Development : Report

Name: Shubham Limbachiya

Banner Id.: 001382964

Course: Artificial Intelligence-01 (CSCI-4740-01)

Objective: Project is based on a Race car simulation where this car should run on different types of tracks which are already given in the repository by creating an Agent.

This car is running on 8 different tracks with the help of 5 lidar sensors and each track have different reward point and bonus after reaching to finish. And to obtain max results the values of the lidar sensors should be good enough to run it smooth and fast on each track.

What have I done: I tried to define values of each Lidar sensors which are- left, midleft, front, midright, right along with defining the maximum velocity to move. Those values were used to define the action that what should the car do (turn left or right, brake, coast or accelerate). Below are the different development stages in which I manipulated the values of all the lidar sensors and some actions with many trials on each track.

DEVELOPMENT VERSIONS

(I) Rule Agent (First):

```
# Controlling the Steering of the car to turn left or right.
        if right<1 or midright<1.2:
          steering='left'
        if right < 1 or midright < 1.2:
            steering = 'left'
        elif left < 1 or midleft < 1.2:
            steering = 'right'
        else:
            steering = 'straight'
# Go ahead or brake controls.
        if front > 1.2 and velo_car < 0.15:
            goahead = 'accelerate'
        elif front < 0.5 or velo_car > 0.40:
            goahead = 'coast'
```

```
    else:
        goahead = 'coast'
```

Values of each track after running atleast 10 times on each:

Tracks	Reward	Avg Speed	Max speed
1,2 (large Circle)	117.40-49	0.140 - 0.147	0.1 - 0.15
3,4 (Medium Circle)	117.10 – 117.45	0.138 – 0.142	0.1 – 0.15
5,6 (Small Circle)	115.58 – 116.28	0.12 – 0.14	0.1 – 0.14
7 (Oval)	116.24 – 117.65	0.1 – 0.145	0.1 0.15
8 (Spiral)	115.4 – 117.40 (Some -ve 49.05)	0.1 – 0.142	0.1 – 0.15

This was the first Initial code which I used in RuleAgent earlier in the racecar, the issue with this code was that it used to give some negative values for the 8th track and its average speed was varying with a huge difference.

(II)My Agent (Friday):

```
# Controlling the Steering of the car to turn left or right.
    if right<1.24 or midright<1.43:
      steering='left'
    elif left<1.24 or midleft<1.43:
        steering = 'right'
    else:
        steering = 'straight'
# Go ahead or brake controls.
    if front > 1.3 and velo_car < 0.25:
        goahead = 'accelerate'
    elif front < 0.5 or velo_car > 0.8:
        goahead = 'brake'
    else:
        goahead = 'coast'
```

Values of each track after running at least 10 times on each:

Tracks	Reward	Avg Speed	Max speed
1,2 (large Circle)	128 – 130	0.24	0.25
3,4 (Medium Circle)	126 – 129	0.24	0.25

5,6 (Small Circle)	122 - 124	0.17 – 0.19	0.25
7 (Oval)	128 - 129	0.24	0.25
8 (Spiral)	-ve 40 ++	8 - 10	0.0

Changes: After that I observed that due to low velocity and less amount of times acceleration action made by the car the total rewards in all the tracks was lesser than expected, So I tried to increase the velocity of car to perform acceleration earlier with increasing the length of sensors distance to work.

Result: This resulted in increasing the rewards for the tracks 1-7 by completing it faster than earlier, but on the other side it was not able to run on the 8th track and always crashes at the smallest curve just after start.

(III) MyAgent(Saturday):

Sensors midleft and midright from 1.43 to1.45.
front from 1.3 to 1.4 and velo_car from 0.25 to 0.12:
Limit to start Coasting at velo_car from 0.8 to 0.4

And added:

```
elif left - right > 1:
    steering = 'left'
elif right - left > 1:
    steering = 'right'
```

Values of each track after running atleast 10 times on each:

Tracks	Reward	Avg Speed	Max speed
1,2 (large Circle)	117-118	0.14	0.15
3,4 (Medium Circle)	116-119	0.147	0.15
5,6 (Small Circle)	119-122	0.17 – 0.19	0.15
7 (Oval)	117	0.148	0.15
8 (Spiral)	117-118 (some -ve)	0.15	0.15

Changes: I decided to change the values of sensors which are causing problems before which are mentioned above and the main change in the behavior of the car was when I decreased the value of max velocity of the car. This time I also tried to check the difference of left and right sensor with a thought that it will keep the car in the center.

