FRAMEWORK AND ALGORITHMS FOR ONLINE INVERSE REINFORCEMENT LEARNING UNDER IMPERFECT OBSERVATIONS

by

Saurabh Arora

(Under the Direction of Prashant Doshi)

Abstract

Autonomous systems predominantly deploy IRL (inverse reinforcement learning) to model the task preferences of a user (often called an expert), as a reward function, by observing the user while executing the task. While IRL is witnessing sustained attention, the related problem of online IRL– where the observations are incrementally accrued, yet the real-time demands of the application often prohibit a full rerun of an IRL method – has received much less attention. Furthermore, most of the current online learning literature assumes perfect noise-free completely perceivable training data along with a prior knowledge of features for the model of task being learned. Unfortunately, these assumptions do not hold in real life applications. The following data imperfections and lack of prior knowledge impact learning accuracy: 1) some of the data in input trajectories is missing, 2) data is mixed up with some data from other sources, 3) input data has perception noise and observation model is unknown, 4) input data has perception noise and manual engineering of system features is not possible. The research contributions from my team address these gaps. Experimental evaluation of these cases on robotic domains (navigation and manipulation) and OpenAI gym domains showed a significant improvement in performance w.r.t. state-of-the-art baselines.

Index words: inverse reinforcement learning, online machine learning, missing data, multi task data, noisy data, perception noise, robotics

FRAMEWORK AND ALGORITHMS FOR ONLINE INVERSE REINFORCEMENT LEARNING UNDER IMPERFECT OBSERVATIONS

by

Saurabh Arora

B.Tech., University of Delhi India, 2008
M.S., University of Delaware, 2016

A Dissertation Submitted to the Graduate Faculty of the
University of Georgia in Partial Fulfillment of the Requirements for the
Degree.

Doctor of Philosophy

Athens, Georgia

2023

FRAMEWORK AND ALGORITHMS FOR ONLINE INVERSE REINFORCEMENT LEARNING UNDER IMPERFECT OBSERVATIONS

by

Saurabh Arora

Major Professor:	Prashant Doshi
Committee:	Khaled Rasheed
	Frederick Maier
	Scott Niekum
	Bikramjit Banerjee

Electronic Version Approved:

Ron Walcott
Dean of the Graduate School
The University of Georgia
August 2023

Dedication

To people who have been there for me!

Acknowledgments

I thank my major adviser Dr Praahant Doshi, committee member Dr Bikramjit Banerjee, lab alumnus Kenneth Bogart for all the guidance and help in conducting research amd writing papers. I also thank all the lab members, especially Ehsan Asali and Prasanth Suresh, for their help in experiments. I also thank Dr Fred Maier for letting us use Turtlebot he had. Lastly, I thank all UGA staff who helped me in this journey.

Contents

List of Figures

List of Tables

Chapter 1

Motivation

At the core of various robotics applications lies the challenge of learning how to map the world state to appropriate actions. A manual development of this mapping, also called a policy, is typically arduous. In the field of RL (Reinforcement Learning), the policy is learned via generating examples of interaction with the environment and receiving rewards as feedback for actions. The mechanism providing the reward (called the reward function) is the motivation behind action selection, and consequently also guides task execution. The function models the motivation or preferences of an agent executing the task at hand. Nevertheless, it is not always straightforward to define a reward function that precisely captures the intended behavior of a task. An alternative to manual designing is to learn the reward function from the examples of task execution, a technique referred as IRL (Inverse Reinforcement Learning).

IRL is the problem of inferring the reward function of an agent (also called an "expert"), given the examples (behavior) of task execution by the expert. It is perceived both as a problem and as a class of methods. A learner uses IRL to model the preferences of an expert by using observations of the expert's behavior, thereby avoiding a manual specification of said preferences.

Why does knowing about IRL matter? Learning the reward function from the demonstration is significant for multiple reasons:

1. In a task defined as a MDP, solving forward reinforcement learning problem involves computing an optimal policy (or controller) given a pre-defined reward function. The manual specification of the reward values for most real-life applications needs a skill set that is rarely available. Learning a reward function from a demonstration of an optimal execution of a task avoids the need for manual specification.

