Maximum Likelihood Approach
For
Model-Free Inverse Reinforcement Learning

by

Vinamra Jain

(Under the Direction of Prashant Doshi)

Abstract

Preparing an intelligent system in advance to respond optimally in every possible situation is difficult. Machine learning approaches like Inverse Reinforcement Learning can help learning behavior using a limited number of demonstrations. We present a model-free technique by applying maximum likelihood estimation to an IRL problem. To make our approach model-free, we model the environment using the canonical Markov Decision Process tuple, except we exclude the transition function. We define our reward function as a linear function of a known set of features. We use a modified Q-learning technique,called Q-Averaging. The direction for optimization is guided by the gradient of likelihood function for current feature weights until the unknown reward function is identified.

Experimental results over a grid world problem supports our model-free representation of an IRL technique. We also extend our experiments to real-world freeway merging problem of autonomous cars and the results are significant.

Index words: Maximum Likelihood, Inverse Reinforcement Learning, Model Free, Markov Decision Process, Q-Averaging

Maximum Likelihood Approach For Model-Free Inverse Reinforcement Learning

by

Vinamra Jain

B.E., Rajiv Gandhi Technical University, 2014

A Thesis Submitted to the Graduate Faculty
of The University of Georgia in Partial Fulfillment
of the
Requirements for the Degree

Master of Science

Athens, Georgia

2017

Maximum Likelihood Approach

For

Model-Free Inverse Reinforcement Learning

by

Vinamra Jain

Major Professor: Prashant Doshi

Committee: Yi Hong
Frederick Maier

Electronic Version Approved:

Suzanne Barbour
Dean of the Graduate School
The University of Georgia
December 2017

Dedication

To my Mom and Dad.

Acknowledgments

I would first express my deepest gratitude to my major professor Dr. Prashant Doshi for his countless motivation and support. The weekly meetings and discussions held by Prof. Doshi kept me pushing towards the success of this thesis. I would also like to thank Dr. Yi Hong and Dr. Frederick Maier for being a part of my thesis committee. Prof. Maier also helped me in getting access to some of the great resources of university.

I would like to thank Roi Ceren and Helene Halstead for helping me proofread my thesis document. I would like to thank Sanath Bhat for his insights and expertise. His guidance helped me to understand my work even deeper.

I would like to thank my parents and my sister for never losing faith in me and always encouraging me to achieve my goals. I would like to extend a very special thanks to Lovina Dmello for forcing me to keep moving ahead even in difficult times. I would like to thank Jugal Panchal and Chirag Jain for their unfailing support.

Finally, I must express my profound gratitude to the Thinc Lab and Department of Computer Science with all faculties and students for all the stimulating discussions and insightful comments throughout my master's degree.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter, we catalog the purpose and significance of this thesis. Section 1.1 describes the problem that is solved during the course of thesis. Section 1.2 illustrates the motivation behind this thesis. The freeway merging problem involving autonomous cars and the need of solving that problem using our approach is well discussed in motivation. The contributions are noted in Section 1.3 and the structure of thesis is outlined in Section 1.4.

1.1 Problem

Machine Learning is the technology that enables computers to become intelligent. Google's self-driving cars and robots are programmed using machine learning algorithms to learn how to make optimal decisions in any given environment. One way of programming an agent is by a Reinforcement Learning (RL) algorithm. In each time-step, the agent makes a decision and performs an action, this results in some specific rewards. If rewards are positive, the agent is more likely to perform similar actions in the future. If rewards are negative, the agent tries to avoid similar actions for this state. Hence, in reinforcement learning, the agent's action in future situations are determined by the rewards achieved in the past for similar situations. However, explicitly defining a reward function is not always easy. Also, RL algorithms usually require a large number of iterations before converging to a near-optimal policy, which is not efficient.

Another way of programming an agent to learn how to perform is using Inverse Reinforcement Learning (IRL). Here, the reward function is not defined explicitly; instead, it is expressed in term of features affecting the reward of an agent. IRL is the inverse of RL as the input and output of each are interchanged. The input to an RL is the rewards from previous actions and the output is the learned optimal policy, whereas the input to IRL is an optimal policy (referred as expert's trajectories) and the output is the learned reward function as shown in figure 1.1. It is also categorized as supervised learning since the expert's demonstrations play a crucial role in an agent's learning. IRL problems are mostly modeled as a Markov Decision Process (MDP). In this thesis, expert's trajectories are modeled as a likelihood function. The solution of the IRL problem over a likelihood function is expected to return the reward function of the expert.

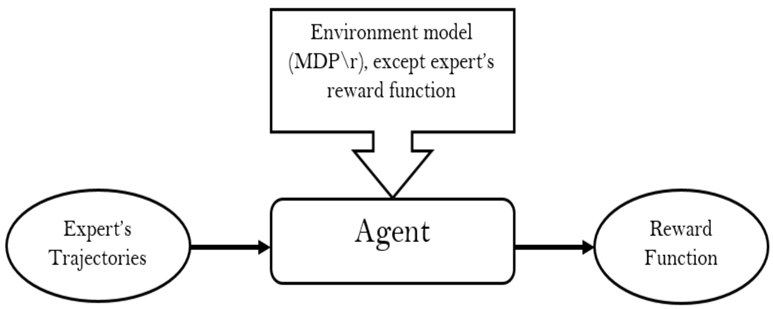

Figure 1.1: The typical framing of an Inverse Reinforcement Learning (IRL) scenario: an agent takes expert's trajectories as input and with prior knowledge of expert's environment, it tries to infer the expert's reward function.

1.2 Motivation

Motivated by a freeway merging domain involving autonomous cars, we develop a model which can be used by an autonomous car in making optimal decisions about merging onto a freeway. The industries investing in self-driving cars are highly concerned with the safety of passengers but they also want an optimal mobility of the vehicle. Researchers focusing on autonomous vehicles have raised the freeway merging problem as one of the significant unresolved challenges. The preferences of a driver for allowing a car to merge on the freeway occasionally changes depending on multiple factors. We present a novel approach to developing this preference model by maximizing the likelihood of trajectories of vehicles on rightmost lane of a freeway using our IRL algorithm. This model can be used by autonomous cars for making strategic decisions.

The complexity of manually specifying rewards in this domain urge us to prefer inverse reinforcement learning over reinforcement learning. Also, IRL helps us to learn the behavior of experts using their trajectories as input. In the freeway merging domain we have the trajectories of drivers of vehicles in the rightmost lane of the freeway and they are assumed to behave optimally in the environment. Hence, using these trajectories as input to an IRL setting, we can learn the preference model of these drivers. The modeling of the transition function in presence of stochastic human drivers in the environment may compromise the safety of passengers in autonomous cars. This inspires the need to develop a model-free approach to perform IRL.

1.3 Contribution

This thesis has three contributions:

1. Most of the previous work in the field of IRL depends heavily on the system's ability to learn transition model from a limited number of trajectories, if not available in environment model of the domain. We devise a model-free IRL approach by dropping the need for a transition function from the standard maximum likelihood IRL approach.

2. We incorporate a modified Q-learning algorithm, dubbed Q-Averaging, to remove the max operator from the canonical Q-learning algorithm. This would resolve the issue of Q-function being non-differentiable. Also, using Q-Averaging helps us eliminate the dependency of transition function without affecting the learning abilities of an agent.

3. We illustrate the validity of our algorithm on a real-world domain of freeway merging for autonomous cars. The vehicle trajectory data used for learning the behavior is extracted from NGSIM Interstate-80 freeway dataset.

1.4 Structure of Thesis

This thesis is structured as follows. In Chapter 2, we discuss a few concepts that, in general, which will make the thesis more comprehensible. It includes topics like the Markov Decision Process (MDP), Reinforcement Learning (RL), Inverse Reinforcement learning (IRL), with details about two existing IRL algorithms which are used in the body of work. Chapter 3 outlines a survey of related works in the field of IRL. Chapter 4 describes the main algorithm of the thesis. It also includes a mathematical model for our approach. The problem domains and datasets are illustrated in Chapter 5. Chapter 6 demonstrates the experiments and results of the IRL problem and their comparisons with existing methods. Finally, this document concludes in Chapter 7.

1.5 Summary

In this chapter, the focus was to give a very broad idea of the context of this work. We started by describing the problem statement and the motivation behind selecting IRL over RL and the rationale for pursing a model-free approach. In the middle of this chapter we discussed the contributions we made in this thesis. We concluded by giving the basic outline of the rest of the thesis.

Chapter 2

Background

In this chapter, we define several terms and concepts which build the foundation to comprehend the later sections of thesis. We start by defining the Markov Decision Process (MDP) in Section 2.1. In Section 2.2, we discuss the concept of maximum likelihood estimation. In Section 2.3 and Section 2.4, we describe reinforcement learning (RL) and inverse reinforcement learning respectively, followed by their closely related algorithms. The term model-free IRL is discussed significantly in Section 2.5 and the concept of gradient-based optimization is covered in Section 2.6.

2.1 Markov Decision Process

In domains of robotics and automated control systems, the problem of sequential decision making for stochastic environments is often modeled mathematically as the Markov Decision Process (MDP). Sequential decision making requires optimization to maximize the utility from agent's actions in past. MDPs are helpful in exploring optimization problems solved using reinforcement learning, inverse reinforcement learning, and many other dynamic programing techniques. An MDP is defined as tuple $\langle S, A, T, R, \gamma \rangle$, where

- S is a finite set of states.
- A is a finite set of actions.
- T is the state transition probability function, $T : S \times A \times S \to [0, 1]$

  $$T(s' \mid s, a) = P(s_{t+1} = s' \mid s_t = s, a_t = a)$$

  $T(s' \mid s, a)$ gives the probability of reaching s' from s executing action a.

- R is the reward function. Reward functions can be modeled as $R(s, a, s') : S \times A \times S \rightarrow \mathbb{R}$ or as $R(s, a) : S \times A \rightarrow \mathbb{R}$ depending upon the environment in play.

  $R(s, a, s')$ is the reward expected when an agent in state s takes an action a and lands in state s'.

  $R(s, a)$ is the immediate reward associated with the agent executing an action a being in state s.

- γ is the discount factor, parameter that determines the importance of future rewards.

$$\gamma \in [0, 1]$$

The solution of an MDP is a policy that associates an action with every state that the agent might reach. The utility of a state sequence is the sum of all the rewards over the sequence, often discounted over time. The goal is to solve the MDP to find an optimal policy that maximizes the utility of the state sequences.

The utility of a state is the expected utility of the state sequences encountered when an optimal policy is executed when starting in that state. The value iteration algorithm for solving MDPs works by iteratively solving the equations relating the utility of each state to those of its neighbors, whereas the policy iteration algorithm alternates between calculating the utilities of states under the current policy and improving the current policy with respect to the current known utilities.

2.2 Maximum Likelihood Estimation

Maximum likelihood estimation is a widely applicable statistical method of estimating unknown parameter values for fixed sets of data and a known statistical model. The likelihood of a set of data is the probability of obtaining that particular set of data, given the probability distribution model. In simple terms, it is the value of parameters which makes the observed data most probable. Maximum likelihood estimation gives a unified approach to estimation, which is well-defined in the case of the normal distribution and many other problems.

Suppose $X_1, X_2, ..., X_n$ is a sample of n independent and identically distributed (i.i.d.) observations. The assumed probability distribution depends on some unknown parameter θ. The goal of maximum likelihood estimation in this case is to find the values of unknown parameters that maximize the probabilistic likelihood of the observed data.

The joint density function of all observations can be denoted as f_θ. For an i.i.d. sample, this joint density function is

$$f_\theta(x_1, x_2, ..., x_n) = f(x_1, x_2, ..., x_n \mid \theta) = f(x_1 \mid \theta) \times f(x_2 \mid \theta) \times \cdots \times f(x_n \mid \theta) \tag{2.1}$$

In the maximum likelihood method, we represent the joint density function as likelihood function, $L(\theta)$,

$$L(\theta; x_1, x_2, ..., x_n) = \prod_{i=1}^{n} f(x_i \mid \theta) \tag{2.2}$$

The value of each $f(x_i \mid \theta)$ is a fraction and multiplying these fractions tends to reach the total value of likelihood towards zero. Rather than maximizing this product, which can be quite tedious and also could lead to extremely small value, we often use the fact that the logarithm is a monotonically increasing function, so it will be equivalent to maximize the log-likelihood:

$$L(\theta; x_1, x_2, ..., x_n) = \sum_{i=1}^{n} \log f(x_i \mid \theta) \tag{2.3}$$

The maximum likelihood estimation method the calculates the value of $\hat{\theta}$ that maximizes the value of $L(\theta)$.

$$\hat{\theta} = \arg\max_{\theta} L(\theta; x_1, x_2, ..., x_n) \tag{2.4}$$

2.3 Reinforcement Learning

Reinforcement learning (RL) is a type of machine learning technique which allows an agent to learn its behavior in order to maximize its performance. The agent does not know a priori which action to take, but instead it must explore which action yields the most reward, based on reward feedback from the environment, also known as a reinforcement signal. This behavior is adaptive in nature. If the problem is modeled with care, some RL algorithms can converge to the global optimum; this is the ideal behavior that maximizes the reward.

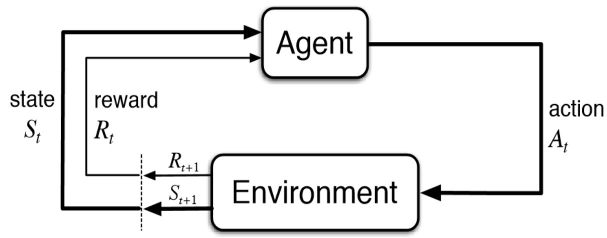

Figure 2.1: The typical framing of a Reinforcement Learning (RL) scenario: an agent takes actions in an environment, which results into a reward and a representation of the state, which are fed back to the agent [21].

Reinforcement signals are different than supervised learning. In supervised learning, an agent learns from the feedback of an expert's behavior, but such feedback is not always available. If no feedback is available, an agent can learn a transition model for its own moves and can perhaps learn to predict the opponent's moves, but the agent will have no grounds for deciding which moves to make. Reinforcement signals from the environment can be received at each time step or together at the end. For example, in games like chess, the reinforcement is received at the end, which helps agents learn what moves not to make when playing the next turn. In games like darts, each point scored is a reward and it helps in improving the agent in the next shot.

Apart from the agent and the environment, the reinforcement learning problem needs to define following four elements as well: a policy, a reward function, a value function, and a model of the environment.

A policy is mapping each state of the environment to an action taken from those states. A policy can be stochastic or deterministic. An optimal policy, usually denoted by π^*, is the best policy, i.e. one that maximizes the cumulative reward over the likelihood of all possible states.

A reward function maps each state or a state-action pair of the environment to a real number. The action selected by the policy results in the reward for that event. As the sole objective of reinforcement learning is to receive maximum reward, if the reward is poor the policy needs to be altered in order to improve the reward.

Value iteration is an algorithm used to calculate the utility of each state from the environment. The utility of a state is the immediate reward for that state plus the expected discounted utility of the next state, assuming that the agent responds according to the most optimal policy available. The value iteration algorithm helps produce an optimal policy that maximizes the accumulated reward.

The environment is modeled as stochastic finite state machine with inputs being actions sent from the agent and outputs being observations and rewards sent to the agent. MDPs are widely used for modeling sequential decision-making environments. Algorithms for solving reinforcement learning problems that use models and planning are known as model-based algorithms, whereas model-free algorithms can be conceived of as trial and error learners with no transition model or planning involved.

