CityU Car Survives Through Hong Kong

LIU Bonan, LIU Yuzhan, PU Ziyi, TIAN Zonghang, XIE Dong

Master of Science in Data Science, City University of Hong Kong

ABSTRACT

This project first simulated the complex traffic situation in Hong Kong as the environment: narrow and sinuous roads layout, obstacles alongside, pedestrians and animals come from all the directions accidentally, besides, limited time for the commute. A 3D back-engine supercar(named UFO) was then modeled in that environment, whose task is to drive to the target terminal point safe and as quickly as possible.

We control UFO through Unity ML-Agent API, deploy the Reinforcement Learning algorithm to help UFO learning to survive from random routes.

Introduction

Theory Support Deep Reinforcement learning algorithms are commonly used in many areas, such as the famous Alpha-Go, real-time gaming, and auto driving.

Motivation We notice that Hong Kong faced a very complex traffic situation than many other cities, even supercities like Shenzhen or Washington. Roads here are much narrow and sinuous, and some segments are located between mountain and sea. The very high density of buildings makes it possible that pedestrians suddenly appear from any horizontal direction, not to say wild pigs and other animals.

Our Target We try to simulate the traffic situation of HK, and 'build' a self-driving car, deploy RL algorithms to help it driving to the target terminal point safe and as quickly as possible.

Figure: Car Agent

Unity 3D Engine

Unity 3D engine can be used to create three-dimensional, two-dimensional, virtual reality, and augmented reality games, as well as simulations and other experiences.

The Unity Machine Learning Agents Toolkit (ML-Agents) is an open-source project that enables games and simulations to serve as environments for training intelligent agents.

We generated random routes and obstacles' locations every time. And assigned a series of sensors(ray casting sensors, trigger collider sensors, and spherical sensors) to the car agent made it agile to drive.

3D Enviroment Captures

(a) Agent's radar (b) Random route1 (c) Random route2 (d) Random route3 (e) Random route4

Experiment Setting and Launch

The below table listed hardware and software settings during our experiment.

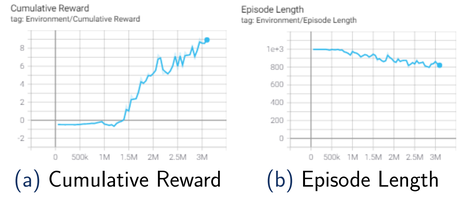
Item	Information
Memory	32G, mini. 16G
CPU	AMD/Intel 8 Core(2.6GHz-2.9GHz)
GPU	NVIDIA RTX 2060 (Optional)
Unity	Version: 2020.3.1f1
ML-Agent	Version: Relese 14
Pedestrians	Red light balls in Unity
obstacles	Static stone balls in Unity
Target Point	Clint('s photo) in Unity

Table: Experiment Setting

RL Algorithm

This project adopted Proximal Policy Optimization(PPO), which is an on-policy algorithm which can be used for environments with either discrete or continuous action spaces. This means that it explores by sampling actions according to the latest version of its stochastic policy. The amount of randomness in action selection depends on both initial conditions and the training procedure. Over the course of training, the policy typically becomes progressively less random, as the update rule encourages it to exploit rewards that it has already found. This may cause the policy to get trapped in local optima.

Results



(a) Cumulative Reward (b) Episode Length

(c) Policy Loss (d) Value Loss

Conclusion

So far, this project shows that given a rule set of reward and penalty, RL algorithms could work well on car self-driving tasks, no matter how the environment is. And we hope this could be a base for future self-driving cars coming into real life and even VR+AI driving school in the Hong Kong market.

Furtherwork

Up to now, due to time and computational resource limitations, this project has several shortcomings that can be improved:

- Because we need more than 2 days to train a single test, so we don't have enough time to train various RL algorithms and make further comparisons(PPO was done, but SAC is still in progress).
- There is only one agent(i.e., the car) in the driving environment. In the future, we hope to add more agents to co-operate in the simulated world. For example, at present, all the red balls are defined by a uniform random function, which we hope can replace by pedestrian agents.
- As for environmental construction, we think we can do more exquisitely to be closer to the real-world layout.

Video Demo

Here are some running captured during our experiments
Click to Open Car Demo