CS 230 - Imitating driver's behavior in urban environment

Category : Imitation Learning

Malik Boudiaf
Department of Aeronautics
and Astronautics
Stanford University
Stanford, USA
Email: mboudiaf@stanford.edu
SUNet ID : mboudiaf

Ianis Bougdal-Lambert
Department of Aeronautics
and Astronautics
Stanford University
Stanford, USA
Email: ianisbl@stanford.edu
SUNet ID : ianisbl

I. Introduction

Accurate human driver models are critical for realistic simulation of driving scenarios, and have the potential to significantly advance research in automotive safety. Older approaches to autonomous driving tried to imitate human drivers' behaviors by describing them as a set of rules to follow. These early rule-based approaches tried to fit parametric model using strong assumptions about road conditions and human's behavior, and failed to generalize well to new driving scenarios.

Recent approaches are more data driven. Imitation learning (IL) uses expert data provided through human demonstrations and try to learn a policy that would be able to reproduce human's behavior in all possible states. Within IL framework, algorithms typically fall within two categories : behavioral cloning (BC), Reinforcement learning (RL).

BC uses a classic supervised learning framework to directly map observed states to actions by fitting the expert data. BC approach suffer, just like early methods, from a limited capacity of generalization. The RL approach, instead, tries to recover a policy as close as possible to an expert policy derived from data. Among RL approaches, the most referenced papers in literature use Inverse Reinforcement Learning (IRL), which assumes that the expert follows an optimal policy with respect to an unknown reward function that we try to recover. Once this reward function is recovered, one may use classic RL to find the target policy. But recovering this reward function can be very computationally expensive, and recent efforts have tried to bypass this step and directly optimize the target policy. Generative Adversarial Imitation Learning (GAIL) [3] is one of these approach.

Most of the latter algorithms have been implemented and tested in highway driving situations in [2]. In this project, we propose to extend the work done in [2] to a more urban scenario. This project will be common to the CS 236 project for both of us.

II. Problem Statement

We regard driving as a sequential decision making task in which the driver obeys a stochastic policy $\pi(a|s)$ mapping observed road conditions s to a distribution over driving actions a. We try to find a policy within a given class of policies $\pi_\theta(a|s)$ parametrized by θ, that best imitates human driving behavior.

III. Approach

A. Dataset

The dataset we will use comes from the NGSIM database that supports traffic modeling simulations [4]. More precisely, the dataset we will use is called *Lankershim Boulevard Dataset*. These data were collected using five video cameras mounted on the roof of a 36-story building located adjacent to the U.S. Highway 101 and Lankershim Boulevard interchange in the Universal City neighborhood, in Los Angeles. NG-VIDEO, a customized software application developed for the NGSIM program, transcribed the vehicle trajectory data from the video. This vehicle trajectory data provided the precise location of each vehicle within the study area every one-tenth of a second, resulting in detailed lane positions and locations relative to other vehicles. A total of 30 minutes of data are available in the full dataset, which are segmented into two 15-minute periods.

The data for Lankershim Blvd was extracted using the NGSIM.jl package developed by the Stanford Intelligent Systems Laboratory (SISL) at Stanford [5]. The code generates two elements from the raw NGSIM data: the roadway's centerlines and the trajectory data.
The raw data does not contain the centerlines *per se* but provides a .dwg file for the roadway. We used AutoCad to generate the centerlines from that .dwg file, and fed the resulting .dxf file to NGSIM.jl.
The trajectory data was provided as a raw CSV file that we had to process and clean before it could be used by the package to extract the trajectories. Both the Julia version used

in the package and the format of the NGSIM datbase have changed since the code was developed, so we had to spend a lot of time updating the code and modifying it so it could work with the new format of the data. The updated NGSIM package can be found on our github repository.

Below are two images of the Lankershim data that we were able to generate. On the first one we can see the empty roadway generated from the .dxf file. Contrary to the Highway 101 and Highway 80 roadways, the Lankershim Boulevard features multiple exit lanes, both to the left and to the right of the road. Intersections are also present at the location of the exits, although they cannot be seen directly on the rendered roadway. The second image shows a screenshot of one frame of the trajectory data, superimposed on the raodway. We can see vehicles going both ways, as well as vehicles entering and exiting the boulevard at the intersections. Note that we only generated one side of the road.

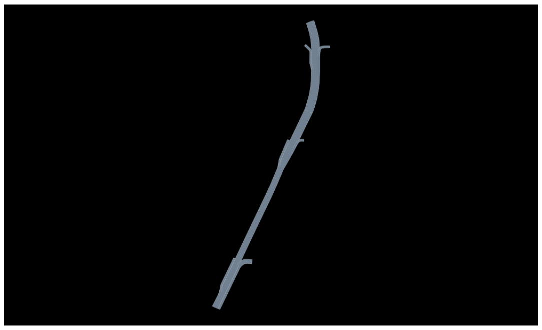


Fig. 1. Section of Lankershim Boulevard rendered using the AutoViz package.