2.3.1 Q-learning

Learning by an agent can be passive or active. In passive learning, the agent learns the utilities of states or state-action pairs using a fixed policy. In contrast, in active learning an agent explores a model of the environment to learn how to behave by altering its policy to maximize the cumulative reward over time. Q-learning is a very popular model-free active learning technique used to solve reinforcement learning problems. A Q-learning agent learns an action-value function, Q, also known as Q-function,

$$Q : S \times A \rightarrow \mathbb{R}$$

giving the expected utility of taking an action in a given state. Q-learning is an off-policy method for Temporal Difference (TD) learning. Off-policy means that the Q-learning calculates an optimal Q-function, Q^*, and hence learns the optimal policy, π^* even when actions are selected in a more random or exploratory fashion rather than directly from the policy in play. The basic Q-update equation for Q-learning is defined as:

$$Q(s,a) \leftarrow Q(s,a) + \alpha(R + \gamma \max_{a'} Q(s',a') - Q(s,a)) \tag{2.5}$$

Equation 2.5 is calculated whenever the agent executes action $a \in A$ from state $s \in S$ and moves to $s' \in S$, receiving the reward stimulus specified in the reward function R. The Q-table is initiated with random values. Then, at each iteration, the agent selects an action and observes the reward and the next state. The action selected by agent at each step is the action that has the highest observed reward. The overall reward resulting from all the actions of agent is accumulated as the weighted sum of individual rewards at each time step. R is the immediate reward received from the behavior of the agent.

The learning schedule $\alpha \in [0,1]$, governs the magnitude of the update. If $\alpha = 0$, then the Q-function will never be updated, and if $\alpha = 1$, only the most recent information is considered.

The learning schedule $\gamma \in [0,1]$, weights the rewards of all future steps reachable from current steps. If $\gamma = 0$, it means that the agent will consider only the current rewards and neglects the future ones, while if $\gamma = 1$, the utilities might reach infinite value for non-terminating or lengthy episodes.

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
    Initialize S
    Repeat (for each step of episode):
        Choose A from S using policy derived from Q (e.g., ε-greedy)
        Take action A, observe R, S'
        $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma \max_a Q(S',a) - Q(S,A)]$
        $S \leftarrow S'$;
    until S is terminal

Figure 2.2: Q-learning algorithm for an exploratory agent [21].

2.4 Inverse Reinforcement Learning

RL problems assume that the reward function is known and fixed, but is not always the case. Stuart Russell [2] proposed the need for a technique that could achieve the same task as RL but without specifying the reward function manually, called Inverse reinforcement learning (IRL). IRL is the problem of learning the most favorable reward function with the help of an expert agent's demonstrations. In IRL, the agent that tries to learn the reward function is usually referred to as the learner, and the agent whose behavior is mimicked by learner is known as the expert. The expert is assumed to behave optimally and, hence, its demonstrations are assumed to generate maximum rewards. The learner does not have access to expert's reward function. In IRL, the environment is modeled as an MDP without the reward function, MDP\r : $< S, A, T, \gamma >$. IRL is based on Learning from Demonstrations (LfD), also known as Imitation Learning or Apprenticeship Learning (AL). Unlike AL, where the goal is to find a policy that performs like the expert, in IRL the goal is to find a reward function that is similar to that of the expert.

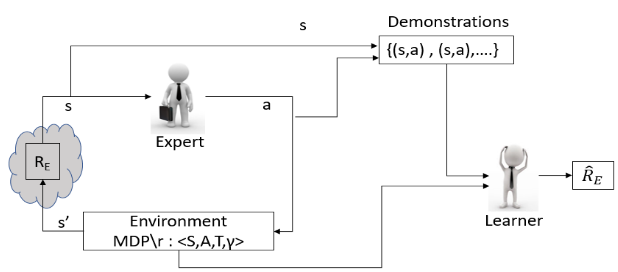

Figure 2.3: Relationship between the RL and IRL problems. The expert tries to learn the optimal policy using RL technique. The learner, however, uses the expert's trajectories (optimal policies) and infers expert's rewards using IRL technique.

Figure 2.3 illustrates the basic difference between RL problems and IRL problems. As the name suggests, IRL is essentially the inverse of RL. The input in RL problems is the reinforcement signal or reward function, R_E, and the output is policy. However, in IRL, the input is a policy or demonstrations and the output is the inferred reward function, $\hat{R}_E$. Demonstrations are assumed to maximize the reward as the expert behaves as in canonical

RL, choosing an action according to the previous rewards. The learner receives the expert's demonstrations and infers a reward function using the IRL method.

In IRL problems, the reward function is widely expressed as weighted sum of binary features [4]. $R(s,a) = \sum_i \phi_i(s,a)\theta_i$, where $\theta_i \in \mathbb{R}$ are weights and $\phi_i(s,a) \to 0, 1$ are binary feature functions for each state-action pair. However, there might be multiple reward functions that corresponds to an expert's behavior. Ng and Russell [3] proposed a solution for removing this degeneracy by formulating the IRL problem as linear program which results in a unique optimal policy.

The demonstrations are a set of trajectories, each of which is a sequence of state-action pairs recorded from expert's behavior.

$$D = \{\zeta_1, \zeta_2, .., \zeta_n\}$$

$$\zeta_i = \{(s_1, a_1)^i, (s_2, a_2)^i, ..., (s_m, a_m)^i\}$$

2.4.1 Bayesian IRL

IRL has always been seen to accomplish either of the two tasks: reward learning or apprenticeship learning. Ramachandran and Amir [9] proposed a different way to model an IRL problem using a Bayesian inferencing approach. As we discussed before, multiple reward functions might explain the expert's behavior. Bayesian IRL (BIRL), allows us to derive a probability distribution over the space of reward functions. The actions of the expert are considered as evidence and the prior knowledge on an expert's reward function can be included in the inference. BIRL relaxes the assumption that the expert always behaves optimally and that its demonstrations will produce maximum rewards.

The mathematical model for BIRL derives a posterior distribution for rewards from the prior distribution. Let us consider an agent E, operating in a $MDP :< S, A, T, \gamma >$. R is the reward function of the expert, chosen from prior distribution P_R. The demonstrations $D_E = \{(s_1, a_1), (s_2, a_2), ..., (s_m, a, m)\}$, recorded from an expert's behavior, is also given as an input to IRL problem. BIRL models the likelihood of an state-action pair given prior distribution as an exponential distribution of the Q-function.The larger the $Q^*(s, a)$, the more likely this state-action pair is in demonstration.

$$P_E((s_i, a_i) \mid R) = \frac{1}{Z_i} e^{\alpha_E Q^*(s_i, a_i, R)} \tag{2.6}$$

where, α_E is a confidence parameter that controls the expert's ability to choose the action with highest value. Similarly, the likelihood of an expert's entire demonstrations is:

$$P_E(D_E \mid R) = \frac{1}{Z} e^{\alpha_E E(D_E, R)} \tag{2.7}$$

where, $E(D_E, R) = \sum_i Q^*(s_i, a_i, R)$ and Z is a normalization constant.

Applying Bayes theorem to calculate the posterior probability of reward function R conditioned on expert's evidence,

$$\begin{aligned} P_E(R \mid D_E) &= \frac{P_E(D_E \mid R) P_R(R)}{P(D_E)} \\ &= \frac{1}{Z'} e^{\alpha_E E(D_E, R)} P_R(R) \end{aligned} \tag{2.8}$$

The normalization constant, Z', is hard to compute, hence the posterior is estimated using a sampling technique. The authors [9] use modified Markov Chain Monte Carlo (MCMC) with a uniform prior for inferencing. Now the two tasks of IRL becomes reward estimation and policy estimation from reward learning and apprenticeship learning, respectively. The reward estimation task can be achieved by minimizing the loss function, calculated as the norm distance between the actual and estimated rewards. This loss function is minimized by setting the estimated rewards as the mean of the posterior from which the actual rewards are drawn. In the case of policy estimation, the loss function is defined as the norm distance between the value of each state achieved by the optimal policy and the value of the expected policy that minimizes the loss over posterior rewards.

Ramachandran and Amir [9], were the first to propose the idea of Bayesian inferencing in IRL problems which later become the framework of many other algorithms [15, 16]

2.4.2 Maximum Likelihood IRL

Since Maximum Likelihood IRL (MLIRL) is just another approach to solve an IRL problem, the framework still remains the same. As such, the expert, learner, and environment are modeled as an $MDP :< S, A, T, \gamma >$, the expert's demonstrations $D_E = \{\zeta_1, \zeta_2 ..., \zeta_n\}$ and other IRL settings. Babes et al. [1] expressed the reward function as $R_\theta(s, a) = \theta^T \phi(s, a)$, where, θ is a set of reward weights and $\phi(s, a)$ is feature set for state $s \in S$ and action $a \in A$ pair. Since the learner is unaware of the expert's reward function, the goal of learner is to use the available information from the environment and the expert's trajectories to estimate the feature weights θ_L that mimic the values that are used to generate those demonstrations.

Algorithm : Maximum Likelihood IRL

Input: MDP$\backslash r$, features ϕ, trajectories $\{\xi_1, \ldots, \xi_N\}$, number of iterations M, step size for each iteration (t) α_t, $1 \leq t < M$.
Initialize: Choose random set of reward weights θ_1.
for $t = 1$ **to** M **do**
  Compute Q_{θ_t}, π_{θ_t}.
  $L = \sum_i \sum_{(s,a)\in\xi} \log(\pi_{\theta_t}(s, a))$.
  $\theta_{t+1} \leftarrow \theta_t + \alpha_t \nabla L$.
end for
Output: Return $\theta_A = \theta_M$.

Figure 2.4: Maximum Likelihood IRL algorithm[1].

Figure 2.4 shows the MLIRL algorithm [1], which starts by assigning a random set of values to the learner's feature weights. This helps in assigning the likelihood to the expert's trajectory. The optimization is guided by the gradient of the likelihood function at current known feature weight values θ_L.

Let us scrutinize the implementation details of the MLIRL [1] approach. First, θ_L is used to calculate the expected values discounted over horizon:

$$Q_{\theta_L}(s, a) = R_{\theta_L}(s, a) + \gamma \sum_{s'} T(s, a, s') \frac{\sum_a Q(s, a) e^{\beta Q(s,a)}}{\sum_{a'} e^{\beta Q(s,a')}} \tag{2.9}$$

The max operator from the conventional Bellman equation was making the likelihood function non-differentiable. In order to use the gradient approach for optimization of likelihood function, it needs to be differentiable. Babes et al. [1] replaces the max operator by using the Boltzmann exploration for calculating the Q-values and thus making the likelihood function differentiable.

Instead of calculating the likelihood of trajectories in [1], authors calculate the log-likelihood of trajectories as we discussed above the advantages of doing so. The log-likelihood function is defined as:

$$L(D \mid \theta) = \log \prod_{i=1}^{N} \prod_{(s,a)\in\zeta_i} \pi_\theta(s,a) = \sum_{i=1}^{N} \sum_{(s,a)\in\zeta_i} \log \pi_\theta(s,a) \tag{2.10}$$

The policy $\pi_\theta(s,a)$ is calculated using the Boltzmann exploration as:

$$\pi_\theta(s,a) = \frac{e^{\beta Q_\theta(s,a)}}{\sum_{a'} e^{\beta Q_\theta(s,a')}} \tag{2.11}$$

Thus, the solution for maximum likelihood in MLIRL [1] is expressed as:

$$\theta_L = \arg\max_\theta L(D \mid \theta) \tag{2.12}$$

Unlike other conventional IRL approaches, MLIRL resolves the issue of receiving multiple reward functions explaining the expert's optimal behavior by searching for only a single optimal reward function. MLIRL even allows to solve the IRL problems with stochastic demonstrations available from expert.

2.5 Model-Free Inverse Reinforcement Learning

IRL has solved the issue of specifying the reward function manually, but applying IRL algorithms requires an optimal policy. This optimal policy can be generated easily by solving different planning or reinforcement learning algorithms with the knowledge of demonstrations. Such algorithms are complex and could degrade the performance of high-dimensional systems with large state spaces or continuous state spaces. To overcome these limitations, an alternate method for these calculations is required, which can be achieved by creating a model-free system which can generate the policy that performs at least as well as expert policy.

Like model-free RL, IRL can also be model-free (i.e. no knowledge of transition function or planning is involved). The MDP of a model-free IRL environment looks like: $< S, A, \gamma >$. Model-free IRL approaches are very helpful in solving IRL problems where the transition model is not available. The accuracy of model-free IRL algorithms over model-based ones is still an open question. One of the model-free approaches is *Relative Entropy Inverse Reinforcement Learning* [7], where authors compare their results with those from model-based approaches. We will further discuss this approach in Section 3.1.

2.6 Summary

In this chapter, we described some concepts which will make the further parts of this thesis easy to understand. We discussed the basic concept of RL and IRL and how to model the environment using an MDP. We showed the basics of maximum likelihood estimation and the significance of the term model-free in context of both RL and IRL. We also discussed the details of BIRL and MLIRL approaches and examined the advantages of each approach.

Chapter 3

Related Work

In this chapter, we will discuss a few concepts which are not used in this work but are similar to topics underlying in this work and worth mentioning. In Section 3.1, we describe the model-free method, *Relative Entropy IRL*. Section 3.2 includes a survey of different IRL problem domains used by researchers to validate their approaches.

3.1 Relative Entropy IRL

Many approaches used to solve an IRL problem are based on the assumption that the dynamic model of the underlying MDP is known or can be learned from sampled trajectories. Learning from limited number of trajectories might be unreliable. Also, these learning methods require planning, which makes the algorithm computationally expensive and cannot be directly applicable to systems with a large or continuous state spaces. Inspired by *Relative Entropy Policy Search* [13] and based on *Maximum Entropy IRL* [6], Boularias et al. [7] proposed a model-free IRL algorithm that not only addresses the issues of learning a model from trajectories but is also able to learn good policies from a limited number of demonstrations. Relative entropy IRL [7], tries to minimize the relative entropy between the empirical distribution of the expert's demonstrations under a baseline policy and under the policy (initially arbitrary) that matches the reward feature counts of the demonstrations. The baseline policy is essentially a distribution over the set of expert trajectories. The gradient descent optimization technique used in the algorithm to minimize the relative entropy was estimated without the help of MDP. The relative entropy here is formulated as KL divergence.

The problem statement in [7] is to minimize the relative entropy which can be expressed mathematically by reformulating Maximum Entropy IRL [6] as:

$$\min_P \sum_{\tau \in \mathcal{T}} P(\tau) \ln \frac{P(\tau)}{Q(\tau)} \tag{3.1}$$

where, $\mathcal{T}$ is set of trajectories, $\mathcal{T} = \{\tau_1, \tau_2, ..., \tau_n\}$, P is probability distribution on the trajectories under current policy, and Q is the probability distribution on trajectories under a baseline policy.

The problem statement is subject to following constraints:

$$\forall i \in \{1, ..k\} : |\sum_{\tau \in \mathcal{T}} P(\tau) f_i^\tau - \hat{f}_i \leq \epsilon_i,$$

$$\sum_{\tau \in \mathcal{T}} P(\tau) = 1,$$

$$\forall \tau \in \mathcal{T} : P(\tau) \geq 0$$

where, f_i^τ is discounted feature expectation of a feature f_i along a trajectory τ, $\hat{f}_i$ is empirical expectation of feature f_i, and ϵ_i is the threshold that can be calculated using Hoeffding's bound.

The solution of the problem statement was given by Dudik and Schapire [14] as the Lagrangian function:

$$L(P, \theta, \eta) = \sum_{\tau \in \mathcal{T}} P(\tau) \ln \frac{P(\tau)}{Q(\tau)} - \sum_{i=1}^{k} \theta_i \left(\sum_{\tau \in \mathcal{T}} P(\tau) f_i^\tau - \hat{f}_i \right) - \sum_{i=1}^{k} |\theta_i| \epsilon_i + \eta \left(\sum_{\tau \in \mathcal{T}} P(\tau) - 1 \right) \tag{3.2}$$

using the Karush-Kuhn-Tucker (KKT) condition,

$$\begin{aligned} \partial_P(\tau) L(P, \theta, \eta) = \ln(P(\tau)/Q(\tau)) - \sum_{i=1}^{k} \theta_i f_i^\tau + \eta + 1 \\ = 0 \end{aligned} \tag{3.3}$$

On solving the above equation, we get:

$$P(\tau) = Q(\tau) exp \left(\sum_{i=1}^{k} \theta_i f_i^\tau - \eta - 1 \right) \tag{3.4}$$

Summing over all the trajectories on both side and solving using $\sum_{\tau \in \mathcal{T}} P(\tau) = 1$, we get the normalization constant, $Z(\theta)$

$$exp(\eta + 1) = \sum_{\tau \in \mathcal{T}} Q(\tau) exp\left(\sum_{i=1}^{k} \theta_i f_i^{\tau} \right) = Z(\theta) \tag{3.5}$$

Therefore,

$$P(\tau \mid \theta) = \frac{1}{Z(\theta)} Q(\tau) exp\left(\sum_{i=1}^{k} \theta_i f_i^{\tau} \right) \tag{3.6}$$

The dual problem resulting from the step above is to maximize the resultant dual function using sub-gradient ascent. The sub-gradient of the dual function cannot be obtained without using the transition function, which is not available. Hence, Boularias et al. [7] presents an alternate method for estimating the gradient using Importance Sampling.

