Continuous Inverse Optimal Control by Energy-based Model

Yifei Xu
Department of Statistics
University of California, Los Angeles
fei960922@ucla.edu

Abstract

Inverse optimal control solves the problem of infering the cost function of a state-control pair from large number of expert trajectories, with the objective to perform ideal control based on the explicit cost function learned. We introduce energy-based generative model to estimate the cost function, and use Langevin Dynamic, a Monte Carlo based sampling algorithm, to directly sample the full trajectory. Conventionally, the cost function is defined on a pair of state and control, with the trajectory cost being the sum of them. Whereas, our method is capable of learning a higher level cost function for the full trajectory. We refine the rule-based cost function with a convolutional neural network to obtain the NN-augmented cost function, which combines the advantages of both model-based and model-free learning. Our method achieves a better control in experiments on real autonomous driving data.

1 Introduction

Optimal control plays a major role in all kinds of robotic projects including autonomous driving. For self-driving, a state is represented by data extracted from multiple resources including camera, radar, lidar, GPS, etc. In order to find the optimal control on the agent, we construct a cost function $C_\theta(x, u)$ on state x and control u. An optimal control given the current state is then that with the lowest cost, which is typically defined as the cost of the future trajectory.

Thus, a key problem in the markov decision process (MDP) described above is the definition on the cost function. In previous literature, both model-based and model-free methods have been applied to the issue, while we are seeking for a balance between flexibility and interpretability.

In early research and most industrial models, the cost function is hand-crafted through human experience. However, this approach is increasingly problematic, as the number of features increases with the availability of data associated with each state. For example, facing an obstacle like a bird, it is hard to decide whether it is big enough to affect the cost function.

As the widespread of deep learning is reshaping the artificial intelligence applications, autonomous driving is also studied without an explicit model. Lu proposed a model-free method to learn

autonomous control on driving from end to end [1]. However, it is not subject to inference such that no convincible explanation can be reached as to why an action, such as turning left, is decided. Therefore, model-free method is not stable or reliable enough in practice.

Whereas, inverse optimal control has the advantage that it learns an explicit cost function from expert trajectories of the state-control. Autonomous driving can hardly be formulated as an MDP, yet there are various prior knowledges for modeling. In specific, it is difficult to formulate the cost function and the final objective in straightforward mathematical terms, while the general dynamic is known and basic rules are in commonsense, for instance, collision must be avoided.

In this paper, we formulate inverse optimal control by energy-based model. Since the training stage is equivalent to a min-max procedure, we can use sampling methods including Langevin Dynamic, gradient descent and iterative Linear Quadratic Regulation. More detailed discussion is presented in Section 3.

In section 4, multiple deep neural network structures are designed as an add-on to rule-based driving cost defined by human, taking the advantage of both deep neural network and rule-based features. The CNN design is considered to be the most accurate, as it takes into account the connection between history trajectories and future trajectories.

We compare the baseline model CIOC with our method empirically in Section 5. Different neural network structures are also discussed. In autonomous simulator, our control is capable of following lane, avoiding collision and even overtaking the slow car ahead.

There are two main contributions of this paper,

(1) Design a deep network based on human defined feature. It not only utilizes the big data by the deep neural network, but also take preliminary knowledges into consideration.

(2) Introduce Langevin Dynamic, a Monte Carlo based sampling method. This method can sample trajectory over non-linear high-dimensional distribution with efficiency.

We also discussed further expansion on multi-agent problem.

2 Related Work

Inverse optimal control (IOC), with an idea similar to inverse reinforcement learning, has seen a variety of methods including maximum margin-based optimization, maximum entropy-based optimization, Bayesian inference, regression and classification [2].

Ziebart proposed an energy-based model for trajectories [3]. They assumed an energy-based distribution on trajectories and learned the model parameters by maximizing the entropy over expert data. Wulfmerer extended the maximum entropy IRL to a deep version [4]. However, these algorithms can only solve finite state control problem, because it uses dynamic programming to calculate the precise normalization term which is intractable in continue case (detailed in section 3).

For continuous inverse optimal control, Levine used Laplace assumption making the trajectory distribution into a gaussian [5]. Nevertheless, the assumption is not always justifiable, especially when the cost function is complex. The emipirical results suggests that our method with sampling method outperformed CIOC method.

Finn proposed guided cost learning, a sample-based method with energy-based model [6][7]. It uses policy search to sample the trajectory, in order to approximate the normalization term in the energy-based model. However, for complex cost functions approximation by policy method is much more difficult.

Jonathan [8] and Yunzhu [9] synthesized and evaluated the full state-control trajectories iteratively by Generative Adversarial Network (GAN). In their setting, the trajectory distribution is captured by GAN, so as to be learned from the images directly. Since it is trained end to end without an explicit model, it is hard to tell why a particular control is performed, since no meaningful cost function is learned.

