Learning Trajectory Prediction with Continuous Inverse Optimal Control via Langevin Sampling

Paper #928

ABSTRACT

Autonomous driving is a challenging multiagent domain which requires optimizing complex, mixed cooperative-competitive interactions. Learning to predict contingent distributions over other vehicles' trajectories simplifies the problem, allowing approximate solutions by trajectory optimization with dynamic constraints. We take a model-based approach to prediction, in order to make use of structured prior knowledge of vehicle kinematics, and the assumption that other drivers plan trajectories to minimize an unknown cost function. We introduce a novel inverse optimal control (IOC) algorithm to learn other vehicles' cost functions in an energy-based generative model. Langevin Sampling, a Monte Carlo based sampling algorithm, is used to directly sample the control sequence. Our algorithm provides greater flexibility than standard IOC methods, and can learn higher-level, non-Markovian cost functions defined over entire trajectories. We extend weighted feature-based cost functions with neural networks to obtain NN-augmented cost functions, which combine the advantages of both model-based and model-free learning. Results show that model-based IOC can achieve state-of-the-art vehicle trajectory prediction accuracy, and naturally take scene information into account.

KEYWORDS

Inverse Optimal Control; Inverse Reinforcement Learning; Autonomous Driving; Trajectory Prediction; Neural Networks

1 INTRODUCTION

Autonomous driving is a challenging multiagent domain which requires optimizing complex, mixed cooperative-competitive interactions. Interacting with other drivers is currently a major impediment to the widespread deployment of driverless autonomous vehicles, and current approaches fall short of a general solution. Rule-based approaches to interaction are too brittle — they require a huge effort to engineer and validate, and they generalize poorly to new scenarios. Learning-based approaches are promising because of the complexity of driving interactions, and the need for generalization. However, learning based systems require a huge amount of data to cover the space of interactive behaviors, and current model-free learning-based methods [5, 6, 19] have not achieved safe, generalizable autonomous interaction thus far.

Learning to predict contingent distributions over other vehicles' trajectories is a standard approach to simplifying the interaction problem, allowing approximate solutions by trajectory optimization with dynamic constraints. Prediction methods generally fall into two categories: model-based and model-free. Model-based methods use prior knowledge of vehicle kinematics as well as the policies and intentions of other drivers to predict physically realistic maneuvers and trajectories [13]. Learning in these models either learns a space of trajectories or policies directly [2], or does inverse optimal control (IOC), learning a cost function that rationalizes observed behaviors [20, 23, 35]. Because they capture the generative structure of vehicle trajectories, model-based methods can potentially learn more, from less data, than model-free methods. However, good cost functions are challenging to learn, and simple, hand-crafted representations may not generalize well across tasks and contexts [16, 28]. In general, model-based methods can be less flexible, and may underperform model-free methods in the limit of infinite data.

Model-free methods [1, 15, 22, 30] take a data-driven approach, aiming to learn predictive distributions over trajectories directly from data. These approaches are more flexible and require less knowledge engineering in terms of the type of vehicles, maneuvers, and scenarios, but the amount of data they require may be prohibitive.

Here we take a model-based approach to prediction, with the goal to achieve the data-efficiency and performance of model-based learning, with the flexibility and generality of model-free learning. Our approach is to combine inverse optimal control (IOC) with model-free deep neural network architectures for cost function learning. Our motivation is that adding structure in the form of kinematic vehicular constraints (which can be modeled very accurately), and the assumption that human drivers optimize their trajectories according to a subjective cost function, provides a powerful inductive bias for learning. Our approach is to focus on learning good cost function representations, and we experiment with various neural network architectures, and model-based and model-free hybrids, to find those which perform the best.

Specifically, we derive a family of algorithms for IOC using energy-based models. We show how training the model is related to a min-max procedure, and how we can use multiple optimal control algorithms for sampling, including Langevin Sampling, gradient descent and iterative Linear Quadratic Regulation. We design multiple deep neural network architectures for cost function learning, some of which augment a set of hand-crafted features for driving using residual learning [17], and others that use convolutional (CNN) architectures to learn non-Markovian cost functions defined over entire trajectories.

We conduct several experiments to evaluate our approach on real-world driving data. We first compare our model with several state-of-the-art prediction techniques on the publicly-available NGSIM dataset [7]. We then evaluate our method on another autonomous highway driving dataset, containing a longer road segment than NGSIM, with more lane curvature as well as multiple highway

Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), N. Agmon, M. E. Taylor, E. Elkind, M. Veloso (eds.), May 2019, Montreal, Canada

https://doi.org/doi

entrances and exits. We compare our method with the CIOC algorithm [23], and we compare a range of different neural network structures in our framework. Finally, we analyze several synthetic examples to show that our IOC-based method predicts a variety of complex maneuvers, including lane-following, collision-avoidance, and overtaking a slower car ahead.

