Reinforcement Learning - Week 3
Q-Learning

Antonio Chiappetta

October 2019

1 Cartpole

1.1 Task 1

The following images show the outcome generated by the application of Q-learning for the *Cartpole* environment, first using a **GLIE** and then a **epsilon greedy** policy. Both simulations were run with a duration of 20000 episodes.

Figure 1: Q-learning with GLIE reaching $\epsilon = 0.1$ after 20000 episodes

1.2 Task 2

The following heatmaps show the optimal value function generated with GLIE in function of x and θ, averaging on $\dot{x}$ and $\dot{\theta}$. The heatmap was first generated before the start, then after 1 episode, at halfway (10k episodes) and after the end of the 20k episodes.

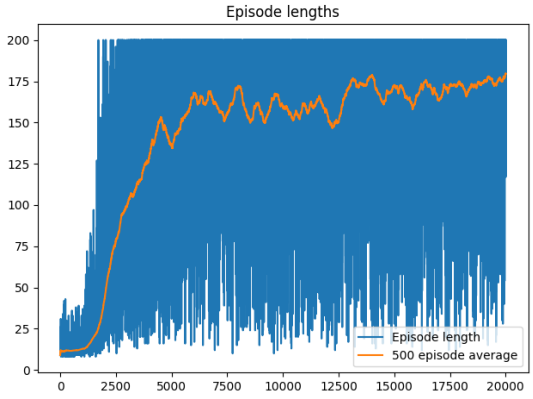

Figure 2: Q-learning with epsilon greedy with $\epsilon = 0.2$ after 20000 episodes

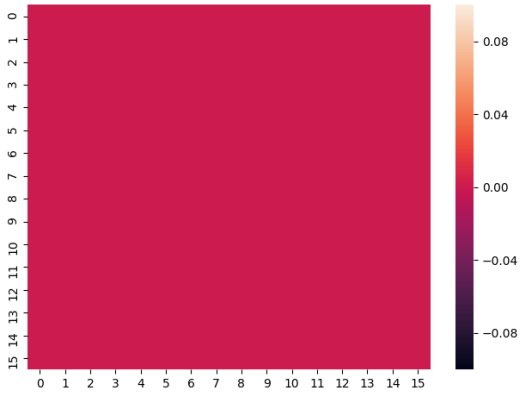

Figure 3: Optimal value function heatmap after 0 episodes

1.3 Question 1

The heatmap starts from being completely uniform and with values all set to 0 before the training.

After the first episode, the few states experienced start having a non-zero value, while the rest of the grid is still quite undefined.

Halfway during the training process states start to look like the ones in the graph following the end of training: states closer in terms of position and velocity to the 0 (center of the grid) have a high value compared to the others all around. In particular, the rectangular shape of the states with high value show how the decay in terms of performance when getting away from the center is higher with velocity (vertical axis) than with position (horizontal axis).

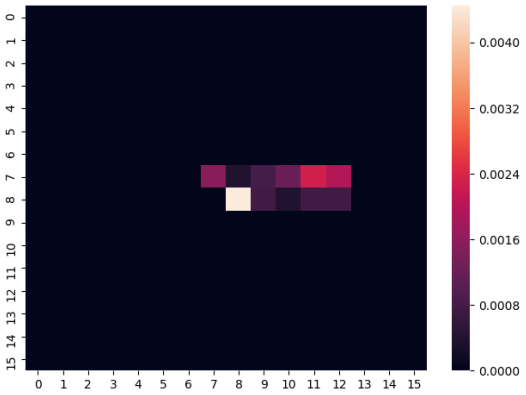

Figure 4: Optimal value function heatmap after 1 episode

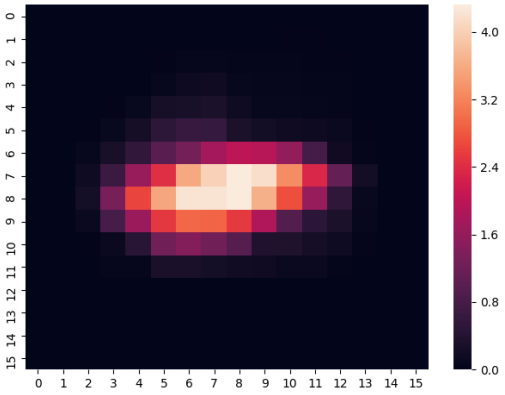

Figure 5: Optimal value function heatmap after 10000 episodes

1.4 Task 3

Using a **greedy** policy the results change to the following. The two images show an execution starting from Q values set to 0, and another starting from 50.

1.5 Question 2

1.5.1 Better model

The model performs way better setting the initial estimate to 50 for all the state-action pairs.

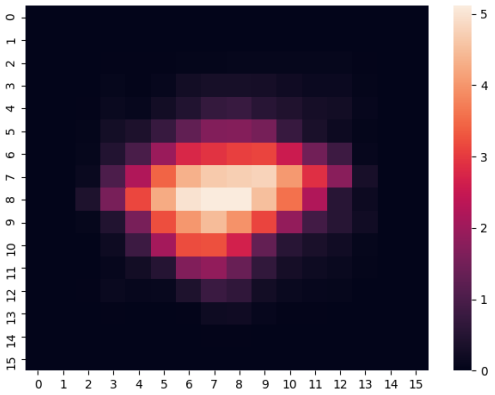

Figure 6: Optimal value function heatmap after 20000 episodes

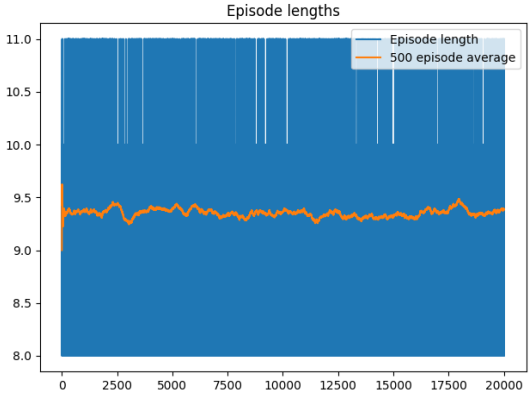

Figure 7: Q-learning with greedy policy with Q values estimates starting from 0

1.5.2 Why

Starting from high values, the policy looks at least once at each state because the rewards it gets will always lower the values of the ones it already visited.