In summary, strictly model-based methods are not flexible enough for more general problem, while model-free methods with deep learning is not subject to inference. We believe that the energy-based model, which is descriptive in nature, could be a solution. A core concern in EBM is how to approximate a non-linear hogh-dimensional distribution with intractable normalization term, yet the existing attempts for IOC are not effective for complex distributions. Whereas, Xie el. introduced Langevin Dynamic, a MCMC based sampling method, to formulate non-linear energy-based model with a variety of data including image data [10], video data [11] and 3D data [12], which learned even high-dimensional distributions effectively. We believe the same idea should be applicable in IOC.

3 Preliminary

3.1 Inverse Optimal Control

We first define continuous markov decision process (MDP) for single agent in IOC. An MDP is defined as,

$$M =< X, U, D, C >$$

where $X \in R^N$ is the state, $U \in R^n$ is the control, D is the dynamic function: $x_{t+1} = f(x_t, u_t)$ and $C(x_t, u_t)$ is the cost function. We assume the dynamic function f is known and the state x and control u are continuous.

A trajectory is a sequence of state and control with a fixed length T. We can define the cost function of a trajectory as, $C_\theta(\tau)$, which can be an arbitrary function. Normally, $C_\theta(\tau) = \sum_{t=0}^{T} C_\theta(x_t, u_t)$.

The objective of optimal control is to output a trajectory with the lowest cost given a known MDP, while that of inverse optimal control is to learn the MDP, derermined by the dynamic function or cost function, that best fits the expert trajectories τ_t given a training set of expert trajectories. In general, the idea is to minimize the expert trajectory costs.

3.2 Autonomous Driving

In autonomous driving, the state x consists of ego vehicle status (position, steering, speed) and environment information (lane, sign and other vehicle status). Control u has two dimensions, steering and acceleration. In our dataset, the environment information are provided as the position of lane, the speed limit, the road boundary and other vehicle status which is estimated through camera and sensor. The trajectory in autonomous driving is a sequence of state and control with the same interval, typically 0.1s.

In cost function parameter learning procedure, our input is expert trajectories. The full trajectories of ego vehicle and other vehicles are provided, so does the lane which the first-frame vehicle can see.

In optimal control procedure, we assume we only have the starting point of a trajectories, for both ego vehicle and other vehicles. We have to predict other vehicles' future trajectories first. Then predict ego vehicle's future trajectory. When evaluating the algorithm, we compare the predicted trajectory with ground truth in testing set. In real car testing, we only use the first frame control (although we predict a trajectory with multiple control). Then we move our car to a new state and predict the new trajectory based on the new state.

The dynamic function is known in our situation. We use bicycle model [13] to form the dynamic function, which considers our vehicle as a two-axis vehicle. It is differentiable.

3.3 Energy-based Model

In this paper, we try to infer the distribution of the trajectory. The distribution is assumed to take the form of an energy-based function where the energy term is the negative cost function.

The probability distribution of the trajectory τ is defined as,

$$P(\tau;\theta) = \frac{1}{Z}\exp(-c_\theta(\tau))q(\tau)$$

where q is reference distribution of trajectory, typically a Gaussian white noise distribution according to the control u, i.e. $q(\tau) \propto exp(-||u||^2/2s^2)$, $c_\theta(\tau)$ is the cost function and,

$$Z = \int \exp(c_\theta(\tau))q(\tau)d\tau = E_q[\exp(-c_\theta(\tau))q(\tau)]$$

is the normalization term. This term constrains $\int P(\tau) = 1$. The probability of taking a trajectory is small if the corresponding cost is big.

In this energy-based model, the energy function is,

$$\mathcal{E}_\theta(\tau) = \frac{||u||^2}{2s^2} - c_\theta(\tau)$$

The goal of inverse optimal control is to find a distribution better represent the expert control. In other word, we want to maximum the log-likelihood on expert trajectories ($\tau_i \in Traj_{obs}$),

$$l(\theta) = \frac{1}{n}\sum \log P(\tau_i;\theta) = \frac{1}{n}\sum(-c_\theta(\tau_i) - log(Z)$$

4 Algorithms

In energy-based model probability density function, the normalization term Z is intractable. In CIOC, it uses Laplace approximation which model the trajectory distribution to be a Gaussian.

