Lab 3
Discrete Model-free methods: Monte-Carlo, Q-learning and Sarsa

October, 2025

1 Goal of the Lab

In this lab you will implement different model-free algorithms for the `RussellGrid` environment. Being model-free, we will not be able to access to the property `self.P` that represent the probability transitions (so I made private P in this new implementation of the Grid). Later you will try your algorithms in the Taxi environment implemented in Gym.

2 Programming Tasks:

In the `lab3.zip` file you will find several files for this project. In particular, you will have to complete the implementations of the files `Monte-Carlo.py`, `Q-learning.py` and `Sarsa.py`. All files follow the same structure:

- First, the code initialize the table of Q-values for each pair of state-action and define the hyper-parameters α (learning-rate, lr) and ϵ (epsilon for exploration).
- Then, the code continues entering into the learning loop that is done for 10.000 episodes. **There you will find the `TODO` section that you have to complete** with the help of the theory slides that explain them.
- Finally, the code visualizes the policy generated, later visualize 10 episodes starting from random positions, and finally it tests the policy learned printing the average in 1000 episodes *without exploration*.

You have to complete, for the three files, the `TODO` part in the inner loop of the learning algorithm. You can find in the slides the codes of the different algorithms. Remember that file `Grid.py` implements in the Gymnasium API the environment methods (`reset`, `step`, `render`, etc.).

3 Experimenting and Reasoning Tasks:

You should observe the performance of the different algorithms and answer the following questions. You should deliver latter your answers in a pdf file.

1. Do all algorithms learn the optimal policy? If you run the algorithms several times, you may obtain different results. This is typical of RL because there is a lot of stochasticity. Do you find more variance of policies in one algorithm? Which one, if it is the case?
2. If the algorithms don't return the optimal policy in some runs, are the policies still reasonable? Why?
3. We are exploring with a epsilon value relatively high (0.2, see line number 18) in all algorithms. Repeat the experiments with different fixed values of epsilon, and also with epsilon starting from 0.2 and decreasing with the number of experiences to 0.01. What do you observe for the different algorithms?

4. In the experiments, the value of `lr` is set to 0.01. This variable represents the α parameter and could be too large. Try to reduce it to 0.001 and 0.0001 and explain what do you observe (and why) during learning and in the final policy obtained.

5. Can you see differences in performance of the algorithms while learning and after learning in testing? Why?

6. Check if the policy learn and V values are approximately the ones obtained by `policy iteration` in the last lab session. Which is the maximum difference you see?

7. Let's make the differences between algorithms more clear. Uncomment line 252 of `Grid.py` to remove stochasticity in the environment, and uncomment line 82 to penalize even more falling in the bad state. Now the environment is deterministic. Run again the algorithms Sarsa and Q-learning and compare the results. Can you see differences in the policies? Can you see differences in the performance of both algorithm during learning *and* in test time? Which is better in each case? Can you explain these differences? (Hint: one method is *on-policy* while the other is *off-policy*)

8. Run Monte-Carlo also in the deterministic case (like in the previous point). It is more like Sarsa or Monte-Carlo? Do you think Monte-Carlo is *on-policy* or *off-policy*? Why?

9. Q-learning is *off-policy*, that means, it learns one policy (the greedy one!) while following another policy (the ϵ-greedy in this case) that generates the data. But notice that Q-learning can learn the optimal policy following any *exploratory* policy, even if it is very bad. To prove that, uncomment line 20 of the file `Q-learning.py` setting epsilon to 1. This way, the policy used to collect the data is the random policy. Run the program again. As expected, the reward collected *during learning* is very bad and does not improve. But wait, How is the policy learned? What happens with the reward collected without exploration? Why?

4 Use your algorithms in other environments (Optional)

Copy your Q-learning algorithm to another file and try to adapt it to work with the Taxi environment implemented in Gym. You will see that you only need to change few things (and remove some lines of the code that are specific to the Grid problem, like showing a map of the policy).

5 Deliver you solution

Create a zip file with the solution code for each algorithm for the Grid file, together with a pdf file answering the questions asked in section 3. If you have also implemented the solution for the Taxi environment, put them also in the zip file.

Finally, deliver the zip file in the *racó* in the "Practicals" section before the deadline specified there.