Multi-Agent GAIL
Implement, Exploration and Further Thoughts

JamesZhutheThird
Shanghai Jiao Tong University
JamesZhutheThird@sjtu.edu.cn

Abstract

Multi-Agent Generative Adversarial Imitation Learning (MAGAIL) can be quite helpful in complex traffic environments. In this project, I completed the code implementation based on PSGAIL, and the visualization results showed good performance. Then I analyzed the experimental results and updated the model, which envisions demonstrated ideal results. Finally, I discuss the limitations of the current project and the future direction of possible improvement.

1 Introduction

Dataset Next Generation Simulation (NGSIM)[6] is a data collection project initiated by the Federal Highway Administration. Researchers widely use it to study driving behaviours such as turning and changing lanes, traffic flow analysis, microscopic traffic model construction, vehicle motion trajectory prediction, driver's intention recognition, autonomous driving decision planning, etc.

I select the I80 road subset of the NGSIM dataset for this multi-agent simulation learning task. The road includes fast lanes, slow lanes and on-ramp, which is highly representative and suitable for rather basic autonomous driving tasks.

Platform The entire project is built and visualized based on Huawei's intelligent car reinforcement learning platform SMARTS. SMARTS (Scalable Multi-Agent RL Training School)[7] is a simulation platform for reinforcement learning and multi-agent research on autonomous driving, focusing on realistic and diverse interactions.

Task This project controls one or more vehicles to simulate driving in the road environment recorded in the dataset. The task is to learn a better strategy that makes the agent vehicle drive as stable and reliable as possible, i.e. driving distance is longer without rushing out of the road and rear-ends collision, or the phenomenon of being rear-ended.

In the case of **single-agent imitation learning**, one specific car is selected as the ego-vehicle, and the strategy controls each round of interaction. The trajectory playback from the dataset is applied to the rest of the cars. In the case of **multi-agent imitation learning**, multiple vehicles are selected as ego-vehicles. During each round of interaction, the ego-vehicle is selected in turn and controlled by the strategy. After all the strategy-controlled ego-vehicles are executed, the trajectory playback is applied to the rest.

2 Related Work

GAIL Imitation learning, that is, learning from the trajectory of experts. This kind of learning is usually offline and does not require interaction with experts. For this task, there are two mainstream solutions. The first is Behavioral Cloning, which is supervised learning; the second is Inverse RL, assuming that the expert data is the optimal solution of the reward function of the environment, and solving the cost function. The former is a compounding error caused by the covariate shift problem; that is, it only learns a single-step decision, and the accumulation will cause a significant deviation, so

it can only be helpful when the amount of data is significant. Since the latter learns the cost function, it will not produce covariate shift, but the complexity is too high, so it is better to learn the action strategy directly. Therefore, Generative Adversarial Imitation Learning(GAIL)[3] adopts the idea of Generative Adversarial Networks, using the strategy as the generator in GAN. The goal is to make the distance between the action sampled by the strategy, and the expert data is as close as possible.

MAGAIL The corresponding multi-agent version of GAIL is MAGAIL[5]. The generator controls the behaviour of all agents in a distributed way, and the discriminator judges the similarity of each agent behaviour to the corresponding expert behaviour. The authors proposed three types of MAGAIL algorithms with different priors on the reward structure-centralized, decentralized and zero-sum. The centralized case assumes that the agents share the same discriminator and fully cooperate. The zero-sum case assumes that the agents are fully competitive, and the discriminator aims to maximize one agent while minimizing another. Furthermore, the decentralized case is the mixed strategy in which each agent has a discriminator, while they are not independently learning as they interact indirectly via the environment.

PSGAIL Based on the centralized case, i.e. all agents share the same reward function, and in a non-competitive relationship, the model derived by MAGAIL on the autonomous driving simulation task is PSGAIL[1], which is based on PS-TRPO (Parameter Sharing Trust region policy optimization).[4, 2] The idea is that all agents share the same network and the same policy to learn together.