Hence, this likelihood can be approximated by doing Taylor expansion on cost function around control u,

$$l(\theta) = \frac{1}{2} C_u^T C_{uu}^{-1} C_u + \frac{1}{2} \log | - C_{uu}| - \frac{d_u}{\log}(2\pi)$$

where $c_u = \frac{\partial C_\theta}{\partial U}, c_{uu} = \frac{\partial^2 C}{\partial U^2}$

However, the assumption is not justifiable in many cases. As an alternative, we use sample-based method to calculate the normalization term. The gradient will be,

$$\frac{\partial}{\partial \theta} l(\theta) = \frac{1}{n} \sum \frac{\partial}{\partial \theta} c_\theta(\tau_i) - E_{P(\tau;\theta)}[\frac{\partial}{\partial \theta} c_\theta(\tau_i)]$$

Here,

$$\frac{\partial}{\partial \theta} \log(Z) = \frac{1}{Z} \frac{\partial}{\partial \theta} \int \exp(c_\theta(\tau)) q(\tau) d\tau = \int \frac{1}{Z} \exp(c_\theta(\tau)) q(\tau) \frac{\partial}{\partial \theta} \exp(c_\theta(\tau)) d\tau$$

$$= \int \frac{\partial}{\partial \theta} \exp(c_\theta(\tau)) P(\tau;\theta) d\tau = E_{P(\tau;\theta)}[\frac{\partial}{\partial \theta} c_\theta(\tau_i)]$$

For the expectation term, we approximate it by sampling through multiple methods.

$$\frac{\partial}{\partial \theta} l(\theta) = \frac{1}{n} \sum \frac{\partial}{\partial \theta} c_\theta(\tau_i) - \frac{1}{\tilde{n}} \sum \frac{\partial}{\partial \theta} c_\theta(\tilde{\tau}_i)$$

where $\tilde{\tau}$ is the sampled trajectories and $\tilde{n}$ is the number of samples.

4.1 Min-max Analysis

We now show that the training procedure can be interpreted as a min-max game. [12] Rewrite the deviation into,

$$\frac{\partial}{\partial \theta} l(\theta) = \frac{\partial}{\partial \theta} \left[\frac{1}{n} \sum \mathcal{E}_\theta(\tau_i) - \frac{1}{\tilde{n}} \sum \mathcal{E}_\theta(\tilde{\tau}_i) \right]$$

Let,

$$V_\theta(\tilde{\tau}) = \frac{1}{n} \sum \mathcal{E}_\theta(\tau_i) - \frac{1}{\tilde{n}} \sum \mathcal{E}_\theta(\tilde{\tau}_i)$$

be the value function. Hence, taking gradient of V is equivalent to taking gradient of the likelihood function.

The learning step attempts to increase the value $V_\theta(\tilde{\tau})$ by updating theta and shifting the mode. It shifts the low energy mode from the synthesized trajectories $\{\tilde{\tau}_i\}$ toward the expert trajectories $\{\tau_i\}$.

The sampling step attempts to decrease the value $V_\theta(\tilde{\tau})$ by selecting τ and seeking the mode. We sample to find the trajectories around the current local mode.

As a result, our training process is to find,

$$\theta = \arg \min_\theta \max \tilde{\tau} V_\theta(\tilde{\tau})$$

4.2 Sampling Algorithm

The state in our model can be divided into two part, vehicle status x_v and environment x_e. Modifying control will change status but not the environment. We sample our trajectories based on input environment. For each expert trajectory with environment, we synthesis one trajectory based on its environment. In other world, we sample a conditional distribution with fixed environment and maximum the conditional likelihood,

$$l(\theta) = \frac{1}{n}\sum_{i=1}^{N} \log P(\tau_i | Xe_i = x_e; \theta)$$

The sampling algorithm will only update the control, which lead to a change toward vehicle status x_v.

We experiment with two sampling methods. The first one is a deterministic method, iterative Linear Quadratic Regulation and second one is a MCMC based method, Langevin Dynamic.

4.2.1 Iterative Linear Quadratic Regulation

Iterative Linear Quadratic Regulation (iLQR) is a variant of Differential dynamic programming (DDP) [14] [15]. Given an initial trajectory, it updates the trajectory by repeatedly solving for the optimal policy under linear quadratic assumptions.

Let (x_t^i, u_t^i) be the i-th iteration trajectory. The dynamic is known, $x_{t+1}^i = f(x_t^i, u_t^i)$. Define $\Delta x_t = x_{t+1} - x_t, \Delta u_t = u_{t+1} - u_t$, then,

$$\Delta x_{t+1} \approx f_{x_t}\Delta x_t + f_{u_t}\Delta u_t$$

$$C_\theta(x_t, u_t) \approx \Delta x_t c_{x_t} + \Delta u_t c_{u_t} + \frac{1}{2}\Delta x_t c_{xx_t}\Delta x_t + \frac{1}{2}\Delta u_t c_{u_t}\Delta u_t + \Delta u_t c_{ux_t}\Delta x_t + C_\theta(x_{t-1}, u_{t-1})$$

where the subscripts denote the Jacobians and Hessians of the dynamic f and cost function C.

