Accounting for Unpredictability in Autonomous Driving Behaviour

by

Sepehr Samavi

A thesis submitted in conformity with the requirements
for the degree of Master of Applied Science
Graduate Department of The Institute for Aerospace Studies
University of Toronto

Abstract

Accounting for Unpredictability in Autonomous Driving Behaviour

Sepehr Samavi
Master of Applied Science
Graduate Department of The Institute for Aerospace Studies
University of Toronto
2021

Autonomous Vehicles (AVs) need to behave like humans when interacting with them. We define unpredictability of surrounding drivers as a measure to take into account for trajectory planning and use Maximum Entropy Inverse Reinforcement Learning (IRL) to demonstrate that incorporating unpredictability into a lane change reward function provides insights on human driving behaviour. We first evaluate the IRL algorithm on a Linear Quadratic Regulator proof of concept. Then we use the IRL algorithm to model reward functions for conducting a lane change maneuver in a highway setting. We investigate whether the unpredictability of surrounding traffic will have an effect on the behaviour of the lane changing car by learning two reward functions from human data, a baseline reward function and a reward function that incorporates unpredictability. Our evaluation confirms that incorporating unpredictability results in modest improvements in explaining the behaviour of human drivers and can result in human-like AVs.

Acknowledgements

As this chapter of my life comes to a close, I have much to be thankful for. First and foremost, I am grateful for the love and support of my parents, Reza Samavi and Mahvareh Ahghari, without which no achievement of mine would be possible. I wish to also thank my supervisor Prof. Angela Schoellig who supported this project from the beginning and Prof. Florian Shkurti whose insight on imitation learning methods was invaluable.

At the beginning of this degree, I had the pleasure of collaborating with the aUToronto team. I would like to thank my colleague Keenan along with the rest of the team for the enjoyable experience of getting our self-driving car, *Zeus*, to work autonomously, whether at our weekend work sessions at UTIAS, or the field trials which spanned two countries and always happened to occur during inclement weather. I would also like to thank Prof. Tim Barfoot for his guidance.

Finally I would like to thank all the students I had a pleasure of working along at the ever-expanding Robotics community at the University of Toronto and at the Vector Institute for Artificial Intelligence. Even on the gloomiest days, our lunch-time conversations were intriguing and uplifting.

قطره‌ی دانش که بخشیدی ز پیش متصل گردان به دریاهای خویش

Cause the drop of knowledge which Thou gavest (us)
heretofore to become united with Thy seas. [1]

Jalāl ad-Dīn Rūmī

[1] Translated from Persian by R. A. Nicholson, *The Mathnawi of Jalalu'ddin Rumi, Vol. 1*, Messrs. Luzac & Co., London, UK 1925.

Contents

1 Introduction **1**

1.1 Contributions 2

1.2 Thesis Structure 3

2 Related Work **4**

2.1 Human-like behaviour models for AVs 4

2.2 Predictability, Legibility and Explainability 6

3 Theory **7**

3.1 Optimal Control Preliminaries 8

3.2 Inverse Reinforcement Learning Preliminaries 10

3.3 Probabilistic Inverse Reinforcement Learning 12

3.3.1 Information Theoretic Preliminaries 12

3.3.2 Maximum Causal Entropy as Softened Bellman Equations 18

3.3.3 Maximum Causal Entropy for IRL 20

3.4 Conclusion 24

4 IRL for LQR **25**

4.1 Background 25

4.1.1 Review of the forward LQR solution 26

4.2 Experimental Setup 27

4.2.1 System and Reward Structure 28

4.3 Synthetic Data Generation Method 29

4.4 Implementation Details 30

4.4.1 IRL Objective and Optimization 30

4.4.2 Numerical Considerations 32

4.5 Results and Discussion 34

4.5.1 Experiment 1: IRL with expert trajectories generated with perfect policy . . . 34
4.5.2 Experiment 2: IRL with expert trajectories generated with stochastic policies . . . 35

5 Unpredictability in Lane Changes **40**
5.1 Overview . . . 40
5.2 Lane Change Problem Formulation . . . 41
5.2.1 Ego-vehicle Dynamics Model . . . 41
5.2.2 Lane Change Environment . . . 42
5.2.3 Baseline Reward Function . . . 42
5.3 Unpredictability Formulation . . . 45
5.3.1 Unpredictability Metric . . . 45
5.3.2 Unpredictability Reward Feature . . . 47
5.4 Dataset Generation . . . 47
5.4.1 Data Smoothing . . . 48
5.4.2 Extraction of Expert Lane Change Trajectories . . . 48
5.4.3 Calculation of Ego-vehicle Dynamics . . . 49
5.4.4 Characterization of Lane Change Environment . . . 49
5.4.5 Dataset Divisions . . . 51
5.4.6 Trajectory Initial Position Normalization . . . 52
5.5 Implementation Details . . . 52
5.5.1 Forward Algorithm Implementation Details . . . 52
5.5.2 IRL Algorithm Implementation Details . . . 52
5.6 Algorithm Verification with Synthetic Data . . . 56
5.6.1 Generating Synthetic Trajectorires . . . 56
5.6.2 IRL Model Training . . . 57
5.6.3 Verification Results . . . 57
5.6.4 Tuning of Scaling Parameter . . . 60
5.7 IRL Model Training . . . 60
5.8 Results and Discussion . . . 62
5.8.1 Quantitative Improvements . . . 63
5.8.2 Qualitative Analysis . . . 65

6 Conclusions **69**
6.1 Future Work . . . 70

Bibliography **71**

A Auxilliary Proofs and Theorem **78**

B Towards an Autonomous Vehicle Scoring Framework **83**

List of Tables

4.1 Results of IRL on our LQR System 35

5.1 Summary of the longitudinal separation critera for trajectory datasets . . 51
5.2 Summary of the lateral separation critera for trajectory datasets 51
5.3 Summary of reward parameter values for synthetically generated data . . 57
5.4 IRL model performance on 127 trials using all five synthetic datasets . . 58
5.5 Summary of IRL parameter values and model performance with different reward coefficients for synthetic dataset 60 60
5.6 Seperate Traffic, Lateral 63
5.7 Combined Traffic, Lateral 64
5.8 Separate Traffic, Longitudinal 64
5.9 Combined Traffic, Longitudinal 65
5.10 Summary of average model performance improvement by incorporating unpredictability. These results only include runs where $\theta_{z,f} > 10^{-}5$. . . . 65
5.11 Learned Parameters for Combined Datasets 65
5.12 Improvement for Combined Training Datasets 66
5.13 Improvement for Combined Test Datasets 66

List of Figures

4.1 The effect of parameter scaling on maximum likelihood offset for a system with a two-dimensional state and single action. The LQR parameter r_1 is fixed and the parameters q_1, q_2 are learned. The blue contour illustrates the likelihood function for parameters scaled by $r_1 = 10^5$ and the red contour lines illustrate the likelihood function without scaled parameters. 33

4.2 Evolution of reward parameter estimates over the IRL optimization process. Note that all the values on the q_1 and q_2 axes were scaled for performing the optimization as described in Section 4.4.1. Unscaled values are illustrated on this graph. 36

4.3 Trajectories generated by LQR with ground truth parameters, $\boldsymbol{\theta}_e$ (blue), initial parameters for IRL, $\boldsymbol{\theta}_0$ (orange), and final paramters generated by IRL, $\boldsymbol{\theta}_f$ (green) (c). The values of the state at each time step are illustrated for the first dimension (a) and the second dimension (b). . . . 38

4.4 Error in learned parameters for different amounts of expert trajectories used for IRL on plotted on a log-log scale. In this set of experiments, all example trajectories start from the same initial condition, $\mathbf{x}_0 = [10, 10]^\top$ 39

4.5 Error in learned parameters for different amounts of expert trajectories used for IRL on plotted on a log-log scale. In this set of experiments, each example trajectories start from a randomly selected initial condition, sampled from a uniform distribution, $\mathbf{x}_0 = [x_0^1, x_0^2]^\top$, where $x_0^1, x_0^2 \sim \mathcal{U}(-10, 10)$ 39

5.1 Blockdiagram of how we intent to learn and compare the reward function learned using only the baseline features and the reward function learned that adds the unpredictability feature. 41

5.2 Illustration of unpredictability metric. We define unpredictability at time k as the mean predictive error of the prediction made by the model at time $k - t_n$. 46

5.3 The distribution of lane labels in the NGSIM I-80 (a) and USA-101 (b) datasets. The ranges of the axes are different in order to highlight the distribution of lane labels along the x-axis. Lanes 1-6 are thru-lanes and Lanes 7 and 8 are on and off ramps, respectively. Traffic flows in the direction of increasing y coordinate. The black lines mark the longitudinal separation defined by us. 50
5.4 Snapshot of a synthetically generated lane change trajectory. Scatter points illustrate the position of the ego-vehicle (blue) and adjacent vehicles (grey) at time step 39. From each scatter point, we illustrate a trail of the past 10 time steps. 57
5.5 Loss optimization for IRL with synthetic dataset 57 and initial condition $\boldsymbol{\theta}_0 = [1, 1, 1, 1]^\top$. 58
5.6 Comparison of the states (5.6a to 5.6d) and actions (5.6e, 5.6f) of the synthetically generated expert lane change trajectory (blue) with trajectories generated by optimizing rewards parametrized by initial guess of parameters, $\boldsymbol{\theta}_0$ (orange), and parameters learned using IRL, $\boldsymbol{\theta}_f$ (green). 59
5.7 IRL loss optimization for baseline model and model incorporating unpredictability for highway `usa101`, traffic congestion time slot `t2`, and geometry `lat5`. 63
5.8 Comparison of the human expert lane change trajectory (blue) with trajectories generated by optimizing rewards parametrized by the baseline reward parameters, $\boldsymbol{\theta}_{f,b}$ (orange), and reward parameters that incorporate unpredictability, $\boldsymbol{\theta}_{f,w}$ (green). 66
5.9 Comparison of the states from human-demonstrated expert lane change trajectory (blue) with trajectories generated by optimizing rewards parametrized by the baseline reward parameters, $\boldsymbol{\theta}_{f,b}$ (orange), and reward parameters that incorporate unpredictability, $\boldsymbol{\theta}_{f,w}$ (green). 67
5.10 Comparison of the actions of the human-demonstrated expert lane change trajectory (blue) with trajectories generated by optimizing rewards parametrized by the baseline reward parameters, $\boldsymbol{\theta}_{f,b}$ (orange), and reward parameters that incorporate unpredictability, $\boldsymbol{\theta}_{f,w}$ (green). 68

Chapter 1

Introduction

An increasing number of autonomous vehicles (AVs) have started to be deployed in mixed traffic situations (e.g., Waymo's robo-taxi pilot), where the AV shares the same road with human-driven vehicles (HVs). The AV research community and the industry (e.g., Waymo, GM, Apple) have put considerable efforts into enhancing control and planning algorithms of these vehicles. Little is known about how the presence of an AV in mixed traffic situations changes the behaviour of the human drivers and in turn how an AV should react to these changes [19]. While driving is considered a nearly ubiquitous task for humans, driving behaviour of individuals varies and has considerable subtlety [21]. For example, an experienced driver with a priori knowledge of the driving environment will behave differently with respect to decelerating and accelerating when performing a lane change. Using this priori knowledge, the driver may decide to cut-off another vehicle (which can be an AV or HV) during the lane change maneuver and introduce a safety risk and in turn garner a reaction from surrounding vehicles.

For the safe introduction of AVs into mixed traffic, behaviour models will need to understand and predict the chain of behavioural actions and reactions between drivers. Specifically, the AV will need to predict the behaviour of other vehicles and act in a way such that surrounding vehicles will be able to predict its behaviour.

In this thesis we propose *unpredictability* as a measure for more human-like AV behaviour planning. We use performance metrics of an off-the-shelf vehicle trajectory prediction model [14] through time as a measure of the unpredictability of a human driver. We assume that if the off-the-shelf model performs poorly on a particular car at a particular time, then that car is behaving unpredictably and vice versa. We propose incorporating unpredictability as a measure of how an AV should act around a particular human driver. We analyze human lane change behaviour using an Inverse Reinforcement Learning (IRL) approach to demonstrate that incorporating this unpredictability measure can

better explain human driving behaviour. Therefore, incorporating the unpredictability measure into a planning framework of an AV can result in more human-like behaviour.

We first summarize the theoretical foundation of Maximum Entropy IRL approaches [60] for dynamical systems with continuous control and action spaces [33]. We then use a Linear Quadratic Regulator (LQR) toy example to investigate how well the IRL algorithm [33] can recover reward parameters in the presence of noise. Finally, we use the IRL algorithm to model reward functions for conducting a lane change maneuver in a highway setting. In order to test the hypothesis that the unpredictability of surrounding traffic will have an effect on the behaviour of the ego-car, we learn two reward functions from human data, a baseline reward function and a reward function that incorporates unpredictability.

1.1 Contributions

The main contribution of this thesis is defining a measure of unpredictability based on an off-the-shelf trajectory prediction model. We present two reward functions for modelling the lane changes: a baseline reward function and a reward function that incorporates unpredictability. We demonstrate that the model incorporating the unpredictability measure can better explain human lane change trajectories.

In addition, we make the following three contributions,

1. We present the underlying theory for IRL for an audience with a background in control theory by compiling the relevant literature from the fields of Information Theory and Optimal Control.

2. We design and conduct an experimental analysis of the performance of the IRL algorithm [33] on an LQR problem. We evaluate how accurately the algorithm can recover reward parameters as a function of noise and the number of synthetic demonstration trajectories. We also present numerical tricks and practical considerations for implementing the algorithm.

3. We build a dataset of smoothed lane change trajectories extracted from the NGSIM [12, 11] datasets. The trajectories include measurements of the state and actions in a kinematic unicycle model.

In addition to the contributions of this thesis listed above, in the Masters program I also made the following contributions.

1. In a project in collaboration with Geotab Inc., we studied the problem of evaluating level 4 autonomous vehicles on public roads using a telematics device. The findings of this study are presented in a technical report. A summary of the report can be found in Appendix B.

2. In a collaborative project for the University of Toronto's self-driving car team, aUToronto, we designed a pedestrain detection and tracking system, published in [9]. I was also deputy lead for the team during the second year of the AutoDrive Challenge. The AV design and competition results for the year were published in [8].

1.2 Thesis Structure

We begin by presenting related work for the main application of this thesis: unpredictability as a measure for more human-like driving in Chapter 2. In Chapter 3 we provide a detailed overview of the theoretical foundations of Inverse Reinforcement Learning (IRL), also known as Inverse Optimal Control (IOC), including the derivation of the algorithm that we use to test the main hypothesis of this thesis. In Chapter 4 we conduct a set of experiments that applies the IRL algorithm to recovering LQR parameters given a varying number of noisy demonstration trajectory data. In Chapter 5, we use the IRL algorithm to model reward functions for conducting a lane change maneuver in a highway setting. We conclude this thesis in Chapter 6.

Chapter 2

Related Work

In this chapter we present literature related to the main application of this thesis: unpredictability as a measure for more human-like driving. We use Inverse Reinforcement Learning (IRL) and Optimal Control to evaluate if human drivers take unpredictability into account when driving. We also evaluate the performance of the IRL algorithm that we use on a Linear Quadratic Regulator (LQR) system. Note that the theoretical background literature related to IRL and Optimal Control and a summary of work related to IRL with LQR systems are presented in Chapters 3 and 4, respectively.

2.1 Human-like behaviour models for AVs

Designing human-like behaviour models for planning and control in Autonomous Vehicles (AVs) that operate in mixed Human-driven Vehicle (HV) and AV environments is explored from different perspectives.

In the classical approach, the planning modules in AVs usually take information about the future of their surrounding environments (e.g. location of adjacent vehicles, pedestrians, road geometry) from prediction models and then produce collision-free trajectories [16, 48, 35]. A major shortcoming of such cascaded prediction and planning approaches is that it does not account for the complex interactions and interdependencies between agents that are operating near each other [49]. Progress has been made in modelling interdependencies between agents within the prediction models (e.g. [2, 14]). Alahi et al. [2] propose a *social* pooling layer within a Long-Short Term Memory (LSTM) pedestrian trajectory prediction network to account for interdependencies between the agents. Deo et al. [14] extend this principle to apply to vehicles on a highway. However these prediction models continue to be completely independent of the planning models in the ego-vehicle.

Among works that incorporate predictions about other agents into planning and control, a number of recent methods use game theoretic techniques to model interactions between AVs and HVs [47, 45, 50, 18]. Sadigh et al. [47] design a game between an AV and an HV to enable the AV to influence the HV's behaviour. The authors in [45] use a game theoretic formulation to generate AV behaviour that is empathetic towards an interacting HV. These models focus on one-on-one interactions, ignoring other surrounding vehicles. Schwarting et al. [50] use psychological metrics to incorporate estimates of HVs' personality into a game theoretic framework for performing highway merging maneuvers and making uncontrolled intersection decisions. Fisac et al. [18] propose a hierarchical game theoretic framework in order to make modelling interactions between agents tractable.

Other methods focus on imitating human driving styles by learning behaviour models from human data using Maximum Entropy Inverse Reinforcement Learning (MaxEnt IRL) [61]. With this method, the behaviour model is represented as a cost function that is learned from expert demonstrations. Kuderer et al. [32] use IRL to learn human-like velocity and acceleration profiles for highway driving. However, their model is focused on modelling the preferred driving style for passengers in the AV rather than interactions with other vehicles. Levine and Koltun [33] extend MaxEnt IRL to be used in cases where expert demonstrations are only locally optimal and noisy. They show that it is possible to learn cost functions for evasive and aggressive driving styles in a simulator.

What is common among the three perspectives described above is that these proposals focus on presenting novel methods for optimizing AV behaviour in order to have human-likeness. We take one step further and study *what* we need to optimize to capture human-like driving behaviour.

Other works that focus on the *what* aspect include Buckman et al. [7] who propose maximizing the visibility of the ego-car in highway and blind intersection scenarios. In a driving simulation, they demonstrate that a visibility-aware AV slows down before entering HVs' blindspots. Sun et al. [53] formalize a notion of being courteous to other interacting drivers. They hypothesize that human drivers are in fact also courteous. The authors test their hypothesis using an IRL analysis, in the same way as this thesis intends to do. Rather than formalizing a notion for courteousness, we define the unpredictability of surrounding traffic as a new parameter that needs to be considered for optimizing AV behaviour.

Hayashi et al. [21] present a Reinforcement Learning (RL) framework for driving in a simulator based on a quantified notion of unpredictability in the scene they are driving in. They claim that the more unpredictable the scene, the more un-humanlike the behaviour

of the ego-car has been. Thus, by minimizing unpredictability one can build a human-like driving model. They define a reward function that is only based on the prediction of adjacent vehicles, then train an RL agent to drive. However their model is exclusively trained and tested in a simulator, and does not use any human data. Therefore, their notion of unpredictability is limited to the driver models that are coded in the simulator.

2.2 Predictability, Legibility and Explainability

The concepts of predictability, legibility, and explainability have also been studied in the context of Human Robot Interaction (HRI) literature [15, 29, 34]. Dragan et al. [15] formalize predictable robot planning as motion that is unsurprising to a human observer and legible robot planning as motion that makes it easy for humans to infer the robot's intent. Koppenborg et al. [29] and Lichtenthaler et al. [34] investigate factors that may affect legibility. The results from these studies are generated from user studies, thus, the scope of their studies is limited by safety considerations. As opposed to conducting human studies, in this thesis we use human-generated driving data and IRL to investigate whether unpredictability affects human driving behaviour.

We define the unpredictability of surrounding traffic as a new parameter that needs to be considered for optimizing AV behaviour. We then use the extension of MaxEnt IRL proposed by Levine and Koltun [33] to learn cost functions for lane changes on a highway, which incorporate unpredictability. A detailed treatment of the theoretical foundations of MaxEnt IRL [61] and its extension to locally optimal demonstrations [33] is presented in Chapter 3.

Chapter 3

Theory

Optimal Control and Reinforcement Learning (RL) are powerful tools for modelling natural learning phenomena. Both methods rely on the assumption that a reward function, as opposed to a policy (i.e. a direct mapping between action and state), is the most succinct and efficient definition of a task. Most methods rely on a known reward function. However, when our goal is to perform a task that is demonstrated successfully in nature, we should consider the reward function as an unknown function and empirically estimate it.

In this chapter we provide the theoretical background of Inverse Reinforcement Learning (IRL), also known as Inverse Optimal Control (IOC). In IRL, we assume that an agent is executing a control policy by optimizing some underlying reward function, $\mathcal{R}^*(\mathbf{x}, \mathbf{u})$. Given expert demonstrations of a task, our goal is to find an estimate of the underlying reward function that results in behaviour that resembles the expert. However given sub-optimal demonstrations in the presence of noise, many different reward parameters could result in behaviour resembling the expert. Thus, IRL is ill-posed in most formulations [43].

The IRL problem was first described by Kalman in 1964 [27], specifically for the case of a linear plant, linear constant scalar control, and full state observability. However, in this thesis we focus on the inverse of more general optimal control and reinforcement learning problems, resembling the formulations presented in litarature by Ng and Russel [43], Abbeel and Ng [1], Ziebart et al. [61], and Levine and Koltun [33].

In Section 3.1 we present an overview of Optimal Control, followed by the formulation of IRL in Section 3.2. Then, in Section 3.3 we present the Maximum Causal Entropy (MCE) probability distribution, in which inference is analogous to solving the optimal control problem. We finally present a probabilistic learning approach to solving IRL for fixed horizon control tasks with continuous state and action spaces and locally optimal

demonstrations.

3.1 Optimal Control Preliminaries

We first review the Optimal Control problem [5] as it relates to this chapter. We define a fixed-horizon control task of length K.

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k, \mathbf{e}_k) \quad \text{dynamics} \tag{3.1a}$$
$$\mathbf{x}_k \in \mathcal{X} \quad \text{state space} \tag{3.1b}$$
$$\mathbf{u}_k \in \mathcal{U} \quad \text{action space} \tag{3.1c}$$
$$\mathbf{e}_k \in \mathcal{D} \quad \text{disturbance space} \tag{3.1d}$$
$$\mathbf{e}_k \sim p(\mathbf{e}_k|\mathbf{x_k}, \mathbf{u_k}) \quad \text{disturbance probability distribution} \tag{3.1e}$$
$$\pi = \{\mu_0, \ldots, \mu_{K-1}\} \quad \text{policy} \tag{3.1f}$$
$$\mathbf{u}_k = \mu_k(\mathbf{x}_k) \in \mathcal{U}(\mathbf{x}_k) \subset \mathcal{U}, \forall \mathbf{x}_k \in \mathcal{X} \quad \text{set of all admissible policies} \tag{3.1g}$$
$$\mathcal{R}_k(\mathbf{x}_k, \mathbf{u}_k) : \mathcal{X} \times \mathcal{U} \mapsto \mathbb{R}^+ \quad \text{stage reward} \tag{3.1h}$$
$$\mathcal{R}_k(\mathbf{x}_K) : \mathcal{X} \mapsto \mathbb{R}^+ \quad \text{terminal reward} \tag{3.1i}$$

We assume that at each time step, $k = 0, \ldots, K-1$, we receive a stage reward, and at the final time step, a terminal reward $\mathcal{R}_K(x_K)$. The optimal control problem is then formalized in the following maximization problem,

$$\begin{aligned} \text{maximize} \quad & \mathbb{E}_{\mathbf{e}_t}\left[\mathcal{R}_K(\mathbf{x}_K) + \sum_{k=0}^{K-1} \mathcal{R}_k(\mathbf{x}_k, \mathbf{u}_k)\right] \\ \text{subject to} \quad & \mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k, \mathbf{e}_k) \\ & \mathbf{u}_k = \mu_k(\mathbf{x}_k) \\ & \mathbf{x}_0 \text{ given} \end{aligned} \tag{3.2}$$

where our task is to find the policy that maximizes the reward, called the optimal policy $\pi^* = \{\mu_0^*(\mathbf{x}_0), \ldots, \mu_{K-1}^*(\mathbf{x}_{K-1})\}$.

Theorem 1 (**Bellman's Principle of Optimality**[5] [a]). *Let $\pi^* = \{\mu_0^*, \ldots, \mu_{K-1}^*\}$ be the optimal policy for the overall problem* (3.2)*. Assume in the process of applying π^*, a state $\mathbf{x}_k$ occurs at time k, with positive probability. Consider the sub-problem*

whereby we are at $\mathbf{x}_k$ *and we want to maximize the sub-problem*

$$\mathbb{E}_{\mathbf{e}_k}\left[\mathcal{R}_K(\mathbf{x}_K) + \sum_{t=k}^{K-1} \mathcal{R}_t(\mathbf{x}_t, \mathbf{u}_t)\right].$$

Then the truncated policy $\{\mu_k^*, \ldots, \mu_{K-1}^*\}$ *is optimal.*

[a] Originally formulated by Bellman [4].

The Principle of Optimality (Theorem 1) allows the maximization over $\pi = \{\mu_0, \ldots, \mu_{K-1}\}$ in 3.2 to be converted to K maximizations, each over a single time step. We consider the *value* of a policy, V_k^π, at time step k.

$$V_k^\pi(\mathbf{x}_k) = \mathbb{E}_{\mathbf{e}_k}[\mathcal{R}_K(\mathbf{x}_K) + \sum_{i=k}^{K-1} \mathcal{R}_i(\mathbf{x}_i, \mathbf{u}_i)] \tag{3.3}$$

The *value function* calculates the expected *reward-to-go* at time step k, i.e. the reward that would be obtained if we continue along the trajectory by following the truncated policy $\{\mu_k, \ldots, \mu_{K-1}\}$. The expectation is over the disturbance probability distribution, $p(\mathbf{e}_k|\mathbf{x_k}, \mathbf{u_k})$. We also define the quality function (Q-function), which depends on the state and action at time step k,

$$Q_k^\pi(\mathbf{x}_k, \mathbf{u}_k) := \mathcal{R}_k(\mathbf{x}_k, \mathbf{u}_k) + \mathbb{E}_{\mathbf{e}_k}[V_{k+1}^\pi(\mathbf{x}_{k+1})]. \tag{3.4}$$

The Q-function calculates the expected *reward-to-go* of choosing an action $\mathbf{u}_k$ at state $\mathbf{x}_k$, given that after the next time step the truncated policy $\{\mu_{k+1}, \ldots, \mu_{K-1}\}$ will be followed. Under the optimal policy, π^*, we can define the value function as a maximization of the quality function,

$$V_k^{\pi^*}(\mathbf{x}_k) = \max_{\mathbf{u}_k} Q_k^{\pi^*}(\mathbf{x}_k, \mathbf{u}_k). \tag{3.5}$$

The value of a state $\mathbf{x}_k$ is therefore determined by choosing the action that maximizes quality. Equations (3.4, 3.5) are commonly referred to as the Bellman Equations. We can find an optimal policy by calculating the Bellman Equations at the end of the trajectory and iterating backwards in time to solve Equation (3.6). This iteration is called the Dynamic Programming Algorithm (DPA).

Definition 1 (**Dynamic Programming Algorithm** [5][a])**.** *Starting at time step* K *with*

$$V_K^{\pi^*}(\mathbf{x}_K) = \mathcal{R}_K(\mathbf{x}_K)$$

and iterating backward in time from $k = K-1, \ldots, 0$, then the value for each time step $\mathbf{x}_k$ is given recursively by,

$$V_k^{\pi^*}(\mathbf{x}_k) = \max_{\mathbf{u}_k} \mathcal{R}_k(\mathbf{x}_k, \mathbf{u}_k) + \mathbb{E}_{e_k}[V_{k+1}^{\pi^*}(\mathbf{x}_{k+1})]. \tag{3.6}$$

If $\mathbf{u}_k^* = \mu_k^*$ for all k optimizes the right side of (3.3) *then the policy $\pi^* = \{\mu_0^*, \ldots, \mu_{K-1}^*\}$ is optimal.*

[a] Originally formulated by Bellman [4].

3.2 Inverse Reinforcement Learning Preliminaries

Inverse Reinforcement Learning (IRL) also known as Inverse Optimal Control (IOC) is defined as the problem of learning the reward function of a task from demonstrations of optimal or near-optimal behaviour [43]. In IRL, we start with System 3.1, but with an unknown reward function. We have access to a set of N demonstration trajectories (aka expert trajectories), $\mathcal{D} = \{(\mathbf{x}^{(1)}, \mathbf{u}^{(1)}), \ldots, (\mathbf{x}^{(N)}, \mathbf{u}^{(N)})\}$, where $\mathbf{x}^{(i)} = [\mathbf{x}_0^\top, \ldots, \mathbf{x}_{K-1}^\top]^\top$ and $\mathbf{u}^{(i)} = [\mathbf{u}_0^\top, \ldots, \mathbf{u}_{K-1}^\top]^\top$ for $i = 1, \ldots, N$. In the broadest sense, the goal of IRL is to find a reward function, $\mathcal{R}_k(\mathbf{x}_k, \mathbf{u}_k)$, under which optimal behaviour [1] matches the demonstrated behaviour of the experts, irrespective of the true process that generated the demonstrations. Recovering such a reward, was first characterized by Ng and Russell [43], however for a systems with finite or infinite state spaces, $\mathcal{X}$, finite action spaces, $\mathcal{U}$, and an infinite time horizons. They show that demonstrated behaviour can be optimal under many different reward weights including the trivial case $\mathcal{R} \equiv 0$, therefore the formulation of IRL is ill-posed [43].

To formulate a general IRL problem, we define $p \in \mathbb{Z}^+$ *reward features*, $\phi_i : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$, where $i = 1, \ldots, p$ is function of the state and input at each time step. We formulate the stage reward as a linear combination of the features, i.e.

$$\mathcal{R}_k(\mathbf{x}_k, \mathbf{u}_k) = \boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{x}_k, \mathbf{u}_k) \tag{3.7}$$

where $\boldsymbol{\phi}(\mathbf{x}_k, \mathbf{u}_k) = [\phi_1(\mathbf{x}_k, \mathbf{u}_k), \ldots, \phi_p(\mathbf{x}_k, \mathbf{u}_k)]^\top$ is a vector of the features at time step k, and $\boldsymbol{\theta} \in \mathbb{R}^{p+}$ is a vector of parameters. For the remainder of this thesis we will focus on reward functions of this form.

Abbeel and Ng [1] formulate the IRL problem as feature matching between the learned model and the observed behaviour. Let $P_{\mathcal{D}}(\mathbf{x}^{(i)}, \mathbf{u}^{(i)})$ be the probability of trajectory i in

[1] i.e. states and actions generated by solving the optimal control problem Equation (3.2) given the learned reward function.

the dataset $\mathcal{D}$. We can write feature matching as,

$$\begin{aligned}\mathbb{E}_{\mathbf{e_k}}[\boldsymbol{\phi}] &= \sum_{i=1}^{N} P_{\mathcal{D}}(\mathbf{x}^{(i)}, \mathbf{u}^{(i)}) \sum_{k=0}^{K-1} \boldsymbol{\phi}(\mathbf{x}_k^{(i)}, \mathbf{u}_k^{(i)}) \\ \mathbb{E}_{\mathbf{e_k}}[\boldsymbol{\phi}] &= \mathbb{E}_{P_{\mathcal{D}}}[\boldsymbol{\phi}].\end{aligned} \tag{3.8}$$

For all features we want the total feature expectation over the control horizon under the learned reward (left) to match the empirical feature counts in dataset $\mathcal{D}$ (right). The authors demonstrate that feature-matching is equivalent to matching the value function (Equation 3.5) over the entire horizon. This finding is summarized in a theorem.

