CS747 Fundamentals of Learning Agents
Assignment 3

Sanchit Jindal 200020120

1 Introduction

In this Assignment we create a self driving car, which starts on a random position on a map and need to reach the road, There are two levels of difficulty in the assignment, in the first part we don't have any mud pits on the plot and the car can go anywhere, in the second part there are 4 mud pits which need to be avoided while travelling

2 Part 1 (Q-learning with function Approximations)

This was my first Idea that I implemented for this question
The Main Algorithm for it is

- Initialize the weights for each action for each of the features I want.
- In each step Calculate the linear Combination of the features for each of the possible pair of actions.
- Select the maximum Q value and return the corresponding action with a probability ϵ or a random action .
- Get the reward and the next state from the environment.
- For the new state calculate the maximum Q value.
- Update the Weights (only for the action taken) as follows:-

$$W = W + \alpha * (R + newQ - oldQ) * \text{features}$$

- This is the principal of gradient descent to get to the optimal value.
- I also store the state transitions in a cache and after update choose some transitions to again use to update weights, This provides stability to the gradient descent and allow for faster learning.
- I also tried various optimizations such as different biases for all the actions and reducing the value of ϵ and learning rate
- I also tried to make some weights only accessible some time such as if the car is near to the right edge but only if the value of y is not in the region of exit
- I also tried tried evaluating the code for different values of the hyperparameters

The Features used by me were

- Distance of the car from the center of the road
- The angle between the direction of the car and the direction of the exit with respect to the car
- If the car goes near the edges other features become non zero, the distance from the edge and the angle the car is pointing away from it.

I used the blog post Going Deeper Into Reinforcement Learning: Understanding Q-Learning and Linear Function Approximations The link is given in references.txt

This Approach did not pan out as even though the car starts to move towards the exit and remain away from the edges due to stochasticity the car didn't reach the exit every time. So I have submitted a deterministic controller function. The Code is commented Out but still available in the run_simulator.py file

3 Part 1 (Deterministic Controller)

In the Deterministic Controller, I have set up rules that the car follows depending on its location and on the orientation
The Algorithm is

- If the Car is in the range such that the y value is $-10 \leq y \leq 10$ then it should just point towards the road and accelerate.
- Otherwise if the car is not in this lane then it moves towards this lane along the y axis and then once in the lane it moves towards the exit.

3.1 Evaluation

Manually setting up the car position the maximum amount that is within the limits give a time steps of around **200**
The Code Passes the given Testcases without any error
I have also run the code on 100 random testcases which all successfully move to the exit.

