Lab 12: Reinforcement Learning

This assignment is due to Gradescope by the beginning of lab next week (2:45p on 4/28). You may work with a partner on this lab – if you do, submit only one solution as a "group" on Gradescope.

Introduction

The goal of this lab is to implement a deep Q-learning agent to play a top-down 2D car racing game. The goal of the game is to drive the car around a randomized track to the finish line as fast as possible. The game simulates some basic physics, such as the car skidding if it attempts to steer too sharply while driving fast or accelerating.

A human player would play this game with the arrow keys (left/right, gas, brake). Your RL model will receive exactly the same information as a human player, i.e. the game screen pixels as an array of shape (96, 96, 3) and a scalar reward metric corresponding to the distance traveled along the track in the elapsed time.

At every time step, the agent may take an action expressed as a 3-element list. The first element corresponds to the position of the steering wheel and must be a float between -1 (full left) and 1 (full right). The second element corresponds to the acceleration pedal and must be a float between 0 (no acceleration) and 1 (full acceleration). The third element corresponds to the brake pedal and must be a float between 0 (no brake) and 1 (full brake).

We will host a single-elimination tournament at the start of next week's lab for extra credit points. Pairs of cars will compete on the same track (generated with the same random seed) and the car that finishes with the highest reward value will advance. You will receive 3 points of extra credit for placing in the quarterfinals, 6 points for placing in the semifinals, 9 points for placing in the finals, and 12 points for winning the tournament.

Provided Files

- `Lab4.pdf`: This file
- `RL.py`: Game simulation
- `Agent.py`: Code scaffold for your agent
- `Lab12_Questions.txt`: Open-ended questions

Instructions

Part 0: Setup

1. `pip install` the following libraries into your virtual environment: `gym`, `pygame`, `box2d`
2. Run `python RL.py --help` for explanations of the command line arguments needed to run the game. You should NOT modify `RL.py`, but you should familiarize yourself with the code so you know how the game simulation interfaces with your agent.
3. Run `python RL.py 1 1000` to make sure you have everything set up properly. This will run the game once for a maximum of 1000 time steps. You should see that the default agent just drives straight off the edge of the map. The bottom of the game displays the current reward in addition to some indicators not available to the model (true speed, four ABS sensors, steering wheel position, gyroscope.

Part 1: Programming

Your task is to complete the `Agent.py` file such that it uses reinforcement learning to drive the car around the track as quickly as possible. This will require completing the provided methods according to their docstrings and adding additional helper methods and class fields of your own. You may modify any part of the `Agent.py` file, as long as you do not change its interface with `RL.py`.

Your agent must perform deep Q-Learning using one (or more) Keras neural networks. Otherwise, you may experiment with whichever deep Q-learning variants, network architectures, hyperparameters, and observation processing techniques you think will be most successful. Your agent will need to explore the environment of the game and convert the results of this exploration into training data for its model.

Deliverables

Submit your final version of `Agent.py` and `Lab12_Questions.txt` to Gradescope.

Grading

Your grade will be based on the following:

- 10 pts: `create_model()` function
- 10 pts: `load()` & `save()` functions
- 10 pts: `act()` function and helpers
- 10 pts: RL effectiveness (does the car make it around the track? How fast?)
- 10 pts: `Lab12_Questions.txt`