Driving in SMARTS with Imitating Learning

Mingyang Jiang
Shanghai Jiao Tong University
jiamiya@sjtu.edu.cn

Abstract

In this paper, I apply imitating learning algorithm on the task of modeling human highway driving behavior in traffic simulator, SMARTS. I first apply GAIL algorithm in this task and then make some improvements about implementing details. I refine my data preprocessing and use a pre-training mode for GAIL. The results on benchmark show that these improvements work well for this task.

1 Introduction

Traffic simulators provide a platform for reinforce learning. With the environment model and designed reward, we can train agent in simulator instead of with raw data from real world. This introduces a problem of designing the reward. In many cases the reward is implicit, which challenges many reinforce learning algorithm when implementing. Imitating learning is a method that does not rely on designed reward. Instead of using reward as a supervised signal, imitating learning aims to train a policy behave like expert.

In this paper, I apply imitating learning algorithm to the task of modeling human highway driving behavior. The work is done on the NGSIM data-set in SMARTS simulator. A clever agent should be trained to control a car drive safely on road. My main works are as follows: First, I apply behavior clone and GAIL algorithm to the simulator. Second, I apply a feature extraction approach and other data preprocessing. Finally, I proposed a behavior clone pretraining mode and adapt it for GAIL training.

The rest of this paper are organized as follows: In the next section, I will introduce the environment and related imitating learning algorithm. Section 3 introduces the details of data prepocessing and algorithm application. The experiments and results are in section 4. In the last section I will conclude my work and propose some further works to make improvements.

2 Related Work

In this paper I use SMARTS simulator with NGSIM data-set. The algorithm applied is based on behavior clone and GAIL.

2.1 Environment

SMARTS(Scalable Multi-Agent RL Training School)[3] is an autonomous driving platform for reinforcement learning research. It simulates the road condition of the real world. In this simulator, cars can be controlled through velocity acceleration and angular speed. A well-trained agent should avoid to collide with other cars and drive smoothly on the road without incidents.

NGSIM(Next Generator SIMulation) is a data-set of car trajectories sampled from a highway in real world. NGSIM180 includes about 2000 trajectories, each about 300 meters long. In simulator, agent car needs to survive between other expert cars that follow the trajectories.

SJTU AI3601 Reinforcement Learning 2021 Final Project.

SMARTS environment returns a reward when agent takes action. However, the reward is just the distance travelled in last step. This reward is not well designed. Both intuition and practice tell that agent trained with the reward usually collides with the front car because it simply obtains high reward through high speed in a step. As a result, I use the provided expert trajectories instead of the reward to train my agent with imitating learning.

2.2 Algorithm

Behavior clone aims to train the agent to take the same action as expert in the same state. For the SMARTS environment, we can train the policy through maximize the likelihood of expert action in agent's action distribution, or directly minimize the distance of action distribution between agent and expert. Behavior clone is an off-policy algorithm, which means a easier training mode and convergence when implementing. However, the main problem of behavior clone is distributional shift. This is because agent cannot clone the action of expert exactly and will finally behave poor in an unseen state.

GAIL(Generative Adversarial Imitation Learning)[1] is another imitating algorithm. The main idea is to train a discriminator to distinguish the expert data and agent data. The discriminator outputs a probability that a data is from expert, and this probability is used as a reward to train an action generator using reinforce learning. A usually used algorithm is PPO(Proximal Policy Optimization)[2]. GAIL has been verified to perform better than behavior clone and some other imitating algorithm in experiments. The performance is highly depended on how appropriate the discriminator is trained. Besides, GAIL trains on policy, which takes more time to interact with simulator, thus slower to converge.

3 Methodology

In this section, I first apply GAIL in the task, but fail to train a clever enough agent. Then I make some improvements about data preprocessing, including feature extraction from the environment observation and extra process on expert actions. These two improvements show better performance when training agent using behavior clone. Finally, I apply the BC-pretrained agent to GAIL, and propose different training modes for policy, value and discriminator networks.

3.1 GAIL

The main process of GAIL is as follows: first, I create a expert data pool using expert trajectories. Each data in the pool is a tuple of expert state, which is transformed from observation in environment, and the expert action. The details of the transformation are in next subsection. For each iteration, first sample expert data batch from expert data pool, while agent samples some trajectories to generate agent data batch of the same size and format. Then discriminator gives each data from two batches a probability that how likely a data is from expert batch. I call this probability "Probability" for simplification in the later section. Finally, use the probability as reward to train the agent with PPO step.

