T10 - Artificial Intelligence

T-AIA-902

Taxi Driver

Reinforcement Learning

1.3.3

Taxi Driver

delivery method:	Github
repository name:	$CourseCode-$GroupName.git
language:	Python or R or anything else that gets the job done

- The totality of your source files, except all useless files (binary, temp files, obj files,...), must be included in your delivery.

You are asked to solve the `Taxi-v3` game, in which the environment can be found in the *Gym* library.

The agent is a taxi that must pick up a passenger present in a random location on the map, and drop them off in one of four possible locations.

You must solve this problem with an **optimized** model-free episodic learning algorithm.

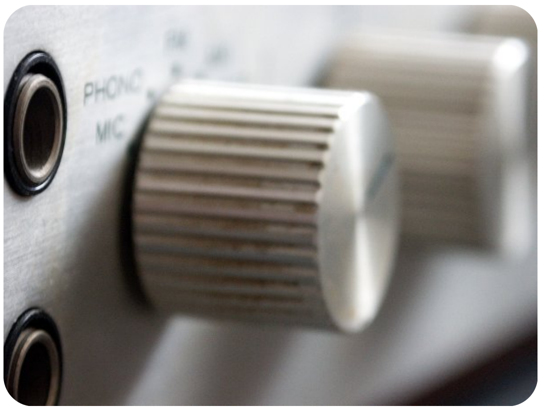
You are free to choose the algorithm, whether on-policy or off-policy.
We recommand Deep Q-Learning, or Monte Carlo-based algorithms, but feel free to find the best possible approach.

If this project is not challenging enough for you, you can create an extension of the game in which the taxi collects 2 people and obtain 4 locations for each of them, the goal being to optimize the route.

Features

Your program must include at least two modes :

- a user mode that allows to tune algorithm parameters
- a time-limited mode where all parameters are optimized to reduce the number of steps to solve the problem in a given time.



Both training and testing the number of episodes must be entered by the user when the program launches.

Output and delivery

Your program should output the mean time for finishing the game and the mean rewards within all the episodes. It should also display random episodes.

To prove your algorithm performances, you must benchmark it against as many metrics as possible, playing around with several parameters and rewards definition.

To do so, you are expected to develop a naive bruteforce algorithm as a comparaison point.

A brute force algorithm can take as much as 350 steps to solve the game when a correct RL model could take 20 steps.
But the most important benchmark is the one you got with your first non-optimized algorithm.
Start from here and then make improvements.

You must report these facts and figures in a document that also includes:

- commentaries about this benchmark that justify your algorithm choices and tuning
- your optimization strategy (concerning parameters and game rewards)
- different relevant graphics and arrays.

You may add other comparaison algorithms to prove the efficiency of your algorithm.
For instance, add a Q-Learning algorithm if you implemented a Deep Q-Learning algorithm.