The Relative Entropy IRL [7] approach was validated using three different problem domains and the results were compared with other well-known approaches. The performances of different IRL methods are compared by calculating the optimal policies using the transition function corresponding to the learned reward functions. In experiments, the relative entropy IRL approach learned the reward functions close to the expert's one in all the three problem domains using a very small number of sampled trajectories.

In contrast to Relative Entropy IRL, our approach tries to relax the assumption that the trajectories are of a fixed horizon. Boularias et al. [7] reformulate the Maximum Entropy IRL [6] as the problem of minimizing the relative entropy between the probability distribution on the trajectories and the distribution on trajectories under a baseline policy. This approach mitigates the issue of learning false reward function which might lead to same expert's policy. We model the IRL problem using maximum likelihood estimation. The issue of learning incorrect reward function is handled by maximizing the likelihood of trajectories.

3.2 Survey of Different IRL Problem Domains

A problem domain is an application that needs to be examined to solve a problem. Problem domains can be thought of as test beds on which experiments are performed or algorithms are executed, and the accuracy of solutions to that problem helps us assess the correctness of an algorithm or method used to solve that problem. Thus, problem domains play a crucial role in evidencing the authentication of an algorithm or hypothesis.

Selection criteria for a problem domain depends primarily on the algorithm used to solve it, or vice versa. For experimenting with an IRL algorithm, we must select a domain where we can have access to the behavior of an expert and partial knowledge of an experts environment. IRL problem domains can be categorized depending on their nature.

3.2.1 Synthetic Toy problems

Synthetic toy problems are not real-world problems, but are created or simulated as an environment with some goals. It's more like a toy or puzzle one can play with to achieve the goal using an IRL algorithm.

Grid World Domain

Grid world are the most commonly used problems to experiment with an IRL algorithm. Grid world represents the environment in form of $n \times n$ grids of equal dimensions mostly. Each grid represents a state, while each movement direction represents an action. Each grid is associated with a reward value (usually its negative reward each state except the goal state). The problem in this domain is to learn the reward associated with each grid from the trajectories of expert using an IRL algorithm. The expert is assumed to behave optimally, i.e. it will prefer to maximize its reward for reaching the goal state from its initial state. The learner tries to do the same and assigns reward values to each grid by learning them from an experts trajectories.

Figure 3.1. (a) shows the basic grid world problem domain. Different colors represent different reward values and are highest for the goal state and lowest for the sink state (both are terminal states), while an arrow means one of the four possible movement directions.

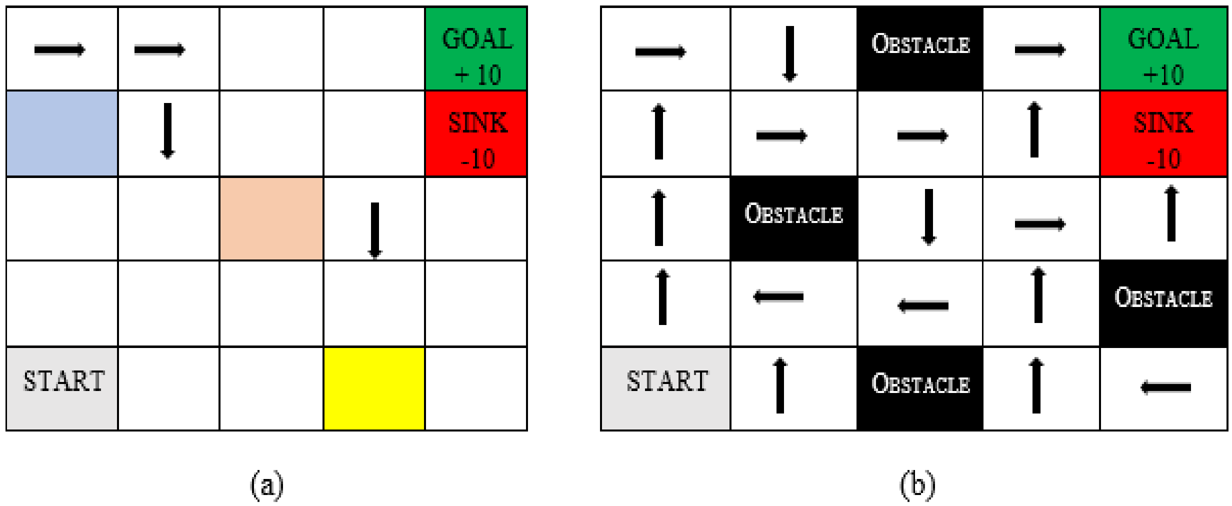

Figure 3.1: (a) Grid World Domain. An agent tries to learn an optimal policy to reach the goal state with minimum cumulative cost. Each grid color has a unique cost associated with it. (b) Gird World Domain with obstacles. The agent is not allowed to pass through obstacle states. The goal still remains the same.

Figure 3.1. (b) is the slight variation of grid world problem, where obstacles are explicitly introduced, indicating those states can never be visited by an expert. If an agent tries to move to an obstacle state, or tries to go out of the assigned grid area, it ends up in the prior state.

Mountain Car Domain

The mountain car problem is commonly used as a benchmark reinforcement learning problem to evaluate learning algorithms. In Algorithms for IRL [3], authors use the same problem to evaluate an IRL algorithm. This problem can be described as a car being placed in a valley, with the goal being to get the car out of the valley. The engine of the car is not powerful enough to drive it out of the valley. Hence, the car must build up a momentum by driving up the opposite side of the valley. The states are defined by the cars x-position, velocity, and actions, which are driving forward, backward, or neutral. The true, undiscounted, reward is

-1 per step until the car reaches the goal at the top of the hill. As in IRL algorithms, the expert is assumed to behave optimally, and the learner tries to achieve the goal by learning rewards from the experts trajectories.

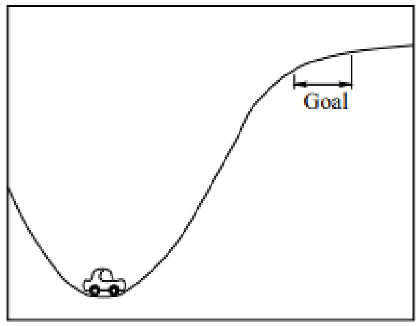

Figure 3.2: Image of mountain car problem [3]. The goal here is to get the car with insufficient engine power out of the valley. This could be achieved by building momentum using actions like driving backward, forward, or neutral. Each action has a cost associated with it.

Role-Playing Games

Role-playing games are also simulated and presented as a problem domain with a set of some experts demonstrations to learn the reward function. Ramachandran and Amir [9] applied their method of reward learning to the very famous role-playing game Dungeons and Dragons. In this game, an agent explores the dungeon, seeking to collect various items of treasure (positive rewards), while avoiding obstacles such as walls or dragons (negative rewards). The state space was represented as m-dimensional binary feature vector indicating the position of the agent and the value of various fluent. The actions are decisions made by the agent such as picking up treasure or other in-game movements.

3.2.2 Autonomous Driving Problems

Autonomous vehicles are no longer part of the realm of fiction, and to improve the efficiency and accuracy of such vehicles, their working environments are often simulated and the issues are resolved using various algorithms. Unlike in previous categories, here we try to solve some real-time issues faced by autonomous vehicles like learning to merge in lanes and driving on a highway. Here, we will discuss two of such domains being used to solve the issues using an IRL algorithm.

Freeway Merging Domain

Merging safely onto a congested freeway from a ramp is still a challenge for an autonomous vehicle in the presence of stochastic human drivers. Many researchers are trying to investigate this problem by representing the similar domains in different models, and trying to solve it using standard algorithms. We are modeling this problem as ABC car model, car B being the autonomous car, and car A and car C are human driven vehicles moving behind and ahead relative to car B, but on the rightmost lane of the freeway. Here, we are trying to solve the freeway merging problem using an IRL algorithm. Car A's trajectories are used to model the reward function, which can later be used by car B (autonomous vehicle) in decision making.

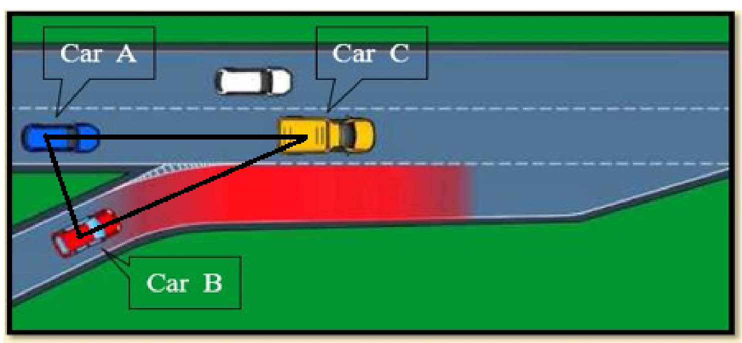

Figure 3.3: ABC model for Freeway Merging Domain. Car B is autonomous car trying to merge onto the freeway in between two human-driven cars traveling on the freeway. The goal here is for the car B to learn the preference of car A's driver and make an optimal decision about when to merge.

The state space is defined as the combination of state variables like the x-distance and velocity between car A and car C, and similarly between car A and car B. Actions are acceleration values of car A, and are discretized as full brake to full acceleration depending upon the bin it lies in. The data used as trajectories of car A is real-world data are taken from the Next Generation Simulation (NGSIM) dataset collected under the supervision of the Federal Highway Administration (FHWA). This dataset was collected from I-80 freeway in San Francisco, CA using six synchronous cameras covering over 1640 feet in length and all seven lanes, including the onramp, over three different time intervals of fifteen minutes each. All the videos from cameras were processed and the dataset is now available and ready to use in tabular format.

Highway Driving Simulator

A driving simulator is a software used to simulate and visualize (often) the real-world driving experience. Pieter and Andrew [4] used a driving simulator to learn different driving styles on highways. They considered five styles: Nice, Nasty, Right lane nice, Right lane nasty, and middle lane. The driving speed was kept constant at 56 MPH during the whole experiment and trajectories were recorded for all five different driving styles. The Markov Decision Process (MDP) of the problem had 5 actions as values, from handling the steering wheel of the vehicle, 3 of which allows driving smoothly on one of the lanes, and 2 causing the vehicle to drive off the road to avoid hitting the cars. The state space was defined indicating the current lane of the car and space between the car in front. Once the trajectories for different styles were available, the learner could mimic them using an IRL algorithm.

3.2.3 Robotics based problems

Robotics is not just about mimicking the event or performing a predefined set of operations. If a robot must perform in an unpredictable or a dynamic environment, it is nearly impossible to prepare it for all possible situations, and there might be times when an autonomous robot might find itself in a situation not considered by its designer. Robot learning allows a robot to adapt to the surrounding environment and behave optimally in unexpected circumstances. To test the learning skills of robots and to evaluate the accuracy of learning algorithm used by robots, we need problem domains which relate to ones in which robots face in the real world.

Robot Grasping Unknown Objects

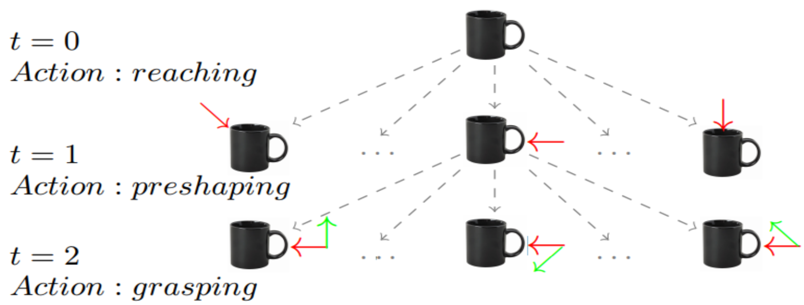

Figure 3.4: Grasping an unknown object as a Markov Decision Process. The process is represented by three steps: reaching, preshaping and grasping. The robot can move ahead at each step or can start over[5].

Boularias et al. [5], discussed the structure and observations of their experiment of a learning algorithm over a problem domain in which the robot tries to learn how to grasp an unknown object. They represented grasping an object as MDP with three steps: reaching, preshaping, and grasping. The reward of each step depends on the current state, and the robot can move ahead or restart at any step. The robot starts from the initial state at t = 0, and the set of actions corresponds to the set of points on the surface of the object. At t = 1, the state is given by a surface point and an approaching direction, the set of actions corresponds to the set of all possible hand orientations. At t = 2, the state is given by a surface point, an approach direction, and a hand orientation. Lastly, the robot either closes its finger and grasps the object, or restarts from the initial state. They used one object and six trajectories leading to a successful grasp from its handle by a robot.

Patrolling Robots

Patrolling robots are autonomous robots trained to patrol in a specified environment, assuring security of that area. These robots are designed in a way so that they can behave optimally in strange situations by learning their moves using a learning algorithm. In experimenting with the Robust IRL algorithm [17], two Turtlebots were used, one as patroller (expert) and other as intruder (learner). The patroller moves around the specified area and the learner is hidden from the sight of patroller. The trajectories of patroller are not directly available to the learner, instead the only observation available is the sound from the drones propeller. Hence, the problem is modeled as Hidden Markov Decision Process (hMDP). The state space is the location and orientation of the drone in the environment. The drone has 3 actions: going forward, turning around, and hovering. The intruder learns the patroller's policy and tries to reach its goal state without being seen by the patroller.

3.3 Summary

In the first part of this chapter, we discussed the details of Relative Entropy IRL approach. In the later part, we categorized the different type of problems which can be solved using an IRL algorithm and cataloged the few domains of each type.

Chapter 4

Maximum Likelihood Approach For Model-Free Inverse Reinforcement Learning (MLMFIRL)

In this chapter, we discuss our approach for solving an inverse reinforcement learning problem when the complete model of the environment is not available directly. We start by defining the likelihood function and its mathematical representation in Section 4.1. In section 4.2, we introduce the Q-Averaging approach to replace the conventional Q-learning equation. Details about the gradient implementation of the likelihood function are cataloged in Section 4.3. The MLMFIRL algorithm is described in Section 4.4 with its analysis in Section 4.5.

4.1 Mathematical Model for MLMFIRL

Like MLIRL [1], our approach also uses a maximum likelihood model to learn an expert's behavior and gradient method to find the optimal solution. However, unlike MLIRL our approach eliminates the dependency on the transition function and makes the method computationally efficient and more reliable for learning with a limited number of demonstrations. The step by step mathematical model of our approach is illustrated below.