iLQR recursively calculates the Q-function from the tail of the trajectory to the head,

$$\begin{aligned}
Q_{xx_t} &= r_{xx_t} + f_{x_t} V_{xx_{t+1}} f_{x_t} \\
Q_{uu_t} &= r_{uu_t} + f_{u_t} V_{xx_{t+1}} f_{u_t} \\
Q_{ux_t} &= r_{ux_t} + f_{u_t} V_{xx_{t+1}} f_{x_t} \\
Q_{x_t} &= r_{x_t} + f_{x_t} V_{x_{t+1}} \\
Q_{u_t} &= r_{u_t} + f_{u_t} V_{x_{t+1}}
\end{aligned}$$

Then we calculate V and K by,

$$\begin{aligned}
V_{xx_t} &= Q_{xx_t} - Q_{ux_t} Q_{uu_t}^{-1} Q_{ux_t} \\
V_{x_t} &= Q_{x_t} - Q_{ux_t} Q_{uu_t}^{-1} Q_{u_t} \\
k_t &= -Q_{uu_t}^{-1} Q_{u_t} \\
K_t &= -Q_{uu_t}^{-1} Q_{ux_t}
\end{aligned}$$

Finally they are used to update the (i+1)-th trajectory by given the ith trajectory.

$$\begin{aligned} x_0^{i+1} &= x_0^i \\ u_t^{i+1} &= u_t^i + k_t + K_t(x_t^{i+1} - x_t^i) \\ x_{t+1}^{i+1} &= f(x_t^{i+1}, u_t^{i+1}) \end{aligned}$$

After several iterations, the trajectory converges to the current local optimal.

4.2.2 Langevin Dynamic

Another sampling method we tried is Langevin Dynamic, which is a Markov Chaine Monte Carlo (MCMC) based method. The iterative function is,

$$U_{\tau+1} = U_\tau - \frac{\delta^2}{2}\left[\frac{U_\tau}{\omega^2} - \frac{\partial}{\partial U}C_\theta(U_\tau)\right] + \delta Noise_\tau$$

The Langevin dynamics consists of a deterministic part, which is the gradient descent for control u, and a stochastic part, which is a Brownian motion that helps the chain to escape spurious local minima.

For each sampling squence, we sample exactly one trajectory.

Notice that the state changes as the control is changed; at the same time, the change of control in the previous frame affects each cost later. Thus the derivative is calculate with chain rule.

$$\frac{\partial}{\partial u_i}C_\theta(U_\tau) = \sum_{i=0}^{T}\frac{\partial C_i}{\partial u_t} = \sum_{i=t}^{T}\frac{\partial C_i}{\partial u_t} = \sum_{i=t}^{T}\frac{\partial C_i}{\partial x_i}\frac{\partial x_t}{\partial u_t}\prod_{j=t}^{i-1}\frac{\partial x_{j+1}}{\partial x_j} + \frac{\partial C_t}{\partial u_t}$$

We can also use gradient descent to find the local mode of the energy function by eliminating the Brownian motion term in Langevin Dynamic.

$$U_{\tau+1} = U_\tau - \frac{\delta^2}{2}\left[\frac{U_\tau}{\omega^2} - \frac{\partial}{\partial U}C_\theta(U_\tau)\right]$$

4.3 Algorithm Flow

The training algorithm of energy-based model is presented as follows,

5 Cost Function

5.1 Rule-based Cost Function

Typically, the final cost function for a trajectory is defined as the sum of the cost for each frame with a state-control pair,

$$Cost_\theta(\tau) = \sum_{(x,u)\in\tau} Cost_\theta(x,u)$$

Algorithm 1 Inverse Optimal Control by EBM

1: **input** expert trajectories τ_i
2: **output** the cost function parameter θ, the synthesized trajectories.
3: Let t $\leftarrow$ 0, initialize θ
4: **repeat**
5: Mode Seeking: Use current cost function to synthesis trajectories by iLQR or Langevin Dynamic
6: Mode shifting: Use synthesis trajectories to update $\theta by \theta^{(t+1)} = \theta^{(t)} + \gamma L'(\theta^{(t)})$
7: t $\leftarrow$ t + 1
8: **until** t = T

For rule-based cost function, we design multiple features and combine them linearly to obtain the final cost function. IOC is used to learn the linear weight for each feature cost.

$$Cost_\theta(x, u) = \sum_{k=1}^{K} \theta_k f_k(x, u)$$

where $f_k(x, u)$ is hand-crafted based on human expertise. See appendix for detailed settings.

5.2 Neural Network Designs

Neural network is add to each frame besides the rule-based function. Three different network structures are designed and compared in the experiments.

The first one is 'NN use parameter'. Instead of linearly combining J functional costs, we input the feature costs into a 2 layer fully-connected neural network and use the output as the final cost. Basically, it is a non-linear combination of the human defined features.

The second one is 'NN as residual'. We feed the raw data into a two layer fully-connected neural network and output a scaler. The final cost is then the sum of the scaler ouput and the original CIOC cost. In this design, neural network works as a residual to correct the cost, so it does not affect the result by much.