2 RELATED WORK

Inverse optimal control (IOC) and inverse reinforcement learning (IRL), have seen a variety of methods including maximum margin-based optimization, maximum entropy-based optimization, Bayesian inference, regression, and classification [3]. Most relevant to our work is the maximum causal entropy framework for IRL [35], which assumes an energy-based distribution on trajectories and learns the model parameters by maximizing the entropy over expert data. Wulfmerer extended maximum entropy IRL to a deep version [31]. However, these algorithms are only applicable in discrete state spaces (or in continuous state spaces with linear dynamics and quadratic costs [26]), because they use dynamic programming to calculate the precise normalization term which is intractable in the continuous case (detailed in Section 3). For continuous IOC (CIOC), Levine used the Laplace approximation to model the trajectory distribution as a Gaussian [23]. However, the Laplace assumption is not always justifiable, especially when the cost function is complex. Our experiments compare our sampling method with the results of CIOC.

Several model-free methods have been proposed which have a deep relationship to model-based IOC. Finn proposed guided cost learning, a sample-based approach to policy learning which uses energy-based models [11, 12]. Policy search is used to learn a network for trajectories sampling, in order to approximate the normalization term in the energy-based model. Ho [18] and Li [25] synthesized and evaluated full state-control trajectories iteratively using generative adversarial imitation learning. In their setting, the trajectory distribution is captured by the generator in a GAN [14], while the cost function is closely related to the representation learned by the GAN discriminator [11]. Our experiments compare the performance of our model with that of GAIL.

Finally, a recent body of research has applied supervised model-free learning to trajectory prediction. These approaches typically use recurrent encoder-decoder architectures [1], augmented with various techniques such as GANs and aggregation functions such as max-pooling [15], spatio-temporal graphs [30], and attention mechanisms [29], ranking and IOC [22], and convolutional pooling [9]. Our experiments compare our model with the convolutional social pooling approach in [9].

A core concern in energy-based models is how to approximate high-dimensional, non-linear distributions with intractable normalization terms. Xie et al. used Langevin Sampling, an MCMC based sampling method to formulate non-linear energy-based models for a variety of data including image data [32], video data [34] and 3D data [33]. Our algorithmic approach is inspired by this technique.

3 PRELIMINARIES

3.1 Inverse Optimal Control

We first formulate the prediction task as an inverse optimal control problem.

The major difference is that, rather than predict the trajectory directly, IOC outputs the control with which we can easily rollout the entire trajectory based on the dynamic model.

We first define continuous MDP for a single agent in IOC,

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

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 both the state x and control u are continuous.

A trajectory is a sequence of states and controls 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 (either dynamic function or cost function is unknown), that best fits the expert trajectories τ_t in the training set. In general, the idea is to minimize the costs of the expert trajectories relative to other trajectories. After the cost function is learned, we can predict the optimal control at current time step, thus go through the dynamic model to rollout the full trajectory.

3.2 Energy-based Models

With the goal to infer the distribution of the trajectory, we assume it takes the form of an energy-based model, where the energy term is the 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 the 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,

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

is the normalization term, which attends to the constrain $\int P(\tau) = 1$. The probability of taking a trajectory is small if the corresponding cost is big.

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

$$E_\theta(\tau) = \frac{||u||^2}{2s^2} - c_\theta(\tau).$$

The goal of IOC is to find a distribution that best fits the expert control. In other word, we maximize 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 the energy-based model, the normalization term Z is intractable. In CIOC, it uses Laplace approximation that models the trajectory distribution as a Gaussian distribution. Hence, the likelihood can

be approximated by the 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 always justifiable. As an alternative, we use sample-based approach to calculate the normalization term, and the gradient is,

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

because

$$\begin{aligned}
\frac{\partial}{\partial \theta} \log(Z) &= \frac{1}{Z} \frac{\partial}{\partial \theta} \int \exp(c_\theta(\tau)) q(\tau) d\tau \\
&= \int \frac{1}{Z} \exp(c_\theta(\tau)) q(\tau) \frac{\partial}{\partial \theta} \exp(c_\theta(\tau)) d\tau \\
&= \int \frac{\partial}{\partial \theta} \exp(c_\theta(\tau)) P(\tau;\theta) d\tau \\
&= E_{P(\tau;\theta)}[\frac{\partial}{\partial \theta} c_\theta(\tau_i)].
\end{aligned}$$

The expectation term can be approximated by sampling,

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

where $\tilde{\tau}$ is the sampled trajectories and $\tilde{n}$ is the number of samples. Different sampling approaches are discussed in Section 4.2.

4.1 Min-max Interpretation

We now show that the training procedure can be approximately interpreted as a min-max game. [33] Rewrite the deviation into

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

Let,

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

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

Mode Shifting The learning step attempts to increase the value $V_\theta(\tilde{\tau})$ by updating θ. It shifts the low energy mode from the current trajectories $\{\tilde{\tau}_i\}$ towards the expert trajectories $\{\tau_i\}$. In other word, with fixed $\tilde{\tau}_i$ and τ_i, we want $\sum E_\theta(\tau_i)$ to be as small as possible by changing θ .

