CS230

Diverse Challenging Driving Tests

Farid Soroush
Department of Mechanical Engineering
Stanford University
fsoroush@stanford.edu

Abstract

Autonomous vehicles industry has been growing so fast in the past few years. Despite this progress, there still exists a lack of robust tests for driving platforms. On the other hand, real-world testing not only is expensive and dangerous for public, but also due to the rare nature of dangerous scenarios, will require billions of miles in order to statistically validate performance claims. In this paper, first we implement Determinantal Point Processes (DPPs) to create the most diverse and challenging driving scenarios to minimize required testing time of autonomous cars. Second, since for high number of scenarios, real human study is expensive and time consuming, we provide a model of human behaviour to reduce testing cost and time. We train a non-linear and high dimensional stochastic human driving model using GAIL algorithm to validate performance of our sampling results using experiments. Using the trained model in the experiments, we show that DPPs sampling generates more diverse and challenging scenarios than naive Monte Carlo.

1 Introduction and Related work

In the past few years, there has been a huge progress in autonomous cars industry. Testing AVs in real environments, which is the most straightforward validation framework for system behavior, requires prohibitive amounts of time due to the rare nature of challenging situation. [1]. In fact, under some scenarios AVs not only need to drive hundreds of billions of miles to create enough data to clearly demonstrate their safety [2], but also formally verifying AV algorithm's "correctness" is difficult [3, 4, 5, 6].

Even though researchers have developed some computational methods for autonomous vehicles driving, testing and verification [7, 8, 9, 10, 11], there exist few works that test the autonomous vehicle based on scenarios from the real world data. In this paper, we use determinantal point process (DPPs) to extract scenarios from a real world data set in order to evaluate autonomous vehicle platforms. DPPs are a class of repulsive point processes that were introduced to machine learning community by a tutorial paper [12]. They have been used in batch generation for mini-batch diversification [13], improving stochastic gradient descent [14] and a couple of other applications in the past few years [15, 16, 17, 18, 19, 20, 21]. Since DPPs samples provide the most diverse and challenging driving scenarios, they reduce the necessary computation time for performance evaluation of the autonomous system.

In order to validate our sampling results, we use a human driving model to drive a car in traffic. Unfortunately, most of the algorithms that have been provided for modelling human driving behaviour are linear and deterministic (whereas human behaviour is nonlinear and stochastic), which leads to low performance in predicting human actions. In contrast, we implement a nonlinear and high dimensional model to imitate human driving policy. Then, we use this model to test and evaluate our scenario sampling methods.

2 Dataset and Features

The data we use is from the Next-Generation Simulation (NGSIM) data set. NGSIM contains highway driving trajectories for US Highway 101 and Interstate 80, and consists of 45 minutes of driving at 10 Hz for each roadway. The US Highway 101 data set covers an area in Los Angeles approximately 640 m in length with five mainline lanes and a sixth auxiliary lane for highway entrance and exit. The Interstate 80 data set covers an area in the San Francisco Bay Area approximately 500 m in length with six mainline lanes, including a high-occupancy vehicle lane and an on-ramp.

Traffic density in the data set transitions from uncongested to full congestion and exhibits a high degree of vehicle interaction as vehicles merge on and off the highway and must navigate in congested flow. The diversity of driving conditions and the forced interaction of traffic participants makes these sources particularly useful for behavioral studies. The trajectories were projected to lanes using centerlines extracted from the NGSIM CAD files. Cars, trucks, buses, and motorcycles are in the data set, but only car trajectories were used for model training. The raw data set consists of 11,850,526 rows and 25 columns. Data was collected through a network of synchronized digital video cameras.

For each vehicle, 66 features are extracted and passed to the policy as the observation. The policy outputs longitudinal acceleration and turn rate values as the vehicle action. These values are used to propagate the vehicle forward in time. Features include 20 lidar distances, 20 lidar distance time derivatives, and several ego and leading vehicle parameters (e.g. relative offset, relative heading, velocity, vehicle length, vehicle width, lane curvature,longitudinal and lateral acceleration, jerk, local and global turn rate, local and global angular rate, time gap, distance to left road edge, distance to right road edge, acceleration of vehicle in front of fore vehicle).

3 Methods

3.1 Markov Decision Processes

An infinite horizon MDP is defined by the tuple (S, A, T, R, γ), where S is the state space, A is the action space, T is the transition model, R is the reward function and γ is the discount factor. A stochastic policy $\pi : S \rightarrow P(A)$ maps each state to the probability of taking each action. The objective in an MDP is to find a policy that maximizes the expected value of each search $V_\pi(s) = E_\pi[g_t|s_t = s]$. A POMDP is a kind of MDP in which the agent receives partial information about the state at each time step. POMDP adds an observation space O and observation model $Z : S \rightarrow P(O)$ to the problem.