Result: This all changes resulted that the car was now running good on all the tracks with good average speed and enough rewards. But sometimes it was still crashing when it just accelerates and turns on a curve on the 8th track which was resulted in a couple of negative rewards.

(IV)MyAgent (Sunday 8/6):

left, right 1.24 to 1, midright and midleft 1.45 to 1.4
front 1.4 to 0.7 and velocity was same as earlier
But the difference of left and right from 1 to 0.5

Values of each track after running at least 10 times on each:

Tracks	Reward	Avg Speed	Max speed
1,2 (large Circle)	117	0.15	0.15
3,4 (Medium Circle)	116	0.147	0.15
5,6 (Small Circle)	119-122	0.13 – 0.15	0.15
7 (Oval)	117	0.148	0.15
8 (Spiral)	117-118 (some -ve)	0.14	0.15

Changes: In this version I tried to reassign the values of all the sensors by performing numerous trials on all the tracks and figured out some values of sensors which can make the car to return more rewards. Also I tried to decrease the velocity threshold again so that the car gets enough time to turn on small curves. As it was working good in small circles but when small was combined with a large one, sometimes it crashes into it which gave many negative results for track 8th as well.

Result: The agent performs well initially by increasing speed and staying close to center but after some iterations it starts drifting far from center which I observed in Left or Right sensors values from tracks sides and then crashes hard with a reward of -49.98 which ruined the whole total reward. But the most important thing I got to know from the simulation that the turning sensor values are good enough to reduce chances of crash if I ignore the speed.

(V)MyAgent(Sunday, 8/6):

```
if front < 0.7:
    goahead = 'brake'
elif front > 1.5: #This change made it to change its velocity depending upon distence of front.
    if velocity < 0.15:
        goahead = 'accelerate'
    else:
        goahead = 'coast'
elif front > 1.0:
    if velocity < 0.12:
        goahead = 'accelerate'
    else:
        goahead = 'coast'
else:
    goahead = 'coast'
```

Values of each track after running atleast 10 times on each:

Tracks	Reward	Avg Speed	Max speed
1,2 (large Circle)	117.23-118	0.148	0.15
3,4 (Medium Circle)	117.6 - 118.16	0.147	0.15
5,6 (Small Circle)	119-121.7	0.16 – 0.18	0.15
7 (Oval)	117.5 – 118.2	0.146	0.15
8 (Spiral)	117.8 -118	0.149	0.15

Changes: As the track 8 has sharp curves and a tight S-shaped layout so the same logic that worked on tracks 1–7 not worked on it. So, I tried to solve that issues by splitting the acceleration, brake and coast actions by separating values of Front sensor which decides the maximum values of velocity for each curves.

Result: The separation of velocity threshold for each length of front sensors made the car more sensitive to continuous curves on the 8th track and also keeping the car mostly at a constant velocity on each of the tracks. This was the code which gave best results among all with least crashes and good rewards. So, I decided to finalize this code as one of the optimal result producing agent.

Conclusion:

In this project I implemented an Agent to control a simulated Race car which used 5 Lidar sensors and different actions of turning and moving ahead, where the goal was to complete various track circuit by avoiding crashes and moving as fast as it can to maximize the rewards.

I learned that how this simple reactive agent works by directly assigning sensor values to perform various actions for which I designed a “steering” and “goahead” logics to define distance of all sensors (left, midleft, front, midright and right) and velocity to accelerate, coast or brake.

Future Work:

While this agent was mostly human tuning I would like to train it with the use of learning agent which can give feedback and learn by its own. Where I can use concepts like Q-learning, Epsolin-Zero. This integration can make the racecar more autonomous and effective which can adapt itself in any curves and can reach to finish line more quickly.