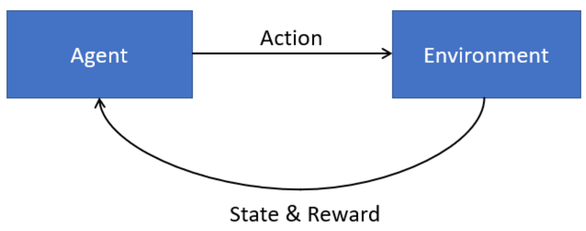
Multi-Agent Reinforcement Learning Benchmark for Traffic Control

Charbel ABI HANA

M1 International Track in Electrical Engineering @ Université Paris-Saclay. Paris, France

Introduction

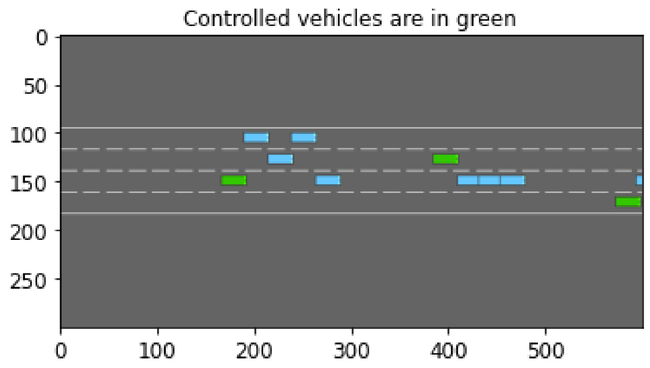
The domain of Reinforcement Learning has become a powerful learning framwork capable of learning complex policies in high dimensional environments. Reinforcement learning models learn from an environment. The environment has a set of rules and is usually assumed to be deterministic. A reinforcement learning model interacts with the environment through an agent. The agent has a state in the environment, and the agent can perform actions that change the state of the agent in the environment.



Road tests for autonomous driving algorithms hasn't been widely applied due to safety reasons, therefore we use intricate simulators like OpenAI's Gym, PettingZoo, SUMO-RL and many others. In this project, we will benchmark known Reinforcement Learning models on a highway environment based on OpenAI Gym in which we simulate a multi-agent environment where multiple autonomously driven cars will navigate as fast as possible on a highway while avoiding collisions with human-driven cars.

The Environment

The environment is based on OpenAI's Gym library for simulation environments. It is a simple python environment simulating autonmously-driven vehicles and human driven vehicles. We can configure the environment to obtain 3 controlled vehicles and 10 human vehicles.



Action/Observation Space

The action space is configured to be discrete. The availability of the actions depends on the positioning of the agent in the environment. The actions that each RL agent in the envrionemnt can take are *lane_left*,*lane_right*, *idle*, *faster* and *slower*. We'll be using the ***KinematicObservation*** which essentially is $V * F$ array where V represents a list of nearby vehicles and F represents the feature being the $x - y$ positions and velocities. We can also add the vehicle's orientation.

Vehicle	x	y	v_x	v_y
ego-vehicle	5.0	4.0	15.0	0
vehicle 1	-10.0	4.0	12.0	0
vehicle 2	13.0	8.0	13.5	0
...	...	...	...	...
vehicle V	22.2	10.5	18.0	0.5

Observation Space

DiscreteMetaAction:

ACTIONS_ALL =

```
{
  0: 'LANE_LEFT',
  1: 'IDLE',
  2: 'LANE_RIGHT',
  3: 'FASTER',
  4: 'SLOWER'
}
```

Action Space

Reward

The general focus is on two features: a vehicle should progress quickly on the road and should avoid collisions. Thus, the reward function is often composed of a velocity term and a collision term:

$$R(s,a) = a * \frac{v - v_{\min}}{v_{\max} - v_{\min}} - b * collision \tag{1}$$

where v, $v_{\min}$ and $v_{\max}$ are the current, minimum and maximum speed of the ego-vehicle respectively, and a, b are two configurable coefficients.

Decentralized Algorithms

Centralized Algorithms

- Dataset management to split source and training datas
- DeepJDOT training stability : best accuracy 25%
- Implementation of EMD (Optimal transport solver)
- Hyperparamters Searching and Fine tuning

Results

- Dataset management to split source and training datas
- DeepJDOT training stability : best accuracy 25%
- Implementation of EMD (Optimal transport solver)
- Hyperparamters Searching and Fine tuning