The third one is 'NN as residual to each'. The build is similar to 'NN as residual', yet we output the J scalers corresponding to J features rather than one single scalar. The J scalers are added to each rule-based feature as residual and output the linear combination. The experiments indicates that this method may distract the human defined features and decrease the accuracy.

Figure 1 shows the network structure. The red block stands the human defined feature and the blue block is neural network layer. We simply use fully connect layers.

5.3 Convolution Neural Network Setting

To use some DDP (differential dynamic programming) method include iLQR. We need the cost function for each state. However, use Langevin dynamic do not need this constrain. We established a convolution neural network to connect the temporal information between states. That is, $Cost_\tau = F(X, U|\theta)$, where F is a fully-convolution neural network. The input of F will be the single frame cost. Figure 2 shows the structure of the CNN.

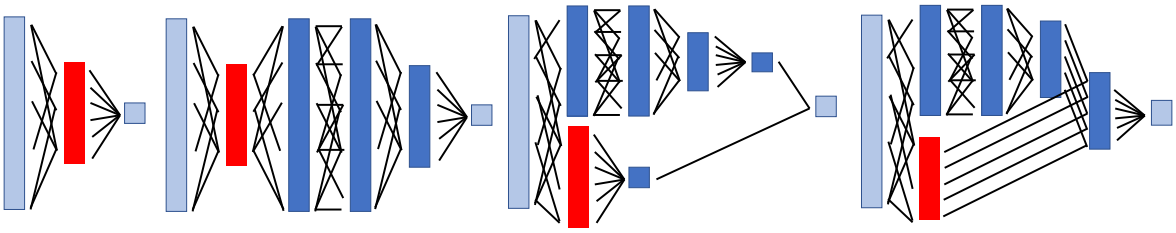

Figure 1: (a) rule-based method (b) NN use parameter (c) NN as residual (d) NN as residual to each

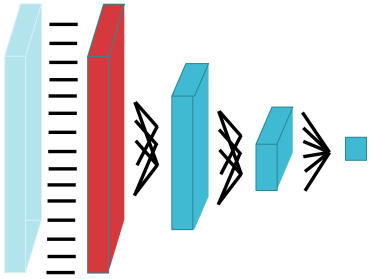

Figure 2: CNN structure

6 Experiments

We implement trajectory planning for experiment.

6.1 Dataset

We use a preprocessed dataset from ISee Inc car recording. The data includes the car status and environment status. Car status include 8 scalers: longitude, latitude, length of car, width of car, height of car, angle, velocity of long., velocity of lat. For ego car, the control is provided by two scalers: the steering and acceleration. Environment status include all lane provided by position list. One problem of this data is the GPS signal has noise. Kalman filter is used to denoise the data. We use rollout data as both training data and testing data. The number of expert trajectories is 1631. The length of trajectories is 30 frames.

Figure 3 shows a typical trajectory include environment data. The blue and green dots are ground truth data for ego vehicle and other vehicles; the yellow and orange dots are smoothed trajectory for ego and other vehicles; the red and purple dots are the lane center they are driving.

6.2 Evaluation

We calculate the L2-error and likelihood between predicted trajectory and ground truth. Since the trajectory position based on GPS has noise while the control is based on internal sensor which is much more precise, we use control to rollout the denoised trajectory as the ground truth.

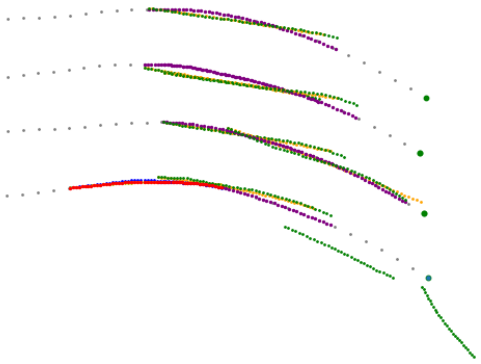

Figure 3: Typical trajectory

Assume $\tau_1 = [(x1_1, y1_1), ..., (x1_n, y1_n)], \tau_2 = [(x2_1, y2_1), ..., (x2_n, y2_n)]$, the l2-error for i-th position is defined as,

$$Err(\tau_1, \tau_2, i) = (x1_1 - x2_1)^2 + (x1_2 - x2_2)^2$$

The likelihood is defined as,

$$L(\tau_1, \tau_2) = \prod_{i=1}^{T} \frac{\phi(x1_i; x2_i, 1)\phi(y1_i; y2_i, 1)}{\phi(x2_i; x2_i, 1)\phi(y2_i; y2_i, 1)}$$

where $\phi(a; \mu, \sigma)$ means the probability for a under gaussian distribution where its mean and standard deviation is μ and σ.

Result with prefect prediction has zero l2-error and likelihood of 1. l2-error is the small the better while likelihood is the bigger the better.