2. The extent of generalization of learned information via learning a reward function is greater than through directly learning a policy

3. The ever-expanding set of potential IRL applications. While introducing IRL, Russell [70] alluded to its potential application in providing computational models of human and animal behavior because these are difficult to specify. In this regard, Baker et al. [10] and Ullman et al. [79] demon-

strate the inference of a human's goal as an inverse planning problem in an MDP. Furthermore, IRL's use toward apprenticeship has rapidly expanded the set of visible applications such as:

(a) Learning to create another agent using the learned preferences of expert. Examples of such applications include helicopter flight control [3], boat sailing [59], socially adaptive navigation that avoids colliding into humans by learning from human walking trajectories [54, 50], house chores like arranging dishes in a dish rack [34], sorting vegetables in vegetable processing facility [7, 6], and so on;

(b) Learning to predict an expert's behavior. In this use, applications include destination or route prediction for taxis [87, 89], footstep prediction for planning legged locomotion [67], the anticipation of pedestrian interactions [90], energy efficient driving [80], and penetration of a perimeter patrol [14] by learning the patrollers' preferences and navigation route.

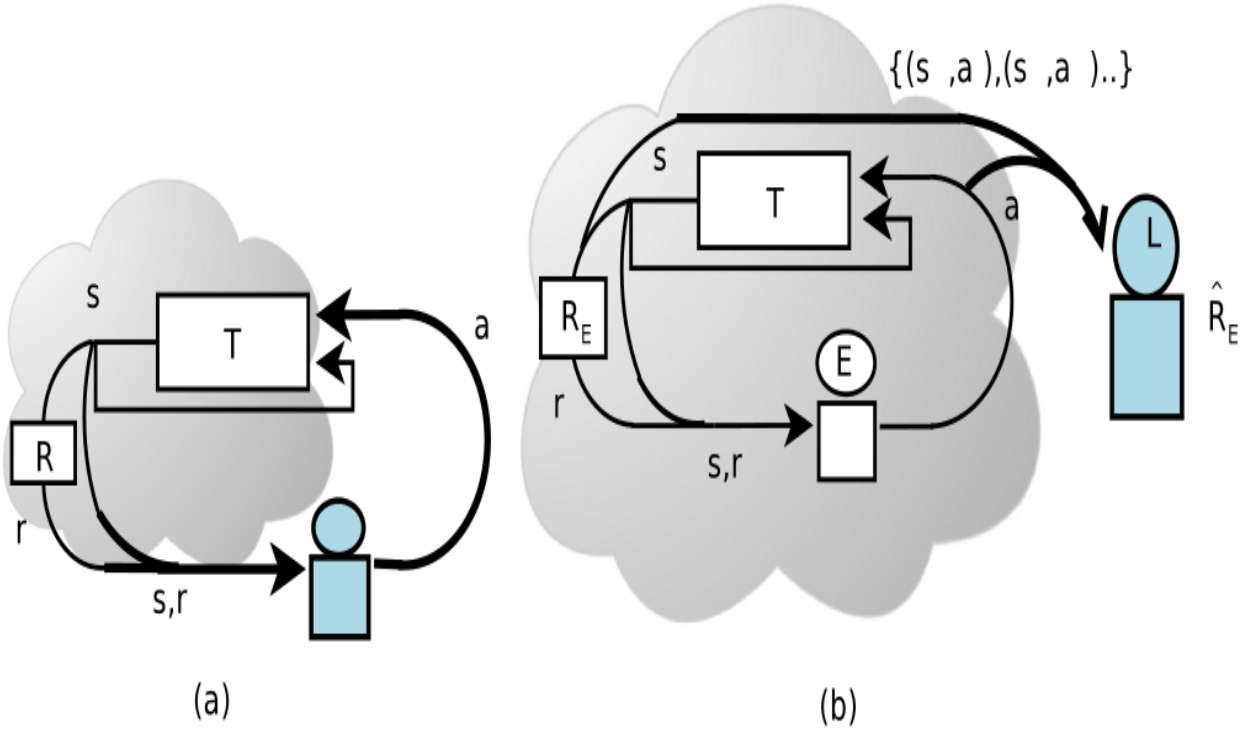


Figure 1.1: (a) A schematic showing the subject agent (shaded in blue) performing RL [48]. In forward learning or RL, the agent chooses an action a at a known state s and receives a reward r in return. Please refer MDP definition in Chapter 2 for the detailed meaning of mathematical terms S, T, R. (b) In inverse learning or IRL, the input and output for the learner L are reversed. L perceives the states and actions $\{(s,a)(s,a)\ldots\}$ of expert E (an alternative to input policy π_E), and infers a reward function, $\hat{R}_E$, of E as an output. Note that the learned reward function may not exactly correspond to the true reward function. Color should be used for this figure in print.