1.5.3 Q values affecting exploration

This increased possibility of exploration makes the agent easily discover the most convenient state-action pairs and use them quickly. We can see from the second figure that episodes start lasting almost 200 timesteps already after 1000 episodes, and around 7500 episodes this value starts stabilizing for the 500 episodes moving average.

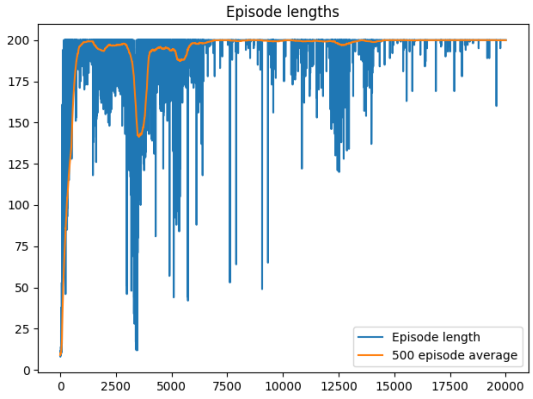

Figure 8: Q-learning with greedy policy with Q values estimates starting from 50

2 Lunar lander

2.1 Task 4

The following images show the application of GLIE and epsilon greedy policies to the Lunar Lander environment, with the same settings as in the previous exercise with the Cartpole environment.

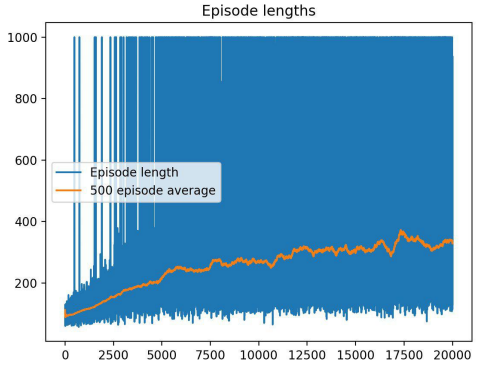

Figure 9: Q-learning episode lengths with GLIE policy with Q values estimates starting from 0

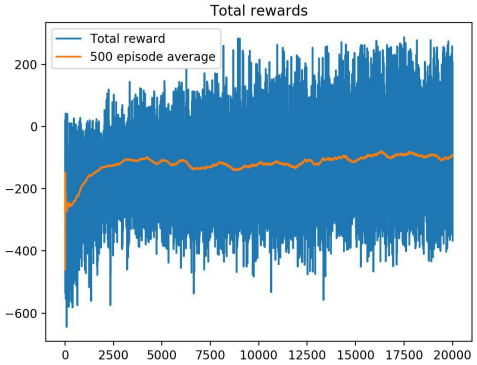

Figure 10: Q-learning rewards with GLIE policy with Q values estimates starting from 0

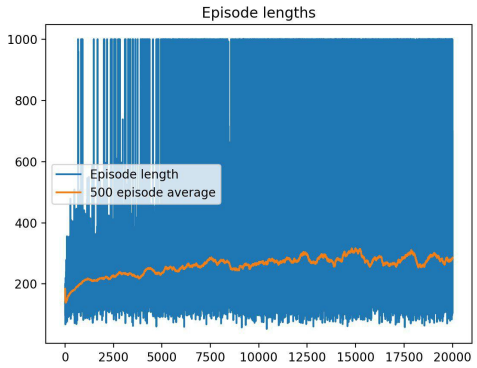

Figure 11: Q-learning episode lengths with epsilon greedy policy with Q values estimates starting from 0

2.2 Question 3

2.2.1 Learn useful behaviour

The lunar lander is not able to learn a useful behaviour with none of the two tested configurations.

2.2.2 Why/why not

The discretized states are not precise enough to model the experience of the lunar lander. This environment is more complicated than the cartpole one, thus this kind of discretization results to be too simplistic.

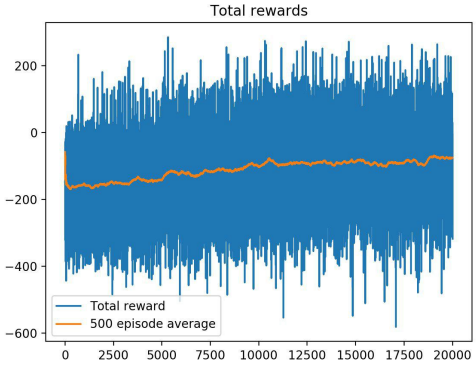

Figure 12: Q-learning rewards with epsilon greedy policy with Q values estimates starting from 0

Since the real state space is continuous, it could be positive to use **function approximation**, e.g. with an artificial neural network, in order to speed up learning and better generalize to unseen states [1].

References

[1] R. S. Sutton and G. Barto. *Reinforcement Learning*. The MIT Press, 2015.