6.3 Methods Comparison

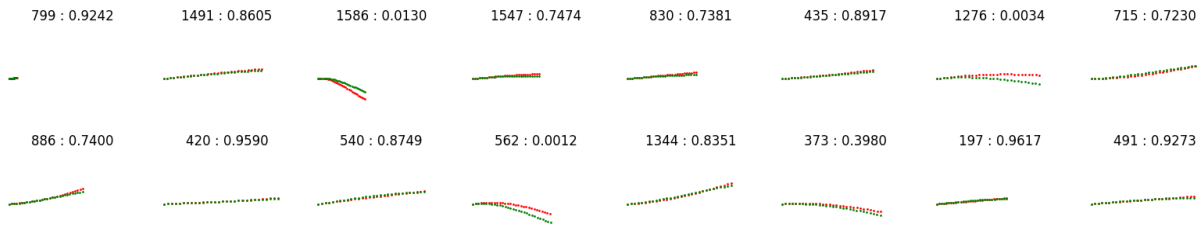

Figure 4: Predicted Trajectory

We first evaluate different inverse optimal control algorithm and different sampling methods. The Continuous Inverse Optimal Control (CIOC) is used for baseline. By using simple model-based cost function without Neural Network, we compare the likelihood result and l2-error result. A typical prediction result is shown in figure 4. We can see that most of the trajectory can be predicted perfectly. A small number of trajectories are not. We called them the corner case. We

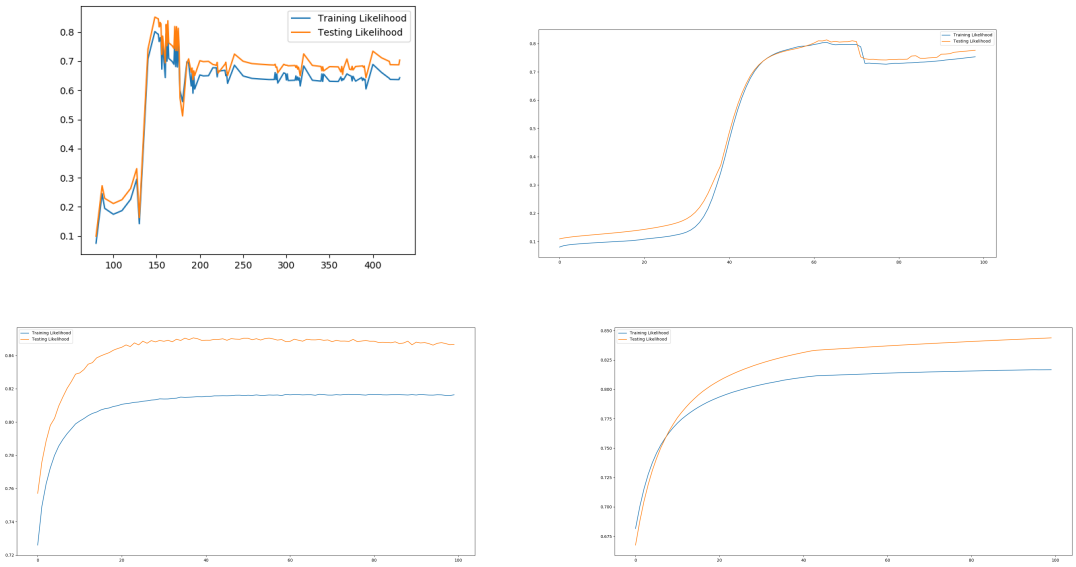

Figure 5: Comparing different methods (Upper-left : Baseline CIOC, Upper-right : iLQR, Bottom-left : Langevin Dynamic, Bottom-right : Gradient Descent)

Table 1: Comparison result using different algorithm with rule-based cost function

Method	CIOC	iLQR	Langevin	Gradient	Half Langevin
Training Likelihood	0.796	0.807	0.816	0.815	0.814
Testing Likelihood	0.837	0.815	0.848	0.849	0.850
Training L2-error	1.16	/	1.849	1.901	1.870
Testing L2-error	0.47	/	0.321	0.328	0.328

found for most of the method, there is no big difference for those good result. Only the corner case can show a result is good or not. We are planning to find more corner case to test.

Due to the limit number of data, the number of corner case is slightly small. As a result, it makes abnormal result that the testing result is better than training result. However, it is still comparable. We set the same random seed making it to have the same training set and testing set. By analysis the predicted trajectory, we do see that better likelihood lead to better prediction in corner case.

Result shows that our method is better. More importantly, it is more stable than the COIC method. Figure 5 shows the training curve showing that CIOC method is not stable enough. For both Langevin Dynamic and gradient descent method got the similar result. 'Half Langevin' stands the sampling step is Langevin dynamic for first half iteration and gradient descent for the remaining.

6.4 Cost Function Comparison

We then compare the result for different cost function setting. Here we provided the result using Langevin Dynamic, which got the best performance in previous comparison.