An important part of GAIL is the discriminator, which gives the probabilities to train the agent. I hope to train an accurate enough discriminator to distinguish data between expert and agent and give the right probabilities. However, a too strong discriminator can cause the agent never receive good reward and hard to learn. In practice, I find my agent can learn to drive farther at the beginning, but soon collapses and the agent probabilities decline to 0. Finally the agent can drive about 100 meters on average and cannot reach farther.

3.2 Improvements on Data Preprocessing

In this section, I make improvements about feature extraction from observation and expert action processing.

3.2.1 Feature Extraction

In last section, I simply concatenate the information from environment observation to create a vector as state. The information includes ego position, speed, acceleration, heading, and these information

of neighboring vehicles. This extraction is not appropriate. I argue that three properties are considered for constructing state vectors:

1. Effectiveness: feature should be useful for drivers. Acceleration of other vehicles, for example, draws little attention when human are driving, and is not an effective feature.

2. Comprehensiveness: information in a state should be enough for a driver to take safe action. Actually, driver needs to know where the road is, so the map information should be included in the feature.

3. Close distribution for expert states and agent states. I hope the discriminator not to distinguish the expert data just using state, but also with action. Some features, such as absolute position of ego car, have different distribution between expert and agent data. Agent reaches larger position-x value harder, which is because agent often terminates halfway when training. Although this difference cannot be eliminated, an intuition is that the discriminator can have more attention on actions if states alone show less difference to discrimination.

Following the properties above, I designed my feature as follows:

1. Lane-view distance: the relative distance of nearest front car and back car in ego lane, left lane and right lane.

2. Lane-view relative speed: the relative speed of nearest front car and back car.

3. ego-lane heading: the orientation of ego lane, corrected by the ego car offset to the the center line of the road. This is computed using the map file in NGSIM.

4. ego-car heading and velocity.

An additional experiment is done to show how feature extraction can influence discriminator. Figure 3 and figure 4 shows the probability given by discriminator only using states. The gap between agent and expert probability is narrowed after using the feature extraction above. This difference cannot be avoid completely. It is unavoidable for agent to be in unseen states that is out of distribution of expert states. However, the appearance of these states will finally decrease to low frequency if agent learns to take right actions in previous states. This can be realized through time-difference update when reinforce learning.

3.2.2 Expert Action Processing

I notices that the expert action is calculated through the two neighboring states, which can cause huge error when the movement of one step is small. I found car with angular speed of more than $30rad/s^2$, which means the car can turn around in a frame. This is impossible, and to deal with it some processes are utilized.

First I replace the extreme angular speed value with the lane heading. Then I filter the expert actions in each trajectory with mean-filter to make action change smoothly. Smoothly changing actions are better for deep network to train, but cause lags in response for car in case of emergency. In that case, I replace the expert acceleration with the original braking deceleration.

After action processing, BC-trained agent can drive about 150 meters.

3.3 GAIL with pre-trained agent

I use the BC-trained policy network as an initialization for GAIL training. This approach combines the benefits of easier convergence from behavior clone and higher limit from GAIL. However, this combination is not that straightforward. Only policy network parameters are trained, the value and discriminator network are initialized randomly. Besides, after BC training, the variance of distribution from policy output can be too small to adapt to later GAIL training.

To deal with the above problems, I make 4 adaptions for pre-trained policy network when apply GAIL: 1. Not to train the policy network at the beginning epochs. 2. Retrain the variance of distribution in policy network output. 3. A self-adapting update strategy for discriminator. When discriminator is too strong that agent probabilities are too low, stop to update the discriminator. Start to update again when average agent probabilities is higher than a threshold. 4. Larger learning rate for value network and smaller learning rate for policy network.

These adaptions are used in the transition from BC-pretraining to GAIL. Otherwise the agent performance will soon go worse when training. After GAIL training, my agent can travel about 170 meters on average.

4 Experiments

I test my agent performance on the benchmark. In the benchmark, 388 cars are tested. Each time agent should control a car and take action based on the observation. An "reach goal" signal will be returned if one car drive through the area without incidents. And trajectories of cars are used to compute metrics of agent performance.

4.1 Metrics

The main metrics are listed as follows:

1. Average Frechet Distance: frechet distance evaluates the distance between agent trajectory and expert trajectory. If agent fails halfway, this distance is about the distance to the destination. When testing, takes average of the metric on all 388 testing cars.

2. Average Distance Travelled: distance travelled is the distance travelled before incidents happen. Incidents include collision, driving off road. If the car drive through the area without incidents, the distance travelled is about 308 meters.