Theorem 2 (Abbeel and Ng [1])**.** *For a system with a reward function of the form* (3.7)*, a policy* π *matches feature counts with observed feature counts i.e.* $\mathbb{E}_{\mathbf{e_k}}[\boldsymbol{\phi}] = \mathbb{E}_{P_{\mathcal{D}}}[\boldsymbol{\phi}^{(i)}]$ *if and only if the policy also matches the total expected value over the entire horizon with the observed value* $V_0^{\pi}(\mathbf{x}_0) = V_k^{\mathcal{D}}(\mathbf{x}_k)$*, on the unknown parameters of a reward linear in features,* ϕ_j*.*

Proof. [60]

$$\begin{aligned}\mathbb{E}_{\mathbf{e}_k}[\boldsymbol{\phi}] &= \mathbb{E}_{P_{\mathcal{D}}}[\boldsymbol{\phi}] \\ \forall \boldsymbol{\theta}, \quad \boldsymbol{\theta}^{\top}\mathbb{E}_{\mathbf{e}_k}[\boldsymbol{\phi}] &= \boldsymbol{\theta}^{\top}\mathbb{E}_{\tilde{P}_{\mathcal{D}}}[\boldsymbol{\phi}] \\ \forall \boldsymbol{\theta}, \quad \mathbb{E}_{\mathbf{e}_k}\left[\boldsymbol{\theta}^{\top}\sum_k \boldsymbol{\phi}_k(\mathbf{x}_k, \mathbf{u}_k)\right] &= \mathbb{E}_{\tilde{P}_{\mathcal{D}}}\left[\boldsymbol{\theta}^{\top}\sum_k \boldsymbol{\phi}_k(\mathbf{x}_k, \mathbf{u}_k)\right] \\ \forall \boldsymbol{\theta}, \quad \mathbb{E}_{\mathbf{e}_k}\left[\sum_k \mathcal{R}_k(\mathbf{x}_k, \mathbf{u}_k)\right] &= \mathbb{E}_{\tilde{P}_{\mathcal{D}}}\left[\sum_k \mathcal{R}_k(\mathbf{x}_k, \mathbf{u}_k)\right] \\ \forall \boldsymbol{\theta}, \quad V_0^{\pi}(\mathbf{x}_0) &= V_k^{\mathcal{D}}(\mathbf{x}_k)\end{aligned}$$

proof of the necessary condition follows the same argument. □

Theorem 2 provides a mechanism for matching the expected total value over the horizon , $V_0^{\pi}(\mathbf{x}_0)$, between our model, parametrized by $\boldsymbol{\theta}$, and the demonstrated trajectories. However, we need to justify why matching the total expected value results in matching the expected behaviour over the entire horizon. From Definition 1 we know that if the value of the policy, $V_k^{\pi}(\mathbf{x}_k)$, is optimal for *every* time step $k = 1, \ldots, K-1$ (Equation (3.3)) then the policy is optimal. From Theorem 1, we know that if a policy, π^* is optimal for a system (3.1) over the entire horizon, then for any subset of the horizon, the associated truncated policy is also optimal. If we assume that the demonstrated trajectories in the dataset $\mathcal{D}$ are optimal and that we find parameters $\boldsymbol{\theta}$ such that we

can match the expected total value over the horizon, then we also match the value at every other time step k, $V_k^\pi(\mathbf{x}_k)$. Therefore matching the expected sum of features over the horizon results in matching the expected behaviour at each time step in the horizon.

The assumption that the demonstrated trajectories in the dataset $\mathcal{D}$ are optimal is very limiting. For real-world problems we will need to account for noise and sub-optimality in the demonstrations, and modelling error in the dynamics. Furthermore, due to the linear structure of the reward function and the selection of the features in the reward function, there may not be a set of parameters that perfectly matches the demonstrated behaviour. To address these concerns, Ng and Russell [43] and Abbeel and Ng [1] propose heuristics to make the best selection. In the next section we focus on a probabilistic approach to address this limitation.

3.3 Probabilistic Inverse Reinforcement Learning

In probabilistic IRL, we assume that there exists an underlying distribution of possible trajectories and our dataset contains samples from this distribution. We model $\mathbf{x}, \mathbf{u}$ as sequences of random variables that describe a trajectory. To simplify notation we can define the tuple $\tau := (\mathbf{x}, \mathbf{u})$ and the probability distribution function $p(\tau)$ to denote the underlying distribution of all trajectories. Our goal is to estimate $p(\tau)$ by using the dataset. In other words, we want to learn a distribution $q(\tau)$ that is the best estimate of $p(\tau)$ given $\mathcal{D}$. Estimating one distribution $p(\tau)$ with another $q(\tau)$ based on data is formulated as an optimization problem. We need to find the parameters, θ of some parametrized probability distribution, $q_\theta(\tau)$, to fit to dataset $\mathcal{D}$. The distribution $q_\theta(\tau)$ needs to account for noise and sub-optimality in the demonstrations, as well as modelling error in the formulation of $q_\theta(\tau)$. In the rest of this section, we will present an information theoretic approach to probabilistically model optimal control problems, then show how IRL can be formulated as a maximum likelihood estimation problem under this probabilistic model.

3.3.1 Information Theoretic Preliminaries

In probabilistic IRL we are interested in selecting the distribution $q_\theta(\tau)$ that is consistent with the incomplete information in the available dataset $\mathcal{D}$. To avoid introducing bias, we use information theory to find a parametrized distribution without making arbitrary assumptions on the parameters. Intuitively we want to structure $q_\theta(\tau)$ such that it has the least commitment to any particular outcome in $\mathcal{D}$. In other words we want the *widest*

possible or *most uncertain* distribution. We use entropy as a measure of the uncertainty of a probability distribution and maximize that measure.

Definition 2. *The **Shannon's Information Entropy** [52] of a discrete random variable,* x*, which can take on the values* $\mathrm{x}_i, i = 1, \ldots, n$ *each with probability* $P(\mathrm{x}_i)$ *is defined as,*

$$\mathbb{H}_P(\mathrm{x}) = \mathrm{E}_{P(\mathrm{x})}[-\log P(\mathrm{x})] = -\sum_{i=1}^{n} P(\mathrm{x}_i) \log P(\mathrm{x}_i). \tag{3.9}$$

Information Entropy (Definition 2) is a unique measurement of the uncertainty in a probability distribution of a discrete random variable [52]. However, in many optimal control applications the state and action spaces are continuous, thus requiring trajectories to be modelled as a continuous random variables. Differential Entropy (Definition 3) was presented by Shannon [25] as the analogous measure of uncertainty in the distribution of a continuous random variable.

Definition 3. *The **Differential Entropy** [13] of a continuous random variable,* $\mathrm{x} \in \mathcal{X} \subseteq \mathbb{R}$ *with probability distribution* $p(\mathrm{x})$ *is defined as,*

$$\mathbb{H}_p^{diff}(\mathrm{x}) = \mathrm{E}_{p(\mathrm{x})}[-\log p(\mathrm{x})] = -\int_{\mathcal{X}} p(\mathrm{x}) \log p(\mathrm{x}) d\mathrm{x} \tag{3.10}$$

Jaynes [25] demonstrated that Differential Entropy lacks some of the properties of Shannon's Information Entropy, most notably invariance under change of parameters. As a result Differential Entropy is not a unique measure of uncertainty in the distribution of a continous random variable. Jaynes instead derives the continuous version of Shannon's Information Entropy by taking (3.9) in the limit of $n \to \infty$ to arrive at

$$\mathbb{H}_p^c(\mathrm{x}) = \mathrm{E}_{p(\mathrm{x})}\left[-\log \frac{p(\mathrm{x})}{m(\mathrm{x})}\right] = -\int_{\mathcal{X}} p(\mathrm{x}) \log \frac{p(\mathrm{x})}{m(\mathrm{x})} d\mathrm{x} \tag{3.11}$$

where $m(\mathrm{x})$ is defined as an invariant measure, which is known to be challenging to determine [25].

In order to find the widest possible distribution for a random variable, we need to maximize the appropriate entropy measure. *The Principle of Maximum Entropy* (Equation (3.12)) formalizes selecting the widest possible distribution that matches observed data. Jaynes [24] presents the solution to the parametrized probability distribution that

maximizes Information Entropy (Definition 2) for a discrete random variable. For continuous random variables, we need to maximize $\mathbb{H}_p^c(\mathrm{x})$. However, taking the maximum of the Differential Entropy (Definition 3), $\mathbb{H}_p^{diff}$, with respect to p is equivalent to taking the maximum of $\mathbb{H}_p^c(\mathrm{x})$ because $m(x)$ does not affect $p(x)$. We present the probability distribution that results from maximizing the Entropy in Theorem 3.

Theorem 3. *The* ***Maximum Entropy Probability Distribution*** *[24] is the distribution with the maximum information entropy subject to matching the expected characteristics, denoted by characteristic functions, $g_j(P(\mathrm{x}))$ with the empirical characteristic averages, $g_j(\tilde{P}(\mathrm{x}))$. It is obtained from the optimization,*

$$\begin{aligned} &\arg\max_P \mathbb{H}_P(\mathrm{x}) \text{ subject to} \\ &\forall j, g_j(P(\mathrm{x})) = g_j(\tilde{P}(\mathrm{x})) \\ &\sum_{i=1}^{n} P(\mathrm{x}_i) = 1 \\ &\forall i, P(\mathrm{x}_i) \geq 0. \end{aligned} \tag{3.12}$$

We focus on probabilistically weighted characteristic functions,

$$g_j(P(\mathrm{x})) = \mathbb{E}_{P(\mathrm{x})}[\phi_j(\mathrm{x})]$$

where, ϕ_j is the j^{th} of p features, i.e. functions of random variable x*. Then the distribution satisfying the maximization will have the following form,*

$$P(\mathrm{x}) = \frac{\exp\left(\sum_j \theta_j \phi_j(\mathrm{x})\right)}{Z(\theta_1, \ldots, \theta_p)} \tag{3.13}$$

The parameters, θ_j, where $j = 1, \ldots, p$ are obtained to satisfy the constraints in (3.12) *and the partition function, Z, is defined as,*

$$Z(\theta_1, \ldots, \theta_p) = \begin{cases} \sum_{\mathrm{x}} exp\left(\sum_j \theta_j \phi_j(\mathrm{x})\right) & \text{if x discrete} \\ \int_{\mathrm{x}} exp\left(\sum_j \theta_j \phi_j(\mathrm{x})\right) & \text{if x continuous} \end{cases}$$

The proof for this theorem can be found in Appendix A.

The Maximum Entropy Probability Distribution in Theorem 3 is immediately applicable to problems where we are constructing a probability distribution to model *a single* random variable. For example we have a dataset containing empirically measured values,

$\tilde{f}(\mathrm{x})$, of a random variable x. We use the Principle of Maximum Entropy to build a probability distribution of x, $p(\mathrm{x})$, and use the distribution to find the expectation of $f(\mathrm{x})$. Then we can check the probability of a statement such as "The expectation of the function, $f(\mathrm{x})$, of random variable, x, is 5.0." However in the most general sense of optimal control, i.e. System 3.1 with a stochastic policy and with stochastic dynamics, we are unable to model a trajectory as a single random variable. Instead we must model the trajectory as two sequences of random variables, a sequence of actions and a sequence of states.

Maximum Entropy for Sequences of Variables

As a robot proceeds through a trajectory, at each time step, an action is selected from a stochastic policy. Then by execution of the action, the robot arrives at a new state. The transition to the new state is governed by stochastic dynamics. Each new action relies on the values of the previous actions and the previous states. To represent a distribution for the sequence of actions over the entire trajectory, we define a causally conditioned probability distribution (Definition 4).

Definition 4. *The probability of a sequence of random variables* $\mathbf{u} = [\mathbf{u}_0^\top, \ldots, \mathbf{u}_{K-1}^\top]^\top$ ***causally conditioned*** *[31] on another sequence of random variables* $\mathbf{x} = [\mathbf{x}_1^\top, \ldots, \mathbf{x}_K^\top]^\top$ *is defined as*

$$P(\mathbf{u}||\mathbf{x}) := \prod_{k=0}^{K-1} P(\mathbf{u}_k|\mathbf{x}_{0:k}, \mathbf{u}_{0:k-1})$$

where $\mathbf{x}_{0:k} = (\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_k)$ *and* $\mathbf{u}_{0:k-1} = (\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_{k-1})$ *indicate concatenations of the sequences* $\mathbf{x}$ *and* $\mathbf{u}$ *for* $k = 0, \ldots, K-1$.

To apply causal conditioning to optimal control, we will let $\mathbf{u}_k$ represent an action at time step k and $\mathbf{x}_k$ represent a state. As the trajectory proceeds in time, we can view each executed action and each visited state as new information about future variables in the sequence. Formally, the new information is presented as a tuple of a state and an action value, $(\mathbf{x}_k, \mathbf{u}_k)$. Each subsequent action is produced by a stochastic policy, modelled by the distribution $P(\mathbf{u}_k|\mathbf{x}_{0:k}, \mathbf{u}_{0:k-1})$ for all time steps $k = 0, \ldots, K-1$. The states are produced by performing the actions and progressing forward in time under the stochastic dynamics, modelled by the distribution $P(\mathbf{x}_k|\mathbf{x}_{0:k-1}, \mathbf{u}_{0:k-1})$. The distribution that corresponds to the dynamics of the system is called the distribution of *side information*, $P(\mathbf{x}_{0:k}||\mathbf{u}_{0:k-1}) = \prod_{k=0}^{K-1} P(\mathbf{x}_k|\mathbf{x}_{0:k-1}, \mathbf{u}_{0:k-1})$. This kind of sequential revelation of

information is *causal*. Past variables reveal information about future variables but not vice versa. In other words, past variables *cause* future variables.

The generalization of Information Entropy to the causal setting is called Causal Entropy (Definition 6). To define Causal Entropy we need to first define Conditional Entropy (Definition 5).

Definition 5. *The* ***Conditional Entropy*** *of a random variable* u *conditioned on a random variable* x *is defined as,*

$$\mathbb{H}_{P(\mathrm{x},\mathrm{u})}(\mathrm{u}|\mathrm{x}) = \mathbb{E}_{P(\mathrm{x},\mathrm{u})}[-\log P(\mathrm{u}|\mathrm{x})]. \tag{3.14}$$

The causal entropy can be decomposed to a sum of the conditional entropies as shown in (3.15).

Definition 6. *Let* $\mathbf{u}$ *be causally conditioned on* $\mathbf{x}$ *as in Definition 4 with joint probability distribution* $P(\mathbf{x}, \mathbf{u})$*. The* ***Causal Entropy*** *[31] of* $\mathbf{u}$ *given* $\mathbf{x}$ *is defined as,*

$$\begin{aligned}
\mathbb{H}_P(\mathbf{u}||\mathbf{x}) &= \mathbb{E}_{P(\mathbf{x},\mathbf{u})}[-\log P(\mathbf{u}||\mathbf{x})] \\
&= -\sum_{\mathbf{x}\in\mathcal{X}}\sum_{\mathbf{u}\in\mathcal{U}} P(\mathbf{x},\mathbf{u}) \log\left(\prod_{k=1}^{K-1} P(\mathbf{u}_k|\mathbf{x}_{0:k}, \mathbf{u}_{0:k-1})\right) \\
&= -\sum_{k=1}^{K-1}\sum_{\mathbf{x}\in\mathcal{X}}\sum_{\mathbf{u}\in\mathcal{U}} P(\mathbf{x},\mathbf{u}) \log\left(P(\mathbf{u}_k|\mathbf{x}_{0:k}, \mathbf{u}_{0:k-1})\right) \\
&= \sum_{k=1}^{K-1} \mathbb{H}_{P(\mathbf{x},\mathbf{u})}(\mathbf{u}_k|\mathbf{x}_{0:k}, \mathbf{u}_{0:k-1})
\end{aligned} \tag{3.15}$$

Ziebart [60] presents the generalization of the *Principle of Maximum Entropy* to the causal setting, called *The Principle of Maximum Causal Entropy.* The Maximum Causal Entropy Probability Distribution (Theorem 4) formalizes selecting the widest possible distribution that matches observed sequentially revealed data.

Theorem 4. *The* ***Maximum Causal Entropy Probability Distribution*** *[60] is the distribution with the maximum causal entropy subject to matching the expected characteristics, denoted by characteristic functions,* $g_j(P(\mathbf{u}||\mathbf{x}))$ *with the empirical characteristic averages,* $g_j(\tilde{P}(\mathbf{u}||\mathbf{x}))$ *and given the distribution of side information*

$\forall k, P(\mathbf{x}_k|\mathbf{x}_{0:k-1}, \mathbf{u}_{0:k-1})$. *It is obtained from the optimization,*

$$\begin{aligned} &\arg\max_P \mathbb{H}_P(\mathbf{u}||\mathbf{x}) \text{ subject to} \\ &\forall j, g_j(P(\mathbf{u}||\mathbf{x})) = g_j(\tilde{P}(\mathbf{u}||\mathbf{x})) \\ &\forall \mathbf{x} \sum_{\mathbf{u}\in\mathcal{U}} P(\mathbf{u}||\mathbf{x}) = 1 \\ &\forall \mathbf{x}, \mathbf{u}, P(\mathbf{u}||\mathbf{x}) \geq 0 \\ &\forall \kappa, \mathbf{u}_{1:\kappa}, \mathbf{x}, \textit{and } \mathbf{x}' \textit{ s.t. } \mathbf{x}_{1:\kappa} = \mathbf{x}'_{1:\kappa} \sum_{\mathbf{u}_{\kappa+1:K}} (P(\mathbf{u}||\mathbf{x}) - P(\mathbf{u}||\mathbf{x}')) = 0 \end{aligned} \tag{3.16}$$

We focus on probabilistically weighted characteristic functions,

$$g_j(P(\mathbf{u}||\mathbf{x})) = \mathbb{E}_{P(\mathrm{x},\mathrm{u})}[\phi_i(\mathbf{x}, \mathbf{u})]$$

where, ϕ_i is the i^{th} of p features, i.e. functions of random variable sequences $\mathbf{x}$ and $\mathbf{u}$; let $\boldsymbol{\phi} := [\phi_1, \ldots, \phi_p]^\top$. Then the distribution satisfying the maximization will have the following recursive form,

$$P_{\boldsymbol{\theta}}(\mathbf{u}_k|\mathbf{x}_{0:k}, \mathbf{u}_{0:k-1}) = \frac{\mathrm{Z}_{\mathbf{u}_k|\mathbf{x}_{0:k},\mathbf{u}_{0:k-1};\boldsymbol{\theta}}}{\mathrm{Z}_{\mathbf{x}_{0:k},\mathbf{u}_{0:k-1};\boldsymbol{\theta}}} \tag{3.17}$$

where,

$$\begin{aligned} \log(\mathrm{Z}_{\mathbf{x}_{0:k},\mathbf{u}_{0:k-1};\boldsymbol{\theta}}) &= \log\left(\sum_{\mathbf{u}_k} \mathrm{Z}_{\mathbf{u}_k|\mathbf{x}_{0:k},\mathbf{u}_{0:k-1};\boldsymbol{\theta}}\right) \\ &= \operatorname*{softmax}_{\mathrm{u}_k}\left(\sum_{\mathbf{x}_{k+1}} P(\mathbf{x}_{k+1}|\mathbf{x}_{0:k}, \mathbf{u}_{0:k-1}; \boldsymbol{\theta}) \log(\mathrm{Z}_{\mathbf{x}_{0:k+1},\mathbf{u}_{0:k};\boldsymbol{\theta}})\right) \end{aligned} \tag{3.18}$$

and,

$$\begin{aligned} \mathrm{Z}_{\mathbf{u}_k|\mathbf{x}_{0:k},\mathbf{u}_{0:k-1};\boldsymbol{\theta}} &= \exp\left(\sum_{\mathbf{x}_{k+1}} P(\mathrm{x}_{k+1}|\mathbf{x}_{0:k}, \mathbf{u}_{0:k-1}) \log(\mathrm{Z}_{\mathbf{x}_{0:k+1},\mathbf{u}_{0:k};\boldsymbol{\theta}})\right) \\ \mathrm{Z}_{\mathbf{x}_{0:K},\mathbf{u}_{0:K-1};\boldsymbol{\theta}} &= \exp(\boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{x}, \mathbf{u})) \end{aligned} \tag{3.19}$$

where $\operatorname{softmax}_x f(x) := \log(\sum_x \exp(f(x)))$ *and* $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_p]^\top$ *are the Lagrange multipliers for the characteristic matching constraint. If we assume that the features decompose into a sum over individual random variables, $\mathbf{x}_k$ and $\mathbf{u}_k$, i.e. $\boldsymbol{\phi}(\mathbf{x}, \mathbf{u}) =$*

$\sum_k \boldsymbol{\phi}_{\mathbf{x}_k,\mathbf{u}_k}$ and the distribution of side information is Markovian, i.e. $P(\mathbf{x}_k|\mathbf{x}_{0:k-1},\mathbf{u}_{0:k-1}) = P(\mathbf{x}_k|\mathbf{x}_{k-1},\mathbf{u}_{k-1})$*, then the recursive form simplifies to,*

$$P_{\boldsymbol{\theta}}(\mathbf{u}_k|\mathbf{x}_k) = \frac{\mathrm{Z}_{\mathbf{u}_k|\mathbf{x}_k,\boldsymbol{\theta}}}{\mathrm{Z}_{\mathbf{x}_k,\boldsymbol{\theta}}} \tag{3.20}$$

where,

$$\begin{aligned}
\log(\mathrm{Z}_{\mathbf{x}_k,\boldsymbol{\theta}}) &= \log\left(\sum_{\mathbf{u}_k} \mathrm{Z}_{\mathbf{u}_k|\mathbf{x}_k,\boldsymbol{\theta}}\right) \\
&= \underset{\mathbf{u}_k}{\mathrm{softmax}}\left(\sum_{\mathbf{x}_{k+1}} P(\mathbf{x}_{k+1}|\mathbf{x}_{0:k},\mathbf{u}_{0:k-1})\log(\mathrm{Z}_{\mathbf{x}_{0:k+1},\mathbf{u}_{0:k}}) + \boldsymbol{\theta}^\top \boldsymbol{\phi}_{\mathbf{x}_k,\mathbf{u}_k}\right) \\
&= \underset{\mathbf{u}_k}{\mathrm{softmax}}\left(\mathbb{E}_{P(\mathbf{x}_k|\mathbf{x}_k,\mathbf{u}_k)}[\log(\mathrm{Z}_{\mathbf{x}_{k+1},\boldsymbol{\theta}})] + \theta^\top \boldsymbol{\phi}_{\mathbf{x}_k,\mathbf{u}_k}\right)
\end{aligned} \tag{3.21}$$

and,

$$\begin{aligned}
\mathrm{Z}_{\mathbf{u}_k|\mathbf{x}_k,\theta} &= \exp\left(\sum_{\mathbf{x}_{k+1}} P(\mathbf{x}_k|\mathbf{x}_k,\mathbf{u}_k)\log(\mathrm{Z}_{\mathbf{x}_{k+1},\theta}) + \theta^\top \boldsymbol{\phi}_{\mathbf{x}_k,\mathbf{u}_k}\right) \\
&= \exp\left(\mathbb{E}_{P(\mathbf{x}_{k+1}|\mathbf{x}_k,\mathbf{u}_k)}[\log(\mathrm{Z}_{\mathbf{x}_{0:k+1},\mathbf{u}_{0:k}})] + \theta^\top \boldsymbol{\phi}_{\mathbf{x}_k,\mathbf{u}_k}\right)
\end{aligned} \tag{3.22}$$

The final constraint in optimization (3.16) ensures the causality of the optimal probability distribution. This constraint forces the probability up to time step κ to be equal for all possible side information values, $\mathbf{x}_{\kappa+1:K}$, that may be revealed after time step κ. The proof for Theorem 4 is presented in Appendix A.

The Maximum Causal Entropy probability distribution provides an information theoretic framework to approximate the probability distribution of sequentially revealed information. In the rest of this section, we will show how this distribution is directly applicable to optimal control.

3.3.2 Maximum Causal Entropy as Softened Bellman Equations

The Maximum Causal Entropy probability distribution in Theorem 4 can be interpreted as a softened version of the Bellman Equations (3.4, 3.5) [60]. First, we re-express the Maximum Causal Entropy distribution as,

$$\begin{aligned}
Q^{soft}_{\boldsymbol{\theta}}(\mathbf{x}_k,\mathbf{u}_k) &:= \log \mathrm{Z}_{\mathbf{u}_k|\mathbf{x}_k,\boldsymbol{\theta}} \\
&= \mathbb{E}_{P(\mathbf{x}_{k+1}|\mathbf{x}_k,\mathbf{u}_k)}[V^{soft}_{\boldsymbol{\theta}}(\mathbf{x}_{k+1})] + \boldsymbol{\theta}^\top \boldsymbol{\phi}_{\mathbf{x}_k,\mathbf{u}_k}
\end{aligned} \tag{3.23}$$

$$\begin{aligned} V^{soft}_{\boldsymbol{\theta}}(\mathbf{x}_k) &:= \log \mathrm{Z}_{\mathbf{x}_k,\boldsymbol{\theta}} \\ &= \underset{\mathbf{u}_k}{\mathrm{softmax}}\, Q^{soft}_{\boldsymbol{\theta}}(\mathbf{x}_k, \mathbf{u}_k). \end{aligned} \tag{3.24}$$

If we take the stage reward function in Equation (3.4) to be $\mathcal{R}_k(\mathbf{x}_k, \mathbf{u}_k) = \boldsymbol{\theta}^\top \boldsymbol{\phi}_{\mathbf{x}_k,\mathbf{u}_k}$ and replace the $\max_{\mathbf{u}_k}$ in Equation (3.5) with a $\mathrm{softmax}_{\mathbf{u}_k}$, we arrive at (3.23, 3.24). Intuitively, **the Maximum Causal Entropy (MCE) probability distribution is a generalization of the Bellman Equations to settings where we do not view the optimality of a trajectory as binary variable, but rather a probabilistic spectrum.** In his spectrum, policies closer to the optimal policy are exponentially more likely. More concretely, as a result of the $\mathrm{softmax}_{\mathbf{u}_k}$ function in the soft value function (3.24), the probability of an action at a particular time step $\mathbf{u}_k$ scales exponentially with,

$$P(\mathbf{u}_k|\mathbf{x}_k) = \exp(Q^{soft}_{\boldsymbol{\theta}}(\mathbf{u}_k, \mathbf{x}_k) - V^{soft}_{\boldsymbol{\theta}}(\mathbf{x}_k)) \tag{3.25}$$

Ziebart [60] illustrates that in the case that $\boldsymbol{\theta}$ is scaled by a constant α, as $\alpha \to \infty$, then the softmax function behaves like the max function. In this limit, the Equations (3.23, 3.24) become equivalent to the Bellman Equations (3.4, 3.5).

Deterministic Dynamics

In many optimal control systems, the dynamics are deterministic, i.e. a simplification of System (3.1) where $\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k)$. In these settings, given an initial condition, $\mathbf{x}_0$, the sequence of states can be deterministically recovered from the sequence of actions. Thus, given the dynamics function, we can describe a trajectory using only the sequence of actions. Then the Maximum Causal Entropy optimization in Equation (3.16) reduces to,

$$\begin{aligned} &\arg\max_P \mathbb{H}_P(\mathbf{u}) \text{ subject to} \\ \forall i, \mathbb{E}_{P(\mathrm{x},\mathrm{u})}[\phi_i(\mathbf{x}, \mathbf{u})] &= \mathbb{E}_{\tilde{P}(\mathrm{x},\mathrm{u})}[\phi_i(\mathbf{x}, \mathbf{u})] \\ \sum_{\mathbf{u}\in\mathcal{U}} P(\mathbf{u}) &= 1 \\ \forall \mathbf{u}, P(\mathbf{u}) &\geq 0 \end{aligned} \tag{3.26}$$

In order to find the optimal probability distribution we will follow the same approach as the proof of Theorem 3. However we replace the expectation matching criteria on a single variable, $\mathbb{E}_{P(\mathrm{x}_i)}[\mathrm{x}_i]$ with $\mathbb{E}_{P(\mathbf{u})}[\sum_k \boldsymbol{\phi}(\mathbf{x}_k, \mathbf{u}_k)]$. We will introduce a vector of Lagrange multipliers $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_p]^\top$ instead of the multiplier μ. The optimal probablity

distribution that we get is,

$$P(\mathbf{u}) = \frac{\exp\left(\boldsymbol{\theta}^\top \sum_{k=1}^{K-1} \boldsymbol{\phi}(\mathbf{x}_k, \mathbf{u}_k)\right)}{Z(\boldsymbol{\theta})} \tag{3.27}$$

where the partition function is the sum over all dynamically possible actions,

$$Z(\boldsymbol{\theta}) = \sum_{\mathbf{u}} \exp\left(\boldsymbol{\theta}^\top \sum_{k=1}^{K-1} \boldsymbol{\phi}(\mathbf{x}_k, \mathbf{u}_k)\right).$$

3.3.3 Maximum Causal Entropy for IRL

With the MCE distribution we can characterize soft-optimal policies probabilistically. We need to apply the soft-optimal policies to IRL. In Theorem 2 we used the Bellman equations (3.4, 3.5) to demonstrate that matching expected feature counts results in matching total expected value and because of the Bellman equations would also result in matching expected behaviour at every time step. We need to show that using the MCE distribution to approximate a dataset of trajectories would also result in matching expected behaviour at every time step.

To this end, we analyze the proof of Theorem 2. We can see that using the softened Bellman equations (3.23, 3.24) in that proof, would result in the same outcome. Therefore, trajectories derived from the MCE distribution that are able to match feature counts with trajectories in a dataset provide the same behavioural guarantees as trajectories derived from solving the Bellman equations. We can summarize this finding in Corollary 4.1.

Corollary 4.1 (Ziebart [60])**.** *The Maximum Entropy Probability Distribution (Theorem 4) provides the same total expected value over the entire horizon as demonstrated variable sequences on any unknown parameters of a reward function linear in* $\boldsymbol{\phi}(\mathbf{x}, \mathbf{u})$.

sketch of proof. We start with the characteristic matching requirement from Theorem 4, $\mathbb{E}_{P(\mathrm{x},\mathrm{u})}[\phi_i(\mathbf{x}, \mathbf{u})] = \mathbb{E}_{\tilde{P}(\mathrm{x},\mathrm{u})}[\phi_i(\mathbf{x}, \mathbf{u})]$, and follow the same argument as the proof of Theorem 2. □

Learning Parameters from a Dataset

In the context of learning from data, the Maximum Causal Entropy distribution is parametrized by the Lagrange multipliers that correspond to the feature matching constraint, $\boldsymbol{\theta}$. Given a dataset, $\mathcal{D} = \{(\mathbf{x}^{(1)}, \mathbf{u}^{(\mathbf{1})}), \ldots, (\mathbf{x}^{(N)}, \mathbf{u}^{(N)})\}$, where $\mathbf{x}^{(i)} = [\mathbf{x}_1^\top, \ldots, \mathbf{x}_K^\top]^\top$

and $\mathbf{u}^{(i)} = [\mathbf{u}_0^\top, \ldots, \mathbf{u}_{K-1}^\top]^\top$ for $i = 1, \ldots, N$ are a set of observed trajectories, we need to solve the following maximization problem.

