Assignment 3

Name - Ujjwal Sharma
Roll No - 23M0837

Solution Overview

The solution to this assignment is based on hand-coded feature extraction and parameter learning for an autonomous agent navigating a road. Our approach focuses on generating relevant features that describe the agent's environment, followed by learning parameters that control the agent's steering and acceleration behaviors. The model leverages hand-crafted features, combining them into subsets to address specific tasks: steering and acceleration.

Feature Extraction

The feature extraction pipeline centers around aligning the agent's position with the center of the road, ensuring that its movements are driven by key features from the road structure.

Road Layout:

We have a 2D grid environment (e.g., a 12x6 grid), with the agent located at position (12, 6) in the 0-indexed grid. The goal is to extract relevant features that represent the agent's surroundings with respect to its location.

Feature Set:

In total, 18 features are extracted, which are grouped into several categories based on their relevance to the steering and acceleration tasks.

For our best model, we extracted 34 features in total.
The features are

- Feature 1-13: Weighted Center of Centers of Each Row
  For each row in the grid, we calculate the "center" of the road, which represents the point where the lane is most concentrated

```
        centerForRow = np.where(state[row] == 1)[0]
         centerPointForRow = 0
         if len(centerForRow) > 0:
             centerPointForRow = np.mean(centerForRow)
```

```
        # add the center points weighted according to the distance from the center and the row
        rowCenteresWeighted.append((row + 1) * (centerPointForRow - mean_point))
```

- Feature 14: Average Center point of the rows: We calculate the average center point across all rows to capture the overall position of the lanes.

- Feature 15: Average center point for the last rows: This feature is the average of the center points for the last 5 rows. This helps capture the short-term trend of the road's layout.

```
avg_center_point_for_last_rows = avg_center_point_for_last_rows / 5
```

- Feature 16: Steering Intensity: The intensity of the steering that the agent should do. It depends upon the weighted centers of the last rows

```
steering_intensity = np.sum(rowCenteresWeighted[8:])/2  # Steering intensity based on the distance from the center of the lane
```

- Feature 17: Acceleration Intensity: The intensity of acceleration that the agent should have at a particular speed. It depends upon 3 things
    - Sensitivity Factor: Based on the agent's current speed.

```
  # Sensitivity factor based on speed
    sensitivity_factor = 1.0 - (speed / 20)
```

    - Speed Difference with max Speed = (20 - speed)
    - The deviation of the center point (6) from the weighted mean point of last rows.

```
    denominator_for_acceleration_intensity = (abs(mean_point - np.mean(rowCenteresWeighted[8:])/5 ) + 1)
```

```
acceleration_intensity = sensitivity_factor *  (20 - speed) / denominator_for_acceleration_intensity # How far the car is from the center of the lane, for smaller speed the acceleration intensity is higher and positive
```

- Feature 18: Speed: Current speed of the car

```
speed = info["speed"]  # Current speed of the car
```

Feature Combination

The extracted features are combined into two subsets: one for steering and one for acceleration. This allows the agent to focus on relevant information for each task.

Steering

For steering, the following features are selected:

- Weighted center of rows (`rowCenteresWeighted`)
- Steering intensity (`steering_intensity`)
- Average center of the last rows (`avg_center_point_for_last_rows`)

```
steering_features = np.array(rowCenteresWeighted +
[steering_intensity,avg_center_point_for_last_rows])  # Features for
steering
```

Acceleration

For acceleration, the following features are selected:

- Average center of the last rows (`avg_center_point_for_last_rows`)
- Average center (`avg_center_point`)
- Speed (`speed`)
- Acceleration intensity (`acceleration_intensity`)

```
acceleration_features = np.array([avg_center_point_for_last_rows,
avg_center_point, speed, acceleration_intensity])
```

Parameters

Once the features are defined, we combine each set of features with a weight vector. Each feature subset is dot-multiplied by a corresponding parameter vector to obtain steering and acceleration outputs. Both steering and acceleration values are passed through the `tanh` activation function to ensure that the final outputs lie between [-1, 1].