The model training process is shown in the figure1. The environment passes the observations to the policy network to get the actions of each agent. The discriminator compares the agent's action with the expert's trajectory to obtain the reward function, which is passed into the value network and updated the strategy network. At the same time, the agent's action will affect the observation of the environment during the next sampling.

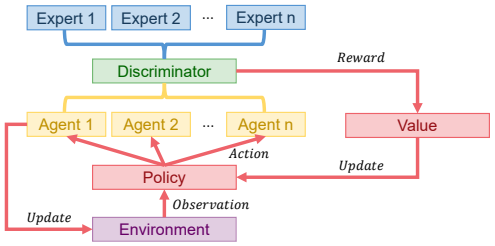


Figure 1: The Structure Demo of Our Model

3 Experiments

3.1 Environment Observations

Thanks to the diversity of the NGSIM dataset, we can collect a wealth of information from the environment. We select 22-dim features from the agent, with 8-dim information eight surrounding neighbours (We divide the agent's surroundings into eight sectors and pick the nearest vehicles.) each. The details of features are shown in table1.

	ego-vehicle	neighbor-vehicle
common	bounding box size(2), heading(1),speed(1)	
special	location(2), lane(4), linear/angular velocity /acceleration/jerk(12)	relative location(2),lane(2)
update	lane(-2)	collision distance(1), ttc(1), ttc with acc(1), lane(-1)

Table 1: The Selection of Observation Features

3.2 Training Configurations

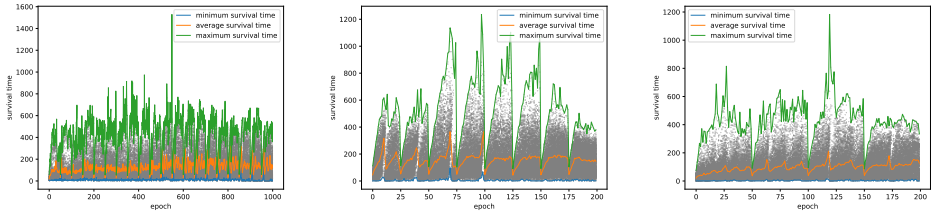
The training is divided into two stages. In the pre-training stage, a smaller number of agents (starting from 2) and a smaller batch size (2000) are used for 1000 iterations, and the number of agents is continuously increased (add 2 more for each 50 epochs). The number of agents is small to ensure that there are as many expert vehicles around the agents at the initial period, and the learning efficiency

is improved. Fast iteration in small batches can effectively speed up pre-training and reach faster convergence.

In the tuning stage, more agents (starting from 100, add 50 more for each 50 epochs) and a larger batch size (10000) are used for 200 iterations. Big batches are used for smoother convergence. More agents are used in this stage to simulate more complex scenarios. This makes it possible to make good decisions even when the roads are all agents.

The total training time on an 8-core 16-thread AMD 3800X CPU is approximately 8 hours.

3.3 Results



(a) pre-train stage (b) fine-tune stage (c) direct fine-tune stage

Figure 2: The Survival Time of Agents (Each time step equals 0.1s)

The result shows that the agents can survive for around 18 seconds, and converge quickly after each increase in the number of agents.

We also compare the results of the direct finetune stage shown in subfigure 3. It is found that the average survival time of direct mass-agent training is not as much as using the pre-trained model. The latter can reach the state of convergence faster, while for direct tuning, more epochs are required to achieve the same effect. This result shows that using the pre-trained model can effectively deal with changing road conditions.

4 Explorations

4.1 Discussion

Combining visualization and survival time distribution, it is found that there is a severe polarization; that is, a large part of the vehicle has a shorter survival time (less than 1 second), but a few vehicles spend more than a few minutes on a 300-meter-long highway. This is not an ideal situation. The specific problems and reasons are analyzed as follows

Polarized Survival Time The long-survived agent adopts a conservative strategy, slows down the vehicle speed, and even stops directly at the ramp with few vehicles. Although this can avoid voluntarily crashing into other vehicles, it will inevitably cause the car behind to rear-end. In order to pass the road section as quickly as possible, agents with a short survival time often adopt a more aggressive strategy, and they cannot estimate whether the lane gap is enough to pass the overtaking.

