Taxi Driver

Introduction

For this project, we were asked to solve a game named “Taxi-v3”. In this game, you play a taxi that must pick up a passenger present in a random location on the map, and drop them off in one of the four possible locations.

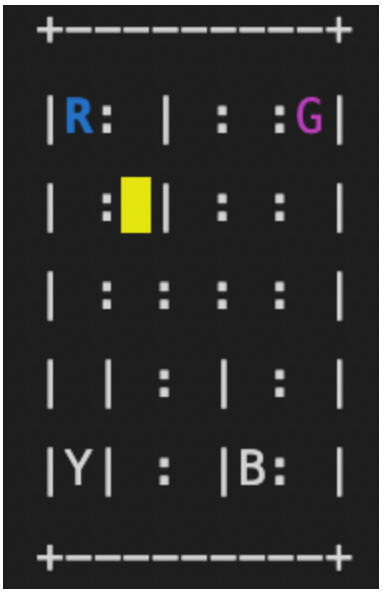
Flow of a game

You start the game by spawning at a random location. Each round, you can choose to play 6 differents moves:

- up
- down
- left
- right
- dropoff
- pickup

Each time you move, you receive a reward of -1. If you dropoff or pickup a passenger at a wrong location, you get a reward of -10. Finally, if you dropoff or pickup at the right location, you get a reward of +20.

With these informations, we now must create an algorithm that solves this problem in the minimum steps possible.



Algorithm selection

Brute force

The first solution and the easiest one is using a brute force algorithm. This algorithm will play a random move each round until it wins.

With this solution, the result is totally random and the algorithm doesn’t learn. It means that each time it plays a game, it won’t be smarter and it won’t have more chances to win.

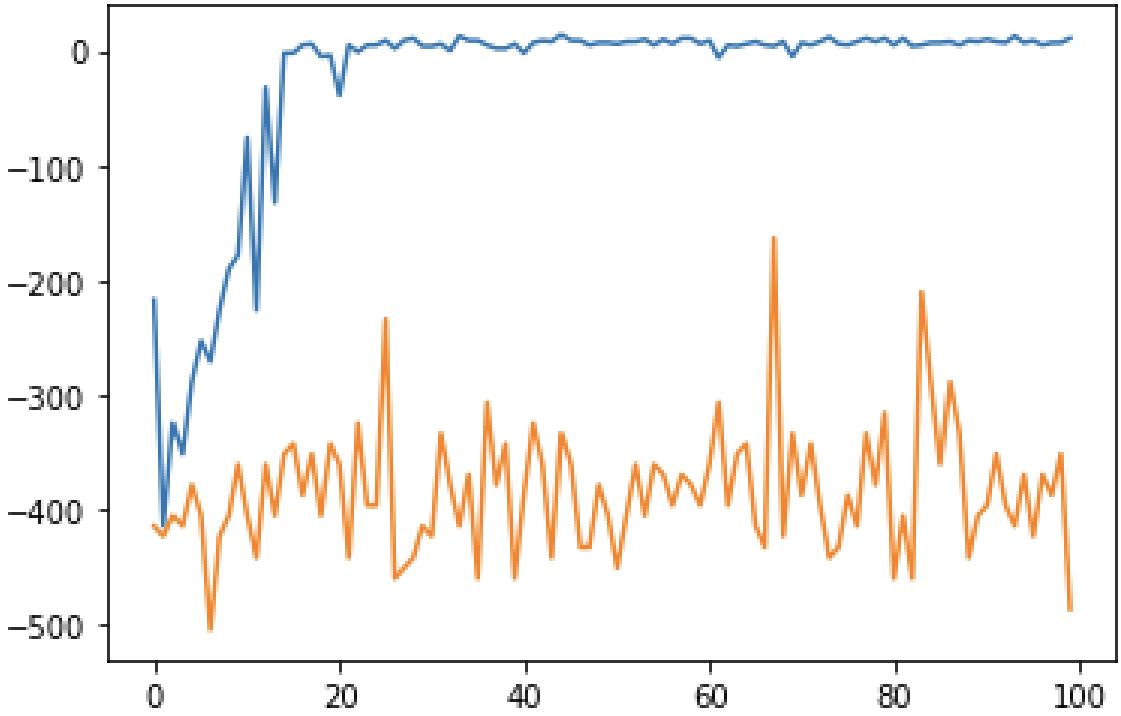
This solution is the fastest and the simplest to develop but we can’t call it AI as it doesn’t learn from his mistakes.

Q-learning

The second solution is named Q-learning. Q-learning is a model-free reinforcement learning algorithm to learn the value of an action in a particular state.

The principle of this algorithm is to create an array with all possible states of the game and in each state, another array of 6 numbers for the 6 possible moves. When the program makes a move, it update the corresponding value using the Bellman equation. With this method, the algorithm learns every round if it was a good or a bad move and as the rounds go by, it gets smarter. It makes less mistakes and begin to win more often and with less moves.

Comparison



In the graph above, we can compare the average reward of both algorithm:

- in orange its the brute force one
- in blue the Q-learning.

We can see that, as the rounds go by, the average reward of the brute force algorithm stays around the same value of -400. This number is random but stays under 0 (which is bad) and doesn’t improve which proves that our algorithm doesn’t learn.

On the other hand, the Q-learning algorithm begins with a low average reward but improves quickly and ends up stagnating around 7,5 of average rewards.

Tuning algorithm

Now that we have our algorithm, we need to find the best parameters. To do so, we started with values thats seems right to us and tested a lot of differents values until we found the best one.

The best values that we found are the following:

```
nb_episodes = 10000
max_steps_per_episode = 100
learning_rate = 0.81
discount_rate = 0.96
exploration_rate = 1
max_exploration_rate = 1
min_exploration_rate = 0.01
exploration_decay_rate = 0.01
```

After 10000 rounds of training, the algorithm is able to win every game in the fewest possible moves.