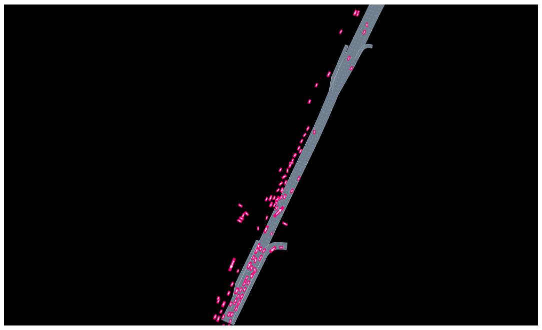
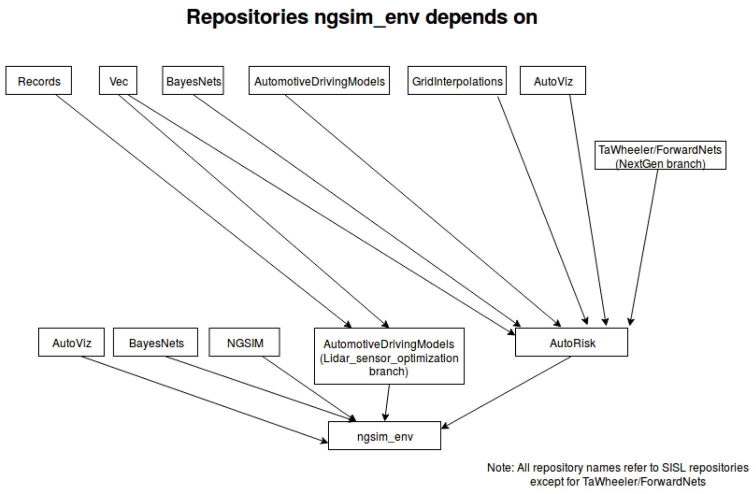


Fig. 2. Section of Lankershim Boulevard with vehicles superimposed. Rendered using the AutoViz package.

B. Environment

Any reinforcement learning simulation requires an environment for the agent to interact with. Autonomous driving environments are very complex, and coding one from scratch would have probably required more than a quarter. Instead, we have decided to use *NGSIM Env* [6], developed by the SISL to carry out our experiments. This environment is specifically designed to enable the training and testing of imitation learning algorithms using NGSIM datasets. It takes care of propagating the environment - ego vehicle + surrounding vehicles - and computing the reward at every time step. The agents' reward is taken to be zero unless it is involved in a collision or drives off-road, in which cases the reward is set to a largely negative value.


Repositories ngsim_env depends on
Records
Vec
BayesNets
AutomotiveDrivingModels
GridInterpolations
AutoViz
TaWheeler/ForwardNets (NextGen branch)
AutoViz
BayesNets
NGSIM
AutomotiveDrivingModels (Lidar_sensor_optimization branch)
AutoRisk
ngsim_env
Note: All repository names refer to SISL repositories except for TaWheeler/ForwardNets
Fig. 3. Repositories *NGSIM env* depends on

Ideally, with the environment already coded, we would only need to focus on the interesting part of the problem: implementing and tuning different architectures for our learning model. However it appeared that most of the code in NGSIM Env was coded in Julia 0.6, when most of the packages it depends on (see Fig.3) have been updated to Julia 0.7, resulting in compatibility issues. For that reason, we had to rewrite a substantial portion of NGSIM Env's code to make it compatible with Julia 0.7. This modification took several full days of coding.

C. Representing a policy

Most recent papers in the literature represent a policy as a neural network that takes as input the current state of the vehicle - and a summary of previous states in recurrent representations - and outputs an action, or at least a distribution from which to sample an action. In [2], two main types of neural networks (NN) are explored : feedforward and recurrent ones, with a common architecture of five feedforward layers that decrease in size from 256 to 32 neurons. The recurrent NN (RNN) adds a layer of 32 gated recurrent unit (GRU) on top of that, while the FFNN adds another feedforward layer.

For this class' part of the project, we would first like to follow the idea of a recurrent network, which seems particularly fit for a driving context, and experiment on different architectures. Ideally, we hope to find more simple and computationally efficient architectures that still achieve similar performances.

In a second time, we would like to explore alternate architectures such as representing a policy by a CNN taking as input a 2D array of the concatenated N last states. This approach is usually used in time series classification but rarely used in the driving context.