- 15 for steering
- 4 for acceleration

```
      steering_unsqueezed = np.dot(steering_features, params[:15])  #
Steering is a linear combination of features
      steering = np.tanh(steering_unsqueezed)  # Apply tanh activation
function to get steering value
      acceleration_unsqueezed = np.dot(acceleration_features, params[15:])
# Acceleration is a linear combination of features

      acceleration = np.tanh(acceleration_unsqueezed)  # Apply tanh
activation function to get acceleration value

      return [acceleration, steering]
```

Fitness

Before optimizing the parameters, the CMA-ES algorithm evaluates the performance of the agent using a **fitness function**. In this assignment, the fitness function is defined as the **average distance traveled by the agent** using a given policy across all six tracks.

```
def fitness(params):
    total_distance = 0.0
    num_tracks = 6  # Evaluate on all 6 tracks
    for track in range(num_tracks):
        env.unwrapped.config["track"] = track
        (obs, info) = env.reset()
        state = obs[0]
        done = False
        while not done:
            action = policy(state, info, False, params)
            (obs, _, term, trunc, info) = env.step(action)
```

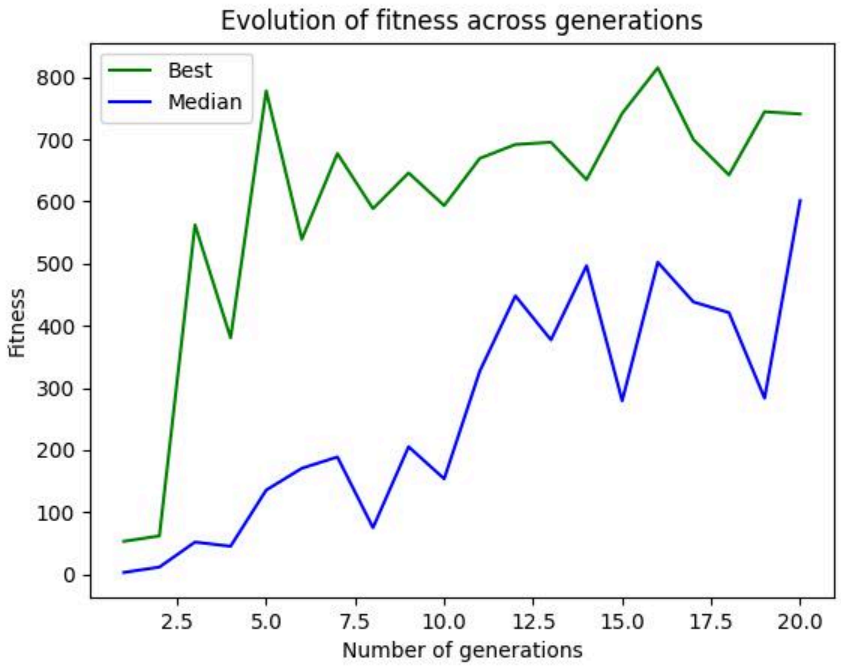
```
            state = obs[0]
            done = term or trunc
        total_distance += info["distance_covered"]
    avg_distance = total_distance / num_tracks
    return -avg_distance  # Minimize negative distance to maximize actual
distance
```

Learning

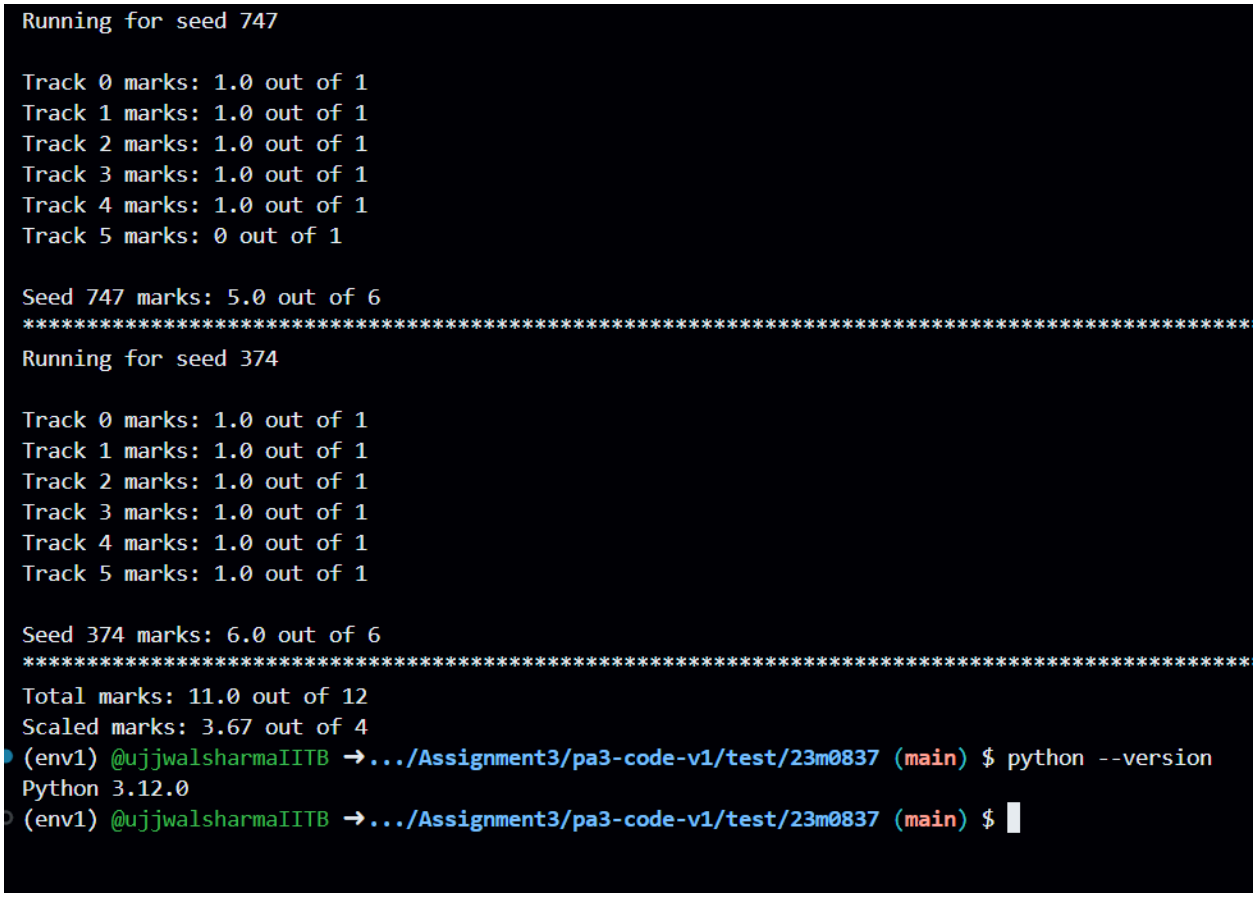
To optimize the parameters, we use the **CMA-ES** (Covariance Matrix Adaptation Evolution Strategy) optimization algorithm. This method is well-suited for black-box optimization problems, where the goal is to minimize a cost function without access to the gradient or explicit model structure.
For running the CMA-ES algorithm, we use these values