Russell [71, 60] introduced the IRL problem: learning the preferences of a reinforcement learning agent (expert) executing a task, by observing an optimal execution of a task by the expert. The interac-

tion of the expert with their environment is modeled as a MDP and the reward function of the MDP represents an expert's preferences for finishing the task.

IRL aims to find a reward function under which the observed behavior (training data) from the expert is optimal [71, 60]. The observed behavior is typically in the form of a set of trajectories, where a trajectory is a sequence of actions the expert takes to finish the task she is executing. After learning the reward function, the learner uses that function to find a decision-making strategy function called a policy (Chapter 2 explains a trajectory and a policy more formally). Notice that IRL inverts the RL problem. Whereas RL seeks to learn the optimal policy given a reward function, IRL seeks to learn the reward function that best supports a given policy or observed behavior. We illustrate this relationship between RL and IRL in Fig. 1.1.

Since its introduction, IRL has witnessed a significantly improved understanding of the inherent challenges and limitations, along with various methods for addressing them. We introduce here some of the prominent methods in IRL literature[1]. Maximum margin prediction methods aim to predict a reward function that makes the expert's policy look better than its alternatives by a margin. e.g. Apprenticeship Learning [1]. As multiple reward functions can explain the expert's behavior, max-margin may introduce a bias into the learned reward function. Thus, multiple methods take recourse to the maximum entropy estimation [45] of a probability distribution parameterized by a reward function. e.g. **MaxEntIRL** [87]. Bayesian IRL [64] is another prominent approach towards IRL. It applies a Bayesian update to an assumed prior distribution over possible hypotheses. Another approach toward IRL utilizes deep learning techniques. Examples of this approach include FIRL, deep MaxentIRL [85], AIRL [] and GAIL.

1.1 Research contributions in thesis

Our research contributions are new online learning methods that address the following gaps noticed in the realms of online learning and imperfect input data (Fig 1.2).

1.1.1 Scarcity of incremental learning methods

To understand this contribution well, it is important to review the differences between batch learning and incremental (online) learning. In batch approach, the learner has to process entire dataset at once. However, in a situation where the data is generated as a function of time or where batch learning is computationally very slow, the learner need to update the model sequentially as the next installment of training information gets available. Latter approach is known as incremental learning or online learning. In incremental learning, it is crucial that the learner does not forget the information learned in past, regardless of how distant the past is. Most IRL methods operate on batches of observations, and, in a single pass, these methods estimate the expert's reward function [2, 9, 21, 27, 64]. These methods may be satisfactory for the applications where the performed task is first observed for some time and this

[1]Further discussions will use the acronyms for some of the methods mentioned here.

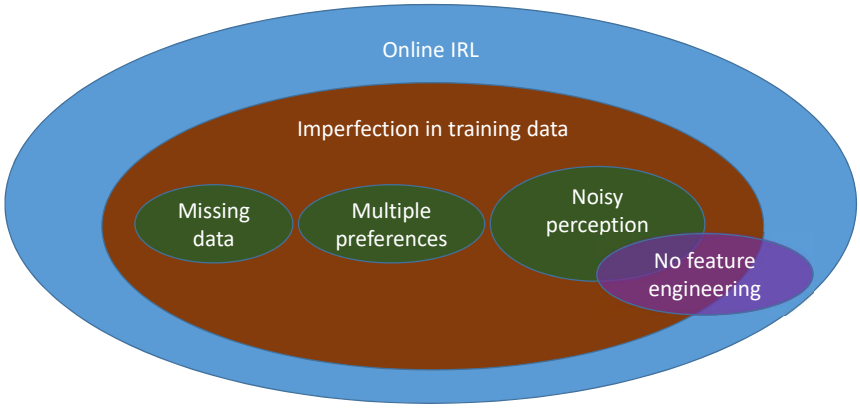


Figure 1.2: Within the realm of online IRL, we focused on applications where demonstration data has three specific kinds of imperfections. For last kind of imperfection, we extended our contribution to a case without need for manual engineering of reward function features.

