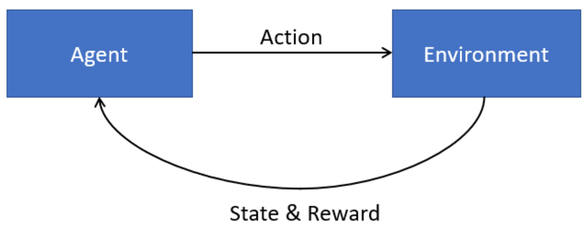
Multi-Agent Reinforcement Learning Benchmark for Traffic Control

Charbel ABI HANA

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

Introduction

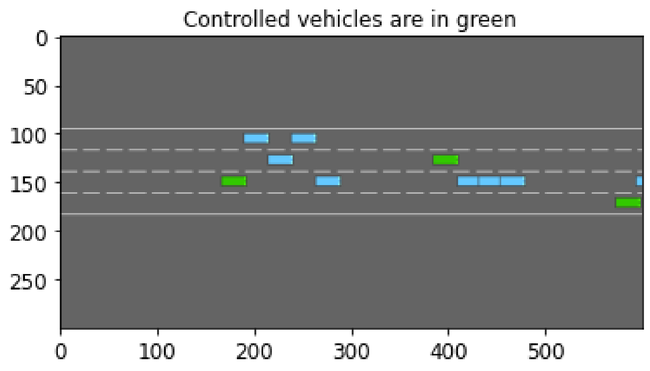
The domain of Reinforcement Learning has become a powerful learning framwork capable of learning complex policies in high dimensional environments. Reinforcement learning models learn from an environment. The environment has a set of rules and is usually assumed to be deterministic. A reinforcement learning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

- IDQN - independent DQN agents, one per autonomously-driven car, each with convolution layers.

Algorithm 1 : Deep Q-Learning Experience Replay Memory

Result : a nearly optimal policy π
Initialize replay memory $\mathcal{D}$ that has a certain length ;
Initialize parameter $\mathbf{w}$;
Initialize parameter $\mathbf{v}$ that calculate TD-Target by $\mathbf{v} \leftarrow \mathbf{w}$;
for *episode*$= 1$ *to* M **do**
  Observe initial state s_0 from environment ;
  for $t = 1$ *to* T **do**
    Select a random action with probability ϵ/m otherwise
    select action $a_t = \arg\max_a \hat{q}(s, \mathbf{a}, \mathbf{w}^t)$;
    Observe reward r_t and next state s_{t+1} from environment ;
    Store (s_t, a_t, r_t, s_{t+1}) tupel in $\mathcal{D}$;
    Sample random batch from $\mathcal{D}$;
    $\mathbf{y} \leftarrow r_t + \gamma \max_{a^\circ} \hat{q}(s_{t+1}, \mathbf{a}^\circ, \mathbf{v})$;
    $\hat{\mathbf{y}} \leftarrow \hat{q}(s_t, \mathbf{a}, \mathbf{w}^t)$;
    $\mathbf{w}^{t+1} \leftarrow \mathbf{w}^t - \alpha \nabla \frac{1}{2}(\mathbf{y} - \hat{\mathbf{y}})^2$;
    Every c time steps, set $\mathbf{v} \leftarrow \mathbf{w}^t$;
  end
end

- IPPO - independent PPO agents with similar neural network as the DQN agents.

Algorithm 2 PPO-Clip

Input: initial policy parameters θ_0, initial value function parameters ϕ_0
for $k = 0, 1, 2, \ldots$ **do**
  Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.
  Compute rewards-to-go $\hat{R}_t$.
  Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based on the current value function V_{ϕ_k}.
  Update the policy by maximizing the PPO-Clip objective:

$$\theta_{k+1} = \arg\max_\theta \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t),\ \ g(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t)) \right),$$

  typically via stochastic gradient ascent with Adam.
  Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg\min_\phi \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(V_\phi(s_t) - \hat{R}_t \right)^2,$$

  typically via some gradient descent algorithm.
end for

Results

The metrics used to compare both algorithms was the score over an episode. The reward is noisy as it is highly dependant of model hyperparameters but an overall convergence can be seen.

Future Improvements

- Add centralized algorithms where agents communicate with one centralized controller
- Extend environment to highway merging and intersection handling