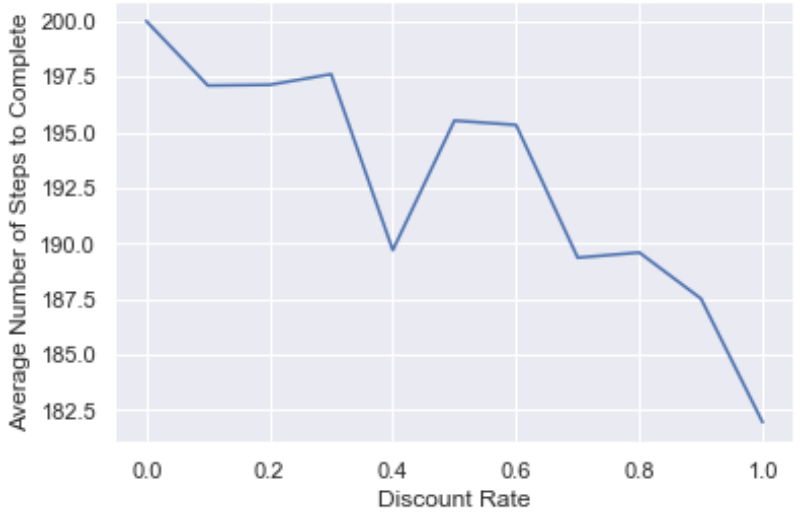
Dogan Aykas
6227725

ACML Lab 3: Reinforcement Learning

For the reinforcement learning algorithm, I have chosen Q-learning in non-deterministic environments formulation. The advantage of this method is that the agent does not need to know the model of the environment that are the transition and reward functions, but it rather calculated the corresponding Q-values for the action given states and iterating through their values. The algorithm for Q-learning is rather straightforward and is relatively easy to implement in the coding. After the implementation, I experimented for the three parameters that are present in the q-learning function, which are: episodes, learning rate and discount rate. The experiments can also be verified from the script provided, since the random seed is fixed in the code.

For the continuous state space that was present in the question, I discretized the two-dimensional state values that consisted of its position and velocity. Since the position range is between [-1.2, 0.6], I have created 19 different states that are discretized by intervals of 0.1 starting from -1.2 to 0.6. Similarly, since the velocity range is between [-0.07, 0.07], I have created 15 different vector states discretized by intervals of 0.01 starting from -0.07 to 0.07. Thus, for the q-values calculated in the code, an array with the size of 19*15*3, with the indices corresponding to each position, velocity and actions.

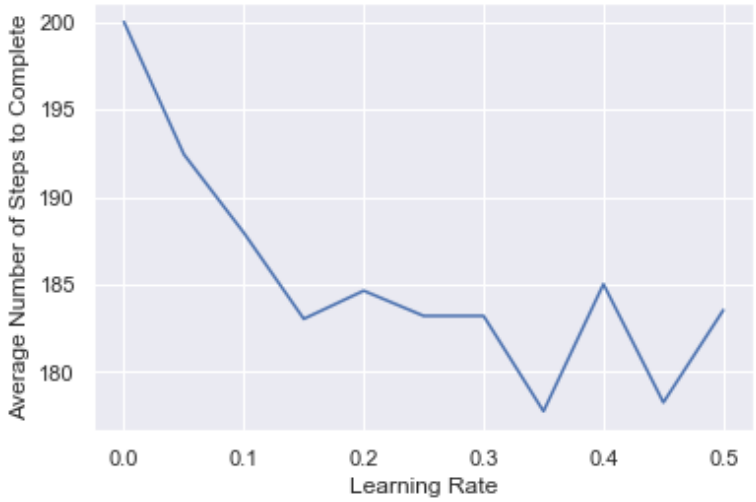
It was difficult to get results at first since the agent was not able to improve its skill reaching to the top. Later, I discovered that the value taken for the discount rate was too low (around 0.2) which was causing the agent to focus more on immediate rewards and rather discarding the future rewards. After a few adjustments, taking the discount rate closer to 1 (0.9), the agent was able to complete the episodes without reaching its maximum moves per episode which is 200. Experiments with different discount rates are depicted below, which are done with 2500 episodes. This many number of episodes is selected to get a faster (and trained to some extent) result than 5000 which seems to be a natural threshold for the training of this mode:



The discount rates of 0, 0.2, 0.4, 0.6, 0.8 and 1 are and their corresponding average steps to complete can be observed. Taking the weight decay value as something close to 1 generally gave

good results according to these observations, thus the value for weight decay is 0.99 in the initial training present in the code.

A similar approach was taken to experiment with the learning rate hyperparameter. For the range of possible learning rates, a narrower range of (0, 0.5) with 0.05 steps of increments are tested and generated results depicted below. Number of episodes for this experiment is also fixed at 2500. Usually, taking the learning parameter something between 0.2 and 0.35 gave the best results in different experiments. For this one, best performance happened when learning rate is taken as 0.35.



Besides discount rate and learning rate, number of episodes is the last hyperparameter to experiment with. As it makes intuitive sense, the agent was able to learn more when the number of episodes is increased. For the experiments, three different number of episodes are used to observe which is a good number for the accuracy and performance. There is a significant increase in the model's performance when the episodes are increased to 500 from 5000. Going from 5000 to 10000 also had a positive impact in terms of average steps being reduced. Creating more than 10000 episodes did not improve the performance, however in the heat map training with 10000 episodes are used for best performance. A table containing the performance for these 3 options, with the optimal values for the remaining hyperparameters from previous experiments are represented below.

	Number of iterations		
	500	5000	10000
Average steps	197.696	179.817	160.774

The discretization of the state space also helped with the visualization of the value function that is the heat map. For each state pair, maximum q value for each action given state is chosen as the value function. A representation of heat function for each state is given in the following page. The calculations related to all the graphs and table can be found in the code.

Dogan Aykas
6227725