Mode Seeking Then we decrease the value $V_\theta(\tilde{\tau})$ by selecting new $\tilde{\tau}$. By sampling we can generate trajectories $\{\tilde{\tau}_i\}$ which are optimal or near-optimal (close to the modes of the cost function).

As a result, our training process is

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

4.2 Sampling Method

4.2.1 Langevin Sampling. The state in our model can be divided into two parts, vehicle status x_v and environment x_e. Modifying control affects the status but not the environment.

We sample the trajectories based on the input environment information. For each expert trajectory, we synthesize one trajectory based on the associated environment. In other words, we sample from the conditional distribution with fixed environment and maximize the conditional likelihood,

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

Our sampling algorithm only updates the control, which leads to a change in status x_v.

The iterative process for Langevin Sampling 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 sequence, 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 calculated by 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 Sampling.

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

4.2.2 Iterative Linear Quadratic Regulation. Iterative Linear Quadratic Regulation (iLQR) is a variant of Differential dynamic programming (DDP) [24] [4]. 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,

$$\begin{aligned}
\Delta x_{t+1} &\approx f_{x_t} \Delta x_t + f_{u_t} \Delta u_t \\
C_\theta(x_t, u_t) &\approx \Delta x_t c_{x_t} + \Delta u_t c_{u_t} + \frac{1}{2} \Delta x_t c_{x x_t} \Delta x_t \\
&+ \frac{1}{2} \Delta u_t c_{u_t} \Delta u_t + \Delta u_t c_{u x_t} \Delta x_t + C_\theta(x_{t-1}, u_{t-1}).
\end{aligned}$$

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,

$$\begin{aligned}
Q_{x x_t} &= r_{x x_t} + f_{x_t} V_{x x_{t+1}} f_{x_t} \\
Q_{u u_t} &= r_{u u_t} + f_{u_t} V_{x x_{t+1}} f_{u_t} \\
Q_{u x_t} &= r_{u x_t} + f_{u_t} V_{x x_{t+1}} f_{x_t} \\
Q_{x_t} &= r_{x_t} + f_{x_t} V_{x_{t+1}} \\
Q_{u_t} &= r_{u_t} + f_{u_t} V_{x_{t+1}}.
\end{aligned}$$

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 $(i+1)$-st trajectory 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.

4.3 Algorithm Flow

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

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 Sampling.
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** $t = T$.

As for prediction after training,

Algorithm 2 Prediction by Optimal Control.

1: **input** the cost function $Cost_\theta$, history status x_0 and environment.
2: **output** the predicted trajectory x_{pre}
3: **if** the history status does not include control **then** Infer the control based on history status list (10 frame)
4: **end if**
5: Use basic prediction method to predict other agent's trajectory. We can use constant velocity / constant acceleration and steering, which is not exact but accurate enough.
6: Use optimal control method like iLQR or Langevin Sampling to find the control $c_{pre} = \arg\min Cost_\theta(x_{pre}, c_{pre})$.
7: Use known dynamic model to rollout the trajectory $x_{pre,i} = f(x_{pre,i-1}, c_i)$.

5 COST FUNCTION

5.1 Feature-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,

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

For feature-based cost function, we design multiple features and combine them linearly to obtain the final cost function. IOC can be 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 a powerful function approximator. It is capable of approximating arbitrarily complex nonlinear function given sufficient training data, and is flexible in incorporating prior information, which in our case is the manually designed features. We design three different network structures as an add-on to the feature-based function in the experiments.

The first one is 'NN as transformer'. 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, we introduce nonlinearity to 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 scalar. The final cost is then the sum of the scalar output and the original CIOC cost. In this design, neural network serves 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 idea is similar to 'NN as residual', yet we output the J scalars corresponding to J features rather than one single scalar. We add each of the J scalars to each feature as residual and output the linear combination. The experiments indicate that this method may distract the human defined features and decrease the accuracy.

Figure 1 shows the network structure. The red block stands for the human defined feature and the blue block the neural network layer, which uses fully connected layers.

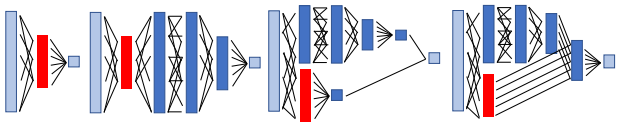

Figure 1: (a) hand-crafted Cost (b) nn as transformer (c) NN as residual (d) NN as residual to each.

5.3 Convolutional Neural Network Setting

To use some DDP (differential dynamic programming) methods including iLQR, we need to specify the cost function for each state.

However, Langevin Sampling does not have this constraint. We use a convolutional structure 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 to F is the single frame cost. Figure 2 displays the structure of the CNN.

