Bachelor project in Computer Science

Offline RL in the CARLA simulator

Remus Stefan Cernat

Abstract

Will be left for last

Table of contents

1 Introduction **3**

2 The Reinforcement Learning problem **5**

3 The CARLA environment **6**

3.1. Client-Server 6

3.2. Sensors and Actions 6

3.3. Functionalities and Related Work 7

4 Data Gathering **8**

4.1. D4RL 8

5 Learning to drive **9**

6 Conclusion **10**

Introduction

The field of machine learning is one that computer scientists pursue in order to solve an innumerable number of problems. Its applications vary widely from image and speech recognition, to medical diagnosis. A sub area of machine learning which takes inspiration from psychological behaviour - where an intelligent agent learns from rewards and punishments, through trial and error - is called reinforcement learning. And a good example of the applications of reinforcement learning - shortened to RL - is autonomous driving. The history of self-driving cars is relatively long, as computer scientists have been tackling it for dozens of years. But in a critical and delicate case such as this, learning through trial and error, is an imprudent dream. In such cases however, one can use an area of reinforcement learning, called *offline* RL. Using offline RL one can gather data in a more cautious manner, such as through human demonstrations. Truly efficient autonomous driving however, is relatively novel. And in a real life environment, the problem is rather complex. As there are numerous applications simulating a driving environment however, the virtual counterpart of the problem, is much more affordable to work with. One example of such a virtual environment would be CARLA; an open-source simulator for autonomous driving research. Although the realism of these simulators is up to debate, they do allow us to tackle the autonomous driving problem in a much safer and simpler environment. This can in turn give us a better understanding of some of the underlying parts of the problem, and pave the way to better research potential.

The Reinforcement Learning problem

Not yet written. Explain RL in general MDP, offline RL, actor critic (other algorithms?)

The CARLA environment

The environment which we will use to gather data and train the agent is one made with the help of an open-source simulator called CARLA. The simulator, whose name is an acronym for “Car Learning to Act” is one made specifically to support development and training of autonomous driving systems. In this section we will be taking a look at the intentions behind the simulator, some of the resources it provides, and how we have used them in our project.

3.1. Client-Server

One of the most fundamental parts of the simulator is its client-server interface. The server takes the commands from the client and runs the simulation, rendering the world with which our agent is supposed to interact. The client on the other hand sends commands to the server and receives certain sensor data. The creation of the client is simple. One must simply specify the IP address, and two TCP ports, which will be used to communicate with the server. Once the client is created, one can create a world using one of several blueprints readily available for use. In our project, we have used a blueprint representing a simple town, which allowed us to mimic a bustling city full of semaphores, pedestrians, and moving vehicles; albeit to a lower extent.

3.2. Sensors and Actions

Once the client and the world have been created, one can also choose a vehicle blueprint in order to produce the driving agent. A sensor can then be created and attached to the agent, which can be used to retrieve information about the world surrounding the vehicle. In our project we have used a sensor called “RBG camera”, which returns several attributes, among which the one which we have used is a raw_data object corresponding to an array of BGRA pixels. The raw_data object can be reconstructed to an RGB image, which in our project corresponds to the state. To send commands and apply actions on the vehicle, one can use the method to pass a “VehicleControl” object to the “apply_control()” method of the actor. This object takes several parameters that mimic typical driving controls. In our project however, we limited the actions to throttle and brake, which are represented as a scalar value between [0.0, 1.0], and steering, which takes the values [-1.0, 1.0].

3.3. Functionalities and Related Work

As the simulator was intended to be used for autonomous driving research, various approaches to machine learning have already been evaluated. The original team behind CARLA has evaluated the simulator with regards to a modular pipeline, imitation learning, and reinforcement learning. They have taken the approach of using the A3C algorithm, on a goal directed navigation. This is different from what we are trying to achieve by simply letting the agent roam freely throughout the world. Based on the methodological differences it will be interesting to take a look at how much our results vary from theirs.

Data Gathering

4.1. D4RL

In order to gather data, we originally wished to implement a simple Q-learning algorithm, and leave it running for enough time to gather a substantial and useful amount of state-action-reward pairs. In consideration of the scope of the project however, we settled for a simpler data gathering method and instead used an open-source benchmark for offline reinforcement learning called d4rl. The d4rl environment can be used to download datasets of observations for numerous tasks, including the CARLA environment. While trying to install d4rl however, we ran into numerous problems related to dependencies. As such, since everything including the code used to create the datasets is open-source, we settled for the next best solution and manually used the code to gather data. The data we gathered contains states represented by a (48, 48, 3)-dimensional RGB image, and as such occupies a large amount of space. In consideration of the space and time used to store and gather the data, we have gathered a total of 100 episodes, each containing 10 000 steps. This is equal to one million (48, 48, 3)-dimensional RGB images, without counting the rewards and actions. The dimensions of the state were chosen to replicate the original d4rl dataset. The data was stored in memory, in three separate files - each representing states, actions or rewards - using the pickle module. Furthermore we made sure that frequent enough interesting events happened throughout the trajectory of the vehicle by adding semaphores, weather changes, and moving actors.

Learning to drive

Not yet written. Talk about recover policy, implementation, results & discussion

Conclusion

Leave for last