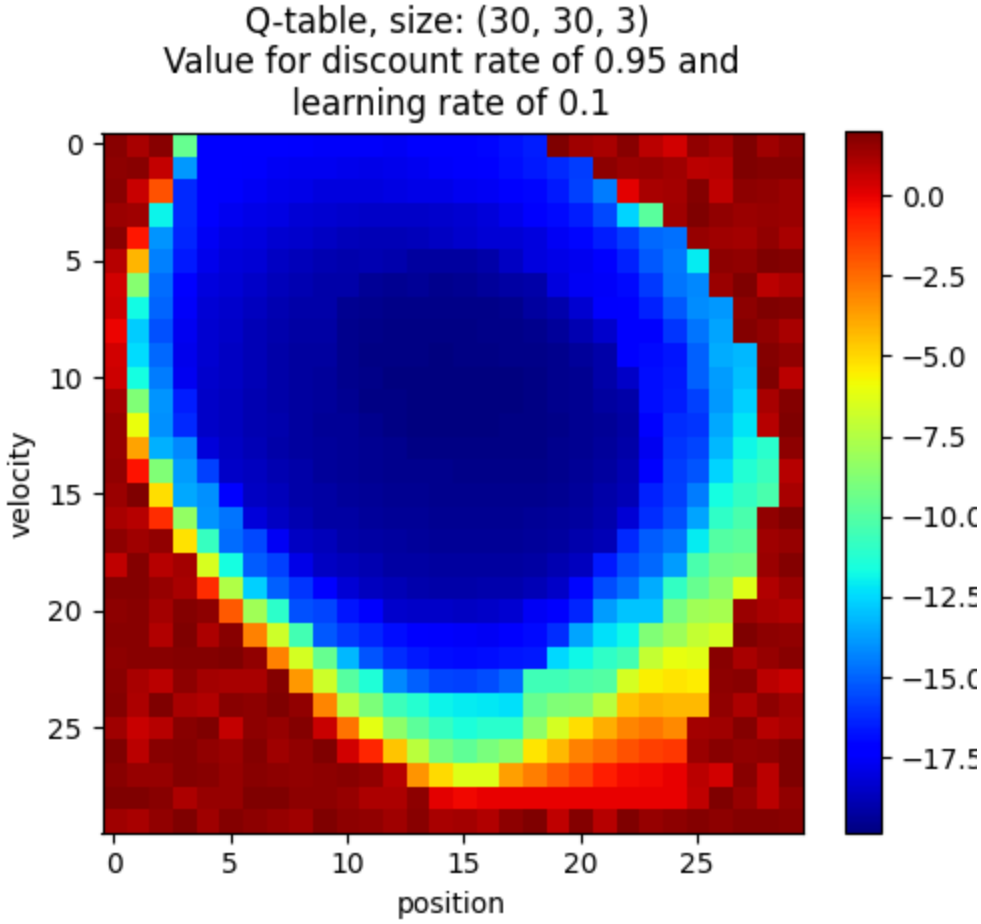

Figure 2: Caption