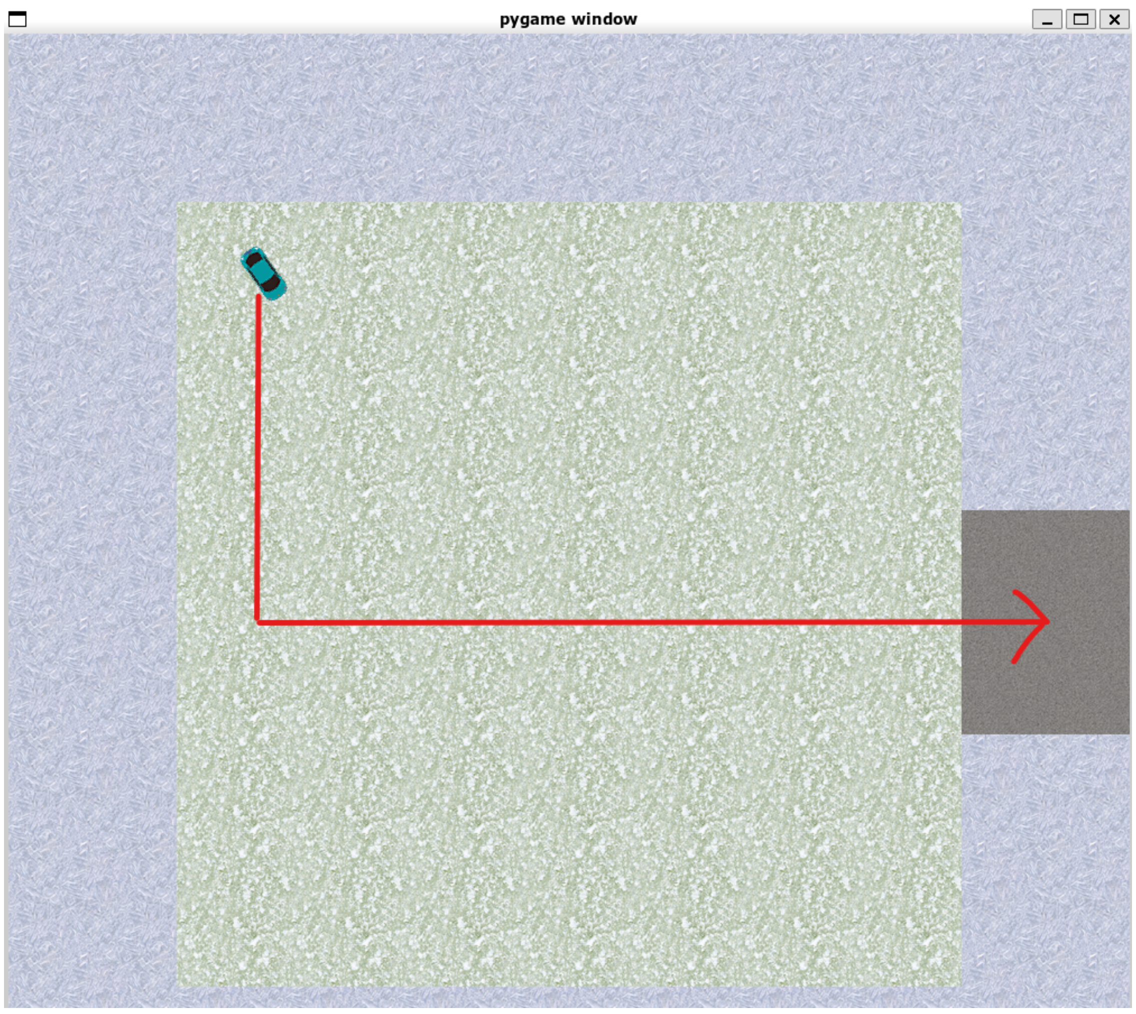

Figure 1: Route without pits

4 Part 2 (Obstacles in the path)

In the Second Part of the Assignment the car should navigate around obstacles in its path to reach the exit. In this part also I have I have implemented a deterministic controller that will move the car depending on the current location and angle. I have exploited the facts that:-

- There are only 4 pits in the map
- There is always only one pit in each quadrant
- The possible location of the pits leave safe space along the axis of the map and along the edges of the map

The Algorithm used by the controller to move the car is:-

- If The Car is in any of the safe spaces left by the pits then the car will have a safe path to the exit and hence can easily move along the path without losing.
- If the car is currently not in a safe spot, that is it is in one of the quadrants then,
- The algorithm checks along the 4 directions *left, down, right, up* to check if the car has a path to an edge (safe space).
- Because of the possible ways of how pits are placed in the map there always will be one direction that is empty.
- The car moves in the direction that is free first checking the right so as to reduce the distance and checking left last
- If in the direction the car falls into a safe space then we already know a path out of there

4.1 Evaluation

The Code is able to navigate the map for the given testcases in the given amount of time,
I have also run the code for 100 random seeds, for all of them the car was successfully able to traverse to the exit,
The Outputs for the 100 cases is available in the file Part2.txt

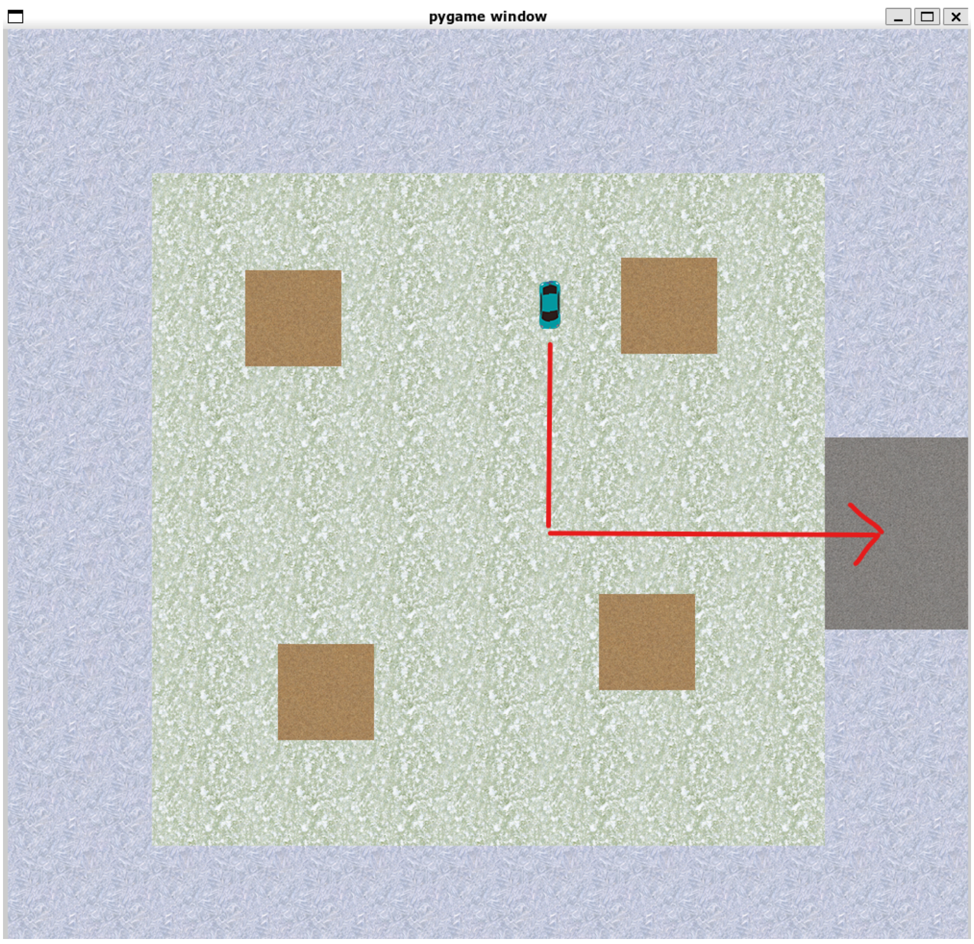

Figure 2: Route with pits

5 File Structure

The Files submitted are:-

```
200020120
|------Part2.txt
|------report.pdf
|------run_simulator.py
|------references.txt
```