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} P_{\boldsymbol{\theta}}(\mathbf{u}||\mathbf{x}) \tag{3.28}$$

where $P(\mathbf{u}||\mathbf{x}) = \prod_{k=1}^{K-1} P(\mathbf{u}_k|\mathbf{x}_{0:k}, \mathbf{u}_{0:k-1})$ and we are given the dynamics. This maximization problem is a maximum likelihood estimate of the parameters. In other words, the optimal parameters $\boldsymbol{\theta}^*$ maximize the likelihood of observing the samples in the dataset under the probability distribution parametrized by $\boldsymbol{\theta}^*$. Ziebart [60] demonstrates that maximizing the causal entropy with feature matching constraints as in Equation (3.16) is equivalent to a maximum likelihood estimate (Equation (3.28)).

Theorem 5 ([60])**.** *Given a dataset* $\mathcal{D} = \{(\mathbf{x}^{(1)}, \mathbf{u}^{(1)}), \ldots, (\mathbf{x}^{(N)}, \mathbf{u}^{(N)})\}$ *with empirical distribution* $\tilde{P}(\mathbf{x}, \mathbf{u})$*, maximizing the causal entropy subject to the characteristic matching constraint,* $\mathbb{E}_{P(\mathbf{x},\mathbf{u})}[\boldsymbol{\phi}] = \mathbb{E}_{\tilde{P}(\mathbf{x},\mathbf{u})}[\boldsymbol{\phi}]$*, as in Equation* (3.16)*, is equivalent to maximum likelihood estimation of* $\boldsymbol{\theta}$ *given* $\mathcal{D}$ *(Equation* (3.28)*).*

The proof for Theorem 5 can be found in [60]. In maximum likelihood estimation, we typically use gradient-based optimization techniques to find the optimal value for model parameters, $\boldsymbol{\theta}^*$ [38]. Theorem 5 demonstrates that in the Maximum Causal Entropy distribution, maximum likelihood estimation of the parameters is equivalent to solving the optimization problem in Equation (3.16). In the proof of Theorem 4, we introduced the Lagrangian of the optimization problem, $\Lambda(P, \boldsymbol{\theta})$, in Equation (A.4). In order to find the optimal parameters we need to apply gradient-based optimization to $\Lambda(P, \boldsymbol{\theta})$. To this end, we find the gradient with respect to each parameter, $\theta_j, j = 1, \ldots, p$,

$$\nabla_{\theta_j} \Lambda(P, \boldsymbol{\theta}) = \mathbb{E}_{P(\mathbf{x},\mathbf{u})}[\boldsymbol{\phi}_j(\mathbf{x}, \mathbf{u})] - \mathbb{E}_{\tilde{P}(\mathbf{x},\mathbf{u})}[\boldsymbol{\phi}_j(\mathbf{x}, \mathbf{u})] \tag{3.29}$$

The second term in in Equation (3.29) is given by the dataset $\mathcal{D}$. Thus, the main algorithmic challenge of IRL is calculating the first term, $\mathbb{E}_{P(\mathbf{x},\mathbf{u})}[\boldsymbol{\phi}_j(\mathbf{x}, \mathbf{u})]$. The algorithms proposed by Ziebart [60] require performing inference in the MCE distribution, or equivalently solving the optimal control problem every time the gradient is required. Although this method is tractable for systems with small finite state and action spaces, it is intractable in continuous settings. Therefore, next we present a method that uses an approximation of $P(\mathbf{x}||\mathbf{u})$ to directly solve Equation (3.28) in the case of deterministic dynamics and infinite state and action spaces.

Continuous Control and Action Spaces and Locally Optimal Examples

Levine and Koltun [33] present the simplification of the MCE distribution for this system,

$$\begin{aligned} \mathrm{P}(\mathbf{u}|\mathrm{x}_0) &= \frac{1}{\mathrm{Z}(\boldsymbol{\theta})} \exp\left(\sum_{k=0}^{K-1} \mathcal{R}_k(\mathrm{x}_k, \mathrm{u}_k)\right) \\ &= \frac{\exp\left(\mathcal{R}(\mathbf{u})\right)}{\int exp\left(\mathcal{R}(\tilde{\mathbf{u}})\right) d\tilde{\mathbf{u}}} \end{aligned} \tag{3.30}$$

where $\mathcal{R}(\mathbf{u})$ denotes the sum of rewards over a trajectory $(\mathbf{x}_0, \mathbf{u})$ and $\mathrm{Z} = \int exp\left(\mathcal{R}(\tilde{\mathbf{u}})\right) d\tilde{\mathbf{u}}$ is the partition function. This is the continuous version of Equation (3.26). Assuming each expert trajectory is independently and identically distributed (i.i.d), we can obtain the probability of all trajectories in dataset, $\mathcal{D}$, by multiplying the probability of each expert trajectory. To find the optimal parameters, $\boldsymbol{\theta}$, we will maximize this probability, which is equivalent to a maximum likelihood estimate (Equation (3.28)).

An analytical solution for Z does not exit. For a numerical approximate solution, we use the *Laplace approximation* [33]. The Laplace approximation for a d-dimensional vector $\mathbf{x}$, constant M, and scalar function $f(\mathbf{x})$ with unique global maximum at $\mathbf{x}_m$ is,

$$\int e^{Mf(\mathbf{x})} d\mathbf{x} \approx \left(\frac{2\pi}{M}\right)^{\frac{d}{2}} \frac{e^{Mf(\mathbf{x}_m)}}{|\mathbf{H}_{\mathbf{x}_m}|^{\frac{1}{2}}} \tag{3.31}$$

where $\mathbf{H}_{\mathbf{x}_m}$ is the Hessian of f at $\mathbf{x}_m$ and $|\cdot|$ denotes the determinant.

We first find the second order Taylor expansion of $\mathcal{R}$ around $\mathbf{u}$.

$$\mathcal{R}(\tilde{\mathbf{u}}) \approx \mathcal{R}(\mathbf{u}) + (\tilde{\mathbf{u}} - \mathbf{u})^\top \frac{\partial \mathcal{R}}{\partial \mathbf{u}} + \frac{1}{2}(\tilde{\mathbf{u}} - \mathbf{u})^\top \frac{\partial^2 \mathcal{R}}{\partial \mathbf{u}^2}(\tilde{\mathbf{u}} - \mathbf{u}) \tag{3.32}$$

where $\frac{\partial \mathcal{R}}{\partial \mathbf{u}}$ denotes the gradient of $\mathcal{R}$ with respect to $\mathbf{u}$ and $\frac{\partial^2 \mathcal{R}}{\partial \mathbf{u}^2}$ denotes the Hessian. For simpler notation, let $\mathbf{g} := \frac{\partial \mathcal{R}}{\partial \mathbf{u}}$ and $\mathbf{H} := \frac{\partial^2 \mathcal{R}}{\partial \mathbf{u}^2}$. We substitute the Taylor expansion into

the partition function

$$\begin{aligned}
\mathrm{P}(\mathbf{u}|\mathrm{x}_0) &\approx \frac{e^{\mathcal{R}(\mathbf{u})}}{\int e^{\mathcal{R}(\mathbf{u})+(\tilde{\mathbf{u}}-\mathbf{u})^\top \mathbf{g}+\frac{1}{2}(\tilde{\mathbf{u}}-\mathbf{u})^\top \mathbf{H}(\tilde{\mathbf{u}}-\mathbf{u})} d\tilde{\mathbf{u}}} \\
&= \frac{1}{\int e^{(\tilde{\mathbf{u}}-\mathbf{u})^\top \mathbf{g}+\frac{1}{2}(\tilde{\mathbf{u}}-\mathbf{u})^\top \mathbf{H}(\tilde{\mathbf{u}}-\mathbf{u})} d\tilde{\mathbf{u}}} \\
&= \frac{1}{\int e^{\frac{1}{2}(\tilde{\mathbf{u}}-\mathbf{u})^\top \mathbf{g}+\frac{1}{2}(\mathbf{H}(\tilde{\mathbf{u}}-\mathbf{u})+\mathbf{g})^\top (\tilde{\mathbf{u}}-\mathbf{u})} d\tilde{\mathbf{u}}} \\
&= \frac{1}{\int e^{\frac{1}{2}(\tilde{\mathbf{u}}-\mathbf{u})^\top \mathbf{g}+\frac{1}{2}(\mathbf{H}(\tilde{\mathbf{u}}-\mathbf{u})+\mathbf{g})^\top \mathbf{H}^{-1}\mathbf{H}(\tilde{\mathbf{u}}-\mathbf{u})} d\tilde{\mathbf{u}}} \\
&= \frac{1}{\int e^{\frac{1}{2}(\tilde{\mathbf{u}}-\mathbf{u})^\top \mathbf{g}+\frac{1}{2}(\mathbf{H}(\tilde{\mathbf{u}}-\mathbf{u})+\mathbf{g})^\top \mathbf{H}^{-1}(\mathbf{H}(\tilde{\mathbf{u}}-\mathbf{u})+\mathbf{g})-\frac{1}{2}(\mathbf{H}(\tilde{\mathbf{u}}-\mathbf{u})+\mathbf{g})^\top \mathbf{H}^{-1}\mathbf{g}} d\tilde{\mathbf{u}}} \\
&= \frac{1}{\int e^{\frac{1}{2}(\tilde{\mathbf{u}}-\mathbf{u})^\top \mathbf{g}-\frac{1}{2}(\tilde{\mathbf{u}}-\mathbf{u})^\top \mathbf{H}^\top \mathbf{H}^{-1}\mathbf{g}+\frac{1}{2}(\mathbf{H}(\tilde{\mathbf{u}}-\mathbf{u})+\mathbf{g})^\top \mathbf{H}^{-1}(\mathbf{H}(\tilde{\mathbf{u}}-\mathbf{u})+\mathbf{g})-\frac{1}{2}\mathbf{g}^\top \mathbf{H}^{-1}\mathbf{g}} d\tilde{\mathbf{u}}} \\
&= \frac{1}{\int e^{\frac{1}{2}(\tilde{\mathbf{u}}-\mathbf{u})^\top \mathbf{g}-\frac{1}{2}(\tilde{\mathbf{u}}-\mathbf{u})^\top \mathbf{H}\mathbf{H}^{-1}\mathbf{g}+\frac{1}{2}(\mathbf{H}(\tilde{\mathbf{u}}-\mathbf{u})+\mathbf{g})^\top \mathbf{H}^{-1}(\mathbf{H}(\tilde{\mathbf{u}}-\mathbf{u})+\mathbf{g})-\frac{1}{2}\mathbf{g}^\top \mathbf{H}^{-1}\mathbf{g}} d\tilde{\mathbf{u}}} \\
&= \frac{e^{\frac{1}{2}\mathbf{g}^\top \mathbf{H}^{-1}\mathbf{g}}}{\int e^{\frac{1}{2}(\mathbf{H}(\tilde{\mathbf{u}}-\mathbf{u})+\mathbf{g})^\top \mathbf{H}^{-1}(\mathbf{H}(\tilde{\mathbf{u}}-\mathbf{u})+\mathbf{g})} d\tilde{\mathbf{u}}}.
\end{aligned} \tag{3.33}$$

In Equation (3.33) we assume that $\mathcal{R}(\mathbf{u})$ has continuous second partial derivatives with respect to the elements in $\mathbf{u}$. Then by Schwartz's Theorem $\mathbf{H}$ is symmetric, i.e, $\mathbf{H}^\top = \mathbf{H}$.

By using the Laplace approximation (Equation (3.31)) to find an approximate solution to the integral in Equation 3.33, we arrive at a closed-form estimate of the probability,

$$\mathrm{P}(\mathbf{u}|\mathrm{x}_0) \approx e^{\frac{1}{2}\mathbf{g}^\top \mathbf{H}^{-1}\mathbf{g}} |-\mathbf{H}|^{\frac{1}{2}} (2\pi)^{\frac{d_\mathbf{u}}{2}} \tag{3.34}$$

where $d_\mathbf{u}$ denotes the dimension of $\mathbf{u}$. We finally obtain the approximate log likelihood,

$$\mathcal{L} = \frac{1}{2}\mathbf{g}^\top \mathbf{H}^{-1}\mathbf{g} + \frac{1}{2} log|-\mathbf{H}| - \frac{d_\mathbf{u}}{2}\log(2\pi) \tag{3.35}$$

In order to maximize the log likelihood, we can use any gradient-based optimization technique. We need the gradient with respect to each parameter $\theta_i, i = 1, \ldots, p$. The gradient is given by,

$$\nabla_{\theta_i}\mathcal{L} = \mathbf{g}^\top \mathbf{H}^{-\top}\frac{\partial \mathbf{g}}{\partial \theta_i} - \frac{1}{2}\mathbf{g}^\top \mathbf{H}^{-\top}\frac{\partial \mathbf{H}}{\partial \theta_i}\mathbf{H}^{-1}\mathbf{g} + \frac{1}{2}\mathrm{tr}\left(\mathbf{H}^{-1}\frac{\partial \mathbf{H}}{\partial \theta_i}\right) \tag{3.36}$$

where $\mathrm{tr}(\cdot)$ indicates the trace of a matrix.

3.4 Conclusion

In this chapter we presented the theory behind Probabilistic IRL. We began by reviewing the optimal control problem and presented a solution, Determinisic Dynamic Programming, to the forward optimal control problem for settings with deterministic dynamics and infinite state and action spaces. Then, we presented a general formulation of the inverse problem, IRL, and explained that this problem is ill-posed. Starting from first principles in Information Theory, we derived the Maximum Causal Entropy probability distribution, in which inference is analogous to solving the optimal control problem. We demonstrated how finding parameters for this distribution is equivalent to solving the IRL problem. Finally, we presented a method for performing IRL for settings with deterministic dynamics and infinite state and action spaces. In the rest of this thesis, we will apply the method to an autonomous driving setting.

Chapter 4

IRL for LQR

The Linear Quadratic Regulator (LQR) is one of the most studied optimal control algorithms. To verify the Inverse Reinforcement Learning (IRL) theory described in Chapter 3 we conduct a set of experiments that applies IRL to recovering LQR parameters given varying amounts of noisy example trajectories data.

4.1 Background

The inverse LQR problem has been used in the literature to study IRL algorithms. Ziebart uses an inverse LQR problem as a practical demonstration of the IRL with the Maximum Causal Entropy (MCE) framework [60]. First, the author shows that the MCE model can be solved analytically in the setting of an LQR problem with linear dynamics that are perturbed by Gaussian noise. The inverse LQR algorithm reported in their thesis relies on the fact that the policy is modelled as a Gaussian distribution. Second, Ziebart experimentally compares the success of the MCE inverse LQR approach with another IRL algorithm on a helicopter hovering problem in an off-the-shelf simulator. Demonstrations are generated by noisily sampling the policy obtained from a forward algorithm. Ziebart's experiment demonstrates that the Maximum Causal Entropy method is applicable to inverse LQR tasks, however the author only presents one experiment. As a result a conclusion cannot be made on how much data is required for successfully recovering LQR parameters under variable amounts of noise.

In our inverse LQR experiments, we use a simpler dynamical system and present more in depth analysis of the effect of the number of example trajectories on the accuracy of inverse solutions under variable amounts of noise. We use the IRL algorithm proposed by Levine et al. [33] (summarized in Section 3.3), which is an extension of the MCE IRL framework to account for continuous state and action spaces and locally optimal

demonstration trajectories. The algorithm performs a linear-quadratic expansion of the dynamics and reward function of a system with non-linear dynamics and/or reward around the example trajectories.

Other works have also evaluated inverse LQR algorithms, albeit using different formulations of the problem. The authors in [6] and [44] use a Linear Matrix Inequality-based approach to inverse LQR. In [44] the inverse method is applied to recovering LQR parameters for a controller that imitates a human's balancing movements when their seat is perturbed externally. The authors present data collected from a participant and demonstrate that their learned controller would respond in a similar way as the participant.

Another formulation of the inverse LQR problem uses Pontryagin's Maximum Principle (PMP) for discrete-time control systems [59]. Their experiments evaluate the effect of the number of example trajectories on the quality of their solution. In contrast to the PMP formulation of IRL used in this proposal, we use the MCE IRL formulation.

In this section we report the results of two groups of experiments. In the first group we demonstrate that the IRL method can be used to exactly recover LQR parameters from demonstration trajectories in the absence of noise and when our problem has linear dynamics and a quadratic reward. Then in the second set of experiments we investigate the effect of varying amounts of noise and number of demonstration trajectories on the accuracy of the parameters recovered by the IRL algorithm.

4.1.1 Review of the forward LQR solution

We focus on a simplification of System (3.1) with deterministic dynamics, and quadratic reward function. We define the linear fixed-horizon control task of length K,

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k) = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k \quad \text{dynamics} \tag{4.1a}$$

$$\mathbf{x}_k \in \mathcal{X} = \mathbb{R}^n \quad \text{state} \tag{4.1b}$$

$$\mathbf{u}_k \in \mathcal{U} = \mathbb{R}^m \quad \text{action} \tag{4.1c}$$

$$\mathcal{R}_k(\mathbf{x}_k, \mathbf{u}_k) = -\mathbf{x}_k^\top \mathbf{Q}\mathbf{x}_k - \mathbf{u}_k^\top \mathbf{R}\mathbf{u}_k \quad \text{stage reward} \tag{4.1d}$$

$$\mathcal{R}_k(\mathbf{x}_K) = -\mathbf{x}_K^\top \mathbf{Q}\mathbf{x}_K \quad \text{terminal reward} \tag{4.1e}$$

$$\pi = \{\mu_0, \ldots, \mu_{K-1}\} \quad \text{policy} \tag{4.1f}$$

where $\mathbf{A} \in \mathbb{R}^{n \times n}$ and $\mathbf{B} \in \mathbb{R}^{n \times m}$ are the system matrix and input matrix, respectively. The matrices $\mathbf{Q} \in \mathbb{R}^{n \times n}$ and $\mathbf{R} \in \mathbb{R}^{m \times m}$ are positive semi-definite matrices that parametrize the reward function.

The goal of the LQR problem is to obtain a policy that, when executed, results in

a trajectory with maximal reward. We can define the Bellman Equations (3.4, 3.5) for the LQR reward (4.1d) and use the Dynamic Programming Algorithm (Definition 1) to obtain an optimal solution for the policy.

$$V_K^{\pi^*}(\mathbf{x}_K) = \mathcal{R}_K(\mathbf{x}_K) = \mathbf{x}_K^\top \mathbf{Q} \mathbf{x}_K \tag{4.2}$$

$$\begin{aligned} V_k^{\pi^*}(\mathbf{x}_k) &= \max_{\mathbf{u}_k} \mathcal{R}_k(\mathbf{x}_k, \mathbf{u}_k) + \mathbb{E}_{e_k}[V_{k+1}^{\pi^*}(\mathbf{x}_{k+1})] \\ &= \max_{\mathbf{u}_k} \{-\mathbf{x}_k^\top \mathbf{Q} \mathbf{x}_k - \mathbf{u}_k^\top \mathbf{R} \mathbf{u}_k\} + V_{k+1}^{\pi^*}(\mathbf{x}_{k+1}) \end{aligned} \tag{4.3}$$

Details for the derivation of the solution are beyond the scope of this thesis and can be found in [5]. The optimal control law for every time step, k, has the following form,

$$\mu_k^*(\mathbf{x}_k) = \mathbf{L}_k \mathbf{x}_k \tag{4.4}$$

where the feedback matrix is given by,

$$\mathbf{L}_k = -(\mathbf{B}^\top \mathbf{K}_{k+1} \mathbf{B} + \mathbf{R})^{-1} \mathbf{B}^\top \mathbf{K}_{k+1} \mathbf{A}. \tag{4.5}$$

The value $\mathbf{K}_k$ for each time step is found by recursive iteration from the final time step such that it is a solution to the Discrete-time Algebraic Ricatti Equation (DARE) [5],

$$\begin{aligned} \mathbf{K}_K &= \mathbf{Q} \\ \mathbf{K}_k &= \mathbf{A}^\top (\mathbf{K}_{k+1} - \mathbf{K}_{k+1} \mathbf{B} (\mathbf{B}^\top \mathbf{K}_{k+1} \mathbf{B} + \mathbf{R})^{-1} \mathbf{B}^\top \mathbf{K}_{k+1}) \mathbf{A} + \mathbf{Q}. \end{aligned} \tag{4.6}$$

4.2 Experimental Setup

We formulate a generic LQR problem to be used in both sets of experiments. We select values for the state matrix, $\mathbf{A}$, input matrix $\mathbf{B}$, and the reward matrices, $\mathbf{Q}$ and $\mathbf{R}$. We use the DARE (Equation (4.6)) to generate a synthetic trajectory for a stabilization task, i.e. driving the state to zero, $\mathbf{x}_K = 0$. We use a dataset of synthetic trajectories as the dataset and use theory presented in Section 3.3.3 to recover the reward parameters.

4.2.1 System and Reward Structure

We set up system 4.1 to be unstable and controllable. For a given state dimension, n, our system matrix takes the following form,

$$\mathbf{A} = \begin{bmatrix} 1.1 & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & 1.1 \end{bmatrix} \in \mathbb{R}^{n \times n} \tag{4.7}$$

where the empty matrix elements are zeros. For a given action dimension, m, the input matrix, $\mathbf{B} \in \mathbb{R}^{n \times m}$ takes the following sparse block-diagonal form,

$$\mathbf{B} = \begin{cases} \mathbf{I}_{n \times n} & \text{if } n = m \\ \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} & \text{if } n > m = 1 \\ \begin{bmatrix} \mathbf{I}_{m \times m} \\ 0 \ldots 1 \\ \vdots \\ 0 \ldots 1 \end{bmatrix} & \text{if } n > m > 1 \\ \begin{bmatrix} & 0 & \ldots & 0 \\ \mathbf{I}_{n \times n} & \vdots & & \vdots \\ & 0 & \ldots & 0 \\ & 1 & \ldots & 1 \end{bmatrix} & \text{if } 1 < n < m. \end{cases} \tag{4.8}$$

We also assume that the state $\mathbf{x}_k$ can be observed. We set up the reward matrices to be diagonal.

$$\mathbf{Q} = \text{diag}(q_1, \ldots, q_n) \tag{4.9a}$$

$$\mathbf{R} = \text{diag}(1, r_2, \ldots, r_m) \tag{4.9b}$$

We require that $q_i \geq 0, i = 1, \ldots, m$ and $r_i \geq 0, i = 2, \ldots, m$ such that the matrices, $\mathbf{Q}$ and $\mathbf{R}$, are positive semi-definite. The parameters of our reward function are

$(r_1, \ldots, r_m, q_1, \ldots, q_n)$. However, any scaling of the parameters,

$$\forall M \in \mathbb{R}^+, (Mr_1, \ldots, Mr_m, Mq_1, \ldots, Mq_n),$$

would result in the same behaviour under an optimal control law. Therefore, for each trajectory an infinite number of reward parameters could be found with IRL. To overcome this problem, we can fix one of the parameters, which has the effect of parameter normalization. Thus, we set $r_1 = 1$.

To match the notation in Section 3.3.3, we can define reward features from the reward function (4.1d). Let $\mathbf{x}_k = [x_{1,k}, \ldots, x_{n,k}]^\top$ and $\mathbf{u}_k = [u_{1,k}, \ldots, u_{m,k}]^\top$. Then, we can define the vector of reward features as,

$$\boldsymbol{\phi}(\mathbf{x}_k, \mathbf{u}_k) = [-u_{1,k}^2, \ldots, -u_{m,k}^2, -x_{1,k}^2, \ldots, -x_{n,k}^2]^\top. \tag{4.10}$$

Then, we can summarize the reward parameters in a vector,

$$\boldsymbol{\theta} = [1, r_2, \ldots, r_m, q_1, \ldots, q_n]^\top. \tag{4.11}$$

Using Equations (4.10, 4.11) we can represent the LQR reward function (4.1d) in the same format as Equation (3.7).

$$\begin{aligned} \mathcal{R}_k(\mathbf{x}_k, \mathbf{u}_k) &= -\mathbf{x}_k^\top \mathbf{Q} \mathbf{x}_k - \mathbf{u}_k^\top \mathbf{R} \mathbf{u}_k \\ &= \boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{x}_k, \mathbf{u}_k) \end{aligned}$$

4.3 Synthetic Data Generation Method

For a horizon length, K, state dimension, n, action dimension m, a set of N initial condition, $\mathcal{I} = \{\mathbf{x}_0\}_1^N$, and ground-truth reward parameters, $\boldsymbol{\theta}_e$, we implement the forward LQR solution described in Section 4.1.1 to generate a dataset of optimal trajectories, $\mathcal{D} = \{(\mathbf{x}^{(1)}, \mathbf{u}^{(\mathbf{1})}), \ldots, (\mathbf{x}^{(N)}, \mathbf{u}^{(N)})\}$. In order to evaluate the performance of our inverse LQR algorithm in a setting with a stochastic policy, we perturb the optimal input by a random noise. Therefore, instead of Equation (4.4) we have the following set up,

$$\mu_k^*(\mathbf{x}_k) = \mathbf{L}_k \mathbf{x}_k + \mathbf{w}_k \tag{4.12}$$

where $\mathbf{w}_k \sim \mathcal{N}(0,)$ is a disturbance sampled drawn from a Gaussian distribution with mean 0 and covariance . For the first experiment, we will set the covariance to zero to have an exact policy. In the second set of experiments we vary the covariance to

investigate the effect of policy stochasticity in the example trajectories on the accuracy of the parameters.

4.4 Implementation Details

Our implementation uses Python (v3.8.5) with Numpy (v1.19.4) [20] for storing vectors and matrices. We use the NLOpt package (v2.6.2) [26] implementation of the Sequential Least-Squares Quadratic Program (SLSQP) algorithm [30] to optimize the IRL objective function. Below, we describe the IRL objective function, then we make note of two numerical tricks that are used in the implementation.

4.4.1 IRL Objective and Optimization

System (4.1) has deterministic dynamics and infinite state and action spaces. We will use the IRL technique described in [33] to recover the reward parameters for this system. In order to estimate the likelihood of a trajectory, we need to calculate the matrices in Equation (3.35). Recall the lifted vector notation for a trajectory, $\mathbf{x} = [\mathbf{x}_1, \dots, \mathbf{x}_K]^\top$, and $\mathbf{u} = [\mathbf{u}_0, \dots, \mathbf{u}_{K-1}]^\top$. Because our system has deterministic dynamics, any trajectory can be fully described by an initial condition, $\mathbf{x}_0$ and a set of actions, $\mathbf{u} \in \mathbb{R}^{Km}$. Therefore, to use the Laplace approximate (Equation (3.31)) we need to find the gradient and Hessian matrices, $\mathbf{g}$ and $\mathbf{H}$, of the reward function with respect to the the input, $\mathbf{u}$. Then, we can use the the log likelihood function described in Equation (3.35) and its gradient with respect to the parameters (Equation (3.36)) to perform IRL. We use the chain rule to find the values of the matrices,

$$\mathbf{g} = \frac{\partial \mathcal{R}}{\partial \mathbf{u}} + \frac{\partial \mathbf{x}^\top}{\partial \mathbf{u}} \frac{\partial \mathcal{R}}{\partial \mathbf{x}} \tag{4.13a}$$

$$\mathbf{H} = \frac{\partial^2 \mathcal{R}}{\partial \mathbf{u}^2} + \frac{\partial \mathbf{x}^\top}{\partial \mathbf{u}} \frac{\partial^2 \mathcal{R}}{\partial \mathbf{x}^2} \frac{\partial \mathbf{x}}{\partial \mathbf{u}} + \frac{\partial^2 \mathbf{x}}{\partial \mathbf{u}^2} \cdot \frac{\partial \mathcal{R}}{\partial \mathbf{x}} \tag{4.13b}$$

where $\cdot$ represents a tensor contraction. Note that for systems with linear dynamics, such as our LQR system, the three-dimensional tensor $\frac{\partial^2 \mathbf{x}}{\partial \mathbf{u}^2} = 0$. For a system with non-linear dynamics, setting the tensor $\frac{\partial^2 \mathbf{x}}{\partial \mathbf{u}^2} = 0$ is equivalent to linearizing the system around the trajectory. Since the LQR reward is quadratic in terms of the trajectory, i.e. Equation (4.10) is quadratic with respect to the elements in $\mathbf{x}$ and $\mathbf{u}$, the Laplace approximation of the distribution Equation (3.30), is exact. Alternative to solving the Gaussian integral for the Taylor approximate, Equation (3.30) itself has a form that can be solved as a Gaussian integral. For simpler notation, let $\mathbf{J} = \frac{\partial \mathbf{x}^\top}{\partial \mathbf{u}}$, $\tilde{\mathbf{g}} := \frac{\partial \mathcal{R}}{\partial \mathbf{u}}$, $\hat{\mathbf{g}} := \frac{\partial \mathcal{R}}{\partial \mathbf{x}}$,

$\tilde{\mathbf{H}} := \frac{\partial^2 \mathcal{R}}{\partial \mathbf{u}^2}$, $\hat{\mathbf{H}} := \frac{\partial^2 \mathcal{R}}{\partial \mathbf{x}^2}$, and $\check{\mathbf{H}} := \frac{\partial^2 \mathbf{x}}{\partial \mathbf{u}^2}$. With these definitions, Equations (4.13a) can be summarized as,

$$\mathbf{g} = \tilde{\mathbf{g}} + \mathbf{J}\hat{\mathbf{g}} \tag{4.14a}$$

$$\mathbf{H} = \tilde{\mathbf{H}} + \mathbf{J}\hat{\mathbf{H}}\mathbf{J}^\top + \check{\mathbf{H}} \cdot \hat{\mathbf{g}}. \tag{4.14b}$$