3.2 Reinforcement Learning

Reinforcement learning (RL) assumes that drivers in the real world follow an expert policy π_E whose actions maximize the expected, global return

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

weighted by a discount factor $\gamma \in [0, 1)$. The local reward function $r(s_t, a_t)$ may be unknown, but fully characterizes expert behavior such that any policy optimizing $R(\pi, r)$ will perform indistinguishably from π_E.

The learned policy must be able to capture human driving behavior, which involves:

• Non-linearity in the desired mapping from states to actions (e.g., large corrections in steering to avoid collisions caused by small changes in the current state).

• High-dimensionality of the state representation, which must describe properties of the ego-vehicle, in addition to surrounding cars and road conditions.

• Stochasticity, because humans may take different actions each time they encounter a given traffic scene.

To address the first and second points, we represent all learned policies π_θ using neural networks. To address the third point, we interpret the network's real-valued outputs given input s_t as the mean μ_t of a Gaussian distribution. Actions are chosen by sampling $a_t \sim \pi_\theta(a_t|s_t)$.

3.3 Imitation Learning

Our goal is to imitate human behaviour to learn his policy in driving. We consider the input data as a sequence of state-action pairs that result from a policy. Although $r(s_t, a_t)$ is unknown, a surrogate reward $\tilde{r}(s_t, a_t)$ may be learned directly from data. Generative Adversarial Imitation Learning (GAIL) algorithm [22] tries to match a generator (G) with state occupancy distribution of the human policy. In GAIL, we have a discriminator D, which tries to distinguish the learned policy π_θ from the human policy π_H. GAIL objective function is

$$\min_{\theta} \max_{\psi} E_{\pi_E} log(D_\psi(s,a)) + E_{\pi_\theta} log(1 - D_\theta(s,a))$$

Generator (G) tries to fool discriminator (D) with state-actions that are not in the data set, but are very close to human behaviour. In order to fit π_θ, a surrogate reward function can be formulated from the previous equation as:

$$\tilde{r}(s_t, a_t|\psi) = -log(1 - D_\psi(s,a))$$

After sufficient training, the generator (G) learns to imitate the human policy.

3.4 Sampling

We use detenminantal point process for sampling scenarios. DPPs are a class of repulsive point process that sample diverse vectors from the data set while considering value of a score function for each vector. We also use Monte Carlo sampling in order to compare the results with DPPs.

3.5 Neural Network Architecture

We use recurrent neural network (RNN) policies, consisting of 64 Gated Recurrent Units (GRUs). The observation is passed directly into the RNN without any initial reduction in dimension. We use recurrent policies in order to address the partial observability of the state caused by occluded vehicles. Policy optimization is performed using an implementation of TRPO from rllab, with a step size of 0.01. For training, we had 1000 iterations with a discount of 0.95 and a batch size of 10000 observation-action pairs.

3.6 Metrics

3.6.1 Safety Metric

Minimum time to collision (TTC) is defined as the time it would take for two cars to intercept one another if they maintain the current heading and speed [23]. We use TTC as the safety measurement (lower TTC means more challenging scenario). The TTC between the ego-vehicle and vehicle i is defined as

$$TTC_i(t) = -\frac{r_t(t)}{\dot{r}_t(t)}$$

where $r_t(t)$ and $r_t(t)$ are distance between vehicles and its time derivative, respectively.

In this paper, vehicles are described as oriented rectangles in the 2D plane. Since we are interested in the time it would take for the ego-vehicle to intersect the polygonal boundary of another vehicle on the road, we utilize a finite set of range and range measurements in order to approximate the TTC metric. For a given configuration of vehicles, we compute N uniformly spaced angles $\theta_1, ..., \theta_N$in the range $[0, 2\pi]$ with respect to the ego vehicle's orientation and cast rays outward from the center of the ego vehicle (Figure 1).

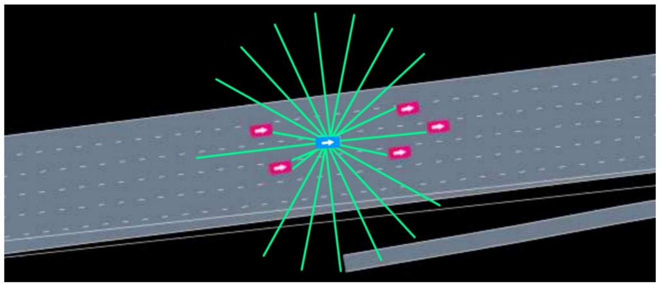


Figure 1: Depiction of lidar sensor input used for GAIL model

For each direction we compute the distance which a ray could travel before intersecting one of the M other vehicles in the environment. These form N range measurements $s_1, ..., s_N$. Further, for each ray s_i , we determine which vehicle (if any) that ray hit; projecting the relative velocity of this vehicle with respect to ego vehicle gives the range-rate measurement $\dot{s}_i$. Finally, we approximate the minimum TTC by:

$$f(X) := \min_{i=1,...,N} \frac{s_i(t)}{\dot{s}_i(t)}$$

3.6.2 Diversity Metric

We assume the Euclidean distance between data samples as the diversity metric. However, we note that our framework extends to other cases, as well. With the Euclidean distance assumption, we use a Gaussian kernel, which is known to be PSD, to directly construct the S-matrix:

$$K_{ij} = exp(-\frac{||y-y||_2^2}{2\sigma^2})$$

Diversity is measured based on determinant of K matrix.

3.6.3 Metric for Experiment Results

To quantify the undesirable traffic phenomena that arise out in challenging driving situations, average acceleration of the ego vehicle in the first 0.3 seconds of each scenario is calculated. Like TTC metric, lower acceleration (higher negative value) means more challenging scenario.

4 Experiments and Results

For sampling and generating scenarios from the data set, DPPs and Monte Carlo were used. TTC and diversity are calculated and for both methods. DPPs perform better than naive Monte Carlo, regardless of the sample size (Figure 2).

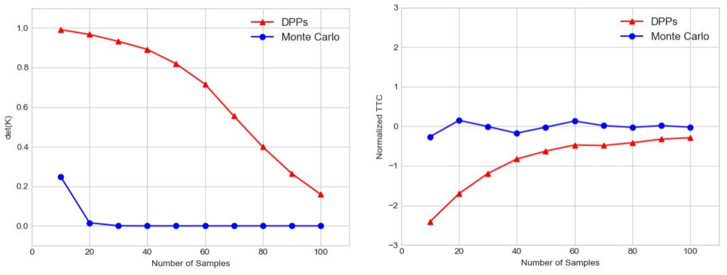
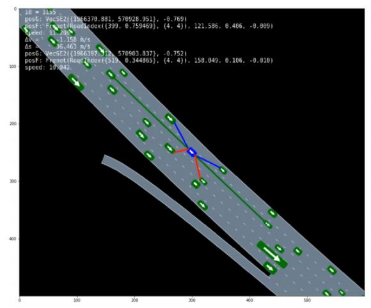


Figure 2: DPPs and Monte Carlo sampling results: (Left) det(K) of DPPs samples is higher (more diverse scenarios) (Right) TTC of DPPs samples is lower (more challenging scenarios)

A generator is trained and the learned weights for policy has been used to drive a car (Figure 3). The full video is available on the Github page of the project. The generator is used to compare results



```
id = 1155
posG: VecSE2({1966370.881, 570928.951}, -0.769)
posF: Frenet(RoadIndex({399, 0.759469}, {4, 4}), 121.586, 0.406, -0.009)
speed: 11.200
Δv =    -1.158 m/s
Δs =    36.463 m/s
posG: VecSE2({1966397.312, 570903.837}, -0.752)
posF: Frenet(RoadIndex({519, 0.344865}, {4, 4}), 158.049, 0.106, -0.010)
speed: 10.842
```

Figure 3: Driving a car using the learned policy (blue, green and red line are lidar measurements), the full video is available on the Github page of the project

of two sampling methods in the simulated environment. Samples from each method are fed to the environment and the learned weights were used to drive a car in each case. Average acceleration of the car in the first 0.3 second of driving is calculated for both methods. Hard-brake cases (high negative acceleration) occurred more in DPPs samples (Figure 4). Therefore, higher performance of DPPs in generating more challenging scenarios is validated using the trained generator (G) that imitates human behaviour.

Even though the current implementation of the human model is stochastic, better stochastic models can be built based on this model. Furthur details can be found in the future work section.

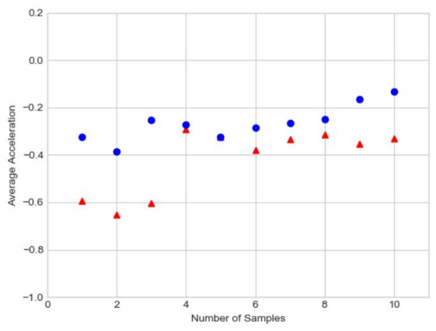


Figure 4: Normalized average acceleration in the first 0.3 seconds of scenario for different sample sizes: human model have higher negative acceleration (more challenging scenarios) in (Red Triangles) DPPs samples than (Blue Circles) Monte Carlo

5 Conclusion and Future Work

In this paper, we provided a novel method to extract near accident scenarios from a driving data set and tested performance of our algorithm. For the evaluating the samples, we trained a high dimensional and non-linear stochastic driving model based on imitation from human driving behaviour. Our work has direct application in testing of autonomous vehicle algorithms and evaluation of their performance in a fast, safe and robust manner.