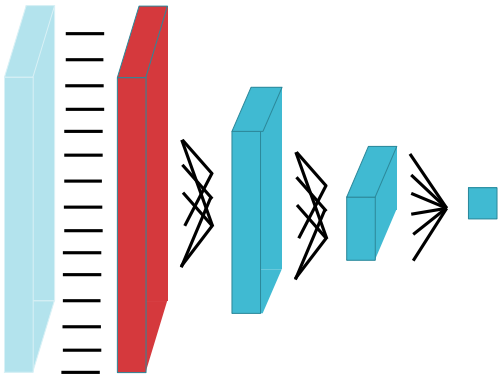

Figure 2: CNN structure.

6 EXPERIMENTS

We quantitatively and qualitatively evaluate the trajectory predictions of our inverse optimal control framework on two driving datasets and several synthetic examples.

In autonomous driving, the state x consists of vehicle status (position, steering, speed) and environment information (lane, signs and other vehicle states). In our datasets the environment information is provided as the position of lane, the speed limit, the road boundary and other vehicle states which is estimated through cameras and other sensors. Control u has two dimensions: steering and acceleration. A vehicle trajectory is a sequence of states and controls with a constant interval between frames, typically 0.1s.

For our cost function learning procedure, our input is a large amount of observed trajectories. The full trajectories of the ego vehicle and other vehicles are provided, as is the lane which the vehicle sees at the first time step (represented as a cubic polynomial).

We assume only the starting point of a trajectory is known for each vehicle. Thus, we need to predict the future trajectories of other vehicles before predicting that of a given vehicle. For simplicity, these other-vehicle trajectory predictions use a constant velocity model. For evaluation, we compare the predicted trajectories with the ground truth in the testing set.

6.1 NGSIM Dataset

We first experiment on the publicly available NGSIM US-101 dataset [7]. This dataset consists of real highway traffic captured at 10Hz over a time span of 45 minutes. We pre-process it into a 5 second / 50 frame blocks. There are 831 total scenes with 96,000 5-second vehicle trajectories.

6.1.1 Pre-processing. In the NGSIM dataset, there are no control values provided. Thus we need to infer the control of each vehicle given the vehicle state, which includes the position, speed and heading angle. Assuming bicycle model [27] dynamics, we perform an inverse-dynamics optimization using gradient descent to minimize the distance between the trajectory reconstructed with the inferred controls, and the real one. Figure 3 shows a typical scene with reconstructed trajectories using inferred controls (green dots) compared to the ground truth (red dots). In the training stage, we infer the controls for every vehicle in the scene.

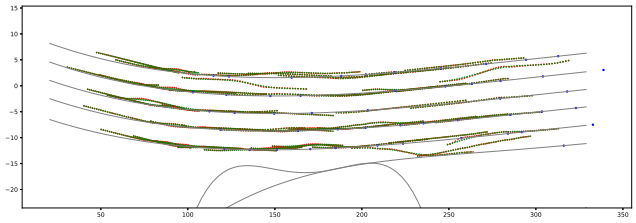

Figure 3: Typical trajectory of NGSIM.

6.1.2 Evaluation Metrics. We use Root-Mean-Square-Error (RMSE) between the predicted trajectory and ground truth to evaluate the performance of our model.

RMSE for i-th position is defined as:

$$RMSE = \sqrt{\frac{1}{N}\sum_k Err(\tau^k_{pre}, \tau^k_{obs}, i)}$$
$$= \sqrt{\frac{1}{N}\sum_k (x^k_{pre,1} - x^k_{obs,1})^2 + (x^k_{pre,2} - x^k_{obs,2})^2}.$$

A small RMSE is desired.

6.1.3 Model Comparison. We compare the following models:

- Constant Velocity: simplest baseline, generates trajectories with constant velocity and zero steering.
- Constant Control: Keeps acceleration and steering constant over the trajectory. Initial controls are estimated from the previous trajectory history.
- C-VGMM + VIM: Uses maneuver-based variational Gaussian mixture models with a Markov random field based vehicle interaction module described in [8].
- GAIL-GRU: Considers a GRU model based on generative adversarial imitation learning described in [21]
- V-LSTM: Uses the maneuver-based LSTM approach described in [10]
- CS-LSTM: Updates V-LSTM by introducing convolutional Social Pooling, as described in [9].

6.1.4 Results. Table 1 and Figure 4 show the RMSE results for different methods. Figure 5 shows randomly selected trajectory predictions. The red dots are ground truth trajectory. Blue, green and yellow dots are iLQR method, Langevin Sampling and constant control, respectively. Orange dots denote other vehicles.