For the LQR system (4.1) we need to calculate the values of the vectors and matrices in Equations (4.14a). First, the matrix $\mathbf{J} \in \mathbb{R}^{Km \times Kn}$ has the following upper block-diagonal form,

$$\mathbf{J} = \begin{bmatrix} \frac{\partial \mathbf{x}_1^\top}{\partial \mathbf{u}_0} & \cdots & \frac{\partial \mathbf{x}_K^\top}{\partial \mathbf{u}_0} \\ \vdots & & \vdots \\ \frac{\partial \mathbf{x}_1^\top}{\partial \mathbf{u}_{K-1}} & \cdots & \frac{\partial \mathbf{x}_K^\top}{\partial \mathbf{u}_{K-1}} \end{bmatrix} = \begin{bmatrix} \mathbf{B}^\top & \mathbf{B}^\top\mathbf{A}^\top & \dots & \mathbf{B}^\top(\mathbf{A}^{K-1})^\top \\ & \mathbf{B}^\top & \dots & \mathbf{B}^\top(\mathbf{A}^{K-2})^\top \\ & & \ddots & \vdots \\ & & & \mathbf{B}^\top \end{bmatrix} \tag{4.15}$$

where the matrices $\mathbf{A}$ and $\mathbf{B}$ are defined in Equations (4.7, 4.8), respectively. Second, the vectors $\tilde{\mathbf{g}} \in \mathbb{R}^{Km}$ and $\hat{\mathbf{g}} \in \mathbb{R}^{Kn}$ have the following form,

$$\tilde{\mathbf{g}} = \begin{bmatrix} \frac{\partial \mathcal{R}}{\partial \mathbf{u}_0} \\ \hline \vdots \\ \hline \frac{\partial \mathcal{R}}{\partial \mathbf{u}_{K-1}} \end{bmatrix} = \begin{bmatrix} -2u_{1,0} \\ -2r_2u_{2,0} \\ \vdots \\ -2r_mu_{m,0} \\ \hline \vdots \\ \hline -2u_{1,K-1} \\ -2r_2u_{2,K-1} \\ \vdots \\ -2r_mu_{m,K-1} \end{bmatrix} \tag{4.16}$$

$$\hat{\mathbf{g}} = \begin{bmatrix} \frac{\partial \mathcal{R}}{\partial \mathbf{x}_1} \\ \vdots \\ \frac{\partial \mathcal{R}}{\partial \mathbf{x}_K} \end{bmatrix} = \begin{bmatrix} -2q_1 x_{1,0} \\ -2q_2 x_{2,0} \\ \vdots \\ -2q_n x_{n,0} \\ \hline \vdots \\ \hline -2q_1 x_{1,K} \\ -2q_2 x_{2,K} \\ \vdots \\ -2q_n x_{n,K}. \end{bmatrix} \tag{4.17}$$

Finally, the matrices $\tilde{\mathbf{H}} \in \mathbb{R}^{Km \times Km}$ and $\hat{\mathbf{H}} \in \mathbb{R}^{Kn \times Kn}$ have the following symmetric forms,

$$\tilde{\mathbf{H}} = \begin{bmatrix} \frac{\partial^2 \mathcal{R}}{\partial \mathbf{u}_0^2} & \frac{\partial^2 \mathcal{R}}{\partial \mathbf{u}_0 \partial \mathbf{u}_1} & \cdots & \frac{\partial^2 \mathcal{R}}{\partial \mathbf{u}_0 \partial \mathbf{u}_{K-1}} \\ \frac{\partial^2 \mathcal{R}}{\partial \mathbf{u}_1 \partial \mathbf{u}_0} & \frac{\partial^2 \mathcal{R}}{\partial \mathbf{u}_1^2} & \cdots & \frac{\partial^2 \mathcal{R}}{\partial \mathbf{u}_1 \partial \mathbf{u}_{K-1}} \\ \vdots & & \ddots & \vdots \\ \frac{\partial^2 \mathcal{R}}{\partial \mathbf{u}_{K-1} \partial \mathbf{u}_0} & \cdots & \cdots & \frac{\partial^2 \mathcal{R}}{\partial \mathbf{u}_{K-1}^2} \end{bmatrix} = \begin{bmatrix} -2\mathbf{R} & & & \\ & -2\mathbf{R} & & \\ & & \ddots & \\ & & & -2\mathbf{R} \end{bmatrix} \tag{4.18}$$

$$\hat{\mathbf{H}} = \begin{bmatrix} \frac{\partial^2 \mathcal{R}}{\partial \mathbf{x}_1^2} & \frac{\partial^2 \mathcal{R}}{\partial \mathbf{x}_1 \partial \mathbf{x}_2} & \cdots & \frac{\partial^2 \mathcal{R}}{\partial \mathbf{x}_1 \partial \mathbf{x}_K} \\ \frac{\partial^2 \mathcal{R}}{\partial \mathbf{x}_2 \partial \mathbf{x}_1} & \frac{\partial^2 \mathcal{R}}{\partial \mathbf{x}_2^2} & \cdots & \frac{\partial^2 \mathcal{R}}{\partial \mathbf{x}_2 \partial \mathbf{x}_K} \\ \vdots & & \ddots & \vdots \\ \frac{\partial^2 \mathcal{R}}{\partial \mathbf{x}_K \partial \mathbf{x}_1} & \cdots & \cdots & \frac{\partial^2 \mathcal{R}}{\partial \mathbf{x}_K^2} \end{bmatrix} = \begin{bmatrix} -2\mathbf{Q} & & & \\ & -2\mathbf{Q} & & \\ & & \ddots & \\ & & & -2\mathbf{Q}. \end{bmatrix} \tag{4.19}$$

We can see that in the case of the LQR System (4.1), the matrices $\tilde{\mathbf{H}}$ and $\hat{\mathbf{H}}$ can be expressed as K block-diagonal repetitions of $-2\mathbf{R}$ and $-2\mathbf{Q}$, respectively. The matrices $\mathbf{R}$ and $\mathbf{Q}$ are defined in Equations (4.9).

We need to maximize the log likelihood (Equation (3.35)) in order to estimate the reward parameters, starting with an initial set of parameters, $\boldsymbol{\theta}_0$.

4.4.2 Numerical Considerations

Calculating $\log|-\mathbf{H}|$

The likelihood equation, Equation (3.35), requires calculating $\log|-\mathbf{H}|$. The matrix $\mathbf{H} \in \mathbb{R}^{Km \times Km}$ is large and can have a very large determinant. We observed that calculating the determinant may cause floating point overflows. However, we only require the log of the determinant, thus we use the following matrix identity to avoid floating point

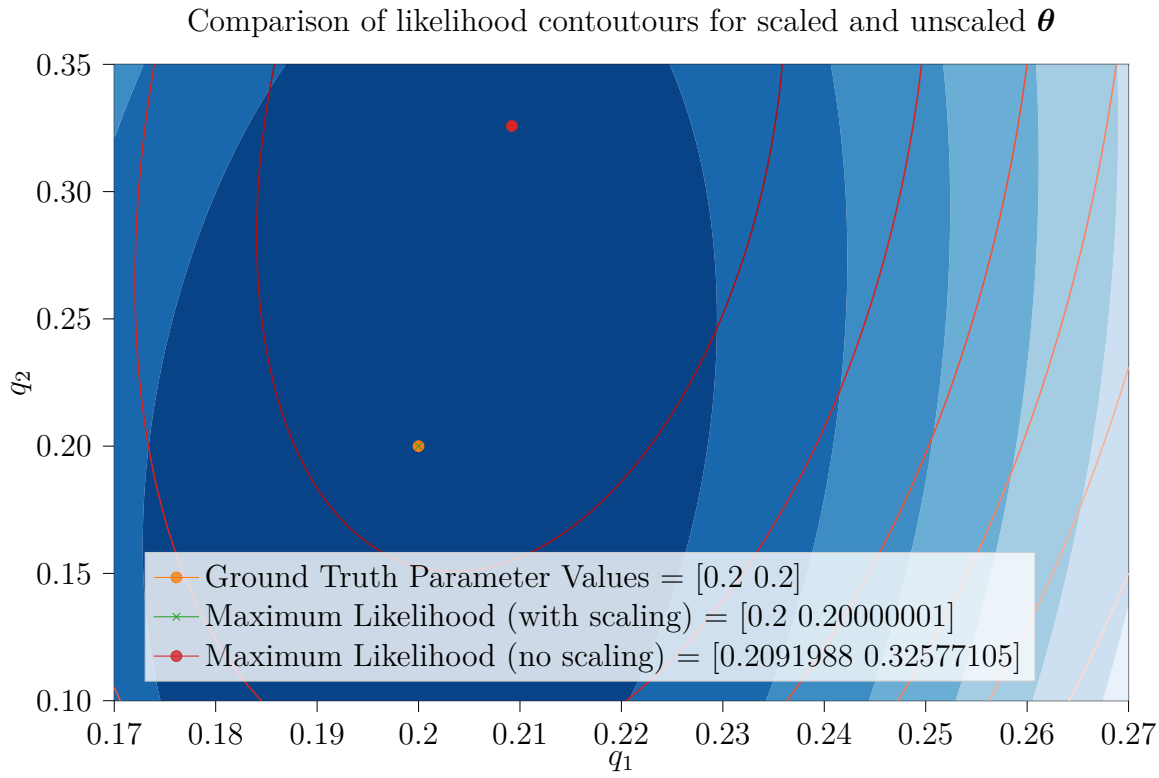


Figure 4.1: The effect of parameter scaling on maximum likelihood offset for a system with a two-dimensional state and single action. The LQR parameter r_1 is fixed and the parameters q_1, q_2 are learned. The blue contour illustrates the likelihood function for parameters scaled by $r_1 = 10^5$ and the red contour lines illustrate the likelihood function without scaled parameters.

overflows,

$$\log|-\mathbf{H}| = 2\text{tr}(\mathbf{L}) \tag{4.20}$$

where $\mathbf{L}$ denotes the Cholesky decomposition of the positive matrix $-\mathbf{H} = \mathbf{L}\mathbf{L}^\top$ and $\text{tr}(\cdot)$ denotes the trace of the matrix.

Scaling of Parameters

By using the Maximum Causal Entropy Probability distribution for the LQR problem, we are estimating the max function in the Bellman Equation (3.5) with a softmax as in Equation (3.24). As discussed in Section 3.3.2, for some function, $f(x)$, $\text{softmax}_x\, \alpha f(x) \to \max_x \alpha f(x)$ as the coefficient of the argument, $\alpha \to \infty$. Therefore, the larger the magnitude of the stage-reward of the LQR problem (Equation (4.1d)), the more accurately the maximum causal entropy probability distribution estimates the Bellman equations. We observed that for problems with small reward magnitudes, the maximum of the likelihood

function, Equation (3.35), is significantly offset from the ground truth parameters. To fix this problem, we increase the magnitude of the reward function by setting $r_1 >> 1$ and scaling the rest of the parameters in $\boldsymbol{\theta}$ accordingly. In Figure 4.1 we illustrate the offset between the likelihood function with unscaled parameters and the ground truth parameter values.

We also need to balance the requirement for a large reward magnitude with the floating point limitations of our computing resources. Practically, we found a good scale that works for a variety of example trajectories is $r_1 = \frac{10^5}{||\bar{\mathbf{x}_0}||}$, where $||\bar{\mathbf{x}_0}||$ is the Euclidean norm of the mean initial condition of the expert trajectories in the dataset.

4.5 Results and Discussion

4.5.1 Experiment 1: IRL with expert trajectories generated with perfect policy

We first present the results of performing IRL on an LQR example with no noise on the policy. We set the horizon length, $K = 100$, state dimension, $n = 2$, action dimension $m = 1$, a single initial condition, $\mathbf{x}_0 = [5, 20]^\top$, ground-truth LQR reward parameters, $\boldsymbol{\theta}_e = [1, q_{1,e}, q_{2,e}]^\top = [1, 0.005, 0.045]^\top$, and initial condition for LQR reward parameters in IRL optimization $\boldsymbol{\theta}_0 = [1, 0.001, 0.0005]^\top$. We use the IRL implementation described in Section 4.2 to obtain an estimate of the LQR reward parameters, $\boldsymbol{\theta}_f = [1, q_{1,f}, q_{2,f}]^\top$.

The results of using the IRL algorithm are summarized in Table 4.1. We report the parameters obtained by IRL, $\boldsymbol{\theta}_f$. We also report three metrics to guage the similarity of a trajectory generated by $\boldsymbol{\theta}_f$ with a trajectory generated by the ground truth, $\boldsymbol{\theta}_e$. We use the solution to LQR, DARE Equation (4.4), to generate a new trajectory in order to validate the behaviour of the system under the obtained parameters. We report the Root Mean Squared Error (RMSE) between the trajectories generated from the ground truth parameters, $\boldsymbol{\theta}_e$ and the obtained parameters, $\boldsymbol{\theta}_f$ for each dimension of the system. We also report the Mean Euclidean Error. As shown in the table, the difference in the parameters and the trajectories are negligible.

Figure 4.2 illustrates a contour plot of the IRL approximate log likelihood (Equation (3.35) for parameter combinations of (q_1, q_2) along with the evolution of the reward parameters over the IRL optimization process. We can observe that the parameters q_1 and q_2 approach the ground truth values $q_{1,e}$ and $q_{2,e}$ as we maximize the IRL approximate log likelihood.

Figure 4.3 illustrates the trajectories generated by LQR with ground truth param-

Table 4.1: Results of IRL on our LQR System

Evaluation Parameters	**Values**
Estimated Parameter Values for $\boldsymbol{\theta}_f$ $(q_{1,f}, q_{2,f})$	$0.00499999, 0.04500091$
Root Mean Squared Error (RMSE) for the x_1 dimension	1.33×10^{-5}
Root Mean Squared Error (RMSE) for the x_2 dimension	3.78×10^{-6}
Mean Euclidean Error (MED)	7.08×10^{-6}

eters, $\boldsymbol{\theta}_e$, initial parameters for IRL, $\boldsymbol{\theta}_0$, and final parameters obtained by IRL, $\boldsymbol{\theta}_f$. Figures 4.3a and 4.3b display the state values for each dimension over the time horizon, and Figure 4.3c displays the trajectories. We can observe that the trajectories generated using $\boldsymbol{\theta}_f$ is nearly identical to the ground-truth trajectory. We can interpret this observation to mean that the RMSE values presented in Table 4.1 are generated from consistently low error along the entire trajectory.

4.5.2 Experiment 2: IRL with expert trajectories generated with stochastic policies

To evaluate the performance of the IRL algorithm under variable amounts of policy noise, we conduct a series of experiments. We set the horizon length, $K = 20$, state dimension, $n = 2$, action dimension $m = 1$, ground-truth reward parameters, $\boldsymbol{\theta}_e = [1, q_{1,e}, q_{2,e}]^\top = [1, 0.2, 0.2]^\top$. Since $m = 1$, the noise on the policy, $w_k \sim \mathcal{N}(0, \sigma^2)$ is a scalar. We conduct experiments with w_k sampled from normal distributions with several standard deviation values: $\sigma^2 = 0.01, 0.05, 0.1, 0.15, 0.2$. For each value of σ^2, we generate trajectory datasets with the number of examples, N, increasing by multiples of 4 i.e. $N = 1, 4, 16, \ldots, 4^8$.

In the first set of experiments, every trajectory in the dataset starts from the same initial condition $\mathbf{x}_0 = [10, 10]^\top$. For each combination of σ^2 and N, we generate 40 datasets of trajectories, $\mathcal{D}$. The 40 datasets are equally divided among four parameter initial conditions, $\boldsymbol{\theta}_0 = [1, q_{1,0}, q_{2,0}]^\top$. We select parameter initial conditions that are 25% deviated from the ground truth values in the q_1, q_2 directions, i.e. $\boldsymbol{\theta}_0$ takes the following values: $[1, 0.15, 0.15]^\top, [1, 0.15, 0.25]^\top, [1, 0.25, 0.15]^\top, [1, 0.25, 0.25]^\top$. We use the IRL implementation described in Section 4.2 to obtain an estimate of the LQR parameters, $\boldsymbol{\theta}_f = [1, q_{1,f}, q_{2,f}]^\top$ based on each dataset. In this set of experiments we generate 1800 datasets, and conduct a total of 1800 respective IRL optimizations.

We evaluate the Euclidean distance of the ground truth parameters from the final parameters in the parameter space, $||\boldsymbol{\theta}_f - \boldsymbol{\theta}_e||$. Figure 4.4 illustrates the Euclidean distances for the results obtained from each IRL optimization. Each point on the plot denotes $||\boldsymbol{\theta}_f - \boldsymbol{\theta}_e||$ for an IRL performed using a particular dataset. We plot the mean

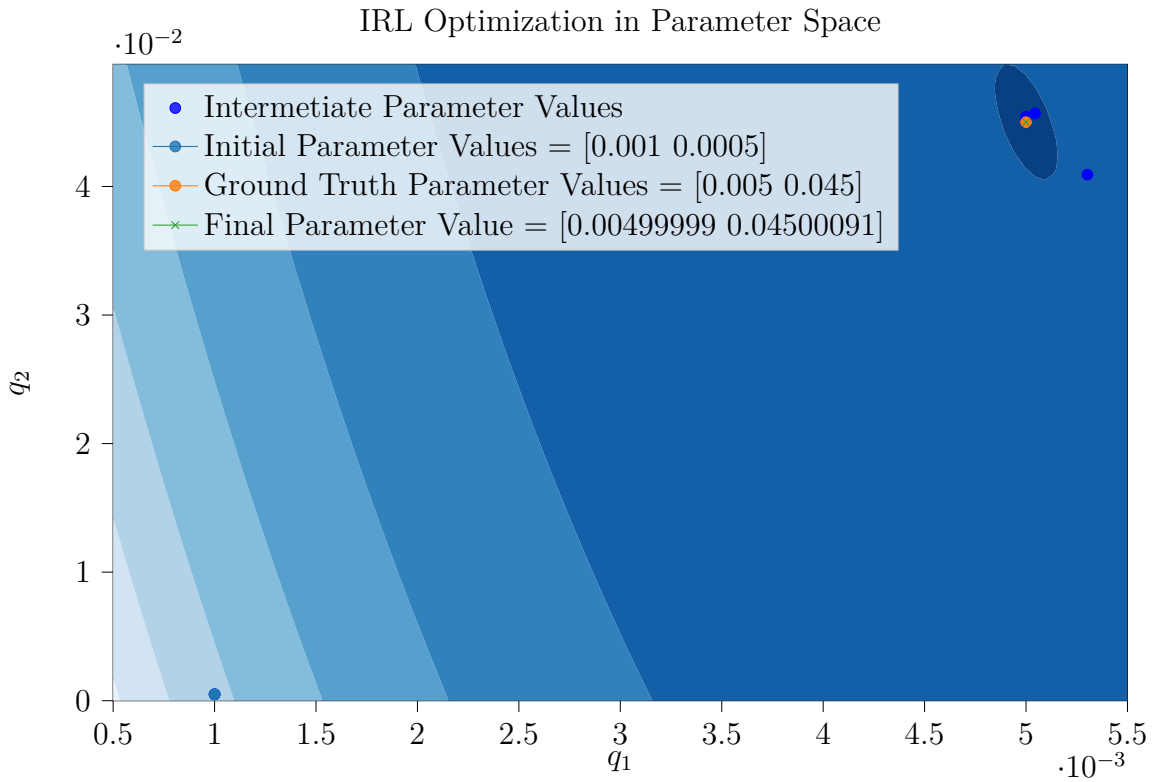


Figure 4.2: Evolution of reward parameter estimates over the IRL optimization process. Note that all the values on the q_1 and q_2 axes were scaled for performing the optimization as described in Section 4.4.1. Unscaled values are illustrated on this graph.

parameter Euclidean distance for each group of experiments with the same number of example trajectories.

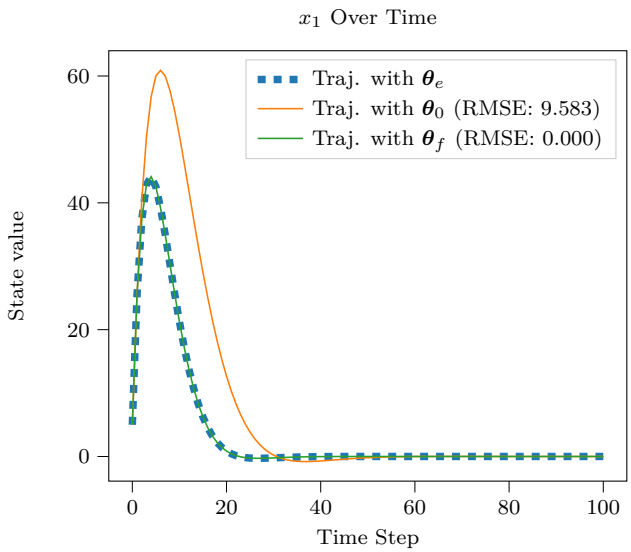
We first note that on average lower σ^2 results in a more accurate parameter set for every N. As expected, less perturbed example trajectories result in better IRL performance. For each σ^2, we can also observe that as the number of expert trajectories increases, the final parameters become more accurate. Since the policies that generate the example trajectories are perturbed by additive white Gaussian noise, we expect that adding more trajectories results in the effect of the noise to be diminished.

Interestingly, we can observe that the standard deviation of the noise, σ^2, has an effect on the speed at which the error in the parameters tends to zero. The higher the standard deviation, the less of an effect increasing the size of the dataset has on the accuracy of the learned parameters. Furthermore, we can also observe that the larger the dataset size, the lower the deviation in the error across different IRL optimizations. In other words, the error in the parameter estimates, $\boldsymbol{\theta}_f$ are more consistent in experiments with a higher number of example trajectories. Therefore for very noisy data, even exponentially

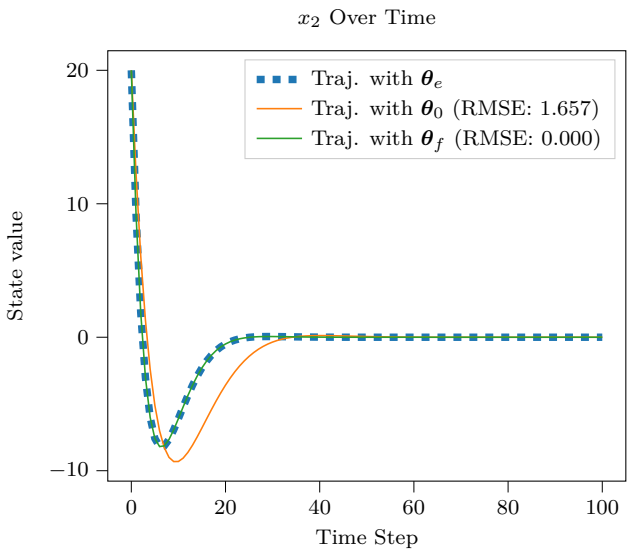
increasing the number of trajectories may have a negligible effect on the accuracy of the parameters obtained by IRL.

We performed another set of 1800 IRL optimizations, with the same experimental parameters, except for the initial conditions, $\mathbf{x_0}$, of the example trajectories in the datasets. In the second set, each of the example trajectory starts from an initial condition that is drawn from a uniform distribution $\mathbf{x}_0 = [x_0^1, x_0^2]^\top$, where $x_0^1, x_0^2 \sim \mathcal{U}(-10, 10)$.

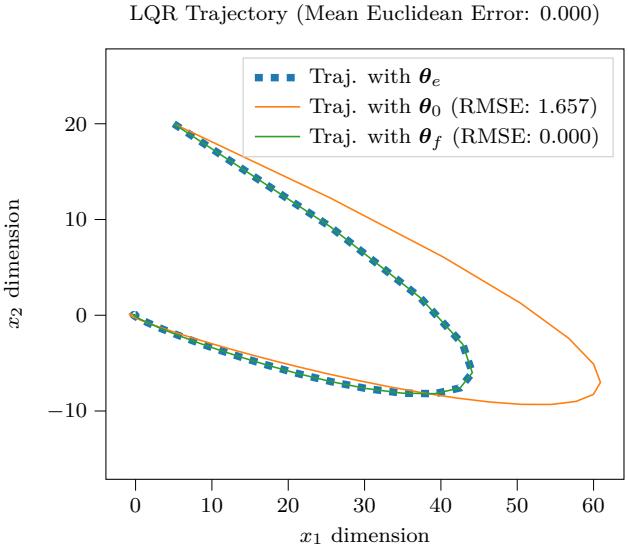
Figure 4.5 illustrates the Euclidean distance of the learned parameters from the ground truth for each group of experiments with the same number of example trajectories. This set of experiments produce worse parameter estimates than the set of experiments where the example trajectories are generated from the same initial condition. We explain the worse performance by the fact that in the second set of datasets we have N different example trajectories each perturbed by noise. In contrast, in the first set of datasets we had N copies of the same trajectory, distinguished from each other only by the effect of noise. The parameter error in this set of experiments also tends more slowly to zero than the previous set. We can conclude that for datasets where the trajectories are generated from random initial conditions, we would need to have much more data to achieve an accurate parameter estimate.



(a) First Dimension



(b) Second Dimension



(c) Trajectory

Figure 4.3: Trajectories generated by LQR with ground truth parameters, $\boldsymbol{\theta}_e$ (blue), initial parameters for IRL, $\boldsymbol{\theta}_0$ (orange), and final paramters generated by IRL, $\boldsymbol{\theta}_f$ (green) (c). The values of the state at each time step are illustrated for the first dimension (a) and the second dimension (b).

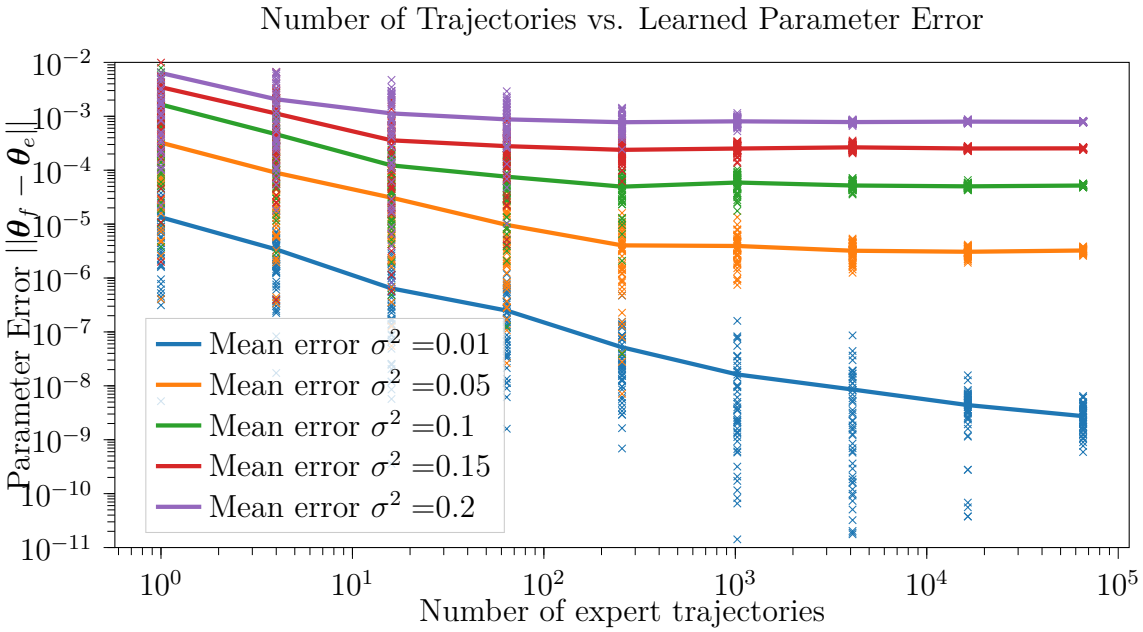


Figure 4.4: Error in learned parameters for different amounts of expert trajectories used for IRL on plotted on a log-log scale. In this set of experiments, all example trajectories start from the same initial condition, $\mathbf{x}_0 = [10, 10]^\top$

.

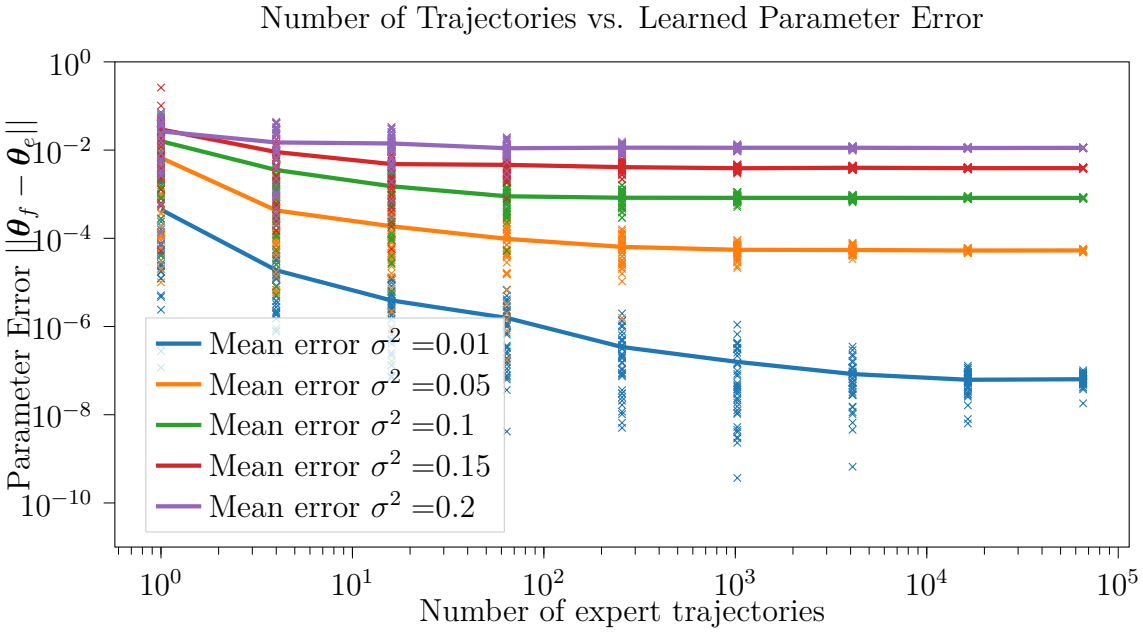


Figure 4.5: Error in learned parameters for different amounts of expert trajectories used for IRL on plotted on a log-log scale. In this set of experiments, each example trajectories start from a randomly selected initial condition, sampled from a uniform distribution, $\mathbf{x}_0 = [x_0^1, x_0^2]^\top$, where $x_0^1, x_0^2 \sim \mathcal{U}(-10, 10)$

Chapter 5

Unpredictability in Lane Changes

In this chapter we will use Inverse Reinforcement Learning (IRL) to model reward functions for conducting a lane change maneuver in a highway setting. The key hypothesis of this project is that the unpredictability of the surrounding traffic will have an effect on the lane change behaviour of the ego-car. In order to evaluate this hypothesis we learn two reward functions using IRL, a baseline reward function and a reward function that includes an unpredictability feature.

5.1 Overview

In this chapter, we describe the formulation of the lane change maneuver as an optimal control problem, the pre-processing of lane change data, implementation of IRL, implementation of the forward optimal control problem, and results of the comparison between the two models. An overview of our experimental methodology is presented as a block diagram in Figure 5.1. Using human expert lane change trajectories from the NGSIM dataset, $\{(\mathbf{x}_e^{(i)}, \mathbf{u}_e^{(i)})\}_{i=0}^{N}$, we learn two parametrizations for two reward functions using IRL, $\boldsymbol{\theta}_{f,b}$ for the baseline reward function and $\boldsymbol{\theta}_{f,w}$ with the additional unpredictability feature. The shaded blocks refer to components related to the baseline reward function, and the plain blocks refer to components that use the reward function that contains the additional unpredictability feature. For each model, we perform optimal control to generate trajectories using the learned reward functions, $\{(\mathbf{x}_{f,b}^{(i)}, \mathbf{u}_{f,b}^{(i)})\}_{i=0}^{N}$ and $\{(\mathbf{x}_{f,w}^{(i)}, \mathbf{u}_{f,w}^{(i)})\}_{i=0}^{N}$, respectively. Then we compare the generated trajectories with the human expert trajectories by calculating the Average Mean Euclidean Error, $\overline{MEE}_{ef,b}$ and $\overline{MEE}_{ef,w}$. We evaluate the improvement in performance of the model that from adding the unpredictability feature to show our hypothesis.