Table 2: Comparison result using Langevin Dynamic with different cost function setting

Method	NN use parameter	NN as residual	NN as residual to each	CNN
Training Likelihood	0.813	0.816	0.808	0.834
Testing Likelihood	0.855	0.850	0.832	0.873
Training L2-error	1.684	1.367	1.157	1.240
Testing L2-error	0.380	0.399	0.415	0.258

Result shows that the CNN setting got the best result. There is a considerable increase in accuracy. IT shows that the connection between multiple frame is important for cost function. We believe a more complex net will handle better result. However, we found that training CNN setting is unstable. Due to the complex setting of cost function, the distribution of trajectory become unsmooth. It makes harder to sample and learn. Figure 6 shows the CNN accuracy in different epoch. Actually, if we set a bigger learning rate, the model will output NaN and making the control to all zero.

For other setting of neural network. We found there is a small promotion in NN use parameter setting, while the NN as residual to each setting decrease the result. Through this experiment, we show that the deep neural network does construct a better cost function. Meanwhile, we show our energy-based model method can handle the neural network setting.

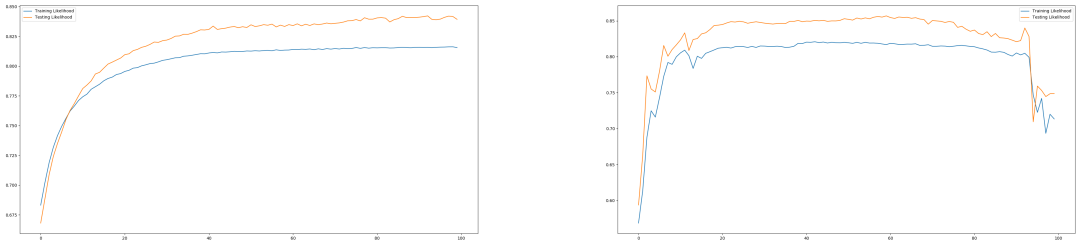

Figure 6: Cost function structure comparsion (left : NN use parameter, right : CNN)

6.5 Simulation

We use ROS to simulate our algorithm, the simulator is provided by Isee Inc. Only rule-based method is tested on simulator. In the simulation, our ego vehicle is able to follow lanes and avoid collisions. Figure 7 is a snapshot of the simulator. (*The snapshot is not runned in our method, but just a sample.)

The white box in the middle is our ego car. The pink dots in front of the ego car is the predicted trajectories. The green lines are lanes provided by the simulator and the hollow box are other vehicles.

7 Discussion and Future Work

In this paper, we implemented continuous IOC with energy-based model. In comparison to CIOC, our method with Langevin Dynamic sampling is proven to achieve a better result with

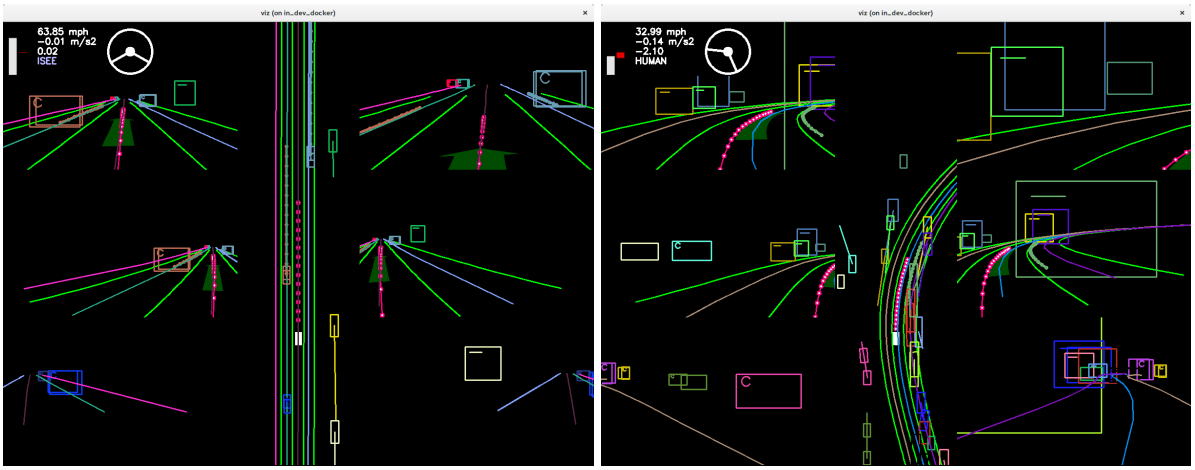

Figure 7: Simulation Result

higher stability. In future works, we plan to do more comparitive experiments with other methods including Guided cost learning and GAIL.

We designed multiple structures of neural network as an add-on to the cost function. The neural network, especially convolutional neural network, greatly improves the prediction in experiments. However, it also introduces unstability to the training. Therefore, our method is not likely to find the global minimum in a higher dimensional space. As a next step, we plan to synthesize more trajectories per epoch and collect more data for training.