gathered data is utilized toward IRL. However, newer applications of IRL impose a different set of requirements. These applications require the learner to continuously observe and repeatedly update the learner's estimation of the inferred reward function. Consider these examples: the task of forecasting a person's goals in an everyday setting by observing her ongoing activities using a body camera [69], or a robotic learner that is continuously monitoring surveillance patrol activities from a given location and learning the patrol pattern for reaching a specific goal as quickly as possible without being detected (application named **Perimeter Patrol**, explained in detail in Chapter 7) [13]. As such, the progress made in making the methods of IRL on-line is rare [46, 69, 35].

1.1.2 Imperfections in online learning data

There are many possible ways in which the expert's demonstration data received by a learner can be faulty. We focus here on a specific subset of those imperfections:

- some of the information is missing from the training data
- expert executes a task in different ways and data for those ways got mixed up before showing data to the learner
- learner perceives data corrupted by factors like sensor's hardware limitations, an inadequate orientation of camera, etc.

For the last type of imperfection, we go one step further by proposing a method that learns under perception noise but without the need for feature engineering.

Missing data

Methods like online linear programming IRL [46], on-line MAXENT IRL [69] assume that input data is not missing any relevant information, which is not always true. The learner may not be able to perceive states completely or may have a partially blocked field of view. As an example, a part of the patrolling path is always occluded from the learner in perimeter patrol (application explained in the previous paragraph). So learner needs a way to learn the patroller's reward function online, and she needs to approximate the missing data. Choi et al. extend apprenticeship learning method to partially observable state-space [28], Bogert et al. [18, 16] formulated MAXENTIRL [87] for settings where the learner is unable to see portions of demonstrated trajectories due to occlusion, and (HIOC) [51] casts MAXENTIRL to the setting modeling learner's sensing-noise as a hidden variable. However, these methods have significant limitations: they do not work in online learning settings where not all demonstration is available and they take a longer time to converge. For time-limited tasks, a slow convergence leads to timeouts, which lowers the performance in solving problem post-learning.

To address the above two gaps, we present the

> first formal framework to facilitate investigations into online IRL

The framework interprets incremental learning as a sequence of sessions and each session adds currently inferred information onto the reward learned previously. We named this framework I2RL (Chapter 3). The framework establishes the notion of incremental IRL casting existing online methods in the formal context. In Chapter 4, we introduce

> a method called LME-I2RL [5] that admits a formally provable convergence of online learning under observations with missing data

thereby addressing the gaps mentioned before. Evidence showed that faster learning leads to better performance of the learner in solving time-limited robotic problems. Also, the method admits convergence guarantees, which is rare in online IRL research.

Data from diverse preferences

An expert (demonstrator of a task) can choose many ways to finish the same task. Some learning applications have data generated from more than one set of preferences (modeled as reward functions). The learner has to first distinguish them and then learn each reward function accurately. Here is an example application from the agriculture industry where data from multiple behaviors easily mix up. The **task of sorting onions** can be accomplished in many ways (inspecting them one at a time, inspecting multiple vegetables simultaneously by rolling hand on them), and the choice depends on the preferences of the sorter between having high precision or high speed of sorting (Fig 1.3). The goal of a learner robot is to observe the sorter execute all possible ways of sorting and then distinguish them, all while receiving training

data continually in installments. In Multi-task IRL, the imperfection in data comes from the demonstrator either switching between multiple ways (preferences) of solving the given problem or performing multiple tasks together.

In a previous approach to multi-task Bayesian IRL, DPM-BIRL [29], a Dirichlet process model is used to perform non-parametric clustering of the trajectories, where each cluster corresponds to an underlying reward function. Different from Bayesian IRL, Babes-VRoman et al. [9] apply the iterative EM-based clustering (EM-MLIRL) by replacing the mixture of Gaussians with a mixture of reward functions, and a reward function of maximum likelihood is learned for each cluster. However, as pointed out by Ziebart et al. [88], such methods use exponentiated Q-function (credit assigned to an action in a specific state) to get the probability of each state-action pair. Due to the locality of the action probability computation, the likelihood of a trajectory is impacted by the number of action choice points (branching) encountered by a trajectory. The bias may cause a trajectory with a higher cumulative return to have a lower likelihood under the learned policy, leading to a sub-optimal learned reward function. Also, MLIRL and DPM-BIRL as is do not admit online learning.