For the future work, an ensemble of generators ($\xi^i, i = 1, ..., m$, where ξ^i is the learned weight of the i^{th} generator) can be trained and a distribution P_0 over ξ can be modeled using a (multivariate normal) parametric bootstrap. The models ξ^i are high dimensional ($\xi \in R^d, m < d$) and graphical lasso can be used to fit the inverse covariance matrix for the ensemble. In this way, a new stochastic model for generator can be obtained and the covariance matrix can be sampled using DPPs in order to get the most diverse driving behaviours.

6 Contributions

The author did not have any other team members.

Code

Github link for the code: https://github.com/FaridSoroush/Driving

References

[1] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. On a formal model of safe and scalable self-driving cars. *arXiv preprint arXiv:1708.06374*, 2017.

[2] N Kalra and SM Paddock. How many miles of driving would it take to demonstrate autonomous vehicle reliability. *Driving to Safety*, 2016.

[3] Matthias Althoff and John M Dolan. Online verification of automated road vehicles using reachability analysis. *IEEE Transactions on Robotics*, 30(4):903–918, 2014.

[4] David Parker. Verification of probabilistic real-time systems. *Proc. 2013 Real-time Systems Summer School (ETR'13)*, 2013.

[5] Sanjit A Seshia, Dorsa Sadigh, and S Shankar Sastry. Formal methods for semi-autonomous driving. In *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–5. IEEE, 2015.

[6] Matthew O'Kelly, Houssam Abbas, Sicun Gao, Shin'ichi Shiraishi, Shinpei Kato, and Rahul Mangharam. Apex: Autonomous vehicle plan verification and execution. 2016.

[7] 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)*, pages 1534–1539. IEEE, 2018.

[8] Li Li, Wu-Ling Huang, Yuehu Liu, Nan-Ning Zheng, and Fei-Yue Wang. Intelligence testing for autonomous vehicles: A new approach. *IEEE Transactions on Intelligent Vehicles*, 1(2):158–166, 2016.

[9] Matthew O'Kelly, Aman Sinha, Hongseok Namkoong, Russ Tedrake, and John C Duchi. Scalable end-to-end autonomous vehicle testing via rare-event simulation. In *Advances in Neural Information Processing Systems*, pages 9849–9860, 2018.

[10] Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. Deeptest: Automated testing of deep-neural-network-driven autonomous cars. In *Proceedings of the 40th international conference on software engineering*, pages 303–314. ACM, 2018.

[11] Dorsa Sadigh, S Shankar Sastry, and Sanjit A Seshia. Verifying robustness of human-aware autonomous cars. *IFAC-PapersOnLine*, 51(34):131–138, 2019.

[12] Alex Kulesza, Ben Taskar, et al. Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*, 5(2–3):123–286, 2012.

[13] Cheng Zhang, Hedvig Kjellstrom, and Stephan Mandt. Determinantal point processes for mini-batch diversification. *arXiv preprint arXiv:1705.00607*, 2017.

[14] Cheng Zhang, Cengiz Öztireli, Stephan Mandt, and Giampiero Salvi. Active mini-batch sampling using repulsive point processes. *arXiv preprint arXiv:1804.02772*, 2018.

[15] Alex Kulesza and Ben Taskar. k-dpps: Fixed-size determinantal point processes. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1193–1200, 2011.

[16] Alex Kulesza and Ben Taskar. Structured determinantal point processes. In *Advances in neural information processing systems*, pages 1171–1179, 2010.

[17] Jennifer Gillenwater, Alex Kulesza, and Ben Taskar. Discovering diverse and salient threads in document collections. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 710–720. Association for Computational Linguistics, 2012.

[18] Zelda E Mariet, Suvrit Sra, and Stefanie Jegelka. Exponentiated strongly rayleigh distributions. In *Advances in Neural Information Processing Systems*, pages 4464–4474, 2018.

[19] Pengtao Xie, Ruslan Salakhutdinov, Luntian Mou, and Eric P Xing. Deep determinantal point process for large-scale multi-label classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 473–482, 2017.

[20] Christophe Dupuy and Francis Bach. Learning determinantal point processes in sublinear time. *arXiv preprint arXiv:1610.05925*, 2016.

[21] Tarun Kathuria, Amit Deshpande, and Pushmeet Kohli. Batched gaussian process bandit optimization via determinantal point processes. In *Advances in Neural Information Processing Systems*, pages 4206–4214, 2016.

[22] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, pages 4565–4573, 2016.

[23] Katja Vogel. A comparison of headway and time to collision as safety indicators. *Accident analysis & prevention*, 35(3):427–433, 2003.