Notice that the results of C-VGMM + VIM, GAIL-GRU, GAIL-GRU and CS-LSTM are retrieved from the original paper, yet we calculate constant velocity to make sure it is a fair comparison. The

Method (RMSE)	1s	2s	3s	4s
MEIOC via Langevin	0.318	0.644	1.149	2.138
MEIOC via iLQR	**0.324**	**0.682**	**1.180**	**2.103**
Constant Control	0.309	0.571	1.140	2.503
Constant Velocity	0.484	1.500	2.911	4.718
C-VGMM + VIM [8]	0.66	1.56	2.75	4.24
GAIL-GRU [21]	0.69	1.51	2.55	3.65
V-LSTM [10]	0.68	1.65	2.91	4.46
CS-LSTM [9]	0.61	1.27	2.09	3.10

Table 1: NGSIM dataset results.

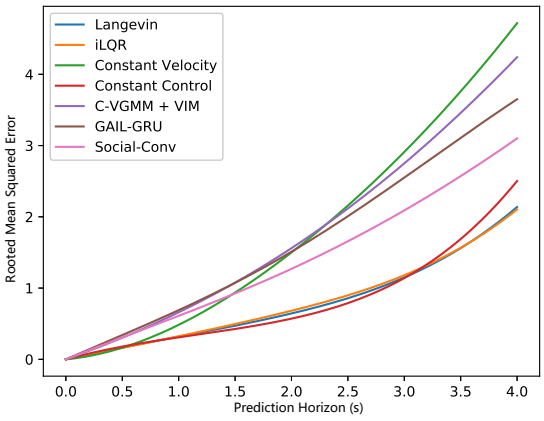

Figure 4: RMSE over time comparison for NGSIM.

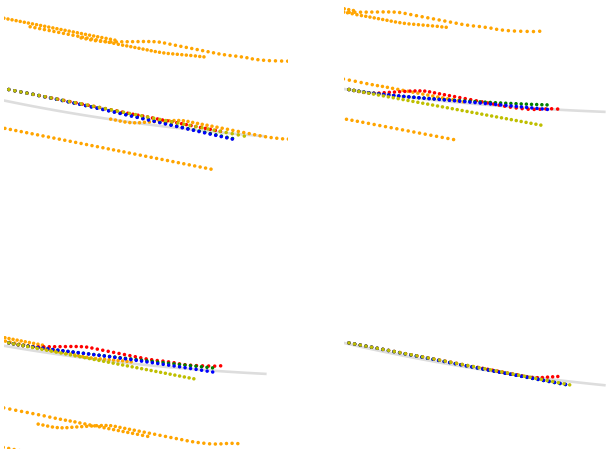

Figure 5: Predicted Trajectory.

obtained constant velocity is 4.718, which is close to that originally reported at around 4.78.

Both optimal control methods, Langevin and iLQR give similar results. The number of Langevin steps is set to 1024, making it possible to achieve the speed 10Hz, which is the lowest requirement for prediction. If we set Langevin step as 10 thousand, the 4s RMSE can be decreased into 2.098, which is better than iLQR.

6.1.5 Discussion. Our method achieved a substantial improvement compared with a state-of-art model-free prediction method. This suggests that representing vehicle kinematics and controls is fundamentally advantageous for prediction. Even with constant controls (acceleration and steering in the first frame), IOC beats all traditional prediction methods.

It is true that there are some differences between the two methods. Our method relies on an accurate measure of speed and heading angle, so that the initial acceleration and steering can be predicted and used in other-vehicle prediction and ego-rollout. The result is very sensitive to initialization, while the NGSIM dataset is collected by a fixed camera so the relative position and speed calculation are quite accurate.

Though the testing data typically should not include the scene in training data, such division is hard for the NGSIM dataset, because it includes the same scene through different times. We tried random splits, but there is no major difference between the training and testing data. As a result, we do use NGSIM data as a whole to train and test, the same as most baseline methods used for comparison.

6.2 Autonomous Driving Dataset

6.2.1 Dataset. In order to evaluate our model in a more diverse range of scenario, we use a dataset collected from an autonomous car during repeated drives on a several-mile stretch of highway, which includes both car states and environment information. In this dataset, we compared our IOC algorithm with CIOC [23], the current state-of-art continuous IOC algorithm discussed in section 4. Compared to NGSIM dataset, this dataset is more challenging for two reasons:

- The lane curvature of this dataset is 3x larger than NGSIM. Figure 6 shows the distribution of curvature.
- The data is noisy – especially for other vehicles. For NGSIM, the camera is set up in a fixed position while the autonomous driving dataset uses car-mounted cameras and LIDAR to estimate the states of other vehicles.

The advantage of this dataset is that it contains a more realistic driving scenario, as the control of the autonomous vehicle is collected by hardware on the car, which is more accurate than inferred controls.

The car state consists of four parts: x, y position, vehicle orientation and velocity. For the autonomous vehicle, the control is provided by two scalars: the steering and acceleration. Environment information consists of all lanes, represented as cubic polynomials. To solve the problem of noisy GPS signal, Kalman filtering is used to denoise the data. We use rollout data for both training and testing. The number of expert trajectories is 44,000, and each has a length of T = 30 frames with 0.1s intervals for a total of 4 seconds.