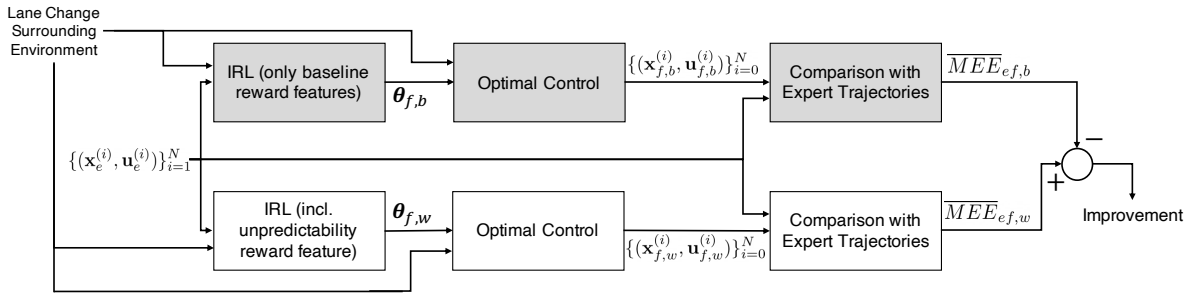


Figure 5.1: Blockdiagram of how we intent to learn and compare the reward function learned using only the baseline features and the reward function learned that adds the unpredictability feature.

5.2 Lane Change Problem Formulation

We formulate a lane change maneuver as an Optimal Control problem that we will solve using trajectory optimization. We model the maneuver as a deterministic discrete control task over a finite horizon of length K. We define the initial condition x_0, continuous states, $\mathbf{x} = [\mathrm{x}_1{}^\top, \ldots, \mathrm{x}_K^\top]^\top$, and continuous control actions $\mathbf{u} = [\mathrm{u}_0{}^\top, \ldots, \mathrm{u}_{K-1}^\top]^\top$. We characterize lane change with time invariant system dynamics, $\mathrm{x}_{k+1} = \mathrm{f}(\mathrm{x}_k, \mathrm{u}_k)$ a stage reward, $\mathcal{R}_k(\mathrm{x}_k, \mathrm{u}_k)$, and a terminal reward $\mathcal{R}_K(\mathrm{x}_K)$. The optimal control actions are found by maximizing the stage reward over the horizon,

$$\begin{aligned} \mathbf{u}^* = \arg\max_{\mathbf{x},\mathbf{u}} \sum_{k=1}^{K-1} \mathcal{R}_k(\mathrm{x}_k, \mathrm{u}_k) + \mathcal{R}_K(\mathrm{x}_K) \\ s.t. \quad \mathrm{x}_{k+1} = \mathrm{f}(\mathrm{x}_k, \mathrm{u}_k). \end{aligned} \tag{5.1}$$

5.2.1 Ego-vehicle Dynamics Model

We model the dynamics of the ego-vehicle with a simple kinematic unicycle model. Then discretize the system to match the NGSIM dataset [12, 11]. We first define the state as,

$$\mathrm{x} = \begin{bmatrix} x \\ y \\ \psi \end{bmatrix} \in \mathcal{X} = \mathbb{R}^3 \tag{5.2}$$

where (x, y) is the two-dimensional position of the ego-vehicle and ψ is its heading. We define the control action as,

$$\mathrm{u} = \begin{bmatrix} v \\ \omega \end{bmatrix} \in \mathcal{U} = \mathbb{R}^2 \tag{5.3}$$

where v is the longitudinal velocity input and ω steering input of the car. Then, the dynamics of the vehicle are defined as,

$$\dot{\mathrm{x}} = \mathrm{f}(\mathrm{x}, \mathrm{u}) = \begin{bmatrix} vcos(\psi) \\ vsin(\psi) \\ \omega \end{bmatrix}. \tag{5.4}$$

We discretize the dynamics model using the forward Euler method with sampling rate $\delta t = 0.1s$ for the NGSIM dataset.

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k) = \mathbf{x}_k + \delta t \begin{bmatrix} v_k cos(\psi_k) \\ v_k sin(\psi_k) \\ \omega_k \end{bmatrix} \tag{5.5}$$

5.2.2 Lane Change Environment

We use the phrase *lane change environment* to refer to both the geometry of the highway in the vicinity of the lane change, as well as the adjacent vehicles that are surrounding the ego-vehicle during the maneuver. The lane change maneuver consists of the vehicle traveling from the *current-lane* (CL) to the *target-lane* (TL). We formulate each lane as a line parametrized by two points, $\mathrm{p}_1^{CL} = [x_1^{CL}, y_1^{CL}]^\top$, $\mathrm{p}_2^{CL} = [x_2^{CL}, y_2^{TL}]^\top$ and $\mathrm{p}_1^{TL} = [x_1^{TL}, y_1^{TL}]^\top$, $\mathrm{p}_2^{TL} = [x_2^{TL}, y_2^{TL}]^\top$, respectively.

We take into account four vehicles that are surrounding the ego-vehicle during the maneuver: the vehicle (1) *preceding in the current lane*, (2) *following in the current lane*, (3) *preceding in the target lane*, and (4) *following in the target lane*. Each adjacent vehicle is parametrized as a point-mass with a two-dimensional position. For the four adjacent vehicles, $i = 1, \ldots, 4$, and timesteps $k = 1, \ldots, K$, we take into account $\mathbf{x}_{s,i,k} = [x_{s,i,k}, y_{s,i,k}]^\top$. For each lane change maneuver, we take the average speed of the four adjacent vehicles as the desired velocity, $v_d = \sum_{i=1}^{4} \sum_{k=1}^{K} v_{s,i,k}$.

5.2.3 Baseline Reward Function

In order to use the IRL algorithm, presented in Section 3.3, we formulate the reward function at each time step as a linear combination of features. The reward function is written as,

$$\mathcal{R}_k(\mathbf{x}_k, \mathbf{u}_k) = \boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{x}_k, \mathbf{u}_k) \tag{5.6}$$

where $\boldsymbol{\phi} := [\phi_1, \ldots, \phi_p]^\top$ are referred to as a set of p features. For $j = 1 \ldots p$, each feature is defined as a function $\phi_j : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ that maps from the state or control action spaces

to a real-valued number at a particular time step k. The parameters $\boldsymbol{\theta} := [\theta_1, \ldots, \theta_p]^\top$ provide the weight of each feature in $\boldsymbol{\phi}$. Only the relative weighting of the reward features has an effect on the maximization of the total reward. Therefore, we fix $\theta_1 = 1$ and vary the other weights accordingly.

Baseline Reward Features

We reviewed features commonly used for trajectory generation in highway driving. Naumann et al. [42] present a detailed survey of features. We present the features that compose our baseline reward function as enumerated below,

1. **Lateral deviation from target lane** [53] is represented as,

$$\phi_d := -exp\left(\frac{d_k}{w}\right) \tag{5.7}$$

where d_k is the distance from the position of the car, $\mathbf{x}_k$, to the centre-line of the target lane. We parametrize the target-lane as a line and w is the average distance between the centre-line of the current lane (CL) and the centre-line of the target lane (TL) in the vicinity of the lane change maneuver. The distance, d_k, is given by the distance from a point to a line,

$$d_k := \frac{|(y_2^{TL} - y_1^{TL})x_k - (x_2^{TL} - x_1^{TL})y_k + x_2^{TL}y_1^{TL} - x_1^{TL}y_2^{TL}|}{\sqrt{(y_2^{TL} - y_1^{TL})^2 + (x_2^{TL} - x_1^{TL})^2}}.$$

Furthermore, since $y_1^{TL} = y_1^{CL}$ and $y_2^{TL} = y_2^{CL}$, we have that,

$$w = \frac{|x_1^{TL} - x_1^{CL}| + |x_2^{TL} - x_2^{CL}|}{2}.$$

2. **Deviation from the mean speed of traffic** is represented as,

$$\phi_v := -(v_k - v_d)^2 \tag{5.8}$$

where v_d is the mean speed of the adjacent vehicles over the entire trajectory.

3. **High angular speed** is represented as,

$$\phi_a := -\omega_k^2 \tag{5.9}$$

4. **Distance from adjacent vehicles**. We evaluated a number of formulations for

the feature that incorporates surrounding vehicles. Since the objective of this thesis is to evaluate how unpredictability of adjacent cars affects driving behaviour, this is the most important feature in our formulation. Our unpredictability feature, which we present in Section 5.3.1, is also an extension of this feature.

- Formulation based on [53].

$$\phi_s := -\sum_{i=1}^{n_s} exp\left(-\sqrt{(x_k - x_{s,i,k})^2 + (y_k - y_{s,i,k})^2}\right) \tag{5.10}$$

- Formulation to take into account the target speed.

$$\phi_s := -\sum_{i=1}^{n_s} exp\left(-\frac{1}{v_d}\sqrt{(x_k - x_{s,i,k})^2 + (y_k - y_{s,i,k})^2}\right) \tag{5.11}$$

- Formulation to take into account the ego-vehicle's speed.

$$\phi_s := -\sum_{i=1}^{n_s} exp\left(-\frac{1}{v_k}\sqrt{(x_k - x_{s,i,k})^2 + (y_k - y_{s,i,k})^2}\right) \tag{5.12}$$

- Formulation to take into account the ego-vehicle's speed.

$$\phi_s := -\sum_{i=1}^{n_s} \left(\frac{1}{\sqrt{(x_k - x_{s,i,k})^2 + (y_k - y_{s,i,k})^2}}\right) \tag{5.13}$$

- Formulation based on [17]. We use this formulation in this thesis.

$$\phi_s := -\sum_{i=1}^{n_s} \left(d_{des} - \sqrt{(x_k - x_{s,i,k})^2 + (y_k - y_{s,i,k})^2}\right)^2 \tag{5.14}$$

  where $d_{des} = d_{min} + t_{headway} v_d$, d_{min} and $t_{headway}$ are hyperparameters, and v_d is the desired speed as defined in Equation (5.8).

- **Formulation based on [17]. We use this formulation in this thesis.**

$$\phi_s := -\sum_{i=1}^{n_s} \left(d_{des} - \sqrt{(x_k - x_{s,i,k})^2 + (y_k - y_{s,i,k})^2}\right)^2 \tag{5.15}$$

  where $d_{des} = d_{min} + t_{headway} v_k$, d_{min} and $t_{headway}$ are hyperparameters, and v_k is the speed of the ego-car.

5.3 Unpredictability Formulation

In this section we describe the unpredictability metric that we use to test the hypothesis that the unpredictability of the surrounding traffic will have an effect on the lane change behaviour of the ego-car. We then describe how we incorporate the unpredictability metric into the lane change reward function structure described in Section 5.2.3.

5.3.1 Unpredictability Metric

The predictability of an agent is often studied in the field of Human-Robot Interaction (HRI) in the context of user studies, which are limited in terms of the scope of behaviours encountered. Chapter 2 summarizes some relevant literature regarding predictability in HRI. However, most AVs that operate in human environments, use a prediction model of the other agents' behaviour [49]. We propose a different approach to measuring predictability – which we call *unpredictability* – that uses the performance of this prediction model. We propose to take an off-the-shelf model of behaviour prediction that has been learned from human data, then to use the performance of that model as a heuristic for measuring predictability.

Behaviour prediction in literature is often performed using camera data. The visual information is fed directly into Long Short-Term memory (LSTM) recurrent neural network architectures [36, 55, 10], or other stochastic methods such as Variational Auto Encoders (VAEs) [3, 46, 58]. However since our data does not include visual information this type of prediction model is incompatible for our purposes. Other proposals use a variety of sensors, such as cameras or LIDAR to represent the surrounding world as a Dynamic Occupancy Grid Map (DOGMa) in 2D [22, 39, 23] or 3D [37]. These models are concerned with predicting the drivable space in the future, and do not explicitly consider the interdependence of agents that participate in the flow of the highway in their design. The final group of models that we investigated look at individual agents and their trajectories [28, 14]. These models predict the future trajectory of agents given a trajectory history. We use the trajectory prediction model presented in [14], which takes into account the behavioural interdependencies between the agents in the highway.

The authors in [14] use an encoder-decoder LSTM model with convolutional social pooling. The encoder LSTMs are responsible for learning vehicle dynamics based on trajectory history input. They use convolutional social pooling layers, an extension of social pooling [2], to take into account the interdependencies between the different vehicles on the highway. Finally, the encoder is responsible for outputting a probability of maneuver classes as well as the most probable trajectory, given a predicted maneuver. We take the

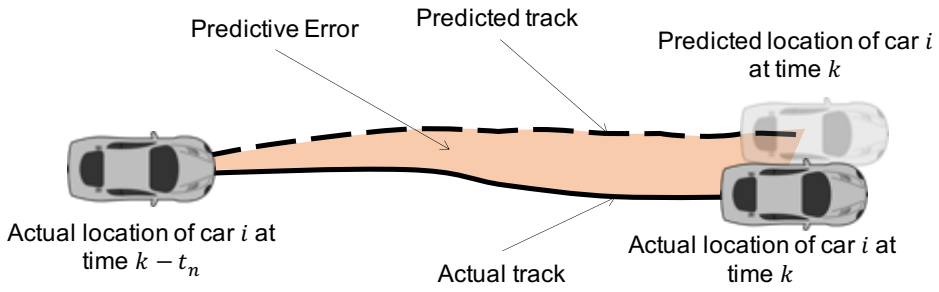


Figure 5.2: Illustration of unpredictability metric. We define unpredictability at time k as the mean predictive error of the prediction made by the model at time $k - t_n$.

highest-probability maneuver's trajectory as the model's prediction. We calculate the Mean Euclidean Error of the prediction with respect to the ground truth and use this value as the unpredictability metric. As illustrated in Figure 5.2, for adjacent vehicle i at time k, we evaluate the predictive error of the prediction made by the model at time $k - t_n$. Then take the average of the error from time $k - t_n$ to k as the unpredictability measure. Concretely, for adjacent car i at time step k the mean Euclidean predictive error is defined as,

$$z_{i,k} := \frac{1}{t_n} \sum_{j=k-t_n}^{k} \sqrt{(x_{i,j} - \check{x}_{i,j}^{(k-t_n)})^2 + (y_{i,j} - \check{y}_{i,j}^{(k-t_n)})^2} \tag{5.16}$$

where t_n is the number of time steps we look to the past, $(\check{x}_{i,j}^{(k-t_n)}, \check{y}_{i,j}^{(k-t_n)})$ is the prediction made at time $k - t_n$ by the model [14] of where car i would be at time j, and $(x_{i,j}, y_{i,j})$ is the observed position of car i at time j.

Unpredictability Metric Normalization

We normalize the unpredictability metric for each adjacent car in each trajectory to lie between $[0, 1]$ given a dataset of trajectories. The normalized unpredictability metric is given by,

$$\hat{z}_{i,k} := \frac{z_{i,k} - z_{i,k,min}}{z_{i,k,max} - z_{i,k,min}}. \tag{5.17}$$

where $z_{i,k,min}$ and $z_{i,k,max}$ represent the minimum and maximum values across every adjacent vehicle in every trajectory in a dataset, respectively.

5.3.2 Unpredictability Reward Feature

In order to incorporate the measure of unpredictability into our baseline lane change reward function, described in Section 5.2.3, we augment the reward function with the following additional feature,

5. **Deviation from unpredictability-weighted desired distance to adjacent cars**. The intuition behind this formulation is that for an adjacent vehicle, i, the driver will favour being farther away, if the adjacent vehicle is behaving unpredictably.

$$\phi_z := -\sum_{i=1}^{n_s} \left(d_{des} - \sqrt{(x_k - x_{s,i,k})^2 + (y_k - y_{s,i,k})^2} \right)^2 \tag{5.18}$$

where

$$d_{des} = d_{min} e^{\hat{z}_{i,k}} + t_{headway} v_k. \tag{5.19}$$

The unpredictability metric, $\hat{z}_{i,k}$, is defined in Equations (5.16) and (5.17), the hyperparameters d_{min} and $t_{headway}$ have the same values as the baseline reward feature ϕ_s (Equation (5.15)), and v_k is the speed of the ego-car.

With this formulation, the desired minimum distance increases exponentially with the unpredictability of the adjacent vehicle.

5.4 Dataset Generation

In this section we explain the data preprocessing tasks performed on the NGSIM data to generate the human expert lane change trajectories. We start with the NGSIM US-101 [12] and I-80 [11] datasets, which contain observations of traffic flow on a roughly $600m$ and $500m$ stretch straight freeway, respectively. Figure 5.3 illustrates the geometry of the freeways. For each freeway, the trajectories are recorded over three 15-minute time slots that correspond to varying levels of congestion. The information that we use from the NGSIM datasets include: 2D position labels of vehicles, lane labels of vehicles, and the velocity of the vehicles. We rely on the training, validation, and testing set splitting from the off-the-shelf prediction model [14] that we use for the unpredictability measure described in Section 5.3.1. Only the lane changes used in the training set for the prediction model are included in our lane change trajectory training set.

In the following subsections we first describe how we smoothed the position labels and extracted the ego-vehicle's dynamics. Then we describe how we selected lane change trajectories and characterized the lane change environment. We also describe how we

divided the data into several different datasets based on geometry, and traffic congestion. Finally, we describe how we normalized the feature values within each dataset.

5.4.1 Data Smoothing

Our primary source of data from the NGSIM dataset are the paths of individual vehicles. Sources of errors in the construction of vehicle trajectories in the NGSIM dataset include mislabelling and noise [54]. The two-dimensional position and lane classification of vehicles are labelled using bounding boxes on images from static video cameras. In some frames, the bounding boxes are slightly translated with respect to the actual position of the vehicle in the image [41]. In a trajectory that transitions from a well-labelled frame to a misaligned frame, there is a jump in the position signals of the vehicle. Additionally, random noise in the labelling position information contributes to the creation of small jumps in the signal. Since some of the values we use are derived from finite differences of the position information, such small jumps can potentially cause large jumps in derivative data. In order to reduce the effect of noise and mislabelling, we filter the position values. Following [54], we smoothed the paths with a symmetric exponential moving filter. Let $(x_{k,meas.}, y_{k,meas.})$ be the originally measured position in the NGSIM datasets. Then, the smoothed values are defined by,

$$x_{k,smooth} = \frac{1}{\sum_{i=k-D}^{k+D} e^{-|k-i|/\Delta}} \sum_{i=k-D}^{k+D} x_{k,meas.} e^{-|k-i|/\Delta}$$

$$y_{k,smooth} = \frac{1}{\sum_{i=k-D}^{k+D} e^{-|k-i|/\Delta}} \sum_{i=k-D}^{k+D} y_{k,meas.} e^{-|k-i|/\Delta}$$

where $\Delta := \frac{T}{\delta t}$ is the smoothing width. We used $T = 0.5s$ and the sampling rate for the NGSIM datasets is $\delta t = 0.1s$.

5.4.2 Extraction of Expert Lane Change Trajectories

In order to build a dataset of lane change trajectories, we find every instance in the NGSIM datasets with a lane change. In the NGSIM datastets, the location of each car at each time step is assigned to a lane label. Figure 5.3 illustrates the locations of every point assigned to each lane label for the two highways.

To find trajectories that contain a lane change, for each vehicle in the NGSIM datasets we find the time steps, t_{lc}, where the lane label changes. Then, we extract the trajectory of the car on the interval $t \in [t_{lc} - \Delta t, t_{lc} + \Delta t]$. A typical lane change maneuver takes

$5 - 6s$ on the NGSIM dataset [54]. We pad the trajectory with one second of lane following before and after the lane change, thus we choose $\Delta t = 4s$. Since the sampling rate for the NGSIM datasets $\delta t = 0.1$, each lane change trajectory contains a total of 79 time steps. We use the initial pose of the ego-car as the initial condition for the ego-car, $\mathbf{x}_0 = [x_0, y_0, \psi_0]^\top$. With this formulation the length of each trajectory becomes 78 time steps.

Since we are only interested in modelling normal lane changes, we exclude every trajectory with more than one lane change. Specifically if a lane change occurs within $6s$ of another lane change, we discard both lane changes from our dataset.

5.4.3 Calculation of Ego-vehicle Dynamics

We fit the trajectory data provided in the NGSIM datasets to the ego-vehicle dynamics model described in Section 5.2.1. The NGSIM dataset provides position at each timestep, lane ID, and speed information for each vehicle. However, the provided velocity data is unfiltered and noisy [54]. Furthermore we found that the velocity values were not dynamically consistent with the location values. We only use the smoothed location values that we found in Section 5.4.1, (x_k, y_k) at each timestep k. We calculated the speed of the ego-car using finite differences,

$$v_k = \frac{\sqrt{(x_{k+1} - x_k)^2 + (y_{k+1} - y_k)^2}}{\delta t}.$$

The dynamics model also requires the heading values for the state of the ego-car and angular velocity for the control actions of the ego-car. We used the $atan2(\cdot, \cdot)$ function to obtain heading values that lie between $\psi_k \in [-\pi, \pi]$,

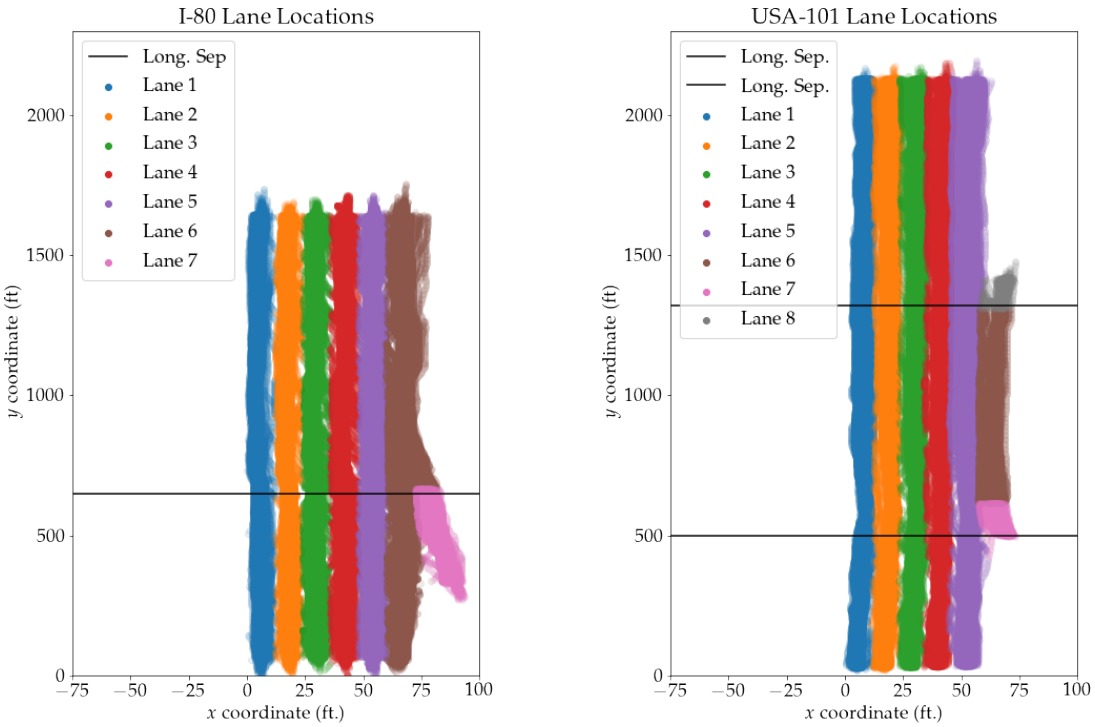
$$\psi_k = atan2(y_{k+1} - y_k, x_{k+1} - x_k).$$

Then, we used finite differences to calculate the angular velocity,

$$\omega_k = \frac{\psi_{k+1} - \psi_k}{\delta t}.$$

5.4.4 Characterization of Lane Change Environment

For the lane change environment, as described in Section 5.2.2, we require characterization of the centre-line of the current lane (CL) and the target lane (TL) characterized as two points, $\mathrm{p}_1^{CL} = [x_1^{CL}, y_1^{CL}]^\top$, $\mathrm{p}_2^{CL} = [x_2^{CL}, y_2^{TL}]^\top$ and $\mathrm{p}_1^{TL} = [x_1^{TL}, y_1^{TL}]^\top$,



(a) I-80 Lane Geometry

(b) USA-101 Lane Geometry

Figure 5.3: The distribition of lane labels in the NGSIM I-80 (a) and USA-101 (b) datasets. The ranges of the axes are different in order to highlight the distribition of lane labels along the x-axis. Lanes 1-6 are thru-lanes and Lanes 7 and 8 are on and off ramps, respectively. Traffic flows in the direction of increasing y coordinate. The black lines mark the longitudinal separation defined by us.

$\mathrm{p}_2^{TL} = [x_2^{TL}, y_2^{TL}]^\top$, respectively. The NGSIM dataset does not provide geometric information about lane boundaries or a lane's centre-line, however the location of every car at every time step in the entire dataset is assigned to a lane label. For each maneuver we locally approximate the target lane and current lane as a line on the road, assuming a flat plane. We perform linear least squares regression on the coordinates of all recorded positions assigned to the target lane accross the entire dataset within the vicinity of the lane change of interest to obtain a line characterizing the centre-line for each lane. We then store each line as two points.

We also require the trajectories of adjacent vehicles during the lane change maneuver. The trajectories include the positions of the vehicles, $\mathbf{x}_{s,i,k} = [x_{s,i,k}, y_{s,i,k}]^\top$, as well as their velocities, $v_{s,i,k}$, where $i = 1 \ldots 4$ is the index for each adjacent car and $k = 1 \ldots K$ is the index for the time step with $K = 78$. For each vehicle position, the NGSIM dataset includes the identifier of the vehicles preceding and following the ego-vehicle

Table 5.1: Summary of the longitudinal separation critera for trajectory datasets

Geometric Separation	**Description**	**Numerical Criteria (I-80)**	**Numerical Criteria (USA-101)**
long0	before on-ramp	$y_k \in [0, 650)$	$y_k \in [0, 500)$
long1	between on-ramp and off-ramp	$y_k \in [650, 2500)$	$y_k \in [500, 1320)$
long2	after off-ramp	*not applicable*	$y_k \in [1321, 2500)$

Table 5.2: Summary of the lateral separation critera for trajectory datasets

Geometric Separation	**Description**
lat5	lane change originates in lane 5
lat4	lane change originates in lane 4
lat3	lane change originates in lane 3
lat2	lane change originates in lane 2

in its lane. For each lane change trajectory, we extract the smoothed positions of the vehicles preceding and following the ego-vehicle, before and after the lane change. Since we are only interested in modelling normal lane changes, we only include trajectories with exactly four adjacent vehicles. We also discard any lane change trajectory if the position of any adjacent vehicles is unavailable for any frame in the trajectory. For the velocity, since we are only interested in the average velocity over the trajectory, we use the original velocity values provided by the dataset.

5.4.5 Dataset Divisions

In order to gain more insight on the driving behaviour in different parts of the highway, we separate the dataset based on the geometry of the highways and the time period when the data was recorded. Each lane change trajectory dataset that we generate contains trajectores from one highway, one geometric separation, and one traffic time period.

The NGSIM datasets are originally divided into two separate highways, I-80 and USA-101, and three 15-minute time periods, which we call $t0, t1, t2$, corresponding to increasing congestion levels. We also define an additional traffic time period, t_a which combines trajectories from all three congestion levels. Tables 5.1 and 5.2 summarize the geometric separation criteria we used. Each of our lane change trajectory datasets is either divided based on longitudinal criteria or lateral criteria, but not both. As a result some trajectories may be included in multiple datasets. Figure 5.3 illustrates the longitudinal and lateral separations with respect to the geometry of the highway.

5.4.6 Trajectory Initial Position Normalization

For each lane change trajectory, we normalize all position values with respect to the initial position of the ego-vehicle, $[x_0, y_0]^\top$, such that every trajectory begins at $\mathbf{x}_0 = [0, 0, \psi_0]^\top$. The normalized position include the positions of the ego-vehicle, the positions of the adjacent vehicles, and the points characterizing the current lane and target lane.

5.5 Implementation Details

5.5.1 Forward Algorithm Implementation Details

We formulate the forward optimal control problem as a constrained trajectory optimization problem starting from an initial state, $\mathbf{x}_0$, that is known a priori. Recall the lifted vector notation for describing a trajectory, $\mathbf{x} = [\mathbf{x}_1^\top, \ldots, \mathbf{x}_K^\top]^\top$ and $\mathbf{u} = [\mathbf{u}_0^\top, \ldots, \mathbf{u}_{K-1}^\top]^\top$, we introduced in Section 4.2. Using the lifted vector notation, the optimization problem can be written as,

$$\begin{aligned} &\max_{\mathbf{x},\mathbf{u}} \mathcal{R}(\mathbf{x}, \mathbf{u}) \\ &\text{subject to } \mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k),\ k = 0 \ldots K-1 \end{aligned} \tag{5.20}$$

To find the optimal trajectory, we need to solve for $\mathbf{x}, \mathbf{u}$ such that the reward function, $\mathcal{R}$ is maximized, subject to the dynamics, $f(\mathbf{x}_k, \mathbf{u}_k)$, of the ego-vehicle. Both the objective function and the dynamics are nonlinear.

Our implementation uses Python (v3.8.5) with Numpy (v1.19.4) [20] for storing vectors and matrices. We use the SciPy Package (v1.4.1) [56] and the function `scipy.minimize.optimize()` to solve (5.20). We are required to provide an initial guess for the vectors $\mathbf{x}, \mathbf{u}$ before optimizing. When optimizing a reward function to compare the resulting trajectory with an expert trajectory, $\mathbf{x}_e, \mathbf{u}_e$, we use the expert trajectory as the initial guess. When generating synthetic data, we set the initial guess for the inputs as the ego-vehicle proceeding straight in the direction initial lane at the desired velocity. We use the default hyperparameters for `scipy.minimize.optimize()`, except for the maximum iterations, which we set to `maxiter=1e4`.

5.5.2 IRL Algorithm Implementation Details

Our implementation uses Python (v3.8.5) with Numpy (v1.19.4) [20] for storing vectors and matrices. We use the NLOpt package (v2.6.2) [26] implementation of the Sequential

Least-Squares Quadratic Program (SLSQP) algorithm [30] to optimize the IRL objective function. Below, we describe the IRL objective function, then we make note of two numerical tricks that are used in the implementation.