The following items are given as input to MLMFIRL:

- Expert's MDP : <set of states S, set of actions A, discount factor γ >
- Expert's trajectories, $\mathcal{T} = \{\zeta_1, \zeta_2, ..., \zeta_N\}$
- Features, $\Phi = \{\phi_1, \phi_2, ..., \phi_d\}$

The goal of MLMFIRL approach is to learn the feature weight vector $\vec{\theta} :< \theta_1, \theta_2, ..., \theta_d >$ that maximizes the likelihood of the expert's trajectories. The problem statement can be expressed as:

$$\vec{\theta} = \arg\max_{\vec{\theta}} L(\vec{\theta}) \tag{4.1}$$

where, $L(\vec{\theta})$ is the log-likelihood of the trajectories in $\mathcal{T}$.

$$L(\vec{\theta}) = \log P(\mathcal{T} \mid \vec{\theta}) \tag{4.2}$$

Since all the trajectories in $\mathcal{T}$ are independent of each other given $\vec{\theta}$ and are equally likely, we can unscramble $P(\mathcal{T} \mid \vec{\theta})$ as:

$$P(\mathcal{T} \mid \vec{\theta}) = \prod_{i=1}^{N} P(\zeta_i \mid \vec{\theta}) \tag{4.3}$$

Since the expert is assumed to execute a policy that does not depend on the actions and observations of previous time step, we can apply following conditional independence rule:

$$P(\zeta_i \mid \vec{\theta}) = \prod_{(s,a)\in\zeta_i} P((s,a) \mid \vec{\theta}) \tag{4.4}$$

$P((s,a) \mid \vec{\theta})$ is the probability of taking an action $a \in A$ in state $s \in S$ given $\vec{\theta}$, i.e. policy value for (s,a) given $\vec{\theta}$. We denote the policy value for any (s,a) as $\pi_{\vec{\theta}}(s,a)$. Using equations (4.3) and (4.4) in equation (4.2) we have the log-likelihood function as:

$$L(\vec{\theta}) = \log \prod_{i=1}^{N} \prod_{(s,a)\in\zeta_i} \pi_{\vec{\theta}}(s,a) = \sum_{i=1}^{N} \sum_{(s,a)\in\zeta_i} \log \pi_{\vec{\theta}}(s,a) \tag{4.5}$$

We model $\pi_{\theta}(s,a)$ as the Boltzmann exploration policy:

$$\pi_{\vec{\theta}}(s,a) = \frac{e^{\beta Q_{\vec{\theta}}(s,a)}}{\sum_{a'} e^{\beta Q_{\vec{\theta}}(s,a')}} \tag{4.6}$$

where, β is the Boltzmann temperature, that controls the degree of confidence in agent's ability to choose actions based on Q values. The Q-value of a state-action pair, (s,a), is the optimal value which can be achieved using the conventional Q-learning equation [20]:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\big(R(s,a) + \gamma \max_{a'} Q(s',a') - Q(s,a)\big) \tag{4.7}$$

where, α is the learning schedule, γ is the discount factor, and $R(s,a)$ is the immediate reward for taking action a in state s. We define our reward function as $R(s,a) = \sum_{i=1}^{d} \theta_i \phi_i(s,a)$

For optimization we use the gradient ascent approach. The optimization is achieved by using the gradient of likelihood function at its current known feature weight values in order to update the feature weights until a locally optimal parameter value is achieved.

$$\nabla L(\vec{\theta}) = \big\{\nabla L_1(\vec{\theta}), \nabla L_2(\vec{\theta}), ..., \nabla L_i(\vec{\theta})\big\} = \left\{\frac{\partial L(\vec{\theta})}{\partial \theta_1}, \frac{\partial L(\vec{\theta})}{\partial \theta_2}, ..., \frac{\partial L(\vec{\theta})}{\partial \theta_i}\right\}$$

$$\theta_i = \theta_i + \alpha_t \nabla L_i(\theta) \tag{4.8}$$

where, α_t is step size of iteration t and $\nabla L_i(\theta)$ is gradient of likelihood function w.r.t. θ_i.

$$\nabla L_i(\vec{\theta}) = \sum_{i=1}^{N} \sum_{(s,a)\in\zeta_i} \frac{1}{\pi_{\vec{\theta}}(s,a)} \frac{\partial \pi_{\vec{\theta}}(s,a)}{\partial \theta_i} \tag{4.9}$$

Since $\pi_{\vec{\theta}}$ is a function of Q-function, we can write the partial derivative of $\pi_{\vec{\theta}}$ as:

$$\partial \pi_{\vec{\theta}}(s,a) = \frac{\partial \pi_{\vec{\theta}}(s,a)}{\partial Q(s,a)} \cdot \frac{\partial Q(s,a)}{\partial \theta_i} \tag{4.10}$$

If we can compute the gradient of the Q-function, we can use it to differentiate all of the above equations to achieve the optimal values of feature weights. However, the "max" operator in standard Q-learning (equation 4.7) makes it non-differentiable w.r.t. θ_i. This makes the gradient of the likelihood function non-differentiable and the use of the gradient ascent method for optimization impractical. We propose a method called Q-Averaging.

4.1.1 Q-Averaging

To address the issue we described above about the likelihood function being non-differentiable due to the "max" operator in equation 4.7, we propose an approach to replace the "max" operator with an average operator in equation 4.7. We call this approach as Q-Averaging because it is Q-learning with averaging.

$$Q(s,a) \leftarrow Q(s,a) + \alpha\Big(R(s,a) + \gamma \frac{\sum_{a'} Q(s',a')}{|A|} - Q(s,a)\Big) \tag{4.11}$$

where, $|A|$ is number of actions applicable in state s'.

Using equation 4.11 in our approach, makes the likelihood function differentiable. The "max" operator in equation 4.7 is responsible for selecting the action which produces the maximum Q-value in state s', i.e. the most favorable action. To support our hypothesis about replacing the standard Q-learning with Q-Averaging, we performed few experiments. We used both the standard Q-learning and the Q-Averaging approaches to solve an RL problem and compared the results for both. We performed the experiment over different RL domains like grid world, mountain car, etc., and observed that the learner achieved the similar policies but with a lower magnitude of rewards. Also, the convergence in case of Q-Averaging took more iterations than in standard Q-learning.

To conclude, the Q-Averaging approach makes the likelihood function differentiable without affecting the learning ability of learner at the cost of few more iterations than the standard Q-learning.

4.2 Gradient Implementation Details

We have likelihood function and policy from Section 4.1 as

$$L(\vec{\theta}) = \sum_{i=1}^{N} \sum_{(s,a)\in\zeta_i} \log \pi_{\vec{\theta}}(s,a)$$

$$\pi_{\vec{\theta}}(s,a) = \frac{e^{\beta Q_{\vec{\theta}}(s,a)}}{\sum_{a'} e^{\beta Q_{\vec{\theta}}(s,a')}}$$

and Q-function from Section 4.2 as:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\Big(R(s,a) + \gamma \frac{\sum_{a'} Q(s',a')}{|A|} - Q(s,a)\Big)$$

Also, we have expert's MDP:$< S, A, \gamma >$, expert's trajectories $\mathcal{T}$ and feature set Φ.

We have randomly initialize the feature weight vector $\vec{\theta}^0 :< \theta_1^0, \theta_2^0, ..., \theta_d^0 >$ and calculate the reward function as:

$$R_0(s,a) = \sum_{i=1}^{d} \theta_i^0 \phi_i(s,a)$$

For t^{th} iteration,

$\forall (s, a) \in \mathcal{T}$,

$$Q_t^0(s, a) = R_t(s, a) = \sum_{i=1}^{d} \theta_i^t \phi_i(s, a)$$
$$\frac{\partial}{\partial \theta_i} Q_t^0(s, a) = \frac{\partial}{\partial \theta_i} R_t(s, a) = \phi_i(s, a)$$

For k^{th} iteration, towards Q-value convergence:

$$Q_t^k(s, a) = Q_t^{k-1}(s, a) + \alpha \left(R_t(s, a) + \frac{\gamma}{|A|} \left[\frac{1 - \gamma^{k-1}}{1 - \gamma} \right] \sum_{a'} Q_t^{k-1}(s', a') \right) - Q_t^{k-1}(s, a)$$
$$\frac{\partial}{\partial \theta_i} Q_t^k(s, a) = \frac{\partial}{\partial \theta_i} Q_t^{k-1}(s, a) + \alpha \left(\frac{\partial}{\partial \theta_i} R_t(s, a) + \frac{\gamma}{|A|} \left[\frac{1 - \gamma^{k-1}}{1 - \gamma} \right] \sum_{a'} \frac{\partial}{\partial \theta_i} Q_t^{k-1}(s', a') - \frac{\partial}{\partial \theta_i} Q_t^{k-1}(s, a) \right)$$

$$Q_t(s, a) = Q_t^*(s, a)$$
$$\frac{\partial}{\partial \theta_i} Q_t(s, a) = \frac{\partial}{\partial \theta_i} Q_t^*(s, a)$$
$$Z_t(s) = \sum_{a'} e^{\beta Q_t(s, a')}$$
$$\pi_t(s, a) = \frac{e^{\beta Q_t(s, a)}}{Z_t(s)}$$
$$L_t(\vec{\theta}) = \sum_{i=1}^{N} \sum_{(s,a) \in \zeta_i} \log \pi_t(s, a)$$

$L_t(\vec{\theta})$ is the likelihood of trajectories in $\mathcal{T}$ after t^{th} iteration, given feature weights. Now we will apply gradient ascent approach to update the value of feature weights. To do so, we will calculate the gradient value of the likelihood function.

$$\nabla L_t(\vec{\theta}) = \left\{ \frac{\partial L_t(\vec{\theta})}{\partial \theta_1}, \frac{\partial L_t(\vec{\theta})}{\partial \theta_2}, ..., \frac{\partial L_t(\vec{\theta})}{\partial \theta_i} \right\}$$

$$\frac{\partial}{\partial \theta_i} L(\vec{\theta}) = \sum_{i=1}^{N} \sum_{(s,a) \in \zeta_i} \frac{1}{\pi_t(s,a)} \frac{\partial \pi_t(s,a)}{\partial \theta_i}$$

$$\frac{\partial}{\partial \theta_i} \pi_t(s,a) = \frac{\beta Z_t(s) e^{\beta Q_t(s,a)} \frac{\partial}{\partial \theta_i} Q_t(s,a) - e^{\beta Q_t(s,a)} \frac{\partial}{\partial \theta_i} Z_t(s)}{Z_t^2(s)}$$

$$\frac{\partial}{\partial \theta_i} Z_t(s) = \beta \sum_{a'} e^{\beta Q_t(s,a)} \frac{\partial}{\partial \theta_i} Q_t(s,a)$$

$$\forall i, \theta_i^{t+1} = \theta_i^t + \alpha_t \nabla L_i(\vec{\theta})$$

Optimal feature weight vector, $\vec{\theta}^* =< \theta_1^*, \theta_2^*, ..., \theta_i^* >$

4.3 MLMFIRL Algorithm

Algorithm 1 MF-MLIRL algorithm

```
Initialize θ⃗ : ⟨θ_1, θ_2, ..., θ_d⟩ randomly.
Initialize local variables L and L' with zero
repeat
    L ← L'
    R(s,a) = Σ_{i=1}^{d} θ_i φ_i(s,a)
    for all (s,a) ∈ {(s,a)|(s,a) ⊒ ζ_i, ζ_i ∈ τ, i ∈ {1,2,...,N}} do
        Q*(s,a) ← Q-Averaging (equation ??)
        π(s,a) = e^{βQ*(s,a)} / Σ_{a'} e^{βQ*(s,a')}
    end for
    L(θ⃗) = Σ_{i=1}^{N} Σ_{(s,a)∈ζ_i} log π(s,a)
    L' ← L(θ⃗)
    for all θ_i ∈ θ⃗ : do
        θ_i ← θ_i + α_n ∇_i L(θ⃗)
    end for
    δ = |L' − L|
until δ < ε(1 − γ)/γ
return θ⃗
```

The input to the MLMFIRL algorithm is set of expert's trajectories, environment model as MDP, features affecting the reward functions, and other controlling parameters like learning rates, step size, and the tolerance error to set convergence criteria. The algorithm uses the inputs and calculates the likelihood of expert's trajectories using the randomly initialized feature weights and optimizes them using gradient ascent approach. This process continues unless the convergence criteria are achieved. We scrutinize the algorithm piecewise below.

In the first step, we initialize the feature weight vector with random values. In the second step, we initialize two variables to store log-likelihood values of trajectories with zero. L stores the log-likelihood value from $(t-1)^{th}$ iteration and L' stores the value of likelihood calculate in t^{th} iteration. Steps 3.a to 3.f are repeated until the convergence criteria are satisfied. The rewards for each state-action pair is calculated as vector multiplication of binary features and feature vector weights. In step 3.c, the Q-values are calculated using the Q-Averaging approach followed by policy calculation using the Boltzmann policy exploration for all the state-action pair in expert's trajectory set. Using the action probability values calculated using the Boltzmann policy exploration for each state-action pair, we calculate the cumulative log-likelihood for the set of expert's trajectories. We update the $\vec{\theta}$, using the gradient ascent approach and perform the same set of operations using updates feature weights values. When the convergence criteria is satisfied, we return the learned feature weight vector that produces the closest optimal policy as of expert's. Also, the learned feature weight vector generates the maximum log-likelihood of the trajectories.

4.4 Analysis of MLMFIRL Algorithm

Analysis of an algorithm is important because by doing so we learn its characteristics needed to evaluate its functionality for various applications or compare it with other algorithms for the same application. An algorithm can be analyzed in many ways but for practical applications or comparisons, we only pay attention to the order of growth of the running time of the algorithm. That is, we learn how efficient the algorithm is mainly when the input size is large. Instead of reporting time of execution in units, we try to learn asymptotic efficiency of an algorithm, i.e. how the running time of an algorithm increases with the increase in the size of inputs. We will be analyzing our algorithm for worst case performance.

Our algorithm MLMFIRL is affected significantly by size of the set of trajectories as it is crucial in calculating the likelihood function of trajectories given current feature weights. Let N be the number of trajectories and $|\zeta_m|$ be the size of longest trajectory. The algorithm also iterates over the action space of size $|A|$. The asymptotic efficiency for worst-case performance of MLMFIRL algorithm is

$$\mathcal{O}(N|\zeta_m||A|)$$

4.5 Summary

In this chapter, we discussed the detailed mathematical model of MLMFIRL approach. We raised the non-differentiability issue with the standard Q-learning and proposed an alternate approach, dubbing Q-Averaging. Details on gradient implementation were also illustrated followed by the MLMFIRL algorithm and its analysis. Theoretically, we validated the MLIRL approach in this chapter. Experimental results and comparisons were also favorable (See Chapter 6).

Chapter 5

Domain Setup and Dataset

In this chapter, we will discuss the freeway merging domain and Next Generation SIMulation(NGSIM) I-80 dataset used for our experiments. As discussed in Chapter 1, freeway merging problem involving autonomous vehicles is the motivation behind this research. During busy hours, when the freeways are congested with vehicles, drivers of the rightmost lane have different preferences about allowing the vehicle on the on-ramp to merge. The goal of this thesis is to learn those preferences. To do so, we defined the ABC model in Section 5.1 and used trajectory data from NGSIM dataset. In Section 5.2, we will describe NGSIM program and details on metadata for I-80 dataset. Steps on extracting trajectories from one big dataset are illustrated in Section 5.3. Our environmental model for our test domain is discussed in Section 5.4.

5.1 ABC Model

The freeway merging domain as discussed in Section 3.2 is a real-world problem faced by autonomous vehicles in making decisions about when to merge, keeping in consideration stochastic behavior of human drivers on the freeway. Solving the freeway merging problem requires modeling of the traffic. Here, we model this problem using an ABC model as shown in Figure 5.1. Vehicle B is an autonomous vehicle that is about to merge onto the freeway. A is the vehicle on rightmost lane of the freeway but relatively behind B. C is also the vehicle on rightmost lane of the freeway but relatively ahead of B. The problem is that vehicle B must merge between A and C but the preferences of A's driver about allowing B to merge is unpredictable. Our objective is to model the variation in the preferences of A's driving model as it detects B using MLMFIRL settings.