Energy-based model is very promising for inverse optimal control, because as a descriptive model it represents the whole trajectory distribution in theory. We believe it can also be extended to multi-agent situations .

In autonomous driving, there are different moving objects including other vehicles, pedestrians and other moving. Currently, we only predict their trajectories individually, while their interactions are neglected, e.g., the car behind should decelerate when the front one decelerates. Therefore, multi-agent planning is a direction to be explored. One possible implementation is to calculate the joint trajectory distribution for the moving agents and sample the multiple trajectories at the same time. Assume we have K agents and each of them has a trajectory τ_i, then,

$$P(\tau_1, ..., \tau_K|\theta) = \frac{1}{Z} e^{\sum_{i=0}^{K} C_\theta(\tau_i)}$$

The cost function of each agents share the same parameters. Notice that the cost function for one vehicle is dependent on the information of the others.

Another direction could be Partially Observable MDP (POMDP). We regard other vehicle's intension as part of an unobserved state, and use POMDP setting to infer their intension.

Our future research will focus on various generalization of energy-based model and its variants, with the belief that it has great potential in the field of optimal control and reinforcement learning.

8 Acknowledgement

The data is provided by Isee Inc.

We thank Ying Nian Wu, Yibiao Zhao, Chris Baker, Tianyang Zhao, Xiao Li, Yilun Zhou, Lu Lyu for their help with experiments, data, algorithms etc.

This template is NIPS2017 template downloaded from the official website.

This paper is for Yifei Xu's PhD written qualifying examination only. Without permission, it should not be posted to public or viewed by people other than those in the Department of Statistics at University of California, Los Angeles.

References

[1] Lu Chi and Yadong Mu. Deep steering: Learning end-to-end driving model from spatial and temporal visual cues. *arXiv preprint arXiv:1708.03798*, 2017.

[2] Saurabh Arora and Prashant Doshi. A survey of inverse reinforcement learning: Challenges, methods and progress. *arXiv preprint arXiv:1806.06877*, 2018.

[3] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.

[4] Markus Wulfmeier, Peter Ondruska, and Ingmar Posner. Maximum entropy deep inverse reinforcement learning. *arXiv preprint arXiv:1507.04888*, 2015.

[5] Sergey Levine and Vladlen Koltun. Continuous inverse optimal control with locally optimal examples. *arXiv preprint arXiv:1206.4617*, 2012.

[6] Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *International Conference on Machine Learning*, pages 49–58, 2016.

[7] Chelsea Finn, Paul Christiano, Pieter Abbeel, and Sergey Levine. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *arXiv preprint arXiv:1611.03852*, 2016.

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

[9] Yunzhu Li, Jiaming Song, and Stefano Ermon. Infogail: Interpretable imitation learning from visual demonstrations. In *Advances in Neural Information Processing Systems*, pages 3812–3822, 2017.

[10] Jianwen Xie, Yang Lu, Song-Chun Zhu, and Yingnian Wu. A theory of generative convnet. In *International Conference on Machine Learning*, pages 2635–2644, 2016.

[11] Jianwen Xie, Song-Chun Zhu, and Ying Nian Wu. Synthesizing dynamic patterns by spatial-temporal generative convnet. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7093–7101, 2017.

[12] Jianwen Xie, Zilong Zheng, Ruiqi Gao, Wenguan Wang, Song-Chun Zhu, and Ying Nian Wu. Learning descriptor networks for 3d shape synthesis and analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8629–8638, 2018.

[13] Philip Polack, Florent Altché, Brigitte d'Andréa Novel, and Arnaud de La Fortelle. The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles? In *Intelligent Vehicles Symposium (IV), 2017 IEEE*, pages 812–818. IEEE, 2017.

[14] Weiwei Li and Emanuel Todorov. Iterative linear quadratic regulator design for nonlinear biological movement systems. In *ICINCO (1)*, pages 222–229, 2004.

[15] Alberto Bemporad, Manfred Morari, Vivek Dua, and Efstratios N Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1):3–20, 2002.

A Details on Rule-based Function

Firstly, we define raw input R as a 37-dimension vector which is,

- 0:6 : status for ego vehicle (x, y position, angle, speed, delta angle, delta speed)
- 8:12 : lane coefficient which represent the lane shape
- 12:32 : other vehicle * 5 (x, y, angle, speed)
- 32:36 : goal state which is a position in front of the ego vehicle
- 36 : speed limit

10 functioned cost defined in CIOC are,

- 0:2 : Goal cost (x, y)
- 2:4 : L2 norm for control
- 4:6 : L2 norm for delta control
- 6 : Obstacle : if the distance from ego vehicle to obstacle is samll, there is a cube penalty
- 7 : speed limit
- 8 : direction compare to lane direction
- 9 : distance to lane center.