D. Training a policy

Two main types of training have currently been identified in imitation learning : behavioral cloning and reinforcement learning.

In Behavioral Cloning (BC), we basically try to find the policy $\pi_\theta(a|s)$ that best matches human behavior by solving a regression problem. This approach seems to exhibit a very limited generalization capacity and faces the well identified problem of cascading errors [7]. Hence, this approach doesn't seem very appropriate in urban situations where the dataset will never be able to cover the full space of very erratic and nuanced states and actions.

Reinforcement learning (RL) approaches, however, generally do better at dealing with the cascading error problem, by maximizing a global expected return over the whole trajectory, rather than taking local actions to solve immediate problems, but may lead to bigger ones down the road. More specifically, in this approach, we assume the driver follows an expect policy π_E whose actions maximize an expected global return :

$$R(\pi, r) = \mathbb{E}[\sum_{t=0}^{T} \gamma^t r(s_t, a_t)] \qquad (1)$$

Where $r(s_t, a_t)$ is a reward function which we know nothing about.

In the CS 230 part of the project, we will mainly investigate the BC approach and try to find the best way to map states directly to actions. We will also try to incorporate a latent variable model to our policy model, to try to map the raw states to more meaningful representations that may capture the similarity between situations. The CS 236 part of our project will mainly investigate the RL approach, and more specifically the Generative Adversarial Imitation Learning (GAIL) framework.

We will also compare our results against a simple baseline consisting of a static Gaussian Model, that is to say an unchanging Gaussian distribution $\pi(a|s) = \mathcal{N}(a|\mu, \Sigma)$ fit using maximum likelihood.

IV. Evaluation of models

Several metrics have been proposed in the literature. For now, we decided to use two of them to evaluate the relative performance of each model :

- *Root-Weighted Square Error (RWSE)* : Captures the deviation of a trajectory generated by our policy from the true trajectory – as given by data – in the same conditions. This metric should yield plots on the RWSE of position, speed, and center lane offset, as a function of time. More precisely, imagine you have i expert trajectories. For each expert trajctory, you sample n trajectories using the same initial state (basically, you try to replicate you true i^{th} trajectory). Then, the RWSE for variable v at time t of the different trajctories will be computed as follows :

$$RWSE(t) = \sqrt{\frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}(v_t^{(i)} - \hat{v_t}^{(i,j)})^2} \qquad (2)$$

- *KL divergence* : The KL divergence will be used here to measure the distance between the real-world distribution and the generated distribution over several quantities like speed or acceleration. Concretely, to do so, we will sample turn rates and accelerations from our model, and form a piece uniform distribution using B equally spaced bins. We will do the same for the data distribution. This will enable us to compute the KL divergence in closed form (assuming we have taken the same bins for the two distributions) between the two distributions :

$$D_{KL}(p_{data}||p_{model}) = \sum_{b=1}^{B} p_{b,data} log(\frac{p_{b,data}}{p_{b,model}}) \qquad (3)$$

V. Future work

So far, we have done the most tedious part of the project, which consisted in processing the dataset, and debugging/setting up the simulation environment. We hope to have our first baseline model work and to obtain our first results before Thanksgiving's break. From then on, we expect we'll be able to quickly iterate over different policy architectures and come up with a selection of the best performing ones.

VI. Contributions

So far, Malik Boudiaf focused mainly on updating, adapting and testing the code in NGSIM Env, while Ianis Bougdal-Lambert focused primarily on adapting the code in NGSIM and generating the Lankershim roadway and trajectories.

References

[1] Mykel J. Kochenderfer
Decision Making Under Uncertainty Theory and Application
MIT Lincoln Laboratory Series , 2015.

[2] A. Kuefler, J. Morton, T. Wheeler, and M. Kochenderfer
Imitating Driver Behavior with Generative Adversarial Networks
2017 IEEE Intelligent Vehicles Symposium (IV)

[3] J. Ho and S. Ermon
Generative adversarial imitation learning
Advances in Neural Information Processing Systems (NIPS), 2016

[4] *Next Generation Simulation (NGSIM)*, US Department of Transportation, Federal Highway Administration
https://ops.fhwa.dot.gov/trafficanalysistools/ngsim.htm

[5] *NGSIM.jl*, Stanford Intelligent Systems Laboratory
https://github.com/sisl/NGSIM.jl

[6] *NGSIM Env*, Stanford Intelligent Systems Laboratory
https://github.com/sisl/ngsim$_e nv$

[7] S. Ross and J. Bagnell
Efficient reductions for imitation learning
International Conference on Artificial Intelligence and Statistics, 2010.