IRL Objective and Optimization

Just as with the IRL implementation for the LQR system in Section 4.4.1, we use the IRL technique described in [33] to recover the reward parameters. The unicycle model of a car that we described in Section 5.2.1 has deterministic nonlinear dynamics and infinite state and action spaces. In order to estimate the likelihood of a trajectory, we need to calculate the matrices in Equation (3.35). Recall the lifted vector notation for a trajectory, $\mathbf{x} = [\mathbf{x}_1^\top, \dots, \mathbf{x}_K^\top]^\top$, and $\mathbf{u} = [\mathbf{u}_0^\top, \dots, \mathbf{u}_{K-1}^\top]^\top$. Due to the deterministic dynamics of the system, any trajectory can be fully described by an initial condition, $\mathbf{x}_0$ and a set of actions, $\mathbf{u} \in \mathbb{R}^{Km}$. Therefore, to use the Laplace approximate (Equation (3.31)) we need to find the gradient and Hessian matrices, $\mathbf{g}$ and $\mathbf{H}$, of the reward function with respect to the input, $\mathbf{u}$. To simplify notation, let $\mathbf{J} = \frac{\partial \mathbf{x}^\top}{\partial \mathbf{u}}$, $\tilde{\mathbf{g}} := \frac{\partial \mathcal{R}}{\partial \mathbf{u}}$, $\hat{\mathbf{g}} := \frac{\partial \mathcal{R}}{\partial \mathbf{x}}$, $\tilde{\mathbf{H}} := \frac{\partial^2 \mathcal{R}}{\partial \mathbf{u}^2}$, $\hat{\mathbf{H}} := \frac{\partial^2 \mathcal{R}}{\partial \mathbf{x}^2}$, and $\check{\mathbf{H}} := \frac{\partial^2 \mathbf{x}}{\partial \mathbf{u}^2}$. With these definitions, the matrices in the log likelihood equation for IRL (Equations (3.35)) can be summarized as,

$$\mathbf{g} = \tilde{\mathbf{g}} + \mathbf{J}\hat{\mathbf{g}} \tag{5.21a}$$

$$\mathbf{H} = \tilde{\mathbf{H}} + \mathbf{J}\hat{\mathbf{H}}\mathbf{J}^\top + \check{\mathbf{H}} \cdot \hat{\mathbf{g}} \tag{5.21b}$$

where $\cdot$ represents a tensor contraction between the three-dimensional tensor $\check{\mathbf{H}}$ and the vector $\hat{\mathbf{g}}$. We set $\check{\mathbf{H}} = 0$, which is equivalent to linearizing the dynamics [33]. We calculate the values of the vectors and matrices in Equations (5.21). First, the matrix $\mathbf{J} \in \mathbb{R}^{Km \times Kn}$ has the following upper block-diagonal form,

$$\mathbf{J} = \begin{bmatrix} \frac{\partial \mathbf{x}_1^\top}{\partial \mathbf{u}_0} & \cdots & \frac{\partial \mathbf{x}_K^\top}{\partial \mathbf{u}_0} \\ \vdots & & \vdots \\ \frac{\partial \mathbf{x}_1^\top}{\partial \mathbf{u}_{K-1}} & \cdots & \frac{\partial \mathbf{x}_K^\top}{\partial \mathbf{u}_{K-1}} \end{bmatrix}. \tag{5.22}$$

We use the chain rule to calculate $\frac{\partial \mathbf{x}_i^\top}{\partial \mathbf{u}_j} = \frac{\partial \mathbf{x}_i^\top}{\partial \mathbf{x}_{i-1}} \frac{\partial \mathbf{x}_{i-1}^\top}{\partial \mathbf{x}_{i-2}} \dots \frac{\partial \mathbf{x}_{j+1}^\top}{\partial \mathbf{u}_j}, \forall i > j \geq 0$. We also have that $\frac{\partial \mathbf{x}_i^\top}{\partial \mathbf{u}_j} = 0, \forall j \geq i$. Therefore, the matrix $\mathbf{J}$ is upper block-diagonal.

Second, the vectors $\tilde{\mathbf{g}} \in \mathbb{R}^{Km}$ and $\hat{\mathbf{g}} \in \mathbb{R}^{Kn}$ have the following form,

$$\tilde{\mathbf{g}} = \begin{bmatrix} \frac{\partial \mathcal{R}}{\partial \mathbf{u}_0} \\ \hline \vdots \\ \hline \frac{\partial \mathcal{R}}{\partial \mathbf{u}_{K-1}} \end{bmatrix} \tag{5.23}$$

$$\hat{\mathbf{g}} = \begin{bmatrix} \frac{\partial \mathcal{R}}{\partial \mathbf{x}_1} \\ \hline \vdots \\ \hline \frac{\partial \mathcal{R}}{\partial \mathbf{x}_K} \end{bmatrix} \tag{5.24}$$

Finally, the matrices $\tilde{\mathbf{H}} \in \mathbb{R}^{Km \times Km}$ and $\hat{\mathbf{H}} \in \mathbb{R}^{Kn \times Kn}$ have the following symmetric forms,

$$\tilde{\mathbf{H}} = \begin{bmatrix} \frac{\partial^2 \mathcal{R}}{\partial \mathbf{u}_0^2} & \frac{\partial^2 \mathcal{R}}{\partial \mathbf{u}_0 \partial \mathbf{u}_1} & \cdots & \frac{\partial^2 \mathcal{R}}{\partial \mathbf{u}_0 \partial \mathbf{u}_{K-1}} \\ \frac{\partial^2 \mathcal{R}}{\partial \mathbf{u}_1 \partial \mathbf{u}_0} & \frac{\partial^2 \mathcal{R}}{\partial \mathbf{u}_1^2} & \cdots & \frac{\partial^2 \mathcal{R}}{\partial \mathbf{u}_1 \partial \mathbf{u}_{K-1}} \\ \vdots & & \ddots & \vdots \\ \frac{\partial^2 \mathcal{R}}{\partial \mathbf{u}_{K-1} \partial \mathbf{u}_0} & \cdots & \cdots & \frac{\partial^2 \mathcal{R}}{\partial \mathbf{u}_{K-1}^2} \end{bmatrix} \tag{5.25}$$

$$\hat{\mathbf{H}} = \begin{bmatrix} \frac{\partial^2 \mathcal{R}}{\partial \mathbf{x}_1^2} & \frac{\partial^2 \mathcal{R}}{\partial \mathbf{x}_1 \partial \mathbf{x}_2} & \cdots & \frac{\partial^2 \mathcal{R}}{\partial \mathbf{x}_1 \partial \mathbf{x}_K} \\ \frac{\partial^2 \mathcal{R}}{\partial \mathbf{x}_2 \partial \mathbf{x}_1} & \frac{\partial^2 \mathcal{R}}{\partial \mathbf{x}_2^2} & \cdots & \frac{\partial^2 \mathcal{R}}{\partial \mathbf{x}_2 \partial \mathbf{x}_K} \\ \vdots & & \ddots & \vdots \\ \frac{\partial^2 \mathcal{R}}{\partial \mathbf{x}_K \partial \mathbf{x}_1} & \cdots & \cdots & \frac{\partial^2 \mathcal{R}}{\partial \mathbf{x}_K^2} \end{bmatrix}. \tag{5.26}$$

The individual elements of the vectors and matrices were calculated symbolically using MATLAB and hardcoded in vectorized form in the Python code.

We use the symbolically calculated vectors and matrices to define the negative of the log likelihood (Equation (3.35)), and minimize the equation in order to estimate the reward parameters. We set the gradient of the negative log likelihood with respect to the parameters $\boldsymbol{\theta}$ using Equation (3.36).

Feature Normalization

The order of magnitude of the different features vary for any trajectory. We observed that this variation resulted in floating-point overflows in the $\mathbf{H}$ matrix. Therefore, we use min-max normalization to map all features to lie between $[0, 1]$ in the dataset. We redefine each feature, ϕ_i, as

$$\hat{\phi}_i := \frac{\phi_i - \phi_{i,min}}{\phi_{i,max} - \phi_{i,min}}. \tag{5.27}$$

where $\phi_{i,min}$ and $\phi_{i,max}$ are constants corresponding to the minimum and maximum values of ϕ_i found in any timestep in any of the trajectories in the dataset, respectively.

Numerical Stability Normalization Feature

The log-likelihood, (Equation (3.35)), includes a log determinant term, $\log|-\mathbf{H}|$. Thus we require $\mathbf{H}$ to remain negative-definite for all values of the parameters that are traversed during the optimization. Following [33], we introduce a dummy regularization feature, ϕ_r, to our reward function in order to ensure numerical stability. The gradient of the feature is zero, $\mathbf{g}_r = 0$, and its Hessian is the identity matrix, $\mathbf{H}_r = -\mathbf{I} \in \mathbb{R}^{Km \times Km}$. Before starting the optimization with arbitrary initial parameters, $\boldsymbol{\theta}_0$, we set the parameter for the regularization feature, $\theta_r = 1E - 10$. Then we loop over every trajectory in the dataset and calculate $\mathbf{H}$. If we reach a trajectory with $\mathbf{H}$ that is not negative-definite, then we double θ_r and restart looping over the trajectories with the new θ_r value. We repeat this process until we reach a θ_r value such that $\mathbf{H}$ is negative-definite for every trajectory in the dataset. In our optimization we set $\theta_r = 0$ as a constraint. With this formulation, we observed that the optimization is numerically stable with our datasets, and gently arrives at optima with $\theta_r = 0$, regardless of the initial condition $\boldsymbol{\theta}_r$.

Scaling of Parameters

Following the justification Section 4.4.2, we multiply the parameters by a constant reward coefficient. Taking into account the need for a sufficiently large reward coefficient and the floating point limitations of our computer, we found a good coefficient that works for a variety of synthetic trajectories is $r_1 = 10^6$. We present the details about selecting the value in the next section (Section 5.6).

Outlier Rejection

Sources of errors in the construction of vehicle trajectories in the NGSIM dataset include mislabelling and noise [54, 40, 41]. The two-dimensional position and lane classification of vehicles are labelled using bounding boxes on images from static video cameras [54]. In some frames, the bounding boxes are slightly translated with respect to the actual position of the vehicle in the image. In a trajectory that transitions from a well-labelled frame to a misaligned frame, there is a jump in the position signals of the vehicle. Additionally, random noise in the labelling position information contributes to the creation of small jumps in the signal. To construct the lane change trajectory datasets, we calculate heading and linear velocity from finite differences of the position and we calculate the angular velocity based on finite differences of the heading (second order finite difference of the position). Therefore, the position jumps translate into large jumps in the other signals. The reward features that we use rely on the velocity and angular veclity as well

as the positions of the ego-vehicle, and four adjacent vehicles. Trajectories that include these jumps will generate inaccurate feature values and can result in the failure of the IRL algorithm due to numerical overflows. If the algorithm succeeds despite the jumps, they can result in an inaccurate learned model.

We use outlier values of the reward features as a heuristic to removing trajectories that contain the position jumps. For each feature, ϕ_i, we calculate the median, $\phi_{i,med}$ and standard deviation, σ_i over every frame in every trajectory in the dataset. We then reject any trajectory that contains a timestep with a feature value greater than $\phi_{i,med}+r_{reject}\sigma_i$, where $r_{reject} \in \mathbb{R}$ is a hyperparameter.

5.6 Algorithm Verification with Synthetic Data

In order to verify the features in our reward function and our IRL algorithm, we generate datasets consisting of synthetic lane change trajectories. We demonstrate that given appropriate parameters, a reward function consisting of our features can generate a variety of qualitatively good lane change trajectories. Furthermore, we use the IRL algorithm to learn reward parameters from the synthetic datasets that we use as a ground-truth. We generate new trajectories using the learned parameters. We demonstrate that the learned parameters produce nearly identical trajectories as the parameters originally used to genereate the datasets.

5.6.1 Generating Synthetic Trajectorires

To create a synthetic dataset, we first generate the lane change surrounding environment, which consists of the current lane, the target lane, and the adjacent vehicles. We extract six environments from the NGSIM datasets by taking the average over every trajectory in each highway and traffic congestion level for each element in the environment. We choose a set of parameters for our reward function. Given the reward parameters, we use the forward algorithm described in section 5.5.1 to generate each of the six synthetic trajectories in the dataset. The initial condition for the forward optimization (Equation (5.20)), $(\mathbf{x}_{init}, \mathbf{u}_{init})$, are set to the ego-car driving straight in the current-lane at the mean velocity of the highway, v_d.

Table 5.3 lists the parameter values, $\tilde{\boldsymbol{\theta}}_{synth} = [\tilde{\theta}_d, \tilde{\theta}_v, \tilde{\theta}_a, \tilde{\theta}_s,]^\top$, we use. The chosen values were found to produce qualitatively good results. Since the features cannot be normalized before generating the synthetic dataset, the elements of $\tilde{\boldsymbol{\theta}}_{synth}$ vary by orders of magnitude. Figure 5.4 illustrates a snapshot of Trajectory 3 from Dataset 57.

Table 5.3: Summary of reward parameter values for synthetically generated data

Dataset ID	$\tilde{\theta}_d$	$\tilde{\theta}_v$	$\tilde{\theta}_a$	$\tilde{\theta}_s$
56	1	5	50	0.01
57	1	0.1	50	0.001
58	1	1	50	0.01
59	1	10	200	0.05
60	1	5	200	0.01

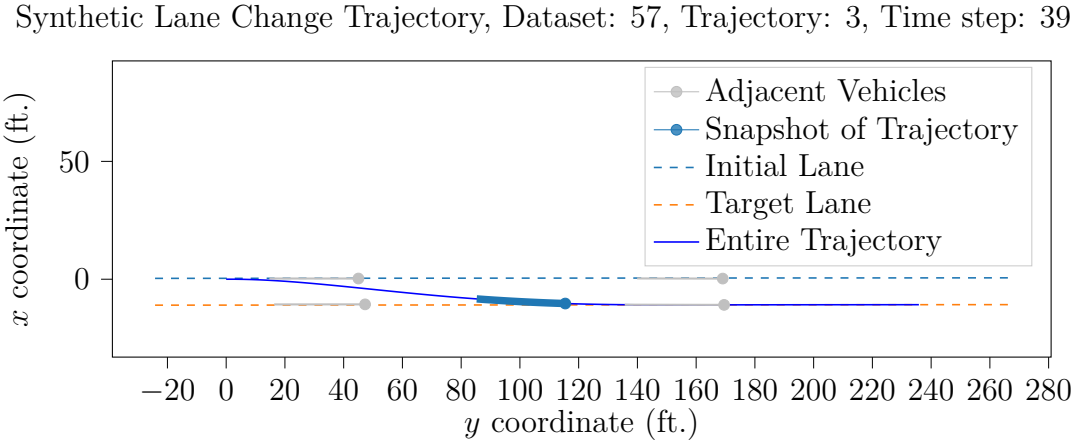


Figure 5.4: Snapshot of a synthetically generated lane change trajectory. Scatter points illustrate the position of the ego-vehicle (blue) and adjacent vehicles (grey) at time step 39. From each scatter point, we illustrate a trail of the past 10 time steps.

5.6.2 IRL Model Training

We define the loss function by taking the negative of the likelihood function, as described in Section 5.5.2. Starting from an initial reward parametrization, $\boldsymbol{\theta}_0$, we use the synthetic trajectory datasets in the IRL implementation described in Section 5.5.2 to recover optimal parameters, $\boldsymbol{\theta}_f$. We performed a hyperparameter sweep over $\boldsymbol{\theta}_0$. For each element, we used the values, $\theta_i = 0.01, 1, 100$, where $i = v, a, s$. With our five datasets and 27 initial parameter values, we learn 135 IRL trials. Figure 5.5 illustrates the learning curve for one of the initial conditions.

5.6.3 Verification Results

To evaluate the reward function learned using the IRL algorithm, we will generate trajectories using the initial and learned parameters, and compare with the synthetic expert trajectories. More concretely, given the lane change environment for each trajectory in the datasets, we use the forward optimal control implementation described in section 5.5.1 to generate trajectories that are optimal with respect to a reward function parametrized by $\boldsymbol{\theta}_0$ and $\boldsymbol{\theta}_f$. We then compare these trajectories to the expert trajectories in the dataset.

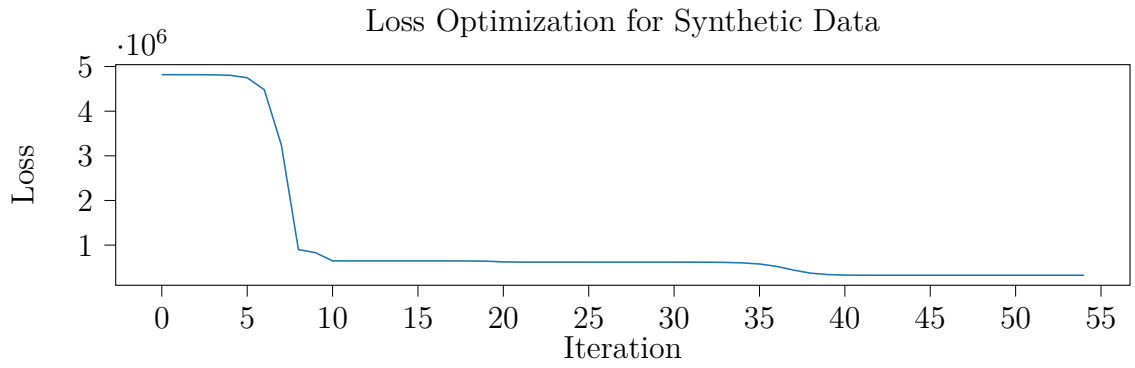


Figure 5.5: Loss optimization for IRL with synthetic dataset 57 and initial condition $\boldsymbol{\theta}_0 = [1, 1, 1, 1]^\top$.

Table 5.4: IRL model performance on 127 trials using all five synthetic datasets

Measure (ft.)	**Minimum**	**Mean**	**Maximum**
$\overline{MEE}_{e0}$	1.061	6.226	21.738
$\overline{MEE}_{ef}$	0.037	0.053	0.081
Improvement	1.024	6.173	21.673

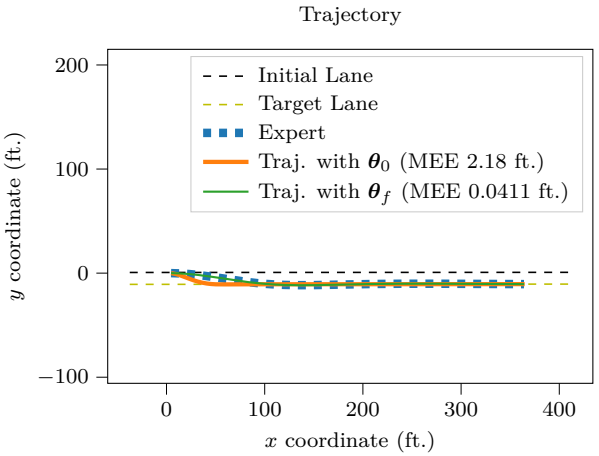
We illustrate an example trajectory for dataset 56 and trajectory 0 in Figure 5.6. The trajectory generated using $\boldsymbol{\theta}_0$ behaves significantly differently than the expert, whereas the trajectory generated by $\boldsymbol{\theta}_f$ is nearly identical.

We use Mean Euclidean Error (MEE) to measure the closeness of each of the generated trajectories with the corresponding synthetic expert trajectory. The MEE between the trajectories generated using the learned parameters and expert parameters (labelled with $\cdot_f$ and $\cdot_0$, respectively) and the synthetic expert trajectory (labelled with $\cdot_e$) is defined as the mean Euclidean distances between the two trajectories at each time step, $k = 1, \ldots, K$.

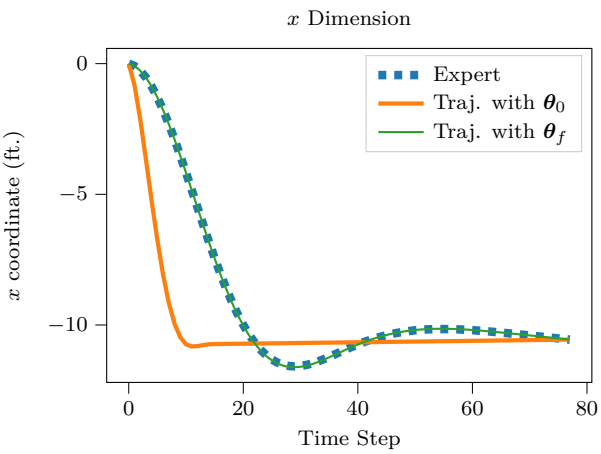
$$\begin{aligned} MEE_{0f} &= \frac{1}{K}\sum_{k=1}^{K}\sqrt{(x_{k,e}-x_{k,0})^2+(y_{k,e}-y_{k,0})} \\ MEE_{ef} &= \frac{1}{K}\sum_{k=1}^{K}\sqrt{(x_{k,e}-x_{k,f})^2+(y_{k,e}-y_{k,f})} \end{aligned} \tag{5.28}$$

For each IRL trial, we calculate the initial mean MEE, $\overline{MEE}_{e0} = \frac{1}{N}\sum_{n=1}^{N} MEE_{e0}$, and mean final MEE, $\overline{MEE}_{ef} = \frac{1}{N}\sum_{n=1}^{N} MEE_{ef}$, over the $N = 6$ trajectories in each dataset. We present a summary of the results from the 127 trials in Table 5.4.

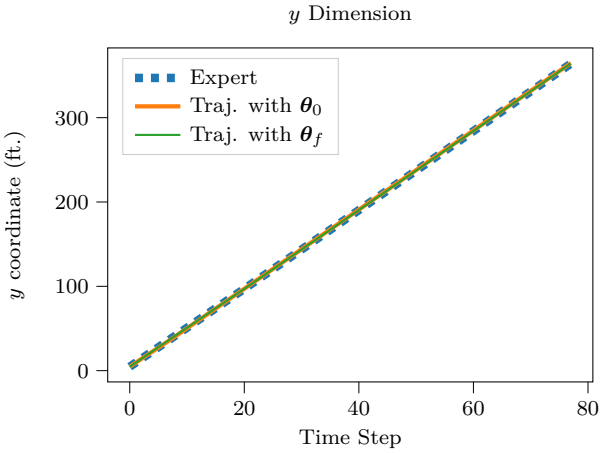
All trajectories optimal with respect to $\boldsymbol{\theta}_f$ are closer to the synthetic trajectories than the trajectories optimal with respect to $\boldsymbol{\theta}_0$. Therefore, our IRL algorithm successfully learns a parametrization of the reward function.



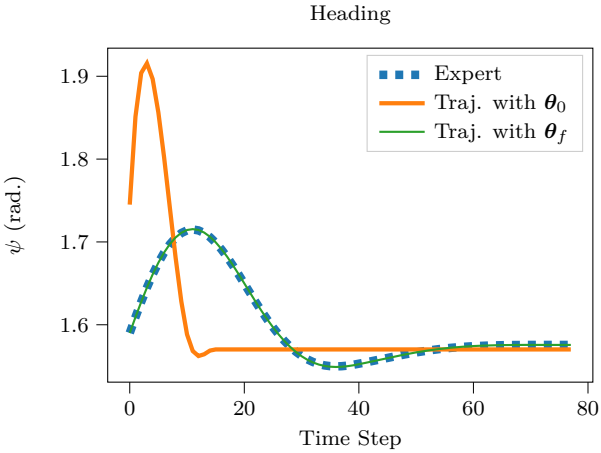
(a) Trajectory



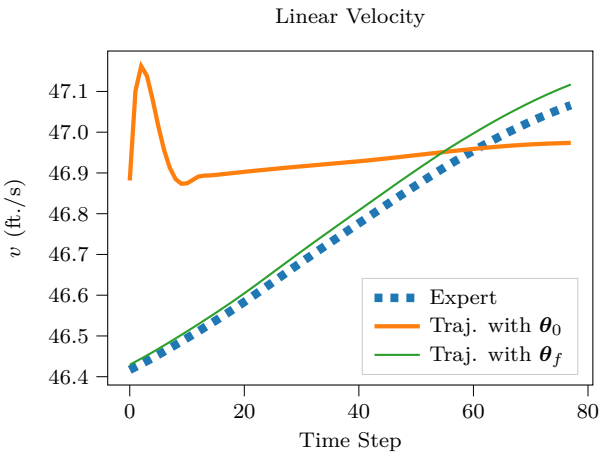
(b) First Dimension



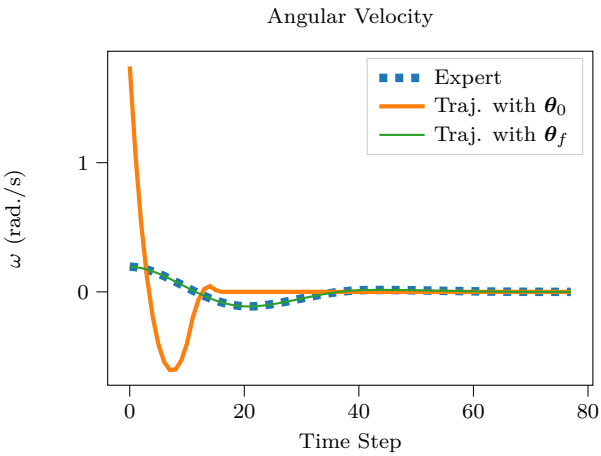
(c) Second Dimension



(d) Heading



(e) Linear Velocity



(f) Angular Velocity

Figure 5.6: Comparison of the states (5.6a to 5.6d) and actions (5.6e, 5.6f) of the synthetically generated expert lane change trajectory (blue) with trajectories generated by optimizing rewards parametrized by initial guess of parameters, $\boldsymbol{\theta}_0$ (orange), and parameters learned using IRL, $\boldsymbol{\theta}_f$ (green).

5.6.4 Tuning of Scaling Parameter

Using the synthetic trajectories we conducted an experiment to find an appropriate value of a constant coefficient for multiplying with the reward. Table 5.5 summarizes the parameter values and $\overline{MEE}_{ef}$. We observe that larger coefficient values correspond to model parameters that result in lower $\overline{MEE}_{ef}$. However, larger parameter values also risk numerical instability during the inverse learning and forward model calculation process.

Table 5.5: Summary of IRL parameter values and model performance with different reward coefficients for synthetic dataset 60

Reward Coef. Value	$\theta_{f,d}$	$\theta_{f,v}$	$\theta_{f,a}$	$\theta_{f,s}$	$\overline{MEE}_{ef}$
10^{-3}	1	61733.2839	20846.2645	395367.524	1.74658071
10^{-2}	1	6172.44888	2084.93245	39531.12	1.74582973
10^{-1}	1	616.372973	208.801559	3947.52966	1.73600614
10^{0}	1	60.8414295	21.2115526	389.671796	1.64238031
10^{1}	1	5.96743042	2.65820004	38.3600134	0.95471478
10^{2}	1	1.64997207	1.16035927	10.9252732	0.12925804
10^{3}	1	1.33506801	1.04798686	8.94777948	0.04477793
10^{4}	1	1.30489093	1.03722661	8.75856694	0.03823539
10^{5}	1	1.3018855	1.03615529	8.73972518	0.03793444
10^{8}	1	1.30155203	1.03603642	8.73763463	0.03800752
10^{9}	1	1.30155173	1.03603631	8.73763274	0.03783267
10^{10}	1	1.3015517	1.0360363	8.73763256	0.03800511
10^{11}	1	1.3015517	1.0360363	8.73763254	0.03790087

5.7 IRL Model Training

In order to compare the baseline model with the model that incorporates unpredictability, we performed IRL and forward optimal control using the same dataset and the same hyperparameters for the pair of models, as illustrated in the block diagram in Figure 5.1.

In the IRL optimization, for each pair of models (baseline vs. with unpredictability), we set the following hyperparameter values:

1. Initial parameter guess $\boldsymbol{\theta}_0 = [1, 1, 1, 1]^\top$ for the baseline model and $\boldsymbol{\theta}_0 = [1, 1, 1, 1, 1]^\top$ for the model that incorporates unpredictability. We performed a hyparparameter sweep over the IRL training step for the synthetic data where we altered the value for each initial parameter between $\theta_{i,0} = 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}$ where $i = v, a, s, z$. We found that training runs with $\boldsymbol{\theta}_0$ where the elements, $\theta_{i,0}$, had orders of magnitude of difference, resulted in numerical instability. For all cases

that were numerically stable, the resulting final parameters were nearly identical. Thus, we select $\boldsymbol{\theta}_0 = [1, 1, 1, 1]^\top$, which was numerically stable in all cases.

2. Relative tolerance for optimizer $ftol = 10^{-18}$. We performed a sweep of this value on the synthetic dataset. We found that lower tolerances did not improve the model learned using IRL.

3. Initial value for numerical stability normalization feature, $\theta_{r,0} = 10^{-10}$.

4. Tolerance for numerical stability normalization feature constraint, $tol_{\theta_r} = 10^{-12}$.

5. Reward coefficient $R_{coeff} = 1$. We performed a sweep over the IRL training step with the following values of $R_{coeff} = 10^{-2}, 10^{0}, 10^{2}, 10^{4}, 10^{6}$. We found that with the real data, $R_{coeff} = 10^{0}$ resulted in a similar order of magnitude for the final parameters as higher R_{coeff} did for synthetic data. Furthermore, we found that parameters learned using $R_{coeff} = 10^{0}$ consistently resulted in the lowest MEE values.

6. Minimum distance for *distance to adjacent car* (Equation (5.15)) and *unpredictability* (Equation (5.18)) features, $d_{min} = 15$.

7. Headway time for *distance to adjacent car* (Equation (5.15)) and *unpredictability* (Equation (5.18)) features, $t_{headway} = 1$.

8. Outlier rejection radius $r_{reject} = 3$. We balanced the tradeoff between rejecting a sufficient amount of outliers due to measurement error for numerical stability with the diversity of the trajectories that remained in the dataset.

9. Unpredictability measure (Equation 5.16) time, $t_n = 0.2s$. We evaluated model performance improvement for the values, $0.2s$, $0.4s$, $0.6s$, $0.8s$, $1.0s$. We observed improvement across all values. We report detailed results for $t_n = 0.2s$, which resulted in the highest average improvement.

We performed IRL optimization on trajectory training datasets that consist of lane change trajectories sorted by the following criteria,

- Highway: I-80 or USA-101.
- Traffic congestion time slot: low congestion ($t0$), medium congestion ($t1$), high congestion ($t2$) or combined (ta).

- Geometry: lateral sorting by lane of origin (*lat*) or longitudinal sorting by location of lane change compared to ramps (*long*).

This sorting results in a total of 51 separated trajectory training datasets. A detailed explanation of the criteria for sorting can be found in Section 5.4.5. We also generated 3 combined training datasets, one for each highway (I-80 and USA-101) and a combined dataset with all the available trajectories. For the combined criteria, we also generated 3 corresponding held-out testing datasets. The training-testing split is consistent with the data used to train the off-the-shelf prediction model [14] that we used in the unpredictability measure described in Section 5.3.1.

5.8 Results and Discussion

We have a total of 51 separated trajectory training datasets and 3 combined trajectory training datasets. For each dataset, we conduct the experiment illustrated in Figure 5.1.

IRL: First, for each training dataset we conduct two IRL optimizations in parallel to obtain baseline reward parameters, $\boldsymbol{\theta}_{\boldsymbol{f},\boldsymbol{b}}$, and reward parameters that include unpredictability, $\boldsymbol{\theta}_{\boldsymbol{f},\boldsymbol{w}}$. We use our IRL implementation, described in Section 5.5.2 to train the pair of models for each dataset. Figure 5.7 illustrates the loss curve during the IRL training step for highway `usa101`, traffic congestion time slot `t2`, and geometry `lat5`. Because of the extra feature, the IRL model incorporating unpredictability has an extra degree of freedom compared to the baseline model. The extra degree of freedom results in a higher number of iterations before convergence. Furthermore, as a sanity check we observe that the loss for the model with unpredictability is lower, as expected due to the extra degree of freedom.