These are because the insufficient observed features of neighbours are obtained without processing and extraction, and it is challenging to learn the critical information to avoid collisions directly.

Gathered Vehicles Although the convergence is fast, it is more because multiple vehicles gather at the same time to protect the internal vehicles by sacrificing the peripheral vehicles. The external vehicles collide quickly, so that the internal vehicles obtain a larger safety zone. It is difficult for the internal and external vehicles to learn more from experts.

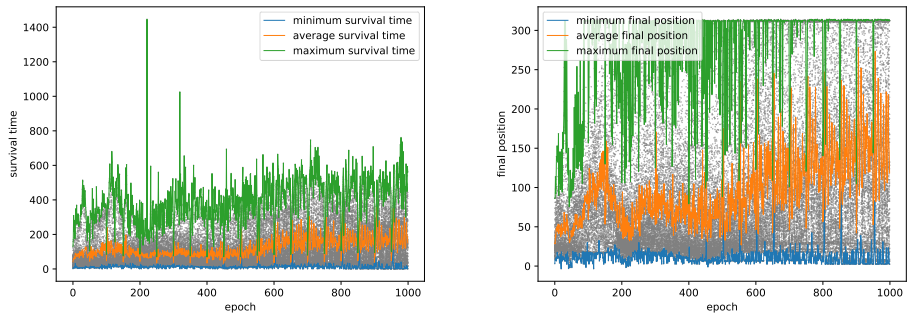
Furthermore, the agent is smooth sailing most of the time, failing to efficiently use the short critical time before the collision for learning.

4.2 Improvement

In order to solve the problems, I added multiple features to the observation, including the closest distance to the surrounding vehicles, collision probability and time with and without the influence of acceleration, and remove the redundant lane information, and the updated observation feature dimension is $20 + 8 \times 10 = 100$ (see table 1).

At the same time, I use the code provided by TA to randomly select vehicles on the road and take over by agents to simulate the actual application scenario of the automatic cruise when driving on the highway.

4.3 Results



(a) The Survival Time of Agents (b) The Final Position of Agents

Figure 3: The Results of Updated Method

The results show that the performance of the updated model has some improvement. The average survival time has increased to approximately 200 time-steps. Vehicles can survive for longer distances, and more vehicles can reach the 300-meter finish line.

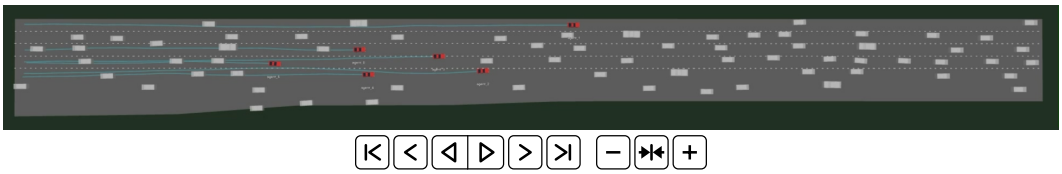


Figure 4: The Envision of The Model(5x Speed Up)

5 Further Thoughts

Despite the expected results, the model still has many shortcomings. The analysis and thinking are as follows.

Lack of diversity in dataset The provided dataset only selects small cars, and as a highway, the vehicle condition is relatively simple. Due to the lack of turns, intersections and roundabouts, the road conditions are also simple. Therefore, it is difficult for the agent to learn how and when to pull over, change lanes, make a U-turn, etc.

Equally Treated Observations The neighbour vehicle selection algorithm provided by the SMARTS platform cannot handle with the situation where the vehicle density of the left and right lanes is different. The subsequent work may consider the dynamic selection of the neighbour vehicles that are the most threatening to the ego-vehicle based on the expert vehicle's speed, distance, and intention. For example, ignore the slow-moving vehicles behind and focus on the vehicles that are changing lanes and accelerating on both sides.