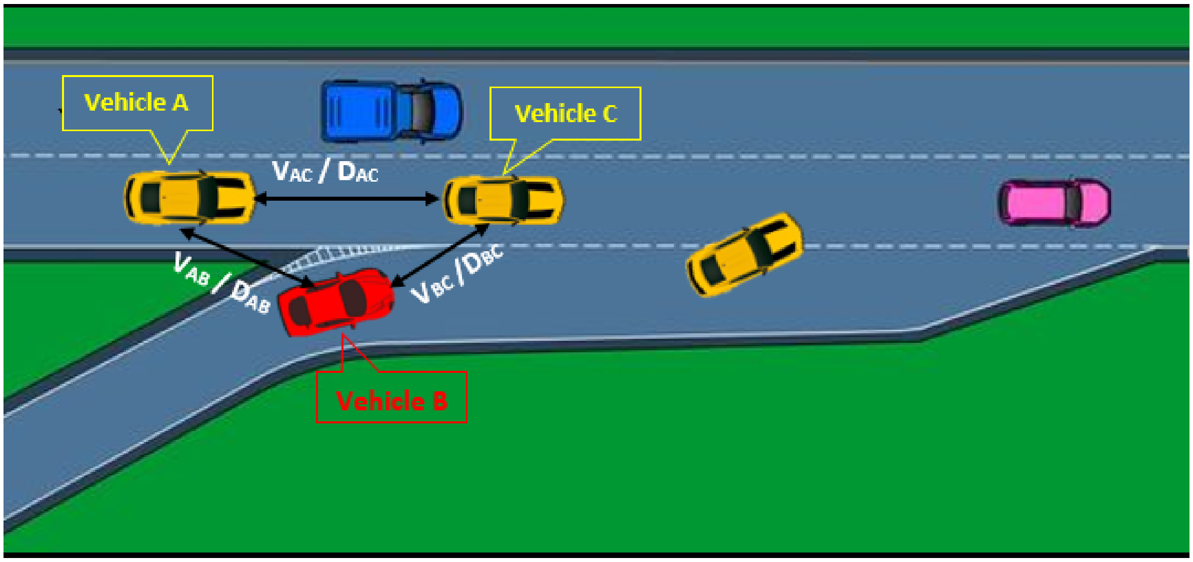

Figure 5.1: Detailed ABC model to represent the freeway merging problem. B is an autonomous vehicle about to merge onto the freeway. Relative variables like velocity and distance between any two vehicles plays crucial role in defining the state of each vehicle.

In Figure 5.1, A, B and C are vehicle fitting the characteristics of each vehicle in ABC model as discussed above. To define these vehicles, we used real-world freeway data from Interstate-80 collected under NGSIM program.

5.2 NGSIM Program and I-80 Dataset

5.2.1 The NGSIM Program

The Next Generation SIMulation (NGSIM) program was launched by United States Department of Transportation (US DOT) Federal Highway Administration (FHWA)'s Traffic Analysis Tools Program to develop algorithms in support of traffic simulation, with a primary focus on microscopic modeling. The detailed and high-quality real-world vehicle trajectory datasets collected under NGSIM turned out very useful in understanding microscopic driver behavior. Through the NGSIM program, FHWA developed several driver behavioral algorithms to describe the interaction of travelers, vehicles, and highway systems. The NGSIM products are freely available at FHWA website along with supporting documentation. The Interstate-80 (I-80) [24] freeway dataset was the first dataset collected under the NGSIM program.

5.2.2 I-80 Dataset

On April 13, 2005, the researchers for the NGSIM program collected detailed vehicle trajectory data on eastbound I80 in the San Francisco Bay area in Emeryville, CA. Seven synchronized digital video cameras were mounted on the top of a 30-story building adjacent to the freeway to record vehicle passing through over approximately 500 meters (1640 feet) in length. The study included all 6 freeway lanes and an additional onramp merging to the freeway.

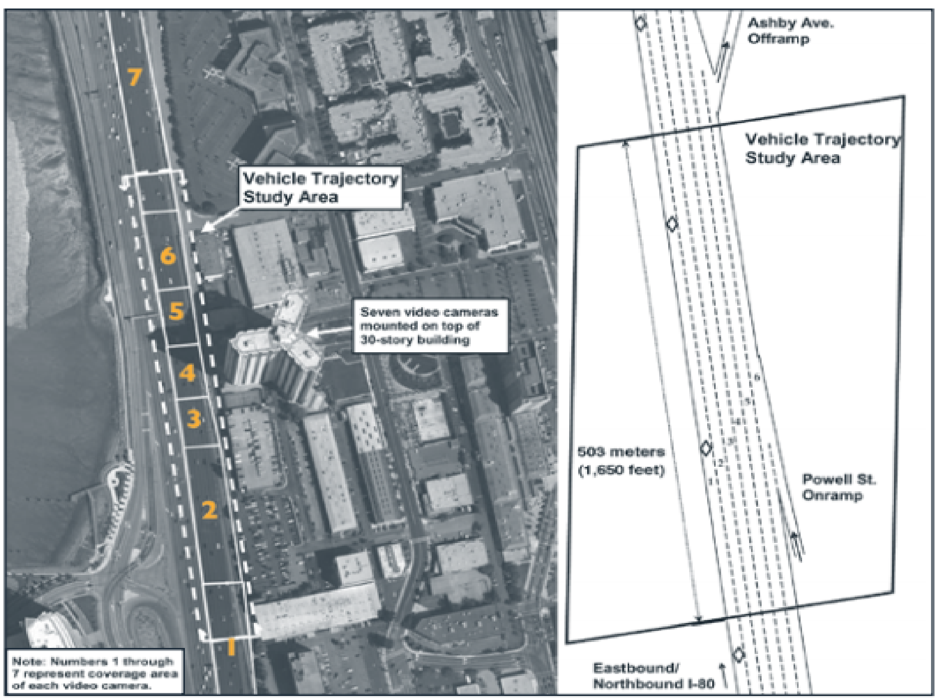

Figure 5.2: Left: The aerial photo of I-80 showing the study area covered during data collection. Right: Schematic drawing describing all the lanes of I-80 freeway including the onramp.[24]

The full I-80 freeway dataset includes total 45 minutes of data, recorded in three 15-minutes time intervals: 4:00pm - 4:15pm; 5:00pm - 5:15pm; and 5:15pm -5:30pm. These periods represent the buildup of congestion, or the transition between uncongested and congested conditions, and full congestion during the peak period.

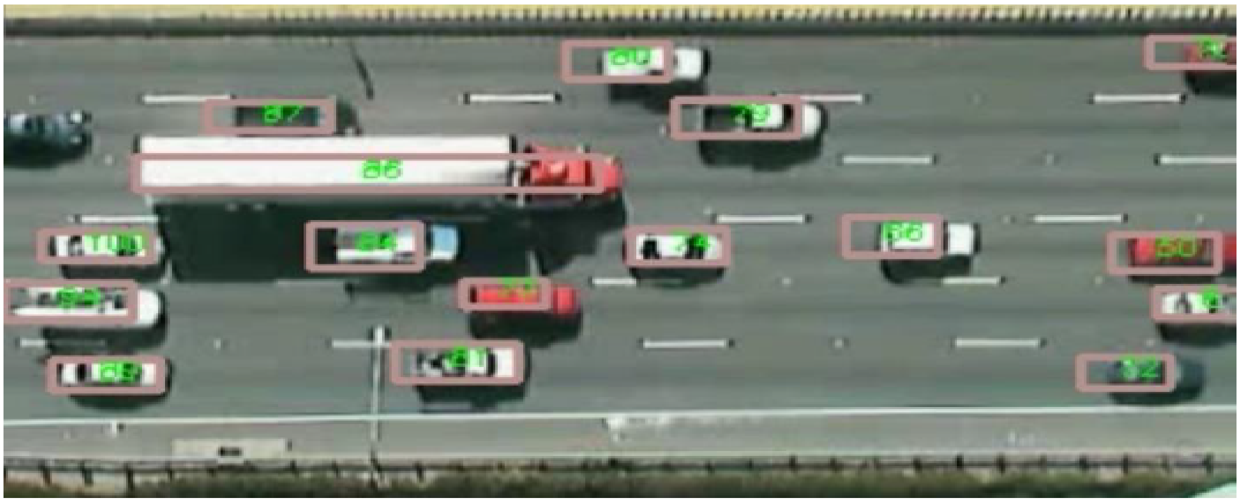

Figure 5.3: Snapshot of the processed video from NGSIM I-80 freeway merging study. The processing of video helped in detecting all the vehicles in each frame and assigned them unique IDs.

NG-VIDEO is a software application developed for the NGSIM program to transcribe the vehicle trajectory data from the video. The dataset catalogs details like location, lane position etc. for each vehicle within the study area every one-tenth of a second.

The full I-80 dataset is freely available at the NGSIM website. It includes vehicle trajectory data, computer-aided design, and geographic information system files, aerial orthorectified photos, freeway loop detector data within and surrounding the study area, raw and processed video, signal timing settings on adjacent arterial roads, traffic sign information and locations, weather data, and aggregate data analysis reports.

Metadata Details for I-80 Dataset

The fully transcribed I-80 dataset consists of around 4.5 million rows each having 18 columns. Each row is a unique tuple corresponding to 18 different useful piece of information about one vehicle in one frame, i.e. recorded every one-tenth of a second. Below are the details [25] on the significance of each column:

- Column 1: Unique vehicle identification number for each vehicle in study area ascending by the time of entry.
- Column 2: Frame ID incremented every 1/10 of a second.
- Column 3: Total number of frames in which the vehicle appears in this dataset.
- Column 4: Global Time (Epoch Time) in *milliseconds*.
- Column 5: Lateral (X) coordinate of the front center of the vehicle with respect to the leftmost edge of the section in the direction of travel in $feet$.
- Column 6: Longitudinal (Y) coordinate of the front center of the vehicle with respect to the entry edge of the section in the direction of travel in $feet$.
- Column 7: X Coordinate of the front center of the vehicle based on CA State Plane III in NAD83 in $feet$.
- Column 8: Y Coordinate of the front center of the vehicle based on CA State Plane III in NAD83 in $feet$.
- Column 9: Length of vehicle in $feet$.
- Column 10: Width of vehicle in $feet$.
- Column 11: Vehicle type: 1-motorcycle; 2: auto/car, 3: truck.
- Column 12: Instantaneous velocity of the vehicle in $feet/second$.

- Column 13: Instantaneous acceleration of the vehicle in $feet/second^2$.

- Column 14: Current lane position of vehicle. Lane 1 is the leftmost lane and lane 6 is rightmost. Lane 7 is onramp and lane 9 is right shoulder.

- Column 15: Vehicle ID of the lead vehicle in the same lane. 0 signifies now preceding vehicle.

- Column 16: Vehicle ID of the vehicle following the subject vehicle in same lane. Again, 0 means lo following vehicle.

- Column 17: Spacing provides the distance between the front-center of a vehicle to the front-center of the preceding vehicle in $feet$.

- Column 18: Headway: Headway provides the time to travel from the front-center of a vehicle (at the speed of the vehicle) to the front-center of the preceding vehicle. A headway value of 9999.99 means that the vehicle is traveling at zero speed (congested conditions) in $seconds$.

Appendix A includes the snapshots of transcribed NGSIM I-80 freeway dataset for a better understanding of vehicle trajectory data.

5.3 Vehicle Trajectory Data Extraction

The I-80 dataset received from NGSIM includes all the vehicles that passed through the study area during the time interval. For modeling freeway merging problem with ABC model, we need data for only those vehicles that fit into one of the three A, B or C vehicle roles. The following steps illustrate the extraction of useful vehicle trajectories from full dataset.

1. Select all the tuples with Column 14 values as 6 or 7, i.e. vehicles in lane 6 (rightmost lane of the freeway) or lane 7 (onramp).

2. Identify the vehicle B ID, i.e. select the vehicle which is about to merge to the freeway.

3. Identify the correct vehicle A and vehicle C with the same frame ID as that of the vehicle B. Vehicle A will be the one having Column 6 value (y-coordinate) minimum less than that for vehicle B in same frame whereas vehicle C will have Column 6 value minimum more than that for vehicle B.

4. We back propagate to get complete trajectories for all the three vehicles from the time they entered study area.

5. Join tuples of A, B, and C in same frame.

6. Now we trim the columns. For each vehicle, we only need vehicle ID, Frame ID, local Y, instantaneous velocity, instantaneous acceleration and vehicle type. Hence, we remove rest of the unwanted columns from the extracted tuples.

7. Finally, we extract all trajectories of vehicle A, with complete details of corresponding vehicle B and vehicle C in each frame.

5.4 Model Instantiation

In this section, we define the MDP model of our freeway merging environment. This model is provided as input to our IRL approach which helps the learner to predict the expert's rewards using demonstrations. Since we are modeling vehicle A's environment, all the references will be w.r.t vehicle A.

5.4.1 State Space

State space is a set of all possible states accessible to an agent in the given environment. Every state is defined using some state variables. For our model we are using following 5 state variables:

d_{AC}: Distance Between Vehicle A and Vehicle C

d_{AC} is the horizontal distance between the vehicle A and vehicle C, i.e. difference of column 6 of each vehicle.

$$d_{AC} = Y_A - Y_C$$

We use the extracted vehicle trajectory data to calculate the d_{AC} for each tuple in every trajectory. For our dataset we found the following minimum and maximum values of d_{AC}:

$$min(d_{AC}) = -690.002\,ft. \quad and \quad max(d_{AC}) = -7.703\,ft.$$

We discretized d_{AC} into 5 intervals:

- $d_{AC} < -85.000$
- $-85.000 \leq d_{AC} < -65.000$
- $-65.000 \leq d_{AC} < -50.000$
- $-50.000 \leq d_{AC} < -35.000$
- $-35.000 \leq d_{AC}$

d_{AB}: Distance Between Vehicle A and Vehicle B

This variable gives the horizontal distance between the vehicle A and vehicle B, i.e. difference of column 6 of each vehicle.

$$d_{AB} = Y_A - Y_B$$

The minimum and maximum values of d_{AB} calculate from extracted trajectory data are:

$$min(d_{AB}) = -603.358\,ft. \quad and \quad max(d_{AB}) = -0.001\,ft.$$

We discretized d_{AB} into 5 intervals:

- $d_{AB} < -45.000$

- $-45.000 \leq d_{AB} < -35.000$
- $-35.000 \leq d_{AB} < -25.000$
- $-25.000 \leq d_{AB} < -15.000$
- $-15.000 \leq d_{AB}$

v_{AC}: Relative Velocity of Vehicle A and Vehicle C

v_{AC} gives the instantaneous relative velocity of vehicle A and vehicle C, i.e. difference of column 12 of each vehicle.

$$v_{AC} = v_A - v_C$$

The minimum and maximum values of v_{AC} from extracted data are:

$$min(v_{AC}) = -36.18\, ft./sec. \quad and\, max(v_{AC}) = 32.41 \quad ft./sec.$$

We discretized v_{AC} into 5 intervals:

- $v_{AC} < -5.00$
- $-5.00 \leq v_{AC} < -2.00$
- $-2.00 \leq v_{AC} < 0.00$
- $0.00 \leq v_{AC} < 3.00$
- $3.00 \leq v_{AC}$

v_{AB}: Relative Velocity of Vehicle A and Vehicle B

This variable signifies the instantaneous relative velocity of vehicle A and vehicle B, i.e. difference of column 12 of each vehicle.

$$v_{AC} = v_A - v_B$$

The minimum and maximum values of v_{AB} from extracted trajectory data are:

$$min(v_{AB}) = -52.79\, ft./sec. \quad and \quad max(v_{AB}) = 35.06\, ft./sec.$$

We discretized v_{AB} into 5 intervals similar to those of v_{AC}:

- $v_{AB} < -5.00$
- $-5.00 \leq v_{AB} < -2.00$
- $-2.00 \leq v_{AB} < 0.00$
- $0.00 \leq v_{AB} < 3.00$
- $3.00 \leq v_{AB}$

Vehicle Type

This variable determines the type of vehicle B for each vehicle A. This variable is crucial as the driving preferences of vehicle A's driver usually changes depending upon the type of vehicle trying to merge. For example, a normal human driver might allow a car type vehicle to merge but might not want to get behind a truck, especially during heavy traffic conditions. The type of vehicle can directly be determined from value of column 11 in dataset. We merged the motorcycle and auto/car vehicle types into same category. Hence vehicle type can either be 0, i.e. car or motorcycle, or 1, i.e. truck.