6.2.2 Evaluation Metrics. Other than RMSE, we introduce likelihood between predicted trajectory and ground truth. The reason is that, RMSE gives the same penalty for both x axis and y axis, while we try to give the y-axis (latitude) a bigger penalty, because even a small lateral shift may trigger a collision with an adjacent vehicle.

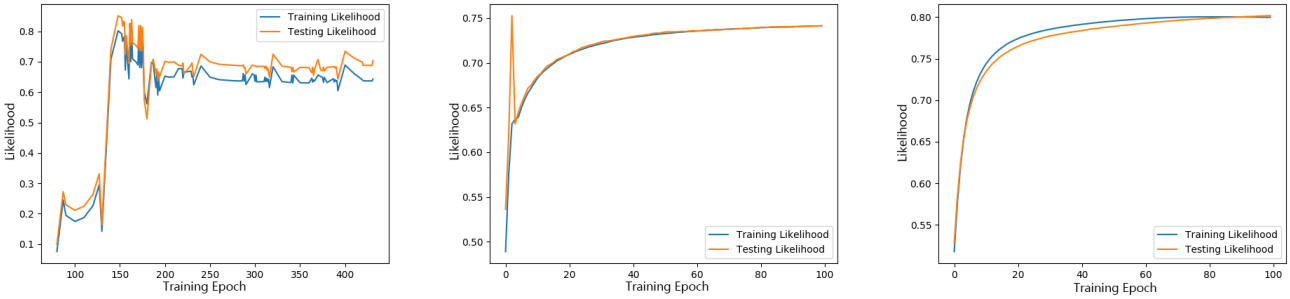

Figure 6: Comparing different methods (Left : Baseline CIOC, Middle: iLQR, Right: Langevin Sampling).

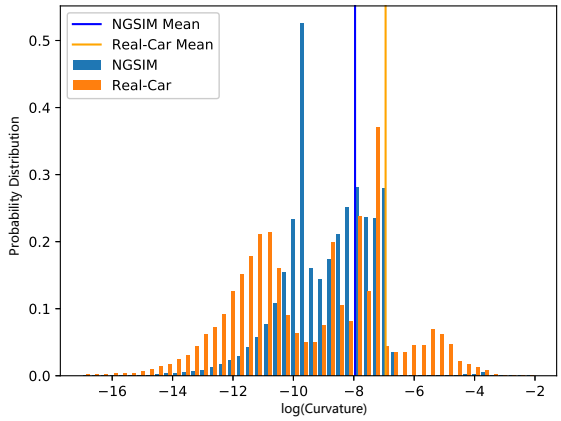

Figure 7: Lane curvature distribution for NGSIM & Autonomous Driving datasets.

Assume,

$$\tau_1 = [(x1_1, y1_1), ..., (x1_n, y1_n)],$$
$$\tau_2 = [(x2_1, y2_1), ..., (x2_n, y2_n)].$$

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 $\phi(a; \mu, \sigma)$ is the probability under a Gaussian distribution with mean μ and standard deviation σ.

The prefect prediction has likelihood of 1. A bigger likelihood is desired.

6.2.3 Model Comparison. On this dataset, we focus on comparison between different IOC based prediction methods. The Continuous Inverse Optimal Control (CIOC) is used as a baseline. Table 2 shows our sample-based method is better and more stable than COIC. The training likelihood curve in Figure 7 also suggests a lower stability for CIOC. Langevin Sampling and gradient descent sampling yield similar results. In 'Lan NO Noise' we eliminate the noise term in Langevin Sampling making it a gradient descent. Figure 8 are randomly selected prediction visualization.

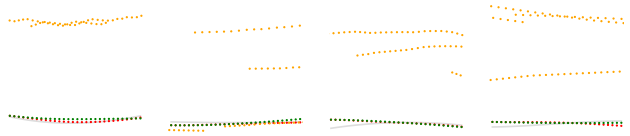

Figure 8: Predicted Trajectories from Autonomous Driving dataset.

Method	CIOC	iLQR	Langevin	Lan NO Noise
Training Likelihood	0.795	0.745	0.799	0.800
Testing Likelihood	0.653	0.745	0.810	0.810
Training RMSE	0.789	0.871	0.767	0.767
Testing RMSE	0.987	0.786	0.660	0.660

Table 2: Comparison result with hand-crafted cost function.

Due to the limited amount of data, the number of corner cases is relatively small. As a result, the testing result can be better than training. However, we could still observe that higher likelihood leads to better prediction in corner cases based on analysis of the predicted trajectories.

6.2.4 Cost Function Comparison. We then evaluate different NN-augmented cost functions. Since the NN-augmented cost function is defined on the full trajectory rather than each single frame, we use Langevin Sampling to train and test.

