Principles of Online Decision-Making (CS-303)

Problem Set 8 - Solutions

Problem 1

In this homework, we keep working with the taxi environment of homeworks 6 and 7. You will implement Q-learning, and observe what happens when the environment, the actions and the rewards are modified.

(a) We want to modify the environment such that the passenger does not appear on the first step, but instead after a random number of steps (as before, they appear on one of the three free squares).
What is the new sate space? How many states does it contain?

The old state space was (taxi positions) x (passenger positions) x (destinations). The new state space is the same, expect that (passenger positions) now has a new element ("nowhere") in addition to the five it already had (the four squares and "in taxi"). The new number of states is 600 instead of 500.

(b) We also add a new action: `wait` (in addition to the 4 movements, pickup and dropoff).
Implement the changes of state and action spaces (adding the `wait` action and the states without passenger) in `taxi_env_modified.py`. The changes we made from the original code are marked by `# PODMnew` and the parts you have to fill are marked by `# PODMexercise`. In both cases we left the original code commented out on the lines above for reference.

See the code.

(c) Run the Monte-Carlo algorithm of homework 7 using this modified environment. For this, you need to replace `taxi.py` by your `taxi_env_modified.py` in the gymnasium package you installed (in the folder `gymnasium/envs/toy_text/`; you need to rename your file `taxi.py`. The folder where the gymnsium is installed can be checked by running `pip show gymnasium`).
How do the modifications affect the policy learned?

The taxi tends to go roughly to go to a specific place and either wait or go back-and-forth at this location until the passenger arrives. Then it gets the passenger and delivers it as with the original setting.

(d) Complete the lines marked by `# PODMexercise` in `taxi_qlearning.py` to complete Q-learning (slide 28 of week 11th), and run the algorithm.

See the code.

(e) When the passenger has not appeared yet, the taxi still tends to move instead of waiting. How can you modify the reward to make it use the `wait` action instead?

In order to push the taxi to wait, we need the reward to be higher for waiting than for moving. For this, we can just increase the reward of `wait` to be greater than -1. However, setting the reward too high can prevent the taxi from moving altogether (for instance a positive reward would intuitively lead the taxi to wait as much as possible before getting the passenger).

(f) Implement the above change in `gymnasium/envs/toy_text/taxi.py` by modifying the function `_build_dry_transitions()` and run it for different reward values. For which rewards does the taxi behave as expected?

See the code.

(g) The default discount factor in the code is 0.95. What do you observe if you keep fixed the rewards you used for the last question and you decrease the discount factor? (You can try values $0.8, 0.7, 0.6$ for instance.) Why does this happen?

When we decrease the discount factor in a setting where the reward is higher for waiting than for moving, we end up learning to not move at all.
This happens because the positive reward $+20$ that we obtain for delivering the passenger is in the future, and thus does not compensate the reward of not moving which we obtain instantly.