Using all the five state variables we can define the state of vehicle A at any instant of time. With 5-5 intervals of d_{AC}, d_{AB}, v_{AC} and v_{AB} and 2 unique values of vehicle type, we have a state space of 1250 states.

5.4.2 Action Space

The instantaneous acceleration values are modeled as actions of the driver. These values are directly available from dataset via column 13 of each tuple. The minimum and maximum value of accelerations from datasets are:

$$min(acc) = -11.20\, ft/sec^2. \quad and \quad max(acc) = 11.2\, ft./sec^2.$$

We discretized the acceleration into five intervals and named them as the following actions:

- High Brake: $-11.20 \leq acc \leq -4.80$
- Low Brake: $-4.79 \leq acc \leq -0.60$
- Zero Acceleration: $-0.59 \leq acc \leq 0.59$
- Low Acceleration: $0.60 \leq acc \leq 4.79$
- High Acceleration: $4.80 \leq acc \leq 11.2$

5.4.3 Feature Space

An agent is assumed to behave optimally in IRL setting, which means that the action sequence of an agent is the result of some parameters that affect the cumulative rewards of the agent. Considering those parameters, we accounted 3 binary features to our model. Any feature is considered active with the value 1 and inactive when the value is 0.

- Feature 1: *Safe.* This feature is inactive only when $d_{AC} > 20\, ft.$ and $acc > 0.6\, ft./sec^2$, i.e. distance from the preceding vehicle is less than 20 ft and vehicle is accelerating. This feature signifies the preference of being safe when active.
- Feature 2: *Time to travel.* This feature is active when $acc > -0.6\ ft./sec^2$, i.e. either the vehicle is accelerating or moving with a constant speed. This feature signifies the importance of time to reach destination.
- Feature 3: *Type of vehicle B.* This feature is active when the vehicle B is a truck.

5.5 Summary

In this Chapter, we defined the characteristics of ABC model for solving a freeway merging problem. We discussed the importance of the freely available Interstate-80 freeway dataset collected under NGSIM program by FHWA in the microscopic modeling of traffic. The vehicle trajectory data required to solve the freeway merging problem using an IRL approach can be extracted as per our requirements. In the last section, we illustrated details of our experimental model.

Since we have discussed the MLMFIRL approach and covered sufficient details on domain setup and dataset, we will be presenting some experimental results with analysis and comparisons in next chapter.

Chapter 6

Experimental Evaluation

In this chapter, we will analyze our model-free MLMFIRL approach with the help of experimental results and compare them with the modeled MLIRL [1] technique. In Section 6.1, we evaluate the performance of both approaches in a grid world environment. In Section 6.2, we show that our approach for solving the freeway merging problem produces more satisfactory results than Babe et al.'s [1] approach. We also justify the validity of our algorithm with the help of few qualitative evaluations in section 6.3.

6.1 Grid World

For evaluating our MLMFIRL approach and comparing it with MLIRL, I used the BURLAP [26] implementation of the grid world environment with grid size 5×5. Figure 6.1 depicts the graphical user interface for our grid world environment. The gray colored circle is an agent which can move along the 25 states using 4 actions. The five different color grids signify the unique location features.

The MDP model for the grid world environment includes 25 states, 4 actions, and the discount factor of 0.99. The 5 location features were initiated with random weights to generate a reward matrix for grid world. We used the Boltzmann temperature (β) as 10.

We recorded 10 trajectories by moving the agent around the grids. These trajectories are used to recover the rewards for each grid using both MLIRL and MLMFIRL approaches. Results for MLIRL were recorded using the BURLAP [26] implementation of MLIRL algorithm. Below are the mean and standard deviation of results recorded from multiple executions using both approaches with same MDP model and demonstrations.

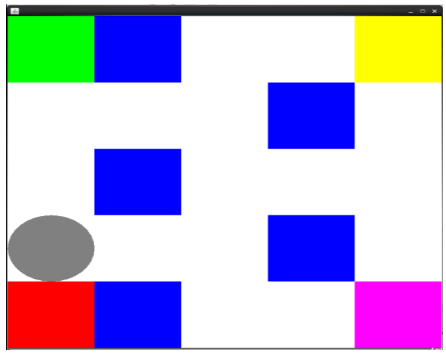

Figure 6.1: The graphical user interface for grid world environment used to demonstrate our approach. The gray circle is an agent exploring the 5×5 grid. Each different color grid represents a unique cost of reaching to that state. The agent tries to learn the cost associated each grid using the expert's trajectories.

Table 6.1 shows the mean and standard deviations of learned feature weights and corresponding maximum log-likelihood values generated using MLMFIRL and MLIRL algorithms. To analyze the results, we compare the mean maximum log-likelihood values of both the approaches. For MLMFIRL, the mean maximum log-likelihood value is -9170.868 with a standard deviation of 195.940. For MLIRL, the mean maximum log-likelihood value is -24142.833 with a standard deviation of 1338.954. Hence within a fixed number of iterations, MLMFIRL not only outperforms MLIRL but also produces more consistent results.

The ideal maximum log-likelihood value is expected to be 0. Here, the main reason for getting high log-likelihood values is because the trajectories we produced does not corresponds to an ideal behavior. We randomly move the agent on grids to produce the trajectories. If we use all the trajectories demonstrating the same behavior, we get lucky to achieve ideal results.

We have analyzed the results using descriptive statistics like mean and standard deviation that describes the results but gives no clue about the significance of results. Hence, we can not generalize the results. We here use another statistical approach known as T-Test to get

Approach	Learned feature weight vector $\vec{\theta} = \langle\theta_1, \theta_2, \theta_3, \theta_4, \theta_5\rangle$ Mean ± Standard Deviation	Maximum log-likelihood Mean ± St. Dev.
Model-Free MLIRL	$\langle -20.979 \pm 0.491, -15.370 \pm 0.244, -15.488 \pm 0.176, -14.740 \pm 0.134, -14.381 \pm 0.212\rangle$	-9170.868 ± 195.941
Model-based MLIRL	$\langle -31.809 \pm 0.530, -20.710 \pm 10.999, -12.010 \pm 10.115, -8.068 \pm 2.535, -16.073 \pm 17.406\rangle$	-24142.883 ± 1338.954

Table 6.1: Comparison of learned feature weights and corresponding maximum log-likelihood values of trajectories for grid world domain using MF-MLIRL and MLIRL algorithms.

the inferential significance of our results. Inferential statistical approaches not only describes our data but also generalizes the results.

Each T-Test has a p-value attached to it. P-value is the probability that the pattern produced by our data could be produced by random data. If $p < 0.05$, results are considered significant. We applied the T-Test to the set of log-likelihood values recorded using MLMFIRL and MLIRL approaches. The resultant p-value was:

$$p = 0.00000000005914$$

Since the p-value was less than 0.05, we conclude that the MLMFIRL approach produces significantly better results than MLIRL approach.

6.2 Freeway Merging Problem

To solve the freeway merging problem, we model the environment and use the extracted NGSIM dataset as illustrated in Chapter 5. Below is the complete details of the experimental setup.

- $MDP :< S, A, \gamma > = < 1250, 5, 0.99 >$
- Features $\Phi = \{\phi_1, \phi_2, \phi_3\}$

Approach	Learned feature weight vector $\vec{\theta} = \langle \theta_1, \theta_2, \theta_3 \rangle$ Mean ± Standard Deviation	Maximum log-likelihood Mean ± St. Dev.
MF-MLIRL	$\langle 0.845 \pm 0.093, 7.975 \pm 0.103, 0.297 \pm 0.180 \rangle$	-47194.575 ± 0.699
MLIRL	$\langle 1.525 \pm 0.052, 17.063 \pm 0.508, 0.062 \pm 0.055 \rangle$	-51055.275 ± 284.681

Table 6.2: Comparison of learned feature weights and corresponding maximum log-likelihood values of trajectories for the freeway merging domain using model-free and model-based algorithms.

- $\mathcal{T} = \{\zeta_1, \zeta_2, ..., \zeta_{260}\}$, i.e. set of 260 expert's trajectory extracted from I-80 NGSIM dataset.
- Boltzmann temperature, $\beta = 0.01$
- Learning rate for Q-Averaging, $\alpha = 0.1$
- Variable step size for gradient ascent.

We used the same setup and the same trajectories to learn the preference models of real drivers of vehicle A on the freeway using two different IRL approaches, modeled MLIRL and model-free MLMFIRL. We recorded our results, below, followed by detailed comparison.

Table 6.2 exhibits the learned feature weights and corresponding maximum log-likelihood values of trajectories using our model-free IRL approach and previously existing MLIRL approach. The mean maximum log-likelihood value for MLMFIRL over multiple executions is -47194.575 with a standard deviation of 0.699. However, the mean maximum log-likelihood value for MLIRL is -51055.275 with a standard deviation of 284.681 which is less than that of MLMFIRL and more varied.

Despite the fact that our approach does not produce ideal log-likelihood, the figures are better than what we get from MLIRL. Also, our model-free approach is more reliable as the learning procedure of transition function for freeway merging domain is questionable.

Figure **??** illustrates the algorithm we used to learn the transition model of freeway merging domain which uses sampling method which is not ideal in the environment of human drivers. Also, learning the transition function using limited trajectories for sampling could be highly inaccurate.

We applied T-Test to samples of log-likelihood values received using MLMFIRL and MLIRL approaches. The resultant p-values was:

$$p = 0.000000083894$$

Since the p-value was less than 0.05, we conclude that the MLMFIRL approach produces significantly better results than MLIRL approach.

Algorithm 2 State Transition Probability

```
for trajectory in all trajectories do
  for t in trajectory do
    s ← current state if trajectory[t][state]
    for a in all actions do
      s' ← sample 100 next states with SampleNextState(t,a)
      vf[s, a, s'] ← visitation frequency of s, a, s'
    end for
    for a in all actions do
      for s' in all states do
        tp(s, a, s') ← vf(s, a, s')/sum(vf[s, a, :])
      end for
    end for
  end for
end for
return tp
SampleNextState(t,a)
s ← trajectories[t][state]
s' ← trajectories[t+1][state]
a ← a + Noise
s'[v_AC] ← s[v_AC] + a * 0.1
return s'
```

The State Transition Probability algorithm is used to recover the transition model of freeway merging problem domain using NGSIM I-80 vehicle trajectories. To recover the unknown state transition model, we traverse through all the states in each trajectory sequence. At each time step in a trajectory, we take the current state and we sample 100 next states for each action executed from current step. We then record the number of transition of (s, a, s') tuple as visitation frequency of the resultant next state. Once we have the visitation frequencies of all next states from each current state in state space and each action in action space, we calculate the transition probabilities for each (s, a, s') by normalizing their visitation frequencies. The next steps are sampled based on motion model calculation in probabilistic robotics[23].

6.3 Qualitative Evaluation

To reinforce the validity of our approach we also performed few supplementary experiments where the output was deterministic. The variation of experiments was carried on the type of trajectories given as input. We first categorized the trajectories into sets with drivers of vehicle A demonstrating the similar behavior in each set. Then, we used our MLMFIRL approach to learn the behavior using similar demonstrations and analyzed the results. Also, we used the same set of results and tried to learn the expert's behavior using MLIRL approach and compared the results with our approach. All the trajectories used for qualitative evaluation are extracted from NGSIM I-80 freeway dataset.

Evaluation I

In the first evaluation, we focused on trajectories where drivers of vehicle A prefers to demonstrate safe driving behavior, i.e. maintaining enough distance from the preceding car and not accelerating. We selected the trajectories that signify safe behavior, even when the driver detects vehicle B as a truck. When the distance from preceding car is too long we found few time steps where vehicle A does accelerates, but the majority of times it prefers being safe.

The first set of column in Table 6.3 shows the learned feature weights and corresponding maximum log-likelihood using MLMFIRL and MLIRL approaches. Here we analyze the results at two-fold. First, the feature weights corresponding to safe driving feature dominates

	Approach	Learned feature weight vector $\vec{\theta} = \langle \theta_1, \theta_2, \theta_3 \rangle$ Mean ± Standard Deviation	Maximum log-likelihood Mean ± St. Dev.
QE I	MF-MLIRL	$\langle 5.955 \pm 0.001, 0.527 \pm 0.00, 2.814 \pm 0.001 \rangle$	-2314.667 ± 0.003
	MLIRL	$\langle 10.250 \pm 0.392, 6.724 \pm 0.304, 8.909 \pm 0.667 \rangle$	-2557.134 ± 3.238
QE II	MF-MLIRL	$\langle 8.472 \pm 0.386, 86.798 \pm 0.008, 9.508 \pm 0.225 \rangle$	-428.822 ± 0.034
	MLIRL	$\langle 7.792 \pm 0.275, 87.719 \pm 6.085, 13.103 \pm 0.488 \rangle$	-546.791 ± 3.162
QE III	MF-MLIRL	$\langle 0.659 \pm 0.292, 20.729 \pm 0.004, 22.809 \pm 0.011 \rangle$	-704.670 ± 0.041
	MLIRL	$\langle 0.998 \pm 0.387, 19.174 \pm 1.828, 29.025 \pm 1.929 \rangle$	-821.379 ± 5.735

Table 6.3: Qualitative evaluation results for MF-MLIRL and MLIRL. QE I corresponds to trajectories demonstrating safe driving. QE II includes trajectories where drivers tend to accelerate in order to reach the destination quickly. QE III is modeling the preferences of drivers when vehicle B is a truck.

the other weights, i.e. the trajectories demonstrate the safe driving behavior. Second, the MLMFIRL approach generates better maximum log-likelihood values than MLIRL approach.

Evaluation II

In the second evaluation, we tried to learn the behavior of drivers using trajectories demonstrating the preference of accelerating in order to reach the destination on time. These trajectories correspond to our second feature, i.e. travel time. Since we used the real drivers' data, finding the trajectories with accelerating actions for each time steps was unrealistic. Hence, we preferred those trajectories that most fit the behavior.

On analyzing the results from the middle rows of Table 6.3, the feature weight values corresponding to the second feature, i.e. travel time to reach the destination is higher than the other feature weights (as expected). Also, the maximum log-likelihood value for this set of trajectories is better in case of MLMFIRL than MLIRL.

Evaluation III

In our third evaluation, we modeled the preferences of vehicle A's drivers when they detect vehicle B as a truck. This evaluation is important as the driving preferences of drivers usually changes when the merging vehicle is a big truck. Drivers tend to accelerate and go ahead of big vehicles.

The last set of columns in Table 6.3 shows the learned feature weights and corresponding maximum log-likelihood for the third set of trajectories using MLMFIRL and MLIRL approaches. As the previous two, here also we analyze the results at two-fold. First, the feature weights corresponding to the third feature, i.e. the type of vehicle B dominates the other two feature weights. Also, as expected the feature weights corresponding to acceleration is higher than the safe driving feature. Second, the MLMFIRL approach generates better maximum log-likelihood values than MLIRL approach.

6.4 Summary

In this chapter, we practically evaluated our model-free MLMFIRL approach for both grid world toy problem domain and the real-world freeway merging problem domain. The results from both the domain were remarkable. We also evaluated our approach using some special test trajectories for a deeper understanding of its functionality. The comparisons illustrated above infers that our approach is better than MLIRL and also more reliable for environments with unknown transition model.

Chapter 7

Conclusion And Future Work

In this thesis, we propose a novel inverse reinforcement learning approach to resolve the issues of existing techniques. Learning the behavior of an expert with complete knowledge of the environment has been solved in contemporary literature. Here, we successfully learn the behavior of expert with only partial knowledge of its environment. For real-world environments, like the one we discussed, it is not easy to learn the transition model accurately. Our solution entirely eliminates the dependency on the transition function from learning via an expert's trajectories, making the approach model-free. In order to accomplish our desired goal, we apply some renowned techniques including maximum likelihood estimation, Q-learning, and gradient ascent. Additionally, to address some mathematical challenges, we introduced some alterations in canonical approaches and justified them.