Forward: Second, for each of the N trajectories in each training dataset, using the pair of learned parameters, we conduct forward optimal control, as described in Section 5.5.1, to obtain trajectories from the baseline model, $\{\mathbf{x}_{f,b}^{(i)}, \mathbf{u}_{f,b}^{(i)}\}_{i=1}^{N}$, and from the model incorporating unpredictability, $\{\mathbf{x}_{f,w}^{(i)}, \mathbf{u}_{f,w}^{(i)}\}_{i=1}^{N}$. We then compare each trajectory generated using the forward optimal control with the respective expert trajectories to obtain the Mean Euclidean Error. We calculate the Average Mean Euclidean Distance $\overline{MEE}_{ef,b}$ and $\overline{MEE}_{ef,w}$ for each dataset. And finally, we calculate the improvement that incorporating unpredictability has on Average Mean Euclidean Distance. For the 3 combined training datasets, we repeat the forward process with the corresponding testing datasets as well.

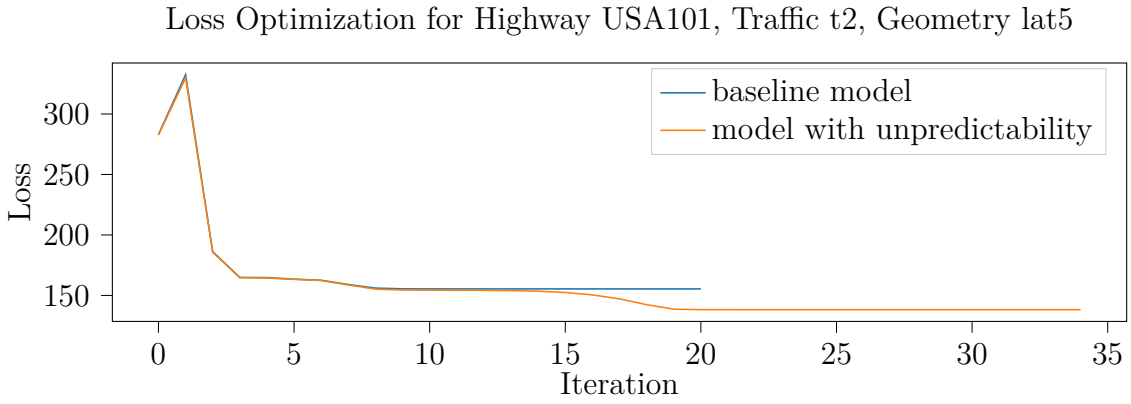


Figure 5.7: IRL loss optimization for baseline model and model incorporating unpredictability for highway `usa101`, traffic congestion time slot `t2`, and geometry `lat5`.

5.8.1 Quantitative Improvements

We report the results for the separated trajectory dataset in Tables 5.6 to 5.9. Each table summarizes results from trajectory datasets where trajectories are organized by non-overlapping sorting criteria, i.e. no two lane change trajectories are repeated in datasets within one table. In Table 5.6, we present results from trajectory datasets that have been divided by highway, traffic congestion time, and lateral geometry. Then in Table 5.7, we present results from trajectory datasets that have the same division criteria, except where trajectories across traffic congestion times are combined. Table 5.8 summarizes results from trajectory datasets that have been divided by highway, traffic congestion time and longitudinal geometry. In Table 5.9, we combine the traffic congestion times for longitudinally sorted trajectories.

Table 5.6: Seperate Traffic, Lateral

Highway	Traff.	Lat	# Traj.	$\boldsymbol{\theta}_{f,b}$	$\boldsymbol{\theta}_{f,w}$	$\overline{MEE}_{ef,b}$	$\overline{MEE}_{ef,w}$	Imp.	% Imp.
i80	t0	lat2	1	1.00, 1.74, 5.04, 0.41	1.00, 1.74, 5.04, 0.41, 0.00	9.56	9.56	0.00	0.00%
i80	t0	lat3	31	1.00, 3.69, 36.30, 1.21	1.00, 3.70, 36.32, 0.61, 3.98	11.89	10.57	1.32	11.11%
i80	t0	lat4	45	1.00, 4.78, 220.38, 1.90	1.00, 4.62, 220.42, 1.29, 27.19	10.48	10.11	0.37	3.54%
i80	t0	lat5	36	1.00, 5.31, 41.22, 1.82	1.00, 5.31, 41.22, 1.82, 0.00	8.55	8.55	0.00	0.00%
i80	t1	lat2	1	1.00, 0.70, 8.35, 0.03	1.00, 0.70, 8.35, 0.03, 0.00	35.69	35.69	0.00	0.00%
i80	t1	lat3	34	1.00, 4.96, 32.48, 3.60	1.00, 4.96, 32.48, 3.60, 0.00	8.40	8.40	0.00	0.00%
i80	t1	lat4	32	1.00, 5.19, 32.58, 3.79	1.00, 5.47, 32.62, 0.17, 70.48	8.02	6.72	1.29	16.14%
i80	t1	lat5	37	1.00, 4.11, 21.57, 4.62	1.00, 4.11, 21.57, 4.62, 0.00	9.33	9.33	0.00	0.00%
i80	t2	lat2	2	1.00, 6.04, 15.95, 10.30	1.00, 10.36, 17.13, 17.84, 301.29	6.69	7.05	-0.36	-5.41%
i80	t2	lat3	25	1.00, 4.60, 65.25, 5.32	1.00, 4.60, 65.25, 5.32, 0.00	7.46	7.46	0.00	0.00%
i80	t2	lat4	37	1.00, 5.71, 21.92, 2.32	1.00, 5.55, 22.16, 1.99, 13.93	7.29	6.92	0.38	5.18%
i80	t2	lat5	47	1.00, 5.42, 81.00, 4.90	1.00, 5.42, 81.00, 4.90, 0.00	8.57	8.57	0.00	0.00%
usa101	t0	lat2	8	1.00, 5.21, 19.82, 0.62	1.00, 5.21, 19.82, 0.62, 0.00	7.57	7.57	0.00	0.00%
usa101	t0	lat3	19	1.00, 2.76, 18.76, 0.61	1.00, 2.76, 18.76, 0.61, 0.00	12.73	12.73	0.00	0.00%
usa101	t0	lat4	21	1.00, 4.15, 11.75, 0.00	1.00, 4.15, 11.75, 0.00, 0.00	10.15	10.15	0.00	0.00%
usa101	t0	lat5	22	1.00, 4.10, 10.61, 3.04	1.00, 4.10, 10.61, 2.89, 0.45	9.06	8.94	0.13	1.41%
usa101	t1	lat2	15	1.00, 3.54, 20.76, 1.37	1.00, 3.54, 20.76, 1.37, 0.00	8.78	8.78	0.00	0.00%
usa101	t1	lat3	25	1.00, 4.29, 14.76, 0.94	1.00, 4.25, 14.77, 0.49, 8.51	9.86	9.45	0.42	4.21%
usa101	t1	lat4	30	1.00, 4.14, 20.31, 2.07	1.00, 4.14, 20.36, 1.89, 10.18	9.55	9.16	0.39	4.08%
usa101	t1	lat5	30	1.00, 3.50, 66.76, 2.65	1.00, 3.50, 66.76, 2.65, 0.00	9.67	9.67	0.00	0.00%
usa101	t2	lat2	14	1.00, 4.94, 21.16, 2.87	1.00, 5.15, 21.21, 1.96, 6.43	6.48	7.37	-0.89	-13.75%
usa101	t2	lat3	29	1.00, 4.41, 16.79, 4.82	1.00, 4.36, 16.82, 4.34, 23.56	8.38	7.72	0.66	7.93%
usa101	t2	lat4	45	1.00, 3.96, 61.90, 3.57	1.00, 3.99, 62.64, 3.01, 68.17	9.41	9.00	0.40	4.28%
usa101	t2	lat5	26	1.00, 4.08, 30.58, 1.06	1.00, 3.65, 30.93, 1.40, 338.12	9.93	7.94	1.99	20.03%

Table 5.7: Combined Traffic, Lateral

Highway	Traff.	Lat	# Traj.	$\boldsymbol{\theta}_{f,b}$	$\boldsymbol{\theta}_{f,w}$	$\overline{MEE}_{ef,b}$	$\overline{MEE}_{ef,w}$	Imp.	% Imp.
i80	ta	lat2	4	1.00, 2.29, 14.75, 1.49	1.00, 2.15, 15.06, 0.67, 60.98	15.11	16.90	-1.79	-11.88%
i80	ta	lat3	90	1.00, 4.92, 61.32, 3.69	1.00, 4.92, 61.32, 3.69, 0.00	9.42	9.42	0.00	0.00%
i80	ta	lat4	114	1.00, 5.48, 248.41, 3.51	1.00, 5.27, 249.86, 2.51, 51.65	8.79	8.24	0.55	6.21%
i80	ta	lat5	120	1.00, 5.31, 110.67, 3.82	1.00, 5.31, 110.67, 3.82, 0.00	8.93	8.93	0.00	0.00%
usa101	ta	lat2	37	1.00, 4.96, 23.22, 1.92	1.00, 5.00, 23.25, 1.24, 5.67	7.66	7.45	0.21	2.77%
usa101	ta	lat3	73	1.00, 4.09, 18.14, 1.85	1.00, 4.06, 18.15, 1.36, 6.66	10.25	9.71	0.54	5.25%
usa101	ta	lat4	96	1.00, 4.30, 64.43, 1.73	1.00, 4.27, 64.51, 1.44, 9.78	10.14	9.78	0.35	3.47%
usa101	ta	lat5	78	1.00, 3.99, 85.99, 2.48	1.00, 3.99, 85.99, 2.48, 0.00	9.50	9.50	0.00	0.00%

Table 5.8: Separate Traffic, Longitudinal

Highway	Traff.	Lat	# Traj.	$\boldsymbol{\theta}_{f,b}$	$\boldsymbol{\theta}_{f,w}$	$\overline{MEE}_{ef,b}$	$\overline{MEE}_{ef,w}$	Imp.	% Imp.
i80	t0	long0	24	1.00, 5.53, 22.34, 2.39	1.00, 5.53, 22.34, 2.39, 0.00	10.35	10.35	0.00	0.00%
i80	t0	long1	102	1.00, 4.42, 296.18, 2.18	1.00, 4.36, 296.71, 1.57, 14.77	10.10	9.35	0.75	7.39%
i80	t1	long0	37	1.00, 4.07, 23.09, 2.16	1.00, 4.04, 23.10, 0.95, 24.43	8.93	8.05	0.87	9.79%
i80	t1	long1	86	1.00, 5.31, 49.99, 5.71	1.00, 5.31, 49.99, 5.71, 0.00	8.74	8.74	0.00	0.00%
i80	t2	long0	54	1.00, 6.38, 64.84, 7.86	1.00, 6.44, 65.04, 7.96, 8.76	6.48	6.08	0.40	6.16%
i80	t2	long1	94	1.00, 4.72, 99.82, 3.29	1.00, 4.72, 99.82, 3.29, 0.00	9.70	9.70	0.00	0.00%
usa101	t0	long1	16	1.00, 3.09, 14.43, 2.60	1.00, 3.09, 14.43, 2.60, 0.00	10.47	10.47	0.00	0.00%
usa101	t0	long2	16	1.00, 3.76, 9.63, 0.00	1.00, 3.76, 9.63, 0.00, 0.00	9.77	9.78	0.00	0.00%
usa101	t1	long0	1	1.00, 3.51, 8.20, 1.97	1.00, 0.00, 8.19, 0.13, 60.24	4.55	2.67	1.89	41.42%
usa101	t1	long1	41	1.00, 4.44, 85.28, 2.76	1.00, 4.43, 85.29, 2.55, 3.98	8.99	8.86	0.13	1.45%
usa101	t1	long2	31	1.00, 3.92, 22.03, 1.97	1.00, 3.93, 22.07, 1.72, 4.30	9.86	9.14	0.72	7.32%
usa101	t2	long0	1	1.00, 5.90, 8.90, 0.00	1.00, 5.90, 8.90, 0.00, 0.00	2.70	2.70	0.00	-0.01%
usa101	t2	long1	41	1.00, 4.74, 56.37, 3.11	1.00, 4.84, 56.42, 2.59, 11.27	9.31	8.49	0.82	8.78%
usa101	t2	long2	44	1.00, 4.17, 27.99, 4.66	1.00, 4.06, 28.55, 2.99, 65.97	7.88	7.51	0.36	4.60%

In order to summarize the overall model performance improvement by incorporating unpredictability, we calculate the average percent improvement over all the datasets in each table (5.6 to 5.9), weighted by the number of trajectories in each dataset. Note that the results from datasets in Tables 5.6 to 5.9 where $\theta_{f,z}$ approaches 0 have not been included in the average. As expected, such models result in 0% improvement over the baseline model. These models cannot be used to guage model improvement caused by unpredictability because the unpredictability metric is not taken into account in the forward optimal control step. The average results are shown in Table 5.10. Each row corresponds to the average results from Tables 5.6 to 5.9. Each column corresponds to the calculated weighted average, where the last column represents the average percent improvement. From these results we can conclude that incorporating the unpredictability feature causes modest improvements in the generalization of the IRL models to the training datasets.

We report results from IRL step using the combined trajectory training datasets as presented in Table 5.11. Tables 5.12 and 5.13 summarize the results from performing the forward step on the training and test datasets, respectively.

As shown in Table 5.11, in the parameters of the IRL model incorporating unpredictability for the combined `i80` dataset, $\theta_{f,z}$ approaches 0. As expected the improvement over the baseline model using these parameters is 0 for both the training and test dataset (Tables 5.12 and 5.13). For the `usa101` dataset and the dataset that contains all trajectories we see modest improvements on both the training dataset and the testing

Table 5.9: Combined Traffic, Longitudinal

Highway	Traff.	Lat	# Traj.	$\boldsymbol{\theta}_{f,b}$	$\boldsymbol{\theta}_{f,w}$	$\overline{MEE}_{ef,b}$	$\overline{MEE}_{ef,w}$	Imp.	% Imp.
i80	ta	long0	115	1.00, 5.70, 56.81, 4.48	1.00, 5.69, 56.84, 4.08, 7.95	8.25	7.95	0.30	3.58%
i80	ta	long1	282	1.00, 4.85, 319.35, 3.86	1.00, 4.85, 319.35, 3.86, 0.00	9.56	9.56	0.00	0.00%
usa101	ta	long0	2	1.00, 4.62, 15.77, 1.73	1.00, 4.62, 15.77, 1.73, 0.00	6.10	6.10	0.00	0.00%
usa101	ta	long1	98	1.00, 4.54, 92.11, 2.90	1.00, 4.54, 92.18, 2.40, 7.48	9.38	8.97	0.41	4.36%
usa101	ta	long2	91	1.00, 3.96, 30.97, 1.94	1.00, 3.95, 30.99, 1.72, 3.45	9.68	9.35	0.33	3.45%

Table 5.10: Summary of average model performance improvement by incorporating unpredictability. These results only include runs where $\theta_{z,f} > 10^{-}5$.

Trajectory Division	Average MEE (ft.) (baseline)	Average MEE (ft.) (w/ unpredictability)	Average % Imp.
Separate Traffic, Lateral	9.25	8.62	6.83%
Combined Traffic, Lateral	9.47	9.05	4.43%
Separate Traffic, Longitudinal	8.88	8.28	6.71%
Combined Traffic, Longitudinal	9.04	8.70	3.80%

dataset. The improvements are larger on the testing dataset, meaning that incorporating unpredictability has a greater impact on modelling lane changes when using trajectories that were held out from the training of the prediction model. We expect the performance of the prediction model to be more variable on the testing dataset, making it better at measuring the unpredictability of surrounding traffic.

Table 5.11: Learned Parameters for Combined Datasets

Highway	# Train Traj.	$\boldsymbol{\theta}_{f,b}$	$\boldsymbol{\theta}_{f,w}$
i80	464	1.00, 5.09, 287.50, 4.20	1.00, 5.09, 287.50, 4.20, 0.00
usa101	286	1.00, 4.46, 103.92, 1.94	1.00, 4.44, 104.02, 1.49, 8.32
combined	750	1.00, 4.84, 383.65, 3.08	1.00, 4.82, 383.85, 2.78, 4.66

5.8.2 Qualitative Analysis

We perform a qualitative analysis of the baseline model and the model incorporating unpredictability on the trajectory illustrated in Figure 5.8. In order to illustrate the evolution of the trajectory over the time horizon we also plot individual state values in Figure 5.9. The models in this trajectory were trained using trajectories from highway `usa101`, during traffic time slot `t2`, and geometric separation `lat5`. In this trajectory we observe a large improvement with the model that incorporates unpredictability over the baseline. The model performance for the baseline model is $MEE_{ef,b} = 21.60ft.$, while the performance of the model that incorporates unpredictability is $MEE_{ef,w} = 7.62ft.$,

Table 5.12: Improvement for Combined Training Datasets

Highway	# Train Traj.	$\overline{MEE}_{ef,b}$	$\overline{MEE}_{ef,w}$	Imp.	% Imp
i80	464	9.09	9.09	0.00	0.00%
usa101	286	9.82	9.34	0.47	4.82%
combined	750	9.50	9.20	0.30	3.11%

Table 5.13: Improvement for Combined Test Datasets

Highway	# Test Traj.	$\overline{MEE}_{ef,b}$	$\overline{MEE}_{ef,w}$	Imp.	% Imp
i80	10	11.74	11.74	0.00	0.00%
usa101	9	8.06	7.29	0.77	9.57%
combined	19	10.07	9.66	0.41	4.06%

indicating an improvement of 65 percent. In Figure 5.9b, we observe that the model that incorporates unpredictability produces a significantly closer trajectory in the y dimension. Furthermore, in Figure 5.9a we can observe that the model that incorporates unpredictability is slower to proceed to the target lane. We interpret this behaviour as the model acting more cautiously by incorporating the new unpredictability feature. This interpretation is consistent with the velocity behaviour in Figure 5.10a. Whereas the velocity produced by the baseline model is very high and different from the expert trajectory, the velocity profile generated by the model that incorporates the unpredictability slowly ramps up, similar to the expert's behaviour.

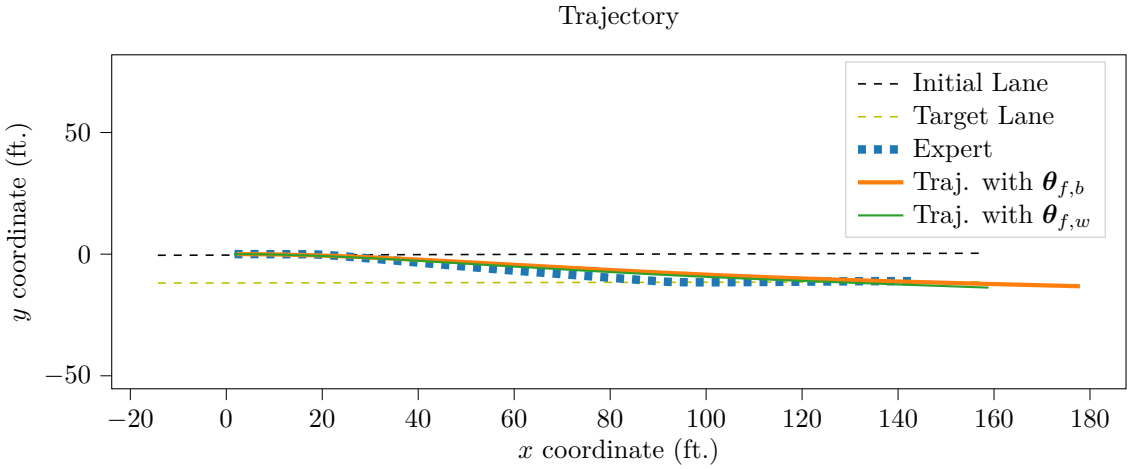
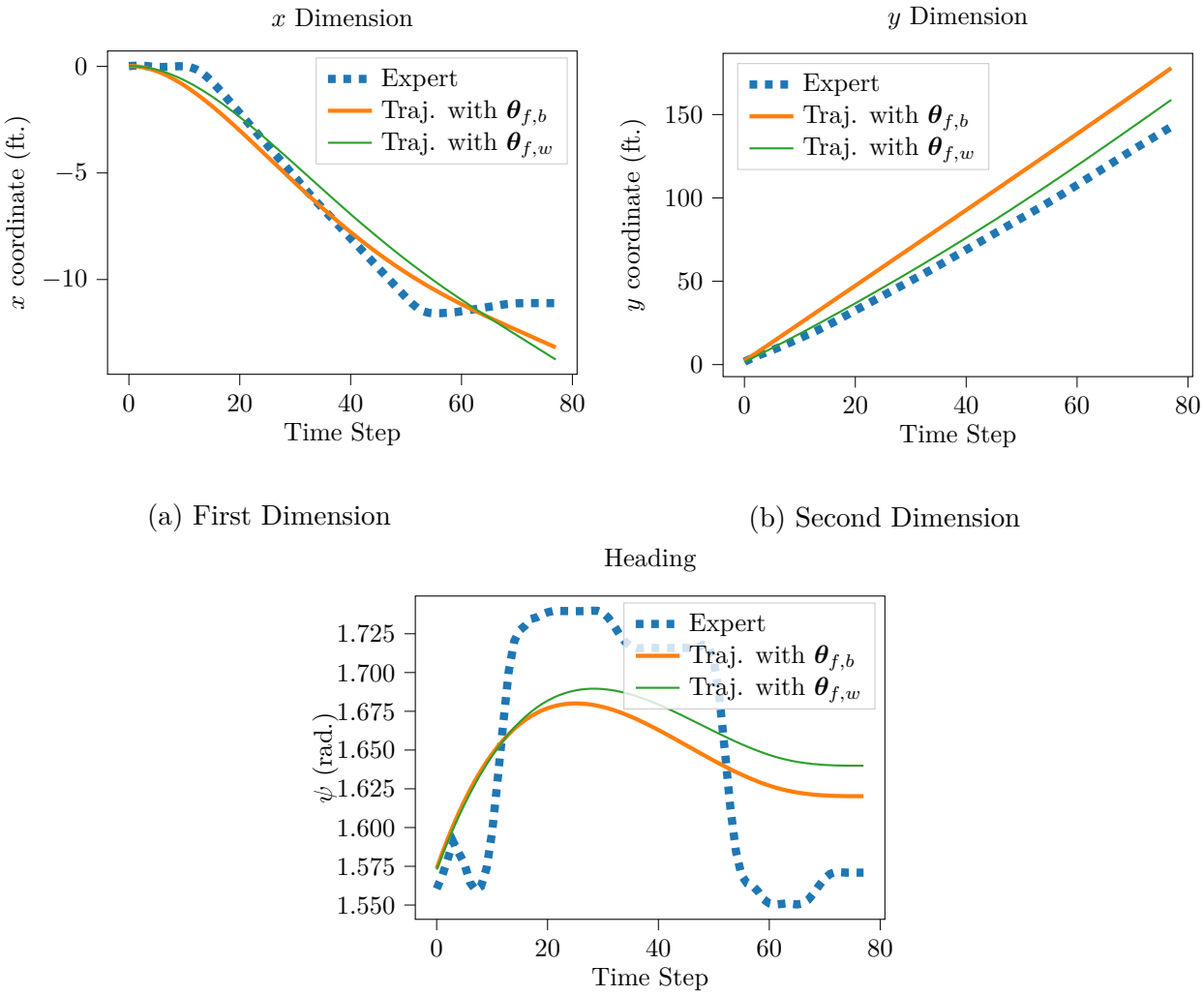


Figure 5.8: Comparison of the human expert lane change trajectory (blue) with trajectories generated by optimizing rewards parametrized by the baseline reward parameters, $\boldsymbol{\theta}_{f,b}$ (orange), and reward parameters that incorporate unpredictability, $\boldsymbol{\theta}_{f,w}$ (green).

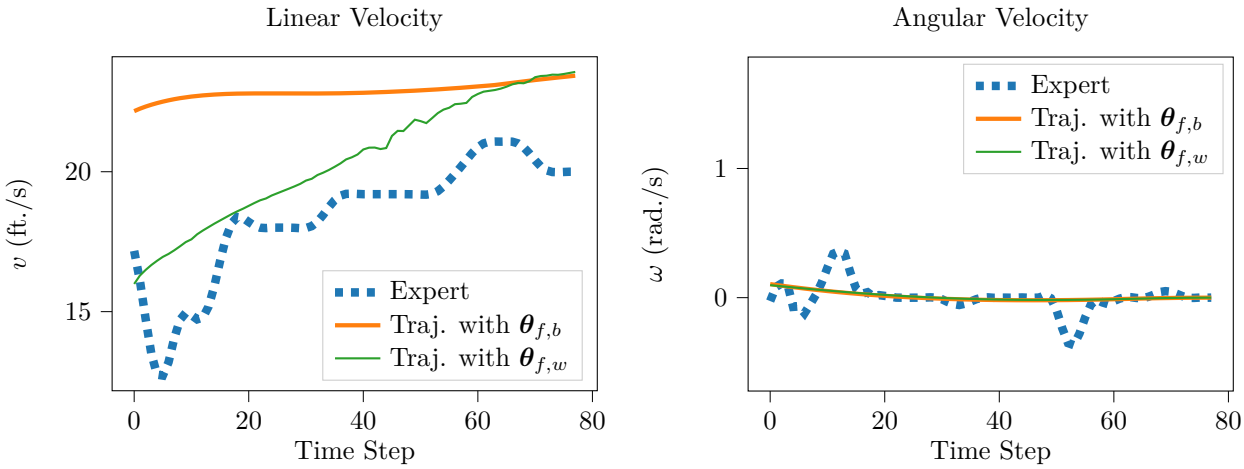


(a) First Dimension

(b) Second Dimension

(c) Heading

Figure 5.9: Comparison of the states from human-demonstrated expert lane change trajectory (blue) with trajectories generated by optimizing rewards parametrized by the baseline reward parameters, $\boldsymbol{\theta}_{f,b}$ (orange), and reward parameters that incorporate unpredictability, $\boldsymbol{\theta}_{f,w}$ (green).



(a) Linear Velocity

(b) Angular Velocity

Figure 5.10: Comparison of the actions of the human-demonstrated expert lane change trajectory (blue) with trajectories generated by optimizing rewards parametrized by the baseline reward parameters, $\boldsymbol{\theta}_{f,b}$ (orange), and reward parameters that incorporate unpredictability, $\boldsymbol{\theta}_{f,w}$ (green).

Chapter 6

Conclusions

In this thesis we proposed *unpredictability* as a measure for more human-like AV behaviour planning. We used the performance metrics of an off-the-shelf vehicle trajectory prediction model through time as a measure of the unpredictability of a human driver. We assumed that if the off-the-shelf model performs poorly on a particular car at a particular time, then that car is behaving unpredictably and vice versa. We proposed incorporating unpredictability as a measure of how an AV should act around a particular human driver. We analyzed human lane change behaviour using an Inverse Reinforcement Learning (IRL) approach to demonstrate that incorporating this unpredictability measure can better explain human driving behaviour.

We first summarized the theoretical foundations of the Maximum Entropy Inverse Reinforcement Learning (MaxEnt IRL) approach and its extension for dynamical systems with continuous control and action spaces and locally optimal demonstration trajectories. We began by reviewing the optimal control problem. Then, we presented a general formulation of the inverse problem, IRL. Starting from first principles in Information Theory, we derived the Maximum Causal Entropy probability distribution, in which inference is analogous to solving the optimal control problem. We demonstrated how finding parameters for this distribution is equivalent to solving the IRL problem.

We used a Linear Quadratic Regulator (LQR) on a linear toy example dynamical system to investigate how well the IRL algorithm can recover reward parameters from synthetically generated demonstration trajectories. After selecting some ground truth LQR parameter values, we produced synthetic trajectory datasets with varying amounts of random noise on the demonstrations and a varying number of trajectories in the datasets. We quantified the quality of the parameters learned from these datasets, in terms of the closeness of the parameter values to the ground truth values and the similarity of resulting optimal trajectories to the demonstrations. We showed that in the case

of noisy data, the improvement in quality for each additional demonstration trajectory in the dataset decreases exponentially.

Finally, we used the IRL algorithm to model reward functions for conducting lane change maneuvers in a highway setting. In order to test the hypothesis that the unpredictability of surrounding traffic would have an effect on the behaviour of the ego-car, we learned two reward functions from human data using IRL, a baseline reward function and a reward function that incorporated unpredictability. We showed modest improvements in the model that incorporated unpredictability. Therefore, we concluded that incorporating the unpredictability measure resulted in more human-like behaviour.

6.1 Future Work

There are a number of directions for extending the work presented in this thesis. In the IRL models presented in this thesis, we focused on reward functions that are composed of a linear combination of reward features. One direction of future work includes incorporating the unpredictability measure into a non-linear reward function. For example, we could include the unpredictability measure as a component in a system that builds reward features from component features [51] or we could include unpredictability as an input to a Deep Neural Network (DNN) representation of a reward function [57].

In this thesis we investigated the effect of unpredictability of adjacent vehicles. Another interesting line of future work would be to investigate whether or not human drivers minimize their own unpredictability when performing driving maneuvers. This investigation would require closing the loop between the prediction model and planning module which would require additional theoretical tools.

Bibliography

[1] Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *International Conference on Machine Learning*, number 346, New York, New York, USA, 2004.

[2] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social LSTM: Human Trajectory Prediction in Crowded Spaces. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 961–971. IEEE, 2016.

[3] Mohammad Babaeizadeh, Chelsea Finn, Dumitru Erhan, Roy H. Campbell, and Sergey Levine. Stochastic Variational Video Prediction. 2017.

[4] Richard Bellman. A Markovian Decision Process. *Journal of Mathematics and Mechanics*, 6(5):679–684, 1957.

[5] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*, volume 1. 2005.

[6] Stephen Boyd, Laurent El Ghaoui, Eric Feron, and Venkataramanan Balakrishnan. *Linear Matrix Inequalities in System and Control Theory*. Society for Industrial and Applied Mathematics, jan 1994.

[7] Noam Buckman, Alyssa Pierson, Sertac Karaman, and Daniela Rus. Generating Visibility-Aware Trajectories for Cooperative and Proactive Motion Planning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3220–3226. IEEE, 2020.

[8] Keenan Burnett, Jingxing Qian, Xintong Du, Linqiao Liu, David J. Yoon, Tianchang Shen, Susan Sun, Sepehr Samavi, Michael J. Sorocky, Mollie Bianchi, Kaicheng Zhang, Arkady Arkhangorodsky, Quinlan Sykora, Shichen Lu, Yizhou Huang, Angela P. Schoellig, and Timothy D. Barfoot. Zeus: A system description of the two-time winner of the collegiate SAE autodrive competition. *Journal of Field Robotics*, (April), 2020.

[9] Keenan Burnett, Sepehr Samavi, Steven L Waslander, Timothy D Barfoot, and Angela P Schoellig. aUToTrack : A Lightweight Object Detection and Tracking System for the SAE AutoDrive Challenge. In *Conference on Computer and Robot Vision (CRV)*, 2019.