- `num_gen = 20`
- `pop_size = 20`
- `Num_policy_params = 19`

Learning Curve



I have tried various features, but these features seem to generalize best.



I have tested my code on both Linux and Windows on Python 3.12.0.

I have added the details of the experiments with complex features in the Appendix. The code for that is inside the complex_features folder.

All the code can be found at the GitHub repository
https://github.com/ujjwalsharmaIITB/FILA-RL-SEM4/tree/main/Assignment3/pa3-code-v1

The final submission is the **main.py** and **cmaes_params.json** in the parent folder.

APPENDIX

I also tried adding more features and different feature combination techniques, and they perform well on the training set, but they tend to overfit and hence do not generalize better than simple features as mentioned above

Additional Features

The features are

- Features: Center of Centers of Each Row: For each row, we also compute a simple "center" value, which represents the deviation of the row's center from the mean of all the row centers.

```
rowCentered.append(centerPointForRow-mean_point)
```

- Curvature Detection: Detect the curvature of the road

```
        curvature = np.mean(rowCentered[9:]) -
np.mean(rowCentered[5:10])
```

- Feature: Average Center for Top Rows: Similar to the last rows, we compute the average of the top rows. This helps capture the long-term trend of the road's layout.

```
avg_center_point_for_top_rows = avg_center_point_for_top_rows / 8
```

Feature Combination

Steering

```
steering_features = np.array(rowCenteresWeighted + [curvature,
steering_intensity, avg_center_point_for_last_rows])  # Features for
steering 28
```

Acceleration

```
acceleration_features = np.array(rowCenteresWeighted + [speed, curvature,
avg_center_point_for_top_rows, avg_center_point_for_last_rows,
acceleration_intensity]) #
```

```
(rl-course) [ssmt@localhost CS747PA3]$
Running for seed 747

Track 0 marks: 1.0 out of 1
Track 1 marks: 0.5 out of 1
Track 2 marks: 1.0 out of 1
Track 3 marks: 1.0 out of 1
Track 4 marks: 0 out of 1
Track 5 marks: 0 out of 1

Seed 747 marks: 3.5 out of 6
****************************************
Running for seed 374

Track 0 marks: 0 out of 1
Track 1 marks: 1.0 out of 1
Track 2 marks: 1.0 out of 1
Track 3 marks: 0.5 out of 1
Track 4 marks: 0.5 out of 1
Track 5 marks: 0 out of 1

Seed 374 marks: 3.0 out of 6
****************************************
Total marks: 6.5 out of 12
Scaled marks: 2.17 out of 4
```

The code for these features is inside the complex_features folder, which is just for reference.

The final submission is the **main.py** and **cmaes_params.json** in the parent folder.