Table 3 shows the result for different cost function setting. As for other neural network structures, a marginal improvement is achieved by 'NN as transformer', while 'NN as residual' decreases the accuracy. CNN is the only method which have overfitting problem, with small training error but large testing error.

In this experiment we showed that the EBM model is able to handle the neural network design effectively. However, current network designs have not achieved a substantial improvement on the accuracy, which is one of our goals in near future.

Method	Human-defined	NN as transformer	NN as residual	NN as residual to each	CNN
Training Likelihood	0.800	0.799	0.80	0.773	0.814
Testing Likelihood	0.810	0.794	0.804	0.772	0.733
Training RMSE	0.767	0.668	0.823	0.802	0.626
Testing RMSE	0.660	0.576	0.700	0.713	0.778

Table 3: Comparison result using Langevin Sampling with different cost function settings.

6.3 Synthetic Examples

Corner cases are important for model evaluation. Therefore, we construct 4 typical corner cases to test our model. Figure 9 shows the predicted trajectory for several synthetic examples.

The upper 2 graphs are sudden-brake, where the orange vehicle in front of the ego car is braking. The left scene has no car alongside, so the predicted control is to change the lane. The right scene has cars alongside, which is why the ego car trigger braking.

The bottom-left graph is another car trying to cut-in to the current lane. Our model plans a trajectory to avoid the collision, and even performs an overtake maneuver.

The bottom-right graph shows a large lane curvature, yet our model still performs good lane following.

For all graphs, green points represent the predicted trajectory of the Langevin Sampling method, orange points show other vehicles and grey lines show the lane.

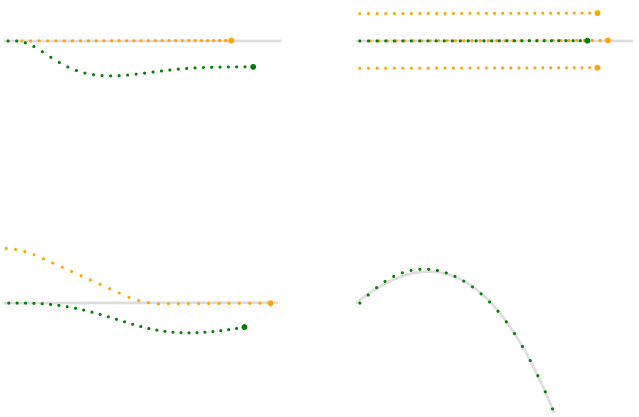

Figure 9: Predicted Trajectory for synthetic examples.

In short, our IOC algorithm via Langevin Sampling is capable of learning a reasonable cost function to avoid collision and handle cut-in situations.

7 CONCLUSION AND FUTURE WORK

In this paper, we implement continuous IOC with energy-based models in order to learn trajectory prediction. In comparison to other prediction methods, IOC-based methods show a large improvement by taking vehicle kinematics into account. In our experiments with an Autonomous Driving dataset, our method with Langevin Sampling is shown to generate better predictions with higher stability.

We design multiple neural network structures to augment a hand-designed cost function. Our experiments show energy-based model along with Langevin Sampling is capable to handle complex cost function. However, current results do not show great improvement thus introduce instability. As a next step, we plan to synthesize more trajectories per epoch and collect more data for training.

EBM is very promising for inverse optimal control, because as a descriptive model it is capable of representing 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, etc. Currently, we only predict the individual trajectories, while we assume other vehicles are driving by constant velocity. However, there is a major problem that vehicles are actually interacting to each other. For example, a vehicle behind may try to overtake if the front one trigger the break. In this case, the ego car should not change the lane. Therefore, multi-agent planning is a direction to explore. One possible implementation is to calculate the joint trajectory distribution for all moving agents, and sample multiple trajectories at the same time. Assume we have K agents and each of them has a trajectory τ_i, then,

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

The cost function of each agent shares the same parameters. Notice that the cost function for one vehicle is dependent on the information on the others. We plan to focus on various generalizations of energy-based models in future work, with the belief that it has great potential for application to trajectory prediction and optimal control.

APPENDIX

Detailed definition of hand-crafted cost function

The human-crafted cost function are defined as 10 components,

- The distance to the goal (x and y).
- The distance to the center of the lane.
- The penalty to collision to other vehicle. It is inversely proportional to distance to other vehicle.
- The L2-norm of acceleration and steering
- The L2-norm for difference of acceleration and steering between two frames.
- The heading angle to lane.
- The difference to speed limit.

The goal is set as keeping the lane by the speed of speed limit. For initialization, we calculate all these components in training sample and normalized them in to same scale.

REFERENCES

[1] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. 2016. Social LSTM: Human Trajectory Prediction in Crowded Spaces. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*.