[10] Wonmin Byeon, Qin Wang, Rupesh Kumar Srivastava, and Petros Koumoutsakos. ContextVP: Fully Context-Aware Video Prediction. *European Conference on Computer Vision (ECCV)*, 2018.

[11] James Colyar and John Halkia. Interstate 80 Freeway Dataset Federal Highway Administration, 2006.

[12] James Colyar and John Halkia. USA 101 Freeway Dataset Federal Highway Administration. Technical report, 2007.

[13] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. 2005.

[14] Nachiket Deo and Mohan M. Trivedi. Convolutional social pooling for vehicle trajectory prediction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 1549–1557, 2018.

[15] Anca D. Dragan, Kenton C.T. Lee, and Siddhartha S. Srinivasa. Legibility and predictability of robot motion. *ACM/IEEE International Conference on Human-Robot Interaction*, pages 301–308, 2013.

[16] Paolo Falcone, Francesco Borrelli, Jahan Asgari, Hongtei Eric Tseng, and Davor Hrovat. Predictive Active Steering Control for Autonomous Vehicle Systems. *IEEE Transactions on Control Systems Technology*, 15(3):566–580, 2007.

[17] Dennis Fassbender, Benjamin C. Heinrich, Thorsten Luettel, and Hans Joachim Wuensche. An optimization approach to trajectory generation for autonomous vehicle following. In *IEEE International Conference on Intelligent Robots and Systems*, volume 2017-Septe, pages 3675–3680, 2017.

[18] Jaime F. Fisac, Eli Bronstein, Elis Stefansson, Dorsa Sadigh, S. Shankar Sastry, and Anca D. Dragan. Hierarchical Game-Theoretic Planning for Autonomous Vehicles. *IEEE International Conference on Robotics and Automation*, pages 9590–9596, 2019.

[19] Yanlei Gu, Yoriyoshi Hashimoto, Li Ta Hsu, Miho Iryo-Asano, and Shunsuke Kamijo. Human-like motion planning model for driving in signalized intersections. *IATSS Research*, 41(3):129–139, 2017.

[20] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, 2020.

[21] Daiki Hayashi, Yunfei Xu, Takashi Bando, and Kazuya Takeda. A Predictive Reward Function for Human-like Driving based on a Transition Model of Surrounding Environment. In *IEEE International Conference on Robotics and Automation*, pages 7618–7624, 2019.

[22] Stefan Hoermann, Martin Bach, and Klaus Dietmayer. Dynamic Occupancy Grid Prediction for Urban Autonomous Driving: A Deep Learning Approach with Fully Automatic Labeling. *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2056–2063, 2018.

[23] Masha Itkina, Katherine Driggs-Campbell, and Mykel J. Kochenderfer. Dynamic Environment Prediction in Urban Scenes using Recurrent Representation Learning. 2019.

[24] Edwin T. Jaynes. Information Theory and Statistical Mechanics. *Physical Review*, 106(4):620–630, 1957.

[25] Edwin T. Jaynes. Information Theory and Statistical Mechanics, 1963.

[26] Steven G. Johnson. The NLopt nonlinear-optimization package, 2020.

[27] R. E. Kalman. When is a linear control system optimal? *Journal of Basic Engineering, Transactions of the ASME*, 86(1):51–60, 1964.

[28] Byeoung Do Kim, Chang Mook Kang, Jaekyum Kim, Seung Hi Lee, Chung Choo Chung, and Jun Won Choi. Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network. In *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, pages 399–404, 2018.

[29] Markus Koppenborg, Peter Nickel, Birgit Naber, Andy Lungfiel, and Michael Huelke. Effects of movement speed and predictability in human–robot collaboration. *Human Factors and Ergonomics In Manufacturing*, 27(4):197–209, 2017.

[30] Dieter Kraft. Algorithm 733: TOMP–Fortran modules for optimal control calculations. *ACM Transactions on Mathematical Software*, 20(3):262–281, 1994.

[31] Gerhard Kramer. Directed information for channels with feedback. *PhD thesis*, (12656), 1998.

[32] Markus Kuderer, Shilpa Gulati, and Wolfram Burgard. Learning driving styles for autonomous vehicles from demonstration. In *IEEE International Conference on Robotics and Automation*, number June, pages 2641–2646. IEEE, 2015.

[33] Sergey Levine and Vladlen Koltun. Continuous inverse optimal control with locally optimal examples. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012*, pages 41–48, 2012.

[34] Christina Lichtenthaler, Tamara Lorenzy, and Alexandra Kirsch. Influence of legibility on perceived safety in a virtual human-robot path crossing task. In *IEEE International Workshop on Robot and Human Interactive Communication*, pages 676–681, 2012.

[35] Alexander Liniger, Alexander Domahidi, and Manfred Morari. Optimization-based autonomous racing of 1:43 scale RC cars. *Optimal Control Applications and Methods*, 36(5):628–647, 2015.

[36] William Lotter, Gabriel Kreiman, and David Cox. Deep Predictive Coding Networks for Video Prediction and Unsupervised Learning. pages 1–18, 2016.

[37] Raquel Luo, Wenjie and Yang, Bin and Urtasun. Fast and Furious: Real Time End-to-End 3D Detection, Tracking and Motion Forecasting With a Single Convolutional Net. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3569–3577, 2018.

[38] David JC MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 7.2 edition, 2007.

[39] Nima Mohajerin and Mohsen Rohani. Multi-Step Prediction of Occupancy Grid Maps with Recurrent Neural Networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10600–10608, 2019.

[40] Marcello Montanino and Vincenzo Punzo. Making NGSIM Data Usable for Studies on Traffic Flow Theory. *Transportation Research Record: Journal of the Transportation Research Board*, 2390(1):99–111, 2013.

[41] Marcello Montanino and Vincenzo Punzo. Trajectory data reconstruction and simulation-based validation against macroscopic traffic patterns. *Transportation Research Part B: Methodological*, 80:82–106, 2015.

[42] Maximilian Naumann, Liting Sun, Wei Zhan, and Masayoshi Tomizuka. Analyzing the Suitability of Cost Functions for Explaining and Imitating Human Driving Behavior based on Inverse Reinforcement Learning. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 5481–5487, 2020.

[43] Andrew Y. Ng and Stuart Russell. Algorithms for Inverse Reinforcement Learning. In *International Conference on Machine Learning*, pages 663–670, 2000.

[44] M. Cody Priess, Richard Conway, Jongeun Choi, John M. Popovich, and Clark Radcliffe. Solutions to the inverse LQR problem with application to biological systems analysis. *IEEE Transactions on Control Systems Technology*, 23(2):770–777, 2015.

[45] Yi Ren, Steven Elliott, Yiwei Wang, Yezhou Yang, and Wenlong Zhang. How Shall I Drive? Interaction Modeling and Motion Planning towards Empathetic and Socially-Graceful Driving. In *IEEE International Conference on Robotics and Automation*, pages 4325–4331, 2019.

[46] Nicholas Rhinehart, Rowan McAllister, Kris Kitani, and Sergey Levine. PRECOG: PREdiction Conditioned On Goals in Visual Multi-Agent Settings. 2019.

[47] Dorsa Sadigh, Shankar Sastry, Sanjit A. Seshia, and Anca D. Dragan. Planning for Autonomous Cars that Leverage Effects on Human Actions. In *Robotics: Science and Systems XII*. Robotics: Science and Systems Foundation, 2016.

[48] Georg Schildbach and Francesco Borrelli. Scenario model predictive control for lane change assistance on highways. In *IEEE Intelligent Vehicles Symposium, Proceedings*, pages 611–616. IEEE, 2015.

[49] Wilko Schwarting, Javier Alonso-Mora, and Daniela Rus. Planning and Decision-Making for Autonomous Vehicles. *Annual Review of Control, Robotics, and Autonomous Systems*, 1(1):187–210, 2018.

[50] Wilko Schwarting, Alyssa Pierson, Javier Alonso-Mora, Sertac Karaman, and Daniela Rus. Social behavior for autonomous vehicles. *Proceedings of the National Academy of Sciences (PNAS)*, pages 1–7, 2019.

[51] Vladlen Koltun Sergey Levine, Zoran Popovic. Feature Construction for Inverse Reinforcement. In *Advances in Neural Information Processing Systems*, pages 1342–1350, 2010.

[52] C. E. Shannon. A Mathematical Theory of Communication. *Bell System Technical Journal*, 27(3):379–423, 1948.

[53] Liting Sun, Wei Zhan, Masayoshi Tomizuka, and Anca D. Dragan. Courteous Autonomous Cars. *IEEE International Conference on Intelligent Robots and Systems*, pages 663–670, 2018.

[54] Christian Thiemann, Martin Treiber, and Arne Kesting. Estimating acceleration and lane-changing dynamics from next generation simulation trajectory data. *Transportation Research Record*, (2088):90–101, 2008.

[55] Ruben Villegas, Jimei Yang, Yuliang Zou, Sungryull Sohn, Xunyu Lin, and Honglak Lee. Learning to generate long-term future via hierarchical prediction. In *International Conference on Machine Learning (ICML)*, volume 7, pages 5429–5449, 2017.

[56] Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J van der Walt, Matthew Brett, Joshua Wilson, K Jarrod Millman, Nikolay Mayorov, Andrew R J Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, \.Ilhan Polat, Yu Feng, Eric W Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E A Quintero, Charles R Harris, Anne M Archibald, Antônio H Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[57] Markus Wulfmeier, Peter Ondruska, and Ingmar Posner. Maximum Entropy Deep Inverse Reinforcement Learning. Technical report, 2015.

[58] Tianfan Xue, Jiajun Wu, Katherine L. Bouman, and William T. Freeman. Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks. *Advances in Neural Information Processing Systems*, pages 91–99, 2016.

[59] Han Zhang, Yibei Li, and Xiaoming Hu. Inverse Optimal Control for Finite-Horizon Discrete-time Linear Quadratic Regulator Under Noisy Output. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, volume 2019-Decem, pages 6663–6668. IEEE, 2019.

[60] Brian D. Ziebart. *Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy*. PhD thesis, Carnegie Mellon University, 2010.

[61] Brian D Ziebart, Andrew Maas, J Andrew Bagnell, and Anind K Dey. Maximum Entropy Inverse Reinforcement Learning. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*, pages 1433–1438, 2008.

Appendix A

Auxilliary Proofs and Theorem

Proof of Theorem 3. We will present the proof for Equation (3.13) in the discrete case. Let P_i represent $P(\mathrm{x}_i)$. Without loss of generality, let $p = 1$, i.e. there is only one characteristic function, $\phi(\mathrm{x}_i)$. We begin by defining the Lagrangian for a generic i, and Lagrange multipliers λ for the total probability constraint, and θ for the characteristic matching constraint. Without loss of generality, assume that the empirical expectation of the characteristic function is $\mathbb{E}_{\tilde{P}(\mathrm{x})}[\phi(\mathrm{x})] = 0$.

$$\mathcal{L}_i(P_i, \lambda, \theta) = P_i \log P_i - \theta P_i \phi(\mathrm{x}_i) - \lambda(P_i - 1).$$

To find the optimal distribution, we equate the gradient with respect to P_i to zero.

$$\nabla_{P_i} \mathcal{L}_i(P_i, \lambda, \theta) = 0 = 1 + \log P_i - \mu\phi(\mathrm{x}_i) - \lambda.$$

We can absorb the 1 into λ to get,

$$\begin{aligned} 0 &= \log P_i - \theta\phi(\mathrm{x}_i) - \lambda \\ P_i &= \exp(-\lambda - \theta\phi(\mathrm{x}_i)) \end{aligned} \tag{A.1}$$

To obtain the values of the Lagrange multipliers, we can plug (A.1) into the equation for the total probability constraint.

$$\begin{aligned} \sum_i P_i - 1 &= 0 \\ \sum_i \exp(-\lambda - \theta\phi(\mathrm{x}_i)) - 1 &= 0 \\ \sum_i \exp(-\theta\phi(\mathrm{x}_i)) &= \exp(\lambda) \\ \lambda &= \log\left(\sum_i \exp(-\theta\phi(\mathrm{x}_i))\right) \\ \lambda &= \log Z(\theta) \end{aligned} \tag{A.2}$$

where we define the partition function $Z(\theta) := \sum_i \exp(-\theta\phi(\mathrm{x}_i))$. Now we can plug (A.2) into (A.1) to obtain,

$$P_i = \frac{\exp(-\theta\phi(\mathrm{x}_i))}{Z(\theta)} \tag{A.3}$$

For multiple characteristic functions $\phi_1, \ldots, \phi_p$, we can show

$$\mathbb{E}[\phi_j(\mathrm{x}_i)] = \sum_i \exp\left(-\sum_j \theta_j \phi_j(\mathrm{x}_i)\right)$$

and $Z(\theta_1, \ldots, \theta_p) := \sum_i \exp(\sum_j -\theta_j\phi_j(\mathrm{x}_i))$, to arrive at (3.13). Furthermore, to expand the proof to the case of continuous random variables, we would replace x_i with continuous value x, P_i with a continuous function p, and the sum $\sum_i$ with an integral over the space of all possible values of x, $\int_{\mathcal{X}}$. □

Proof of Theorem 4. We follow the same argument as the proof of Theorem 3. We define the Lagrangian,

$$\begin{aligned} \Lambda(P, \boldsymbol{\theta}) = \mathbb{H}_P(\mathbf{u}||\mathbf{x}) &+ \sum_j \theta_j(\mathbb{E}_{P(\mathbf{x},\mathbf{u})}[\phi_j(\mathbf{x},\mathbf{u})] - \mathbb{E}_{\tilde{P}(\mathbf{x},\mathbf{u})}[\phi_j(\mathbf{x},\mathbf{u})]) \\ &+ \sum_{k=0}^{K} \sum_{\mathbf{x}_{1:k}} \sum_{\mathbf{u}_{1:k-1}} \lambda_{\mathbf{x}_{1:k},\mathbf{u}_{1:k-1}} \sum_{\mathbf{u}_k} P(\mathbf{u}_k|\mathbf{x}_{1:k}, \mathbf{u}_{1:k-1}). \end{aligned} \tag{A.4}$$

Then we take the gradient with respect to, $P(\mathbf{u}_k|\mathbf{x}_{1:k}, \mathbf{u}_{1:k-1})$ for a generic time step,

k to get

$$\nabla_{P(\mathbf{u}_k|\mathbf{x}_{1:k},\mathbf{u}_{1:k-1})}\Lambda(P,\boldsymbol{\theta}) = 0 = \lambda_{\mathbf{x}_{1:k},\mathbf{u}_{1:k-1}} - P(\mathbf{x}_{1:k},\mathbf{u}_{1:k-1})\left(\sum_{\kappa=k}^{K}\mathbb{E}_{P(\mathbf{x},\mathbf{u})}[\log P(\mathbf{u}_\kappa|\mathbf{x}_{1:\kappa},\mathbf{u}_{1:\kappa-1})|\mathbf{x}_{0:k},\mathbf{u}_{0:k}] - \sum_j \theta_j \mathbb{E}_{P(\mathbf{x},\mathbf{u})}[\phi_j(\mathbf{x},\mathbf{u})|\mathbf{x}_{0:k},\mathbf{u}_{0:k}]\right) \tag{A.5}$$

where the expectations are conditional, i.e. $\mathbb{E}_x[f(x)|y] = \sum_x f(x)P(x|y)$.

$$0 = \lambda_{\mathbf{x}_{1:k},\mathbf{u}_{1:k-1}} - P(\mathbf{x}_{1:k},\mathbf{u}_{1:k-1}) \left(\sum_{\kappa=k}^{K}\sum_{\mathbf{x}}\sum_{\mathbf{u}} P(\mathbf{x},\mathbf{u}|\mathbf{x}_{0:k},\mathbf{u}_{0:k})\log P(\mathbf{u}_\kappa|\mathbf{x}_{1:\kappa},\mathbf{u}_{1:\kappa-1}) + \sum_j \theta_j \mathbb{E}_{P(\mathbf{x},\mathbf{u})}[\phi_j(\mathbf{x},\mathbf{u})|\mathbf{x}_{0:k},\mathbf{u}_{0:k}]\right) \tag{A.6}$$

We can divide by $P(\mathbf{x}_{1:k},\mathbf{u}_{1:k-1})$ and factor the the term $\log P(\mathbf{u}_k|\mathbf{x}_{0:k},\mathbf{u}_{0:k-1})$ out of the first expectation to get

$$\begin{aligned} -\frac{\lambda_{\mathbf{x}_{1:k},\mathbf{u}_{1:k-1}}}{P(\mathbf{x}_{1:k},\mathbf{u}_{1:k-1})} &= -\sum_{\mathbf{x}}\sum_{\mathbf{u}} P(\mathbf{x}_{k+1:K},\mathbf{u}_{k+1:K-1})\log P(\mathbf{u}_k|\mathbf{x}_{1:k},\mathbf{u}_{1:k-1}) \\ &\quad - \sum_{\kappa=k+1}^{K}\sum_{\mathbf{x}}\sum_{\mathbf{u}} P(\mathbf{x}_{k+1:K},\mathbf{u}_{k+1:K-1})\log P(\mathbf{u}_\kappa|\mathbf{x}_{1:\kappa},\mathbf{u}_{1:\kappa-1}) \\ &\quad + \sum_j \theta_j \mathbb{E}_{P(\mathbf{x},\mathbf{u})}[\phi_j(\mathbf{x},\mathbf{u})|\mathbf{x}_{0:k},\mathbf{u}_{0:k}]. \end{aligned} \tag{A.7}$$

But since $\mathbf{u}_k$ is independent from $\mathbf{x}_{k+1:K},\mathbf{u}_{k+1:K-1}$, then we are left with,

$$\log P(\mathbf{u}_k|\mathbf{x}_{1:k},\mathbf{u}_{1:k-1}) = -\frac{\lambda_{\mathbf{x}_{1:k},\mathbf{u}_{1:k-1}}}{P(\mathbf{x}_{1:k},\mathbf{u}_{1:k-1})} + \sum_{\kappa=k+1}^{K}\mathbb{E}_{P(\mathbf{x},\mathbf{u})}[\log P(\mathbf{u}_\kappa|\mathbf{x}_{1:\kappa},\mathbf{u}_{1:\kappa-1})|\mathbf{x}_{0:k},\mathbf{u}_{0:k}] - \sum_j \theta_j \mathbb{E}_{P(\mathbf{x},\mathbf{u})}[\phi_j(\mathbf{x},\mathbf{u})|\mathbf{x}_{0:k},\mathbf{u}_{0:k}]. \tag{A.8}$$

$$P(\mathbf{u}_k|\mathbf{x}_{1:k},\mathbf{u}_{1:k-1}) = \frac{1}{\exp\left(\frac{\lambda_{\mathbf{x}_{1:k},\mathbf{u}_{1:k-1}}}{P(\mathbf{x}_{1:k},\mathbf{u}_{1:k-1})}\right)} \exp\left(\sum_{\kappa=k+1}^{K}\mathbb{E}_{P(\mathbf{x},\mathbf{u})}[\log P(\mathbf{u}_\kappa|\mathbf{x}_{1:\kappa},\mathbf{u}_{1:\kappa-1})|\mathbf{x}_{0:k},\mathbf{u}_{0:k}] - \sum_j \theta_j \mathbb{E}_{P(\mathbf{x},\mathbf{u})}[\phi_j(\mathbf{x},\mathbf{u})|\mathbf{x}_{0:k},\mathbf{u}_{0:k}]\right). \tag{A.9}$$

This is a recursive definition. Starting at the distribution for the final action, $P(\mathbf{u}_{K-1}|\mathbf{x}_{0:K-1}, \mathbf{u}_{0:K-2})$, we can recurse back in time to calculate the distribution for the input at each time step. In order to prove the recursive structure in Theorem 4, we will substitute the recursive partition functions (3.22) into (A.7). The theorem is proven by selecting a value of $\lambda_{\mathbf{x}_{1:k},\mathbf{u}_{1:k-1}}$ based on the recursive partition functions to solve the optimization poblem (Equation (3.16)).

$$
\begin{aligned}
-\frac{\lambda_{\mathbf{x}_{1:k},\mathbf{u}_{1:k-1}}}{P(\mathbf{x}_{1:k},\mathbf{u}_{1:k-1})} &= -\sum_{\mathbf{x}}\sum_{\mathbf{u}} P(\mathbf{x}_{k+1:K},\mathbf{u}_{k+1:K-1}) \log P(\mathbf{u}_k|\mathbf{x}_{1:k},\mathbf{u}_{1:k-1}) \\
&\quad - \sum_{\kappa=k+1}^{K}\sum_{\mathbf{x}}\sum_{\mathbf{u}} P(\mathbf{x}_{k+1:K},\mathbf{u}_{k+1:K-1}) \log P(\mathbf{u}_\kappa|\mathbf{x}_{1:\kappa},\mathbf{u}_{1:\kappa-1}) \\
&\quad + \sum_j \theta_j \mathbb{E}_{P(\mathbf{x},\mathbf{u})}[\phi_j(\mathbf{x},\mathbf{u})|\mathbf{x}_{0:k},\mathbf{u}_{0:k}] \\
-\frac{\lambda_{\mathbf{x}_{1:k},\mathbf{u}_{1:k-1}}}{P(\mathbf{x}_{1:k},\mathbf{u}_{1:k-1})} &= -\log P(\mathbf{u}_k|\mathbf{x}_{1:k},\mathbf{u}_{1:k-1}) - \sum_{\kappa=k+1}^{K} \mathbb{E}_{P(\mathbf{x},\mathbf{u})}[\log P(\mathbf{u}_\kappa|\mathbf{x}_{1:\kappa},\mathbf{u}_{1:\kappa-1})|\mathbf{x}_{0:k},\mathbf{u}_{0:k}] \\
&\quad + \sum_j \theta_j \mathbb{E}_{P(\mathbf{x},\mathbf{u})}[\phi_j(\mathbf{x},\mathbf{u})|\mathbf{x}_{0:k},\mathbf{u}_{0:k}] \\
-\frac{\lambda_{\mathbf{x}_{1:k},\mathbf{u}_{1:k-1}}}{P(\mathbf{x}_{1:k},\mathbf{u}_{1:k-1})} &= -\sum_{\kappa=k}^{K-1} \mathbb{E}_{P(\mathbf{x},\mathbf{u})}[\sum_{\mathbf{x}_{\kappa+1}} P(\mathbf{x}_\kappa+1|\mathbf{x}_{0:\kappa},\mathbf{u}_{0:\kappa}) \log \mathrm{Z}_{\mathbf{x}_{0:\kappa+1},\mathbf{u}_{0:\kappa}} - \log \mathrm{Z}_{\mathbf{x}_{0:\kappa},\mathbf{u}_{0:\kappa-1}}|\mathbf{x}_{0:k},\mathbf{u}_{0:k}] \\
&\quad - \mathbb{E}_{P(\mathbf{x},\mathbf{u})}[\boldsymbol{\theta}^\top\boldsymbol{\phi} - \log \mathrm{Z}_{\mathbf{x}_{0:K},\mathbf{u}_{0:K-1}}|\mathbf{x}_{0:k},\mathbf{u}_{0:k}] + \mathbb{E}_{P(\mathbf{x},\mathbf{u})}[\boldsymbol{\theta}^\top\boldsymbol{\phi}|\mathbf{x}_{0:k},\mathbf{u}_{0:k}] \\
-\frac{\lambda_{\mathbf{x}_{1:k},\mathbf{u}_{1:k-1}}}{P(\mathbf{x}_{1:k},\mathbf{u}_{1:k-1})} &= -\sum_{\kappa=k}^{K-1} \mathbb{E}_{P(\mathbf{x},\mathbf{u})}[\log \mathrm{Z}_{\mathbf{x}_{0:\kappa+1},\mathbf{u}_{0:\kappa}} - \log \mathrm{Z}_{\mathbf{x}_{0:\kappa},\mathbf{u}_{0:\kappa-1}}|\mathbf{x}_{0:k},\mathbf{u}_{0:k}] - \\
&\quad \mathbb{E}_{P(\mathbf{x},\mathbf{u})}[-\log \mathrm{Z}_{\mathbf{x}_{0:K},\mathbf{u}_{0:K-1}}|\mathbf{x}_{0:k},\mathbf{u}_{0:k}] \\
\lambda_{\mathbf{x}_{1:k},\mathbf{u}_{1:k-1}} &= P(\mathbf{x}_{1:k},\mathbf{u}_{1:k-1}) \log \mathrm{Z}_{\mathbf{x}_{0:k},\mathbf{u}_{0:k-1}}.
\end{aligned}
\tag{A.10}
$$

□

Probability of a Policy under the Maximum Causal Entropy Distribution

In optimal control, the optimality of a policy is binary. However with the MCE model, we have a soft interpretation of optimality. Therefore we need to be able to assign a probability to a particular stochastic policy under the MCE model. This probability is used as a measure of optimality.

Let $P(\pi) := \{P_0(\mu_0(\mathbf{x}_0)), \ldots, P_{K-1}(\mu_{K-1}(\mathbf{x}_{K-1}))\}$ be a stochastic policy. At each time step, k, we sample a policy from the probability distribution, $P_k(\mu_k(\mathbf{x}_k))$. The MCE model provides us with a probability distribution for a sequence of actions causally

conditioned on a sequence of states (Definition 4), $P(\mathbf{u}||\mathbf{x})$. We can interpret the distribution as the probability that a given sequence of states and actions may occur under the MCE model. Under policy $P(\pi)$, let $P_\pi(\mathbf{x}, \mathbf{u})$ be the probability of a sequence of states and actions, let $P_\pi(\mathbf{x}_k, \mathbf{u}_k)$ be the marginal probability of a state and action pair at a particular time step, and let $P_\pi(\mathbf{x}_k)$ be the marginal probability of a particular state occurring at a particular time step. Then, the probability of the policy being optimal under the MCE model can be found by using a multinomial distribution with $P(\mathbf{u}||\mathbf{x})$ as the probability of a configuration and $P_\pi(\mathbf{x}, \mathbf{u})$ as the number of times a configuration occurs within a set number of trials [60].

$$
\begin{aligned}
P_{\boldsymbol{\theta}}(\pi) &= \prod_{\mathbf{u}_{0:K-1}, \mathbf{x}_{0:K-1}} P(\mathbf{u}_{0:K-1}||\mathbf{x}_{0:K-1})^{\pi(\mathbf{x}_{0:K-1}, \mathbf{u}_{0:K-1})} \\
\log P_{\boldsymbol{\theta}}(\pi) &= \sum_{\mathbf{u}_{0:K-1}, \mathbf{x}_{0:K-1}} \pi(\mathbf{x}_k, \mathbf{u}_k) \log P(\mathbf{u}_{0:K-1}||\mathbf{x}_{0:K-1}) \\
&= \sum_{k=0}^{K-1} \sum_{\mathbf{u}_k, \mathbf{x}_k} \pi(\mathbf{x}_k, \mathbf{u}_k) \log P(\mathbf{u}_k|\mathbf{x}_k) \\
&= \sum_{k=0}^{K-1} \sum_{\mathbf{u}_k, \mathbf{x}_k} \pi(\mathbf{x}_k, \mathbf{u}_k)(Q^{soft}_{\boldsymbol{\theta}}(\mathbf{u}_k, \mathbf{x}_k) - V^{soft}_{\boldsymbol{\theta}}(\mathbf{x}_k)) \\
&= \sum_{k=0}^{K-1} \sum_{\mathbf{u}_k, \mathbf{x}_k} \pi(\mathbf{x}_k, \mathbf{u}_k) \left(\sum_{\mathbf{x}_{k+1}} P(x_{k+1}|\mathbf{x}_k, \mathbf{u}_k) V^{soft}_{\boldsymbol{\theta}}(\mathbf{x}_{k+1}) + \boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{x}_k, \mathbf{u}_k) - V^{soft}_{\boldsymbol{\theta}}(\mathbf{x}_k) \right) \\
&= \sum_{k=0}^{K-1} \sum_{\mathbf{u}_k, \mathbf{x}_k} \pi(\mathbf{x}_k, \mathbf{u}_k) \boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{x}_k, \mathbf{u}_k) + \sum_{k=0}^{K-1} \sum_{\mathbf{u}_k, \mathbf{x}_k} \pi(\mathbf{x}_k) V^{soft}_{\boldsymbol{\theta}}(\mathbf{x}_k) - \sum_{k=1}^{K-1} \sum_{\mathbf{u}_k, \mathbf{x}_k} \pi(\mathbf{x}_k) V^{soft}_{\boldsymbol{\theta}}(\mathbf{x}_k) \\
&= \mathbb{E}_{\pi(\mathbf{x}_{0:K-1}, \mathbf{u}_{0:K-1})} \left[\sum_{k=0}^{K-1} \boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{x}_k, \mathbf{u}_k) \right] - \sum_{\mathbf{x}_0} P(\mathbf{x}_0) V^{soft}_{\boldsymbol{\theta}}(\mathbf{x}_0)
\end{aligned}
\tag{A.11}
$$

We summarize the result of the derivation in the following theorem,

Theorem 6 ([60]). *The probability of a stochastic policy* $\pi := \{\mu_0(\mathbf{x}_0), \ldots, \mu_{K-1}(\mathbf{x}_{K-1})\}$*, under the Maximum Causal Entropy distribution is given by,*

$$
P^{soft}_{\boldsymbol{\theta}}(\pi) = \frac{\exp\left(\mathbb{E}_{\pi(\mathbf{x}_{0:K-1}, \mathbf{u}_{0:K-1})} \left[\sum_{k=0}^{K-1} \boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{x}_k, \mathbf{u}_k) \right] \right)}{\exp\left(\sum_{\mathbf{x}_0} P(\mathbf{x}_0) V^{soft}_{\boldsymbol{\theta}}(\mathbf{x}_0) \right)}. \tag{A.12}
$$

The denominator is independent of the policy.

Appendix B

Towards an Autonomous Vehicle Scoring Framework

Summary: As autonomous vehicles (AV) continue to proliferate around the world, developing comprehensive operational performance measurements for such vehicles is becoming a major challenge. Municipalities and other levels of government are allowing vehicles with varying levels of autonomous features on their roads. They have an interest in investigating whether or not autonomous vehicles are performing to the level claimed by manufacturers. The goal of this project was to serve as a first step for evaluating autonomous vehicle performance on public roads based on data available from a telematics device.

To this end, we described a framework for evaluating specific maneuvers performed by the autonomous vehicle. We presented three classes of measures: behavioural indicators, safety indicators, and quality indicators. Behavioural indicators are first used to determine what maneuvers have been performed by the AV and to segment the maneuvers for evaluation. Then, safety indicators are used to compare the execution of a maneuver with driving requirements. Finally, quality indicators are used to evaluate how normal or desirable the execution of the maneuver was. We performed an experimental evaluation of the framework using the maneuvers *stopping at a stop sign* and *proceeding through an intersection* using our autonomous vehicle, Zeus, and a GeoTab Go telematics device. We also compared the data collected from the GeoTab device with data collected from our vehicle's sensors. We finally provided a set of findings and recommendations for future investigation. The findings of this study were presented in a technical report.