Performance Evaluation project:
Optimizing cars' trajectory with AI

Ottavy Macéo, Longatte Mathieu, Louison Mocq

Part I
Introduction

The goal of this project is split into five parts:

- Creating racing car environment to simulate simple 2D racing car model.
- Implementing Deep Q-learning and Genetic algorithms to optmize the behaviour of a car on trakcs so that the car can have the best trajectories possible.
- Evalute the performances of Deep Q-learning and Genetic algorithms and compare them.
- Evalute the performances of Deep Q-learning depending of the hyperparameters.
- As a bonus: evaluate the performance of our best car's behaviour.

All the code have been made with python.

Part II
Deep Q-learning

1 Markovian decision porcess

2 What is Q value?

3 What is Q learning

Part III
Genetic algorithms

4 What are genetic algorithms

5 Markov Chain modelisation

6 NEAT

Part IV
Car Racing environment

7 Tracks

A track is originally a .png file wich look like the left image of figure 7. Then, the image is converted to a matrix T such that $T[0][0]$ is the bottom left corner. After that, we crop the image, compute the

starting point and the lines of track (that will be explained in the reward part) to have a final result which look the right image of figure 7.

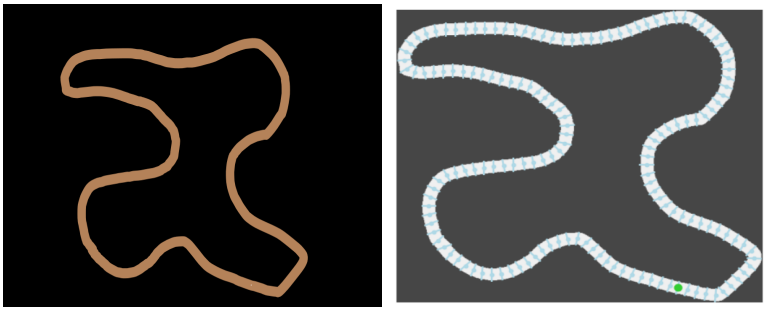

Figure 1: .png and computed track

8 Cars' physics

The Car physic is really simple. It is a 2D cartoon-like physics that act as follow:
The car has to main informations: its speed $\in$ [0, MaxSpeed] and its rotation $\in [0, 360]$. The physics is simple, at each times step, the car move to next coordinates on direction of the car's rotation and of a lenght equal to the car's speed.
If the coordinates of the car is (x, y), its speed is s and its rotation is α, then, after a time step, the coordinate of the car will be:

$$(s.cos(\frac{\pi}{180}\alpha) + x,\ s.sin(\frac{\pi}{180}\alpha) + x)$$

Moreover, at each time step, the car can make some actions:

- it can accelerate, this will increase the car's speed by a constant
- it can brake, this will decrease the car's speed by reduce the car speed by a constant. The car cannot have a negative speed.
- it can turn, i.e. add a constant $\in [\![-K, K]\!]$ to its rotation. K is a constant that is the maximum angle the car can turn per each time step.

The behaviour of the car will need to interact with the track therefore we need to decide what is the state of a car, i.e. how the car see the environment. We could give to our algorithms the track matrices, the informations of the car but this will leed to to many argument because a track can have size 900×600. Therefore we will need to train on all possible state wich will be at least $2^{900\times600}$. Therefore, we decided to give a more realistic state wich represent how a car racer see. Then the state of a car is a array of size 8.

- T_0 is the current speed of the car
- $\forall i \in \{1, ..., 7\}$, T_i is the distance of the car to the next wall in the direction $\alpha + A_{i-1}$ where α is the current rotation of the car and $A = [60, 40, 20, 0, -20, -40, -60]$

Then, the representation looks like figure 8.

9 Technical aspects of the environment

To manipulate our environment, we use the python packages `gymnasium` which provide code convention for those type or environment, i.e. environment where at each time step, you have one action to do. The environment has to have some essential function: `reset()` that reset the environment to be able to do an other simulation, `render()` that render the current state of our environment and the most important one is `step()` that do one step of time, i.e. given an action, the `step()` function figure out

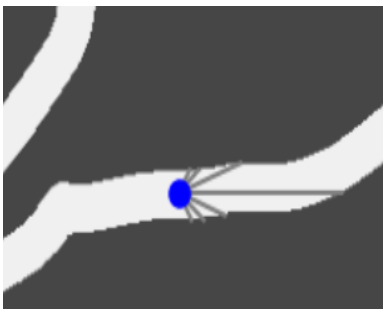

Figure 2: Car state

is the car has crashed or not, move the carn to its next position and return the new state of the car, a reward and if the car has crashed.
Our environment has a variable named `time` wich give us the opportunitie to discretise more or less or time.

10 Rewards

For those type of problem where the AI model has to compute a behaviour, the AI model produces something which look like a function f that take a car state and return an action. We need to specifie to our AI model when it produce a good action and a bad action, for instance, if a car crash, we need to punish the AI model.
We do that thanks to a function reward implemented in the function `step()` of our environment. The reward is an integer, the bigger it is the best the action was. To punished the car when it do something bad we do:

- If the car crashes, we stop the simulation and return a reward of -500
- If the car is not moving, i.e. has a speed of 0, the reward is -10

For the positive reward, we have automatically computed some track line (represented in right image of figure 7). If the car crosses next line, it has a reward of $+10\times$ the number of lines it has cross in the good order with this action. If the car cross a line in the wrong order, it means that it has gone backward, therefore, we punished the car with a reward of -200 and we stop the computation.
On top of that, at each time step, we add to the current reward the speed of the car divided by a constant to encourage the car to go fast.

Part V
Performance Evaluation

11 Algorithms

12 Best car