[2] Georges S. Aoude, Brandon D. Luders, Joshua M. Joseph, Nicholas Roy, and Jonathan P. How. 2013. Probabilistically Safe Motion Planning to Avoid Dynamic Obstacles with Uncertain Motion Patterns. *Auton. Robots* 35, 1 (July 2013), 51–76. DOI: http://dx.doi.org/10.1007/s10514-013-9334-3

[3] Saurabh Arora and Prashant Doshi. 2018. A Survey of Inverse Reinforcement Learning: Challenges, Methods and Progress. *arXiv preprint arXiv:1806.06877* (2018).

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

[5] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. 2016. End to End Learning for Self-Driving Cars. *CoRR* abs/1604.07316 (2016). http://arxiv.org/abs/1604.07316

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

[7] J. Colyar and J. Halkias. 2007. US highway dataset. Federal Highway Administration (FHWA), Tech. Rep. FHWA-HRT-07-030 (2007).

[8] Nachiket Deo, Akshay Rangesh, and Mohan M Trivedi. 2018. How Would Surround Vehicles Move? A Unified Framework for Maneuver Classification and Motion Prediction. *IEEE Transactions on Intelligent Vehicles* 3, 2 (2018), 129–140.

[9] Nachiket Deo and Mohan M Trivedi. 2018. Convolutional Social Pooling for Vehicle Trajectory Prediction. *arXiv preprint arXiv:1805.06771* (2018).

[10] Nachiket Deo and Mohan M Trivedi. 2018. Multi-Modal Trajectory Prediction of Surrounding Vehicles with Maneuver based LSTMs. *arXiv preprint arXiv:1805.05499* (2018).

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

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

[13] Enric Galceran, Alexander G. Cunningham, Ryan M. Eustice, and Edwin Olson. 2017. Multipolicy Decision-making for Autonomous Driving via Changepoint-based Behavior Prediction: Theory and Experiment. *Auton. Robots* 41, 6 (Aug. 2017), 1367–1382. DOI: http://dx.doi.org/10.1007/s10514-017-9619-z

[14] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.

[15] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. 2018. Social GAN: Socially Acceptable Trajectories with Generative Adversarial Networks. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*.

[16] Dylan Hadfield-Menell, Smitha Milli, Pieter Abbeel, Stuart J Russell, and Anca Dragan. 2017. Inverse Reward Design. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Curran Associates, Inc., 6765–6774. http://papers.nips.cc/paper/7253-inverse-reward-design.pdf

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition* (2016).

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

[19] Alex Kendall, Jeffrey Hawke, David Janz, Przemyslaw Mazur, Daniele Reda, John-Mark Allen, Vinh-Dieu Lam, Alex Bewley, and Amar Shah. 2018. Learning to Drive in a Day. *CoRR* abs/1807.00412 (2018). http://arxiv.org/abs/1807.00412

[20] Markus Kuderer, Shilpa Gulati, and Wolfram Burgard. 2015. Learning driving styles for autonomous vehicles from demonstration. In *Proceedings of the IEEE International Conference on Robotics Automation (ICRA)*, Vol. 134. Seattle, USA.

[21] Alex Kuefler, Jeremy Morton, Tim Wheeler, and Mykel Kochenderfer. 2017. Imitating driver behavior with generative adversarial networks. In *Intelligent Vehicles Symposium (IV), 2017 IEEE*. IEEE, 204–211.

[22] Namhoon Lee, Wongun Choi, Paul Vernaza, Christopher B Choy, Philip HS Torr, and Manmohan Chandraker. 2017. Desire: Distant future prediction in dynamic scenes with interacting agents. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 336–345.

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

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

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

[26] Mathew Monfort, Anqi Liu, and Brian D. Ziebart. 2015. Intent Prediction and Trajectory Forecasting via Predictive Inverse Linear-quadratic Regulation. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI'15)*. AAAI Press, 3672–3678. http://dl.acm.org/citation.cfm?id=2888116.2888226

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

[28] Ellis Ratner, Dylan Hadfield-Menell, and Anca D. Dragan. 2018. Simplifying Reward Design through Divide-and-Conquer. In *Robotics: Science and Systems*.

[29] Amir Sadeghian, Vineet Kosaraju, Ali Sadeghian, Noriaki Hirose, and Silvio Savarese. 2018. SoPhie: An Attentive GAN for Predicting Paths Compliant to Social and Physical Constraints. *arXiv, CoRR* abs/1806.01482 (2018).

[30] Anirudh Vemula, Katharina Muelling, and Jean Oh. 2018. Social Attention: Modeling Attention in Human Crowds. In *Proceedings of the International Conference on Robotics and Automation (ICRA) 2018*.

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

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

[33] Jianwen Xie, Zilong Zheng, Ruiqi Gao, Wenguan Wang, Song-Chun Zhu, and Ying Nian Wu. 2018. Learning Descriptor Networks for 3D Shape Synthesis and Analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 8629–8638.

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

[35] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. 2008. Maximum Entropy Inverse Reinforcement Learning.. In *AAAI*, Vol. 8. Chicago, IL, USA, 1433–1438.