3. Success rate: The ratio of cars that finish the travel without incidents.

Probabilities of discriminator are also used to evaluate how well the algorithm works, though there is no direct connection between the agent performance and agent probability.

4.2 Results

Experiments are done on 4 agents of different algorithm versions:

1. GAIL-V0: The original version of my GAIL implementation as a contrast.

2. BC-V0: The original version of my BC implementation as a contrast.

3. BC-V1: The BC with data preprocessing improvements, used as the pre-trained version for GAIL.

4. BC+GAIL: The GAIL with BC as pretraining.

Figure 1 shows the distance travelled of each car when training. The x-label "epoch" is reshape to the same size for 4 algorithms, since BC converges much faster than GAIL when training. I also filter the distance data for each curve, because the distance travelled has a large variance and the original curve fluctuates wildly. We can see a great improvement about 50 meters with data preprocessing. GAIL with pretraining performs about 20 meters better than BC-pretrain. Despite that the improvement is not remarkable, the result verifies that my adaption for pre-training is successful and agent can learning from GAIL when training. This can be also verified through figure 2. After a decline in agent probability, the agent soon learns to behave like expert gradually. The agent probability reaches its highest about 0.45 and fails to go higher.

I also test these agents for accurate metrics on the benchmark. Table 1 shows the results in detail. These results are based on all 388 testing cars, and can exclude the influence of randomness to some extend. The best performance is brought out by "BC+GAIL", which reaches an average distance travelled of more than 200 meters and declines the Average Frechet Distance to less than 100 meters. The success rate shows that 179 of 388 cars travelled safely through the area.

Noticed that the highest score is 30 meters more than that from figure 1, this is because I use 2 action clipping operations when testing: 1. if velocity<0 and speed acceleration<0: acceleration=0. 2. if front car distance<10 and acceleration>0: acceleration=-1. These two clipping operations aim to correct the agent action when obvious mistakes are made. I add these operation because I found one of the testing car travels a distance of -10 meters. An explanation of this phenomenon is I use more relative information when constructing state features, making the agent hard to pay enough attention to absolute speed.

Agent	Average Frechet Distance	Average Distance Travelled	Success rate
GAIL-V0	170.52	131.34	8.51%
BC-V0	173.16	128.62	7.73%
BC-V1	148.51	154.54	21.91%
BC+GAIL	96.37	208.59	46.13%

Table 1: the results on benchmark for 4 implementing versions.

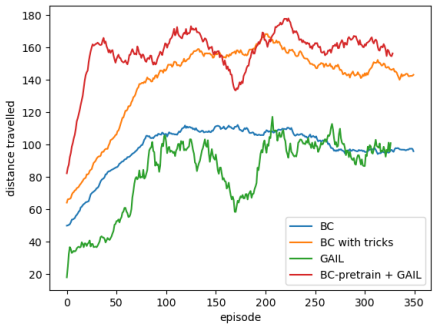

Figure 1: distance travelled on average when training.

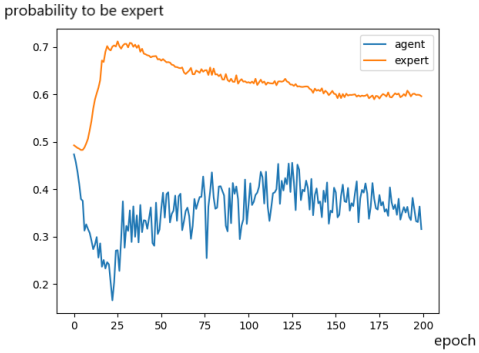

Figure 2: the probabilities given by discriminator when training GAIL.

5 Conclusion and Discussion

In this paper, I apply GAIL to the task of modeling human driving behavior in SMARTS simulator. Since the first implementation is not ideal, I make several improvements on data preprocessing. These improvements work well for behavior cloning. The trained network parameters are used to initialize policy in GAIL. One main problem during the transition from BC to GAIL is untrained networks in GAIL can misguide policy perform worse at the beginning and fail to reach its limits. I then propose an implementation approach to make sure the transition goes well. Finally I test my agent on benchmark and get a relatively good performance.

To make a further improvement, I believe Multi-Agent-GAIL could help agent learn better in this task. I notice that the expert cars in the simulator just drive following replayed trajectories and do not avoid collision with my agent car, even if my car drives reasonably. This is because other cars do not have perception in the simulator. I believe the cooperation between neighboring vehicles can help to keep a safe distance for every car.