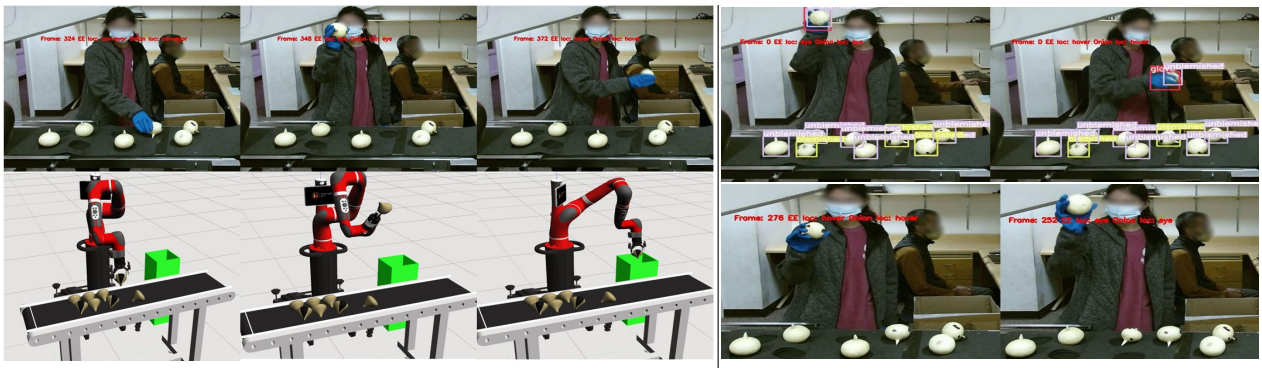
To fill the void in the literature, unlike previous approaches, we present a

> a method called online MME-MTIRL that admits faster online multi-task IRL than the state of the art without the label bias pointed in [88]

This new method for multi-task IRL generalizes the well-known maximum entropy approach to IRL (MaxEntIRL) [88]. We call our method Min-Max Entropy Multi-Task IRL, or MME-MTIRL (Chapter 5) [6]. We formulate the problem as a single entropy-based nonlinear program that combines the MaxEntIRL objective with the objective of finding a cluster assignment distribution having the least entropy. Taking a different viewpoint, Gleave and Habryka [38] propose regularized MaxEnt multi-task IRL that is based on the premise that each reward function being learned is close to the mean across all functions, thus transferring information across tasks. Unlike [38], MME-MTIRL has been tested on a real-life physical cobot and it does not need task specifications to be close to each other. After formulating a batch version of this method, we extend it to online learning to help reduce learning time in time-limited domains. In our evaluations, we found that online MME-MTIRL performed the task better and learned faster than batch MME-MTIRL; which in turn performed better than MLIRL and DPM-BIRL.

Perception noise

In real-life applications, unwanted noise may perturb the perception process and a learner may need to learn incrementally from noisy input because not all the training data is available in one batch. Motivated by the challenging goal of bringing robotic automation to post-harvest processing lines for vegetables, we consider examples of noisy perception by a robot learner observing a human sorting onions by removing the blemished ones. To learn this complicated manipulation task from its observations, the learner needs to unambiguously understand when is an onion considered blemished, and how the various locations of the expert's hand help realize the task (e.g., hand hovering over the table, hand near the eyes of the sorter);



(a) (b)

Figure 1.3: **(a)** (top) A human demonstrator picks an onion, inspects it closely to check if it is blemished, and places it in the bin on finding it to be blemished. (bottom) The learner (Sawyer) robot uses the learned policy to imitate the observed behavior. **(b)** (Examples of noisy perception: top) a blemished onion detected as being unblemished. (bottom-left) An onion held in the hover region. (bottom-right) An onion held at the eye (or in inspection) region.

see Fig. 1.3(a). When the expert places a blemished onion in a bin for bad onions, as the robot's sensing is imperfect, the blemish may not be perceived correctly by the learner's camera. This introduces an error in the perceived state because the learner incorrectly sees the human placing an unblemished onion in a destination meant for bad onions (see Fig. 1.3(b) top). Another example of imperfect perception is shown in Fig. 1.3(b) bottom: although the expert is holding an onion just above the table, the learner may incorrectly perceive it to be in the inspection (or eye) region. This ambiguity occurs because these two regions are in close proximity.