Separate Control v.s. Joint Control As a shared strategy model, PSGAIL has the advantages of adjustable agent number and fast training speed, but this is based on the premise of independent but identical decision-making by each vehicle. The ideal application scenario for multi-agents is a kind of self-driving travel in a fleet and group. We hope that multiple vehicles can make joint decisions to avoid long-distance leading or leaving behind. One specific implementation method is to use the same discriminator when pre-training the model. In the finetuning stage of a fixed number of agents, each agent has its copy of its parameters, and subsequent discriminator training is completed by each agent independently. At the same time, add the relationship observation between agents. The agent must not only meet its own survival time as long as possible and ensure that it is at an appropriate distance from other agents and cares about each other as much as possible.

Heading Weirdness of Envision The SMARTS envision platform seems to use the direction of the coordinate displacement of the two sampling points as the heading of a vehicle. So when the vehicle

is stationary, it may look like rotating due to minor disturbance. Fortunately, the official staff quickly responded to my issue[Github issue 1214] and will add smoothing processing to avoid noise in the upcoming updates.

6 Conclusions

In this project, I implemented the MAGAIL model based on PSGAIL and improved the model based on the analysis of the envisions. This multi-agent algorithm based on a unified policy and discriminator can be effectively applied to simple road conditions such as expressways with a single goal. However, PSGAIL cannot model the relationship between agents, and therefore cannot make joint decisions. In fact, it is just a simple combination of multiple agents and GAILs. In future research, it is hoped that the pre-training and finetuning stages can be distinguished. The former trains a unified model, and the latter is individually tuned for each agent, to finally realize the combination of independent control and joint decision-making. At the same time, build more complex models and observations to deal with more challenging datasets and environments.

Acknowledgments

Special thanks to Prof. Weinan Zhang and T.A. Zhengbang Zhu for their support and help in the past two months, and extra thanks to Zhe Cao, Jiahang Cao, Shenyu Zhang and Ziyuan Li for discussions over implementation.

References

[1] Raunak P. Bhattacharyya, Derek J. Phillips, Blake Wulfe, Jeremy Morton, Alex Kuefler, and Mykel J. Kochenderfer. Multi-agent imitation learning for driving simulation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2018, Madrid, Spain, October 1-5, 2018*, pages 1534–1539. IEEE, 2018.

[2] Jayesh K. Gupta, Maxim Egorov, and Mykel Kochenderfer. Cooperative multi-agent control using deep reinforcement learning. In Gita Sukthankar and Juan A. Rodriguez-Aguilar, editors, *Autonomous Agents and Multiagent Systems*, pages 66–83, Cham, 2017. Springer International Publishing.

[3] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 4565–4573, 2016.

[4] John Schulman, Sergey Levine, Pieter Abbeel, Michael I. Jordan, and Philipp Moritz. Trust region policy optimization. In Francis R. Bach and David M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 1889–1897. JMLR.org, 2015.

[5] Jiaming Song, Hongyu Ren, Dorsa Sadigh, and Stefano Ermon. Multi-agent generative adversarial imitation learning. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 7472–7483, 2018.

[6] USDOT. Next generation simulation (ngsim) vehicle trajectories and supporting data. [Online; accessed 1-1-2022], 2016.

[7] Ming Zhou, Jun Luo, Julian Villella, Yaodong Yang, David Rusu, Jiayu Miao, Weinan Zhang, Montgomery Alban, Iman Fadakar, Zheng Chen, Aurora Chongxi 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 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, 11 2020.

A Appendix

There are three main python programs. You can simply run each program to get the final envision in your browser.

- `prepare_all_in_one.py` generates experts trajectory and observation features in `experts.npy`
- `train_all_in_one.py` runs pre-train stage and finetune stage by default (setting `train_flag = True`). And checkpoints, results and logs are saved in `./output/*` folder. You can load checkpoint and continue training by setting `continue_flag = True`, or you may just want to plot results by setting `plot_flag = True`.
- `eval_all_in_one.py` uses the model trained before to run envision scripts, open `localhost:8081/` in your browser to check the envision.

More demonstrations can be found in `MAGAIL.pptx` file and the codes can be found in [Github].

Thanks for reading this paper and have a nice day `:)`.