As is written in last section, the direct implementation of GAIL is not satisfying enough. This can be explained by the difficulty of training discriminator at an exact intensity. An ideal curve of discriminator probability will converge to 0.5 for both expert and agent data. But in practice I find my agent reach a probability of 0.45 at highest, which means agent do not learn all knowledge from expert. I have tried to make discriminator more stupid to let agent probability go higher, but in fact the agent can learn little. I believe approaches about balancing the discriminator and generator can be applied to this task.

Besides, due to being short of time, further experiments about how implementing details can influence the final results have not been brought out. For example, more ablation experiments such as how feature extraction alone can influence the time of convergence and the final performance are to carry out.

Acknowledgments

I thank TA.Zhengbang Zhu for helping with problems in algorithm implementation and answering my confusion with patience. I thank Ruokun He, Zhe cao, Xingyuan Chen and other peers for helpful idea in environment configuration and implementing details.

References

[1] Jonathan Ho Openai and Stefano Ermon. Generative adversarial imitation learning.

[2] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. 7 2017.

[3] Ming Zhou, Jun Luo, Julian Villella, Yaodong Yang, David Rusu, Jiayu Miao, Weinan Zhang, Montgomery Alban, Iman Fadakar, Zheng Chen, Aurora C Huang, Ying Wen, Kimia Hassanzadeh, Daniel Graves, Dong Chen, Zhengbang Zhu, Nhat Nguyen, Mohamed Elsayed, Kun Shao, Sanjeevan Ahilan, Baokuan Zhang, Jiannan Wu, Zhengang Fu, Kasra Rezaee, Peyman Yadmellat, Mohsen Rohani, Nicolas Perez Nieves, Yihan Ni, Seyedershad Banijamali, Alexander I Cowen-Rivers, Zheng Tian, Daniel Palenicek, Haitham bou Ammar, Hongbo Zhang, Wulong Liu, Jianye Hao, and Jun Wang. Smarts: Scalable multi-agent reinforcement learning training school for autonomous driving.

A Appendix

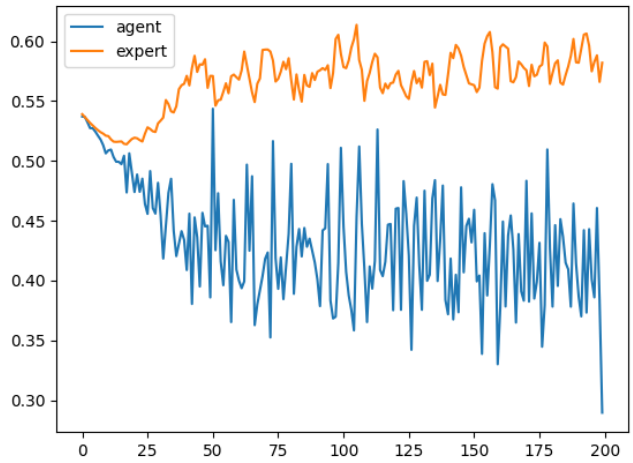

Figure 3: The probability given by the discriminator only using states. This figure is based on the feature of GAIL-v0 when improvements about feature extraction have not been made. Expert and agent data can be easily distinguished.

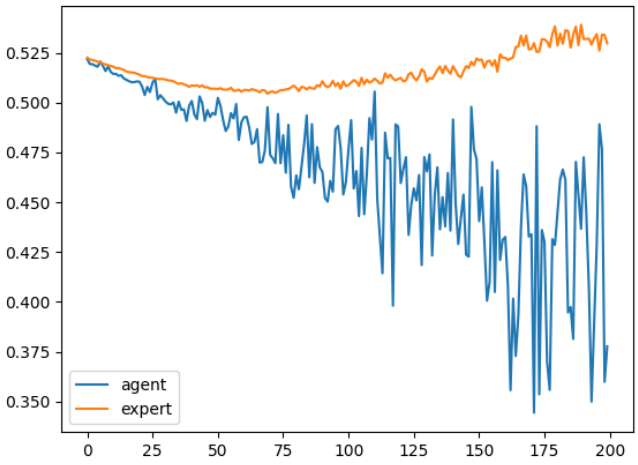

Figure 4: The probability given by the discriminator only using states. This figure is based on the feature of BC-V1 with improvements. We can have a view that expert states cannot be distinguish easily because the expert probability is only slightly higher than 0.5. Agent states still get relatively low probability since agent can step into out-of-distribution states in training.