We showed that MLMFIRL is effective in recovering the expert's reward, even with a limited number of expert's trajectories, outperforming existing IRL algorithm in a grid-world environment. The Q-values for each state-action pair in trajectory set is calculated using the Q-Averaging technique which is then used to produce the optimal action probabilities using the Boltzmann policy exploration technique. We used gradient ascent to iteratively update the feature weight values in the direction of the gradient of the log-likelihood of expert's trajectory. To summarize, the MLMFIRL algorithm is simple, easy to implement, time efficient and even space efficient.

MLMFIRL demonstrates promising results for the freeway merging problem domain using the NGSIM I-80 dataset. The learned model can be used in self-driving cars to make an optimal decision about when to merge by monitoring behavior of cars on rightmost lane of the freeway. The evaluation and comparison of results with other contemporary techniques are significant. Using MLMFIRL, we are not only able to produce better learning results but also more reliable results from a limited number of trajectories.

Future work may include implementing the MLMFIRL approach with other optimization techniques which do not require differentiating the likelihood function. This will allow the use of conventional Q-learning with the “max” operator. Additionally, an avenue may be to replace the “max” operator entirely.

We described the MLMFIRL algorithm for a single expert setting. It would be interesting to predict the behavior in presence of multiple agents. Modeling of a multi-agent environment and their interaction with other experts in the environment might lead to better learning of behavior.

In experimental settings, applying the MLMFIRL approach to a larger dataset collected from different demographic locations is expected to yield even better results. Recent advancements in the field of inverse reinforcement learning and maximum likelihood estimation can be integrated to make the algorithm more efficient and scalable.

Bibliography

[1] Babes, Monica, Vukosi Marivate, Kaushik Subramanian, and Michael L. Littman. "Apprenticeship learning about multiple intentions." *In Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 897-904. 2011.

[2] Russell, Stuart. "Learning agents for uncertain environments." *In Proceedings of the eleventh annual conference on Computational learning theory*, pp. 101-103. ACM, 1998.

[3] Ng, Andrew Y., and Stuart J. Russell. "Algorithms for inverse reinforcement learning." In *Icml*, pp. 663-670. 2000.

[4] Abbeel, Pieter, and Andrew Y. Ng. "Apprenticeship learning via inverse reinforcement learning." In *Proceedings of the twenty-first international conference on Machine learning*, p. 1. ACM, 2004.

[5] Boularias, Abdeslam, Oliver Krmer, and Jan Peters. "Structured apprenticeship learning." *Machine Learning and Knowledge Discovery in Databases* (2012): 227-242.

[6] Ziebart, Brian D., Andrew L. Maas, J. Andrew Bagnell, and Anind K. Dey. "Maximum Entropy Inverse Reinforcement Learning." In *AAAI*, vol. 8, pp. 1433-1438. 2008.

[7] Boularias, Abdeslam, Jens Kober, and Jan Peters. "Relative entropy inverse reinforcement learning." In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 182-189. 2011.

[8] Ho, Jonathan, Jayesh Gupta, and Stefano Ermon. "Model-free imitation learning with policy optimization." In *International Conference on Machine Learning*, pp. 2760-2769. 2016.

[9] Ramachandran, Deepak, and Eyal Amir. "Bayesian inverse reinforcement learning." *Urbana* 51, no. 61801 (2007): 1-4.

[10] Scholz, F. W. "Maximum likelihood estimation." *Encyclopedia of statistical sciences.* Wiley Online Library (1985).

[11] Neu, Gergely, and Csaba Szepesvri. "Apprenticeship learning using inverse reinforcement learning and gradient methods." In *Conference on Uncertainty in Artificial Intelligence*, pp. 31-46. 2007.

[12] Choi, Jaedeug, and Kee-Eung Kim. "Inverse reinforcement learning in partially observable environments." *Journal of Machine Learning Research* 12, no. Mar (2011): 691-730.

[13] Peters, Jan, Katharina Mlling, and Yasemin Altun. "Relative Entropy Policy Search." In *AAAI*, pp. 1607-1612. 2010.

[14] Dudk, Miroslav, and Robert E. Schapire. "Maximum entropy distribution estimation with generalized regularization." In *Proc. COLT*, pp. 123-138. 2006.

[15] Choi, Jaedeug, and Kee-Eung Kim. "Nonparametric Bayesian inverse reinforcement learning for multiple reward functions." In *Advances in Neural Information Processing Systems*, pp. 305-313. 2012.

[16] Lopes, Manuel, Francisco Melo, and Luis Montesano. "Active learning for reward estimation in inverse reinforcement learning." In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 31-46. Springer, Berlin, Heidelberg, 2009.

[17] Shahryari, Shervin. "Inverse reinforcement learning under noisy observations (Robust IRL)." *Masters' Thesis, The University of Georgia.* 2016.

[18] Das, Indrajit. "Inverse reinforcement learning of risk-sensitive utility." *Masters' Thesis, The University of Georgia.* 2016.

[19] Watkins, Christopher JCH, and Peter Dayan. "Q-learning." *Machine learning* 8, no. 3-4 (1992): 279-292.

[20] Russell, Stuart, and Peter Norvig. *Artificial Intelligence: A modern approach*, third edition.Upper Saddle River (2010).

[21] Sutton, Richard S., and Andrew G. Barto.*Reinforcement learning: An introduction.* Vol. 1, no. 1. Cambridge: MIT press, 1998.

[22] Puterman, Martin L. *Markov decision processes: discrete stochastic dynamic programming.* John Wiley & Sons, 2014.

[23] Thrun, Sebastian. "Probabilistic robotics."*Communications of the ACM* 45, no. 3 (2002): 52-57.

[24] United State Department of Transportation (US DOT)."Fact sheet Interstate-80 freeway dataset." *Federal HighWay Administration(FHWA)*, 2006.

[25] Federal HighWay Administration(FHWA)."Vehicle Trajectory File Data Dictionary." *Next Generation Simulation (NGSIM) Program*, 2006.

[26] MacGlashan, James. "The Brown-UMBC Reinforcement Learning and Planning (BURLAP)." *Brown University, University of Maryland, Baltimore County*, August 2017. http://burlap.cs.brown.edu/

Appendix A

NGSIM I-80 Dataset

As described in Section 5.2.2, I-80 dataset was collected under NGSIM program by FHWA in 2005 using seven synchronized digital cameras. The recoded video data was then transcribed into vehicle trajectory data using NG-VIDEO, a customized software application developed for NGSIM program. Below are few snapshots of transcribed I-80 freeway merging dataset.

The complete NGSIM I-80 dataset consists of approximately 4.5 million rows. Each row is a unique combination of 14 columns. We illustrate the detailed significance of each column values in Section 5.2.2.

1	13	884	1113433136200	16.938	49.463	6042842.012	2133118.909	14.3	6.4	2	12.50	0.00	2	0	0	0.00	0.00
1	14	884	1113433136300	16.991	50.712	6042841.908	2133120.155	14.3	6.4	2	12.50	0.00	2	0	0	0.00	0.00
1	15	884	1113433136400	17.045	51.963	6042841.805	2133121.402	14.3	6.4	2	12.50	0.00	2	0	0	0.00	0.00
1	16	884	1113433136500	17.098	53.213	6042841.701	2133122.649	14.3	6.4	2	12.50	0.00	2	0	0	0.00	0.00
1	17	884	1113433136600	17.151	54.463	6042841.597	2133123.895	14.3	6.4	2	12.49	-0.09	2	0	0	0.00	0.00
1	18	884	1113433136700	17.204	55.712	6042841.493	2133125.142	14.3	6.4	2	12.48	-0.08	2	0	0	0.00	0.00
1	19	884	1113433136800	17.257	56.956	6042841.389	2133126.389	14.3	6.4	2	12.52	0.55	2	0	0	0.00	0.00
1	20	884	1113433136900	17.330	58.199	6042841.306	2133127.628	14.3	6.4	2	12.67	2.21	2	0	0	0.00	0.00
1	21	884	1113433137000	17.289	59.463	6042841.107	2133128.871	14.3	6.4	2	13.00	4.43	2	0	0	0.00	0.00
1	22	884	1113433137100	17.197	60.776	6042840.852	2133130.155	14.3	6.4	2	13.49	5.64	2	0	0	0.00	0.00
1	23	884	1113433137200	17.027	62.157	6042840.510	2133131.507	14.3	6.4	2	13.98	4.77	2	0	0	0.00	0.00
1	24	884	1113433137300	16.863	63.592	6042840.166	2133132.922	14.3	6.4	2	14.36	2.84	2	0	0	0.00	0.00
1	25	884	1113433137400	16.752	65.054	6042839.872	2133134.362	14.3	6.4	2	14.58	1.17	2	0	0	0.00	0.00
1	26	884	1113433137500	16.731	66.526	6042839.670	2133135.799	14.3	6.4	2	14.64	0.08	2	0	0	0.00	0.00
1	27	884	1113433137600	16.682	67.993	6042839.442	2133137.232	14.3	6.4	2	14.62	-0.40	2	0	0	0.00	0.00
1	28	884	1113433137700	16.632	69.455	6042839.214	2133138.665	14.3	6.4	2	14.59	-0.54	2	0	0	0.00	0.00
1	29	884	1113433137800	16.583	70.913	6042838.987	2133140.098	14.3	6.4	2	14.52	-0.69	2	0	0	0.00	0.00
1	30	884	1113433137900	16.528	72.367	6042838.753	2133141.539	14.3	6.4	2	14.38	-1.71	2	0	0	0.00	0.00
1	31	884	1113433138000	16.478	73.809	6042838.525	2133142.971	14.3	6.4	2	14.13	-3.49	2	0	0	0.00	0.00
1	32	884	1113433138100	16.453	75.210	6042838.328	2133144.363	14.3	6.4	2	13.75	-4.36	2	0	0	0.00	0.00
1	33	884	1113433138200	16.462	76.559	6042838.171	2133145.699	14.3	6.4	2	13.37	-3.51	2	0	0	0.00	0.00
1	34	884	1113433138300	16.495	77.867	6042838.044	2133146.993	14.3	6.4	2	13.11	-1.72	2	0	0	0.00	0.00
1	35	884	1113433138400	16.534	79.159	6042837.924	2133148.279	14.3	6.4	2	13.00	-0.30	2	0	0	0.00	0.00
1	36	884	1113433138500	16.567	80.454	6042837.797	2133149.573	14.3	6.4	2	12.98	0.16	2	0	0	0.00	0.00
1	37	884	1113433138600	16.601	81.754	6042837.671	2133150.867	14.3	6.4	2	13.00	0.07	2	0	0	0.00	0.00
1	38	884	1113433138700	16.634	83.054	6042837.544	2133152.162	14.3	6.4	2	13.00	0.00	2	0	0	0.00	0.00
1	39	884	1113433138800	16.667	84.354	6042837.417	2133153.456	14.3	6.4	2	13.00	0.00	2	0	0	0.00	0.00
1	40	884	1113433138900	16.701	85.654	6042837.291	2133154.750	14.3	6.4	2	13.00	0.00	2	0	0	0.00	0.00
1	41	884	1113433139000	16.734	86.954	6042837.164	2133156.044	14.3	6.4	2	13.00	0.00	2	0	0	0.00	0.00
1	42	884	1113433139100	16.767	88.254	6042837.037	2133157.339	14.3	6.4	2	13.00	0.00	2	0	0	0.00	0.00
1	43	884	1113433139200	16.801	89.554	6042836.911	2133158.633	14.3	6.4	2	13.00	0.00	2	0	0	0.00	0.00
1	44	884	1113433139300	16.834	90.854	6042836.784	2133159.927	14.3	6.4	2	13.00	0.00	2	0	0	0.00	0.00

4	288	749	1113433163700	52.553	301.726	6042846.483	2133373.265	13.4	5.3	2	0.00	0.00	5	21	27	22.32	9999.99
4	289	749	1113433163800	52.523	301.726	6042846.457	2133373.232	13.4	5.3	2	0.00	0.00	5	21	27	22.61	9999.99
4	290	749	1113433163900	52.588	301.726	6042846.518	2133373.264	13.4	5.3	2	0.00	0.00	5	21	27	22.95	9999.99
4	291	749	1113433164000	52.704	301.789	6042846.618	2133373.404	13.4	5.3	2	1.40	8.05	5	21	27	23.18	16.56
4	292	749	1113433164100	52.896	301.980	6042846.782	2133373.653	13.4	5.3	2	2.06	5.91	5	21	27	23.24	11.28
4	293	749	1113433164200	53.005	302.244	6042846.857	2133373.944	13.4	5.3	2	2.45	1.71	5	21	27	23.27	9.50
4	294	749	1113433164300	53.076	302.524	6042846.898	2133374.201	13.4	5.3	2	2.46	-1.86	5	21	27	23.38	9.51
4	295	749	1113433164400	53.048	302.769	6042846.838	2133374.470	13.4	5.3	2	2.22	-3.09	5	21	27	23.56	10.61
4	296	749	1113433164500	53.047	302.975	6042846.814	2133374.665	13.4	5.3	2	1.95	-2.84	5	21	27	23.67	12.14
4	297	749	1113433164600	53.047	303.148	6042846.796	2133374.815	13.4	5.3	2	1.73	-1.56	5	21	27	23.71	13.70
4	298	749	1113433164700	53.047	303.303	6042846.774	2133374.994	13.4	5.3	2	1.63	-0.28	5	21	27	23.75	14.57
4	299	749	1113433164800	53.020	303.462	6042846.731	2133375.128	13.4	5.3	2	1.61	0.05	5	21	27	23.91	14.85
4	300	749	1113433164900	53.091	303.614	6042846.781	2133375.311	13.4	5.3	2	1.70	1.48	5	21	27	24.21	14.24
4	301	749	1113433165000	53.195	303.774	6042846.866	2133375.479	13.4	5.3	2	2.02	4.32	5	21	27	24.56	12.16
4	302	749	1113433165100	53.383	303.978	6042847.032	2133375.670	13.4	5.3	2	2.58	7.16	5	21	27	24.88	9.64
4	303	749	1113433165200	53.485	304.258	6042847.097	2133375.990	13.4	5.3	2	3.35	8.86	5	21	27	25.10	7.49
4	304	749	1113433165300	53.555	304.646	6042847.123	2133376.367	13.4	5.3	2	4.16	7.57	5	21	27	25.21	6.06
4	305	749	1113433165400	53.539	305.132	6042847.051	2133376.832	13.4	5.3	2	4.63	2.70	5	21	27	25.24	5.45
4	306	749	1113433165500	53.536	305.656	6042846.982	2133377.387	13.4	5.3	2	4.51	-4.76	5	21	27	25.22	5.59
4	307	749	1113433165600	53.533	306.123	6042846.921	2133377.870	13.4	5.3	2	3.79	-10.49	5	21	27	25.21	6.65
4	308	749	1113433165700	53.531	306.442	6042846.881	2133378.193	13.4	5.3	2	2.92	-8.70	5	21	27	25.25	8.65
4	309	749	1113433165800	53.532	306.650	6042846.863	2133378.350	13.4	5.3	2	2.41	-2.05	5	21	27	25.37	10.53
4	310	749	1113433165900	53.532	306.835	6042846.845	2133378.507	13.4	5.3	2	2.46	4.24	5	21	27	25.57	10.39
4	311	749	1113433166000	53.532	307.076	6042846.806	2133378.829	13.4	5.3	2	3.05	7.74	5	21	27	25.78	8.45
4	312	749	1113433166100	53.528	307.422	6042846.744	2133379.313	13.4	5.3	2	3.80	7.63	5	21	27	25.89	6.81
4	313	749	1113433166200	53.528	307.851	6042846.679	2133379.838	13.4	5.3	2	4.43	5.64	5	21	27	25.84	5.83
4	314	749	1113433166300	53.529	308.343	6042846.618	2133380.335	13.4	5.3	2	4.84	2.37	5	21	27	25.68	5.31
4	315	749	1113433166400	53.530	308.857	6042846.557	2133380.831	13.4	5.3	2	4.99	0.06	5	21	27	25.53	5.12
4	316	749	1113433166500	53.529	309.357	6042846.495	2133381.327	13.4	5.3	2	4.99	-0.05	5	21	27	25.47	5.11
4	317	749	1113433166600	53.504	309.856	6042846.408	2133381.820	13.4	5.3	2	4.99	0.05	5	21	27	25.48	5.11
4	318	749	1113433166700	53.576	310.355	6042846.419	2133382.320	13.4	5.3	2	5.00	0.15	5	21	27	25.52	5.10
4	319	749	1113433166800	53.682	310.856	6042846.462	2133382.834	13.4	5.3	2	5.01	0.11	5	21	27	25.54	5.10

