Project

Reinforcement Learning: Searching High-Quality Policies to Control an RC Car from Images

INSTRUCTIONS

You need to deliver:

- a report containing your discussions, algorithms, and results.
- your code commented and cleaned with the associated model weights, the `interface.py` file, the `load_agent.py` file and README that explain how your code runs.
- a `requirements.txt` file in order to install your libraries (with the correct version).

1 DOMAIN (3 POINTS)

We consider the `CarRacing-v3` environment with continuous action space implemented in the Gymnasium library. The Car Racing environment in Gymnasium is a top-down racing simulation where the agent must navigate a track as quickly as possible while avoiding obstacles. The goal is to maximize the reward by visiting track tiles efficiently, with penalties for going off-track or taking too long.
You do not need to code the environment yourself; you can use `https://gymnasium.farama.org/environments/box2d/car_racing/`, where the Car Racing environment is available. We advise you to use Gymnasium with "`import gymnasium as gym`" in your scripts.

To set up the environment, create a conda environment and install the required libraries using the provided `requirements.txt` file. Follow these steps:

1. **Create a Conda Environment:**

```
conda create --name car_racing_env python=3.10
```

You can change the python version, but we advise you to use python 3.10.

2. **Activate the Environment:**

```
conda activate car_racing_env
```

3. **Install the Required Libraries:**

```
pip install -r requirements.txt
```

This will set up your environment with all the necessary dependencies for the Car Racing project.
In your report, describe the domain as seen in the course. The equations of the dynamics are not required.

2 External library algorithm (5 points)

You are asked to plug an algorithm from an external RL library that fits the Gymnasium API. Such library examples are CleanRL and Stablebaseline3.
Justify your algorithm choice based on your domain description. Describe your protocol for training this algorithm and justify each choice. For instance, how you chose the hyperparameters of your algorithm.
Your implementation should run using the `interface.py` script. You should save the weights of your models and modify the script named `load_agent.py` to load your model. You should also provide a `README.md` file with the details to run each experiment.

3 Personal algorithm (8 points)

You are asked to implement your algorithm with a justified protocol for training it. The implementation of your algorithm should work properly in the environment introduced in section 1. You can code an algorithm proposed in the literature. Note that every reference should be cited to not be considered plagiarism.
Your implementation should run using the `interface.py` script. You should save the weights of your models and modify the script named `load_agent.py` to load your model. You should also provide a `README.md` file with the details to run each experiment.

4 Discussion (4 points)

Compare your algorithm in section 2 and section 3. Discuss the performance of these algorithms in the environment in your report.