Max-margin approaches use slack and regularization variables for accommodating noisy input, but they may learn inaccurately [3] or predict the control poorly [65], resulting in unstable learned behavior. More robust approaches like MAXENTIRL, BIRL, MLIRL and GPIRL use probabilistic frameworks to account for perturbation or noise in training data. MMP learns while matching the state visitations in demonstrated trajectory and that of the learned trajectory, an approach not robust to undesirably long input trajectories. [30, 67] solved the issue by modifying the method by applying a functional gradient normalized by the state visitation counts of a whole trajectory. As these methods are not explicitly modeling for noise, their results are mostly wanting. Shahryari and Doshi [74] approach the problem of offline IRL under noisy observation using the *Robust IRL* method (sampling hidden ground truth using an observation model or sensor model), but they assume that the learner knows the observation model. This assumption may not be pragmatic because the observation model is typically a function of obscure variables including inaccuracies in sensing hardware and computer vision. Real-world robotic applications need a method that can learn the unknown observation model under noisy input and use that model to learn a reward function. Furthermore, these methods as such are not incremental learners and therefore are limited to domains where all data is available in one go. On the other hand, in our evaluation

experiments, the IRL methods which are online did not perform well if input data has perception noise (AIRL [35]).

The analysis leads to the requirement of an online method that can learn under perception noise without the knowledge of the observation model. To meet this challenge, in Chapter 6, we present

> a method labeled as RIMEO [7] that generalizes the batch Robust IRL to the incremental setting, and without prior knowledge of the observation model

Our method engages in robust incremental IRL with a maximum entropy estimation of the observation model, and is an instantiation of the I2RL framework. We formulate it as a sequence of incremental learning sessions where each session is a two-step process – maximum entropy learning of the observation model of the learner, followed by inversely learning the reward function of the expert by utilizing the learned observation model.

Using noisy observations of a human engaged in sorting onions, we evaluate the performance of RIMEO in a Gazebo sim of the onion-sorting domain in which Sawyer processes the observations incrementally and eventually imitates the task of sorting the onions. We show that it generates a significantly improved sorting performance compared to a recent end-to-end IRL baseline that accepts raw images as input. As this and other IRL methods do not directly learn an observation model, we also evaluate the comparative advantage of RIMEO against a custom baseline that learns a less sophisticated observation model. Similar results are observed in another application used for evaluation. Furthermore, we formally prove that RIMEO's data likelihood improves monotonically with more sessions, and it admits a probabilistic bound for sample complexity.

Neu et al. [59] emphasized the limitation that the accuracy of many IRL methods [61, 1, 65, 59, 87] closely rely on the correctness of prior knowledge of reward features. Recent advances like regression or deep-learning (deep-learning based online method AIRL [35], regression-based method FIRL, and GCL) have attempted to remove the need for prior knowledge of features, by automatically constructing the features. However, extension or analysis of such methods to online settings under perception noise is missing from the literature. While working on RIMEO, we noticed similar gaps that motivated further research. Firstly, noise-robust methods likeRIMEO need manual engineering of reward features and AIRL does not need that prior engineering. And secondly, under noisy input, AIRL (as a baseline for RIMEO) performs worse than other methods [7]; potentially due to the inability of the existing structure of AIRL to explicitly account for the noise in the input data. To address both problems, we found a way to improve the performance of AIRL under perception noise, which in turn also helped avoid the need for manual feature engineering in online learning under perception noise.

Some researchers attempted to fill these gaps but their modified the goal of learning. Chen et al. [25] proposed a method called 'SSRR noisy-AIRL'. The method first fits the input data to a function that can characterize noise level for trajectories and then uses regression to learn a reward function that makes the learner explore to find a policy with returns higher than sub-optimal input. However, in