1986	7182	1202	1113437485100	30.101	1638.339	6042632.050	2134694.495	13.8	7.3	2	10.25	0.00	3	0	1991	0.00	0.00
1986	7183	1202	1113437485200	30.109	1639.366	6042631.896	2134695.508	13.8	7.3	2	10.23	-0.29	3	0	1991	0.00	0.00
1986	7184	1202	1113437485300	30.118	1640.389	6042631.742	2134696.521	13.8	7.3	2	10.19	-0.54	3	0	1991	0.00	0.00
1986	7185	1202	1113437485400	30.125	1641.404	6042631.587	2134697.534	13.8	7.3	2	10.16	-0.26	3	0	1991	0.00	0.00
1986	7186	1202	1113437485500	30.135	1642.415	6042631.433	2134698.548	13.8	7.3	2	10.16	0.27	3	0	1991	0.00	0.00
1986	7187	1202	1113437485600	30.144	1643.430	6042631.278	2134699.561	13.8	7.3	2	10.19	0.54	3	0	1991	0.00	0.00
1986	7188	1202	1113437485700	30.153	1644.453	6042631.124	2134700.574	13.8	7.3	2	10.23	0.29	3	0	1991	0.00	0.00
1986	7189	1202	1113437485800	30.163	1645.480	6042630.970	2134701.588	13.8	7.3	2	10.25	0.00	3	0	1991	0.00	0.00
1986	7190	1202	1113437485900	30.172	1646.505	6042630.815	2134702.601	13.8	7.3	2	10.25	0.00	3	0	1991	0.00	0.00
1986	7191	1202	1113437486000	30.182	1647.530	6042630.661	2134703.614	13.8	7.3	2	10.25	0.00	3	0	1991	0.00	0.00
1986	7192	1202	1113437486100	30.191	1648.555	6042630.507	2134704.628	13.8	7.3	2	10.25	0.00	3	0	1991	0.00	0.00
1986	7193	1202	1113437486200	30.200	1649.580	6042630.352	2134705.641	13.8	7.3	2	10.25	0.00	3	0	1991	0.00	0.00
1986	7194	1202	1113437486300	30.210	1650.605	6042630.198	2134706.654	13.8	7.3	2	10.25	0.00	3	0	1991	0.00	0.00
1986	7195	1202	1113437486400	30.220	1651.630	6042630.044	2134707.667	13.8	7.3	2	10.25	0.00	3	0	1991	0.00	0.00
1986	7196	1202	1113437486500	30.228	1652.655	6042629.889	2134708.681	13.8	7.3	2	10.25	0.00	3	0	1991	0.00	0.00
1986	7197	1202	1113437486600	30.238	1653.680	6042629.735	2134709.694	13.8	7.3	2	10.25	0.00	3	0	1991	0.00	0.00
1986	7198	1202	1113437486700	30.247	1654.704	6042629.580	2134710.707	13.8	7.3	2	10.25	0.00	3	0	1991	0.00	0.00
1986	7199	1202	1113437486800	30.257	1655.730	6042629.426	2134711.721	13.8	7.3	2	10.25	0.00	3	0	1991	0.00	0.00
1986	7200	1202	1113437486900	30.266	1656.755	6042629.272	2134712.734	13.8	7.3	2	10.25	0.00	3	0	1991	0.00	0.00
1987	6353	366	1113437402200	6.612	61.046	6042830.316	2133129.106	13.3	7.3	2	48.02	0.00	1	1977	0	268.92	5.60
1987	6354	366	1113437402300	6.622	65.546	6042829.762	2133133.572	13.3	7.3	2	48.02	0.00	1	1977	0	269.01	5.60
1987	6355	366	1113437402400	6.625	70.562	6042829.146	2133138.534	13.3	7.3	2	48.02	0.00	1	1977	0	268.63	5.59
1987	6356	366	1113437402500	6.623	74.563	6042828.653	2133142.504	13.3	7.3	2	48.02	0.00	1	1977	0	269.30	5.61
1987	6357	366	1113437402600	6.623	79.563	6042828.038	2133147.466	13.3	7.3	2	48.02	0.00	1	1977	0	269.04	5.60
1987	6358	366	1113437402700	6.592	84.311	6042827.422	2133152.189	13.3	7.3	2	48.02	4.63	1	1977	0	269.08	5.60
1987	6359	366	1113437402800	6.560	89.186	6042826.791	2133157.018	13.3	7.3	2	48.26	-1.40	1	1977	0	269.00	5.57
1987	6360	366	1113437402900	6.527	93.997	6042826.168	2133161.778	13.3	7.3	2	48.16	-0.46	1	1977	0	268.90	5.58
1987	6361	366	1113437403000	6.494	98.811	6042825.544	2133166.549	13.3	7.3	2	48.12	-0.38	1	1977	0	268.74	5.58
1987	6362	366	1113437403100	6.462	103.618	6042824.921	2133171.319	13.3	7.3	2	48.10	0.00	1	1977	0	268.61	5.58
1987	6363	366	1113437403200	6.430	108.428	6042824.298	2133176.090	13.3	7.3	2	48.10	-0.05	1	1977	0	268.60	5.58
1987	6364	366	1113437403300	6.402	113.237	6042823.674	2133180.861	13.3	7.3	2	48.10	0.04	1	1977	1999	268.71	5.59

2029	6899	1825	1113437456800	40.943	989.790	6042745.668	2134055.424	14.8	6.9	2	8.49	-0.11	4	2023	2036	31.95	3.7
2029	6900	1825	1113437456900	40.942	990.640	6042745.542	2134056.265	14.8	6.9	2	8.47	-0.24	4	2023	2036	32.14	3.7
2029	6901	1825	1113437457000	40.943	991.481	6042745.417	2134057.106	14.8	6.9	2	8.50	0.51	4	2023	2036	32.36	3.8
2029	6902	1825	1113437457100	40.943	992.319	6042745.294	2134057.932	14.8	6.9	2	8.69	2.85	4	2023	2036	32.57	3.7
2029	6903	1825	1113437457200	40.943	993.182	6042745.169	2134058.772	14.8	6.9	2	9.12	5.87	4	2023	2036	32.76	3.5
2029	6904	1825	1113437457300	40.943	994.112	6042745.032	2134059.687	14.8	6.9	2	9.75	7.29	4	2023	2036	32.88	3.3
2029	6905	1825	1113437457400	40.943	995.132	6042744.881	2134060.701	14.8	6.9	2	10.38	5.83	4	2023	2036	32.91	3.1
2029	6906	1825	1113437457500	40.942	996.219	6042744.718	2134061.789	14.8	6.9	2	10.81	2.83	4	2023	2036	32.87	3.0
2029	6907	1825	1113437457600	40.943	997.330	6042744.554	2134062.891	14.8	6.9	2	11.00	0.53	4	2023	2036	32.81	2.9
2029	6908	1825	1113437457700	40.942	998.439	6042744.391	2134063.979	14.8	6.9	2	11.03	-0.23	4	2023	2036	32.75	2.9
2029	6909	1825	1113437457800	40.942	999.540	6042744.229	2134065.067	14.8	6.9	2	11.01	-0.12	4	2023	2036	32.70	2.9
2029	6910	1825	1113437457900	40.942	1000.639	6042744.067	2134066.155	14.8	6.9	2	11.00	0.00	4	2023	2036	32.65	2.9
2029	6911	1825	1113437458000	40.942	1001.739	6042743.904	2134067.243	14.8	6.9	2	11.00	0.00	4	2023	2036	32.60	2.9
2029	6912	1825	1113437458100	40.942	1002.839	6042743.742	2134068.331	14.8	6.9	2	11.00	0.00	4	2023	2036	32.55	2.9
2029	6913	1825	1113437458200	40.942	1003.939	6042743.580	2134069.419	14.8	6.9	2	11.00	0.00	4	2023	2036	32.50	2.9
2029	6914	1825	1113437458300	40.942	1005.039	6042743.417	2134070.507	14.8	6.9	2	11.00	0.00	4	2023	2036	32.45	2.9
2029	6915	1825	1113437458400	40.942	1006.139	6042743.255	2134071.595	14.8	6.9	2	11.00	0.00	4	2023	2036	32.41	2.9
2029	6916	1825	1113437458500	40.942	1007.238	6042743.093	2134072.682	14.8	6.9	2	11.00	0.00	4	2023	2036	32.36	2.9
2029	6917	1825	1113437458600	40.941	1008.338	6042742.930	2134073.770	14.8	6.9	2	11.00	0.03	4	2023	2036	32.31	2.9
2029	6918	1825	1113437458700	40.941	1009.438	6042742.768	2134074.858	14.8	6.9	2	11.00	0.00	4	2023	2036	32.24	2.9
2029	6919	1825	1113437458800	40.941	1010.539	6042742.605	2134075.946	14.8	6.9	2	11.00	-0.06	4	2023	2036	32.17	2.9
2029	6920	1825	1113437458900	40.941	1011.639	6042742.443	2134077.034	14.8	6.9	2	10.99	-0.11	4	2023	2036	32.15	2.9
2029	6921	1825	1113437459000	40.941	1012.735	6042742.281	2134078.122	14.8	6.9	2	11.00	0.21	4	2023	2036	32.23	2.9
2029	6922	1825	1113437459100	40.941	1013.831	6042742.119	2134079.204	14.8	6.9	2	11.07	1.13	4	2023	2036	32.44	2.9
2029	6923	1825	1113437459200	40.941	1014.936	6042741.957	2134080.292	14.8	6.9	2	11.25	2.32	4	2023	2036	32.74	2.9
2029	6924	1825	1113437459300	40.941	1016.068	6042741.790	2134081.410	14.8	6.9	2	11.50	2.90	4	2023	2036	33.05	2.8
2029	6925	1825	1113437459400	40.941	1017.247	6042741.618	2134082.566	14.8	6.9	2	11.64	0.56	4	2023	2036	33.32	2.8
2029	6926	1825	1113437459500	40.941	1018.430	6042741.441	2134083.753	14.8	6.9	2	11.56	-2.18	4	2023	2036	33.57	2.9
2029	6927	1825	1113437459600	40.940	1019.572	6042741.262	2134084.946	14.8	6.9	2	11.45	-1.38	4	2023	2036	33.85	2.9
2029	6928	1825	1113437459700	40.944	1020.694	6042741.085	2134086.133	14.8	6.9	2	11.46	1.50	4	2023	2036	34.16	2.9
2029	6929	1825	1113437459800	40.950	1021.832	6042740.908	2134087.320	14.8	6.9	2	11.64	3.28	4	2023	2036	34.45	2.9

3366	3177	291	1113433452600	3.517	1319.416	6042656.832	2134375.476	16.8	6.9	2	60.00	-0.02	1	977	978	130.50	2.1
3366	3178	291	1113433452700	3.503	1325.416	6042655.850	2134381.395	16.8	6.9	2	60.00	0.00	1	977	978	130.70	2.1
3366	3179	291	1113433452800	3.489	1331.416	6042654.868	2134387.314	16.8	6.9	2	60.00	0.00	1	977	978	131.02	2.1
3366	3180	291	1113433452900	3.475	1337.416	6042653.886	2134393.233	16.8	6.9	2	60.00	0.00	1	977	978	131.49	2.1
3366	3181	291	1113433453000	3.460	1343.416	6042652.903	2134399.152	16.8	6.9	2	60.00	0.00	1	977	978	131.96	2.2
3366	3182	291	1113433453100	3.445	1349.417	6042651.921	2134405.071	16.8	6.9	2	59.98	0.25	1	977	978	132.31	2.2
3366	3183	291	1113433453200	3.431	1355.414	6042650.939	2134410.990	16.8	6.9	2	60.03	0.02	1	977	978	132.53	2.2
3366	3184	291	1113433453300	3.417	1361.418	6042649.957	2134416.909	16.8	6.9	2	60.04	-0.21	1	977	978	132.69	2.2
3366	3185	291	1113433453400	3.396	1367.421	6042648.975	2134422.829	16.8	6.9	2	60.01	-0.06	1	977	978	132.83	2.2
3366	3186	291	1113433453500	3.371	1373.422	6042647.992	2134428.748	16.8	6.9	2	60.00	0.00	1	977	978	132.93	2.2
3366	3187	291	1113433453600	3.347	1379.422	6042647.010	2134434.667	16.8	6.9	2	60.00	0.00	1	977	978	133.02	2.2
3366	3188	291	1113433453700	3.323	1385.422	6042646.028	2134440.586	16.8	6.9	2	60.00	0.00	1	977	978	133.13	2.2
3366	3189	291	1113433453800	3.299	1391.418	6042645.046	2134446.505	16.8	6.9	2	60.15	-1.56	1	977	978	133.32	2.2
3366	3190	291	1113433453900	3.274	1397.421	6042644.063	2134452.424	16.8	6.9	2	60.36	-5.96	1	977	978	133.57	2.2
3366	3191	291	1113433454000	3.250	1403.430	6042643.081	2134458.343	16.8	6.9	2	58.53	11.20	1	977	978	133.88	2.2
3366	3192	291	1113433454100	3.212	1409.171	6042642.091	2134464.224	16.8	6.9	2	60.92	11.20	1	977	978	134.50	2.2
3366	3193	291	1113433454200	3.203	1415.372	6042641.146	2134470.009	16.8	6.9	2	63.49	-10.82	1	977	978	134.69	2.1
3366	3194	291	1113433454300	3.186	1421.676	6042640.103	2134476.351	16.8	6.9	2	62.35	-11.20	1	977	978	134.78	2.1
3366	3195	291	1113433454400	3.152	1427.849	6042639.081	2134482.452	16.8	6.9	2	61.20	-10.34	1	977	978	135.02	2.2
3366	3196	291	1113433454500	3.127	1433.927	6042638.089	2134488.425	16.8	6.9	2	58.53	11.20	1	977	978	135.34	2.3
3366	3197	291	1113433454600	3.089	1439.666	6042637.098	2134494.313	16.8	6.9	2	60.94	11.20	1	977	978	135.94	2.2
3366	3198	291	1113433454700	3.085	1445.927	6042636.153	2134500.097	16.8	6.9	2	62.99	-11.20	1	977	978	135.87	2.1
3366	3199	291	1113433454800	3.077	1452.284	6042635.110	2134506.440	16.8	6.9	2	59.98	1.27	1	977	978	135.56	2.2
3366	3200	291	1113433454900	3.044	1458.088	6042634.080	2134512.545	16.8	6.9	2	60.86	11.20	1	977	978	135.75	2.2
3366	3201	291	1113433455000	3.020	1464.198	6042633.117	2134518.290	16.8	6.9	2	62.43	11.20	1	977	978	135.73	2.1
3366	3202	291	1113433455100	3.020	1470.502	6042632.102	2134524.508	16.8	6.9	2	63.91	6.41	1	977	978	135.62	2.1
3366	3203	291	1113433455200	3.003	1476.933	6042631.048	2134530.859	16.8	6.9	2	64.45	7.04	1	977	978	135.36	2.1
3366	3204	291	1113433455300	2.986	1483.411	6042629.985	2134537.265	16.8	6.9	2	66.47	-11.20	1	977	978	134.89	2.0
3366	3205	291	1113433455400	2.983	1490.170	6042628.929	2134543.716	16.8	6.9	2	64.07	-11.20	1	977	978	134.01	2.0
3366	3206	291	1113433455500	2.951	1496.468	6042627.828	2134550.263	16.8	6.9	2	61.50	10.88	1	977	978	133.53	2.1
3366	3207	291	1113433455600	2.926	1502.663	6042626.825	2134556.251	16.8	6.9	2	62.73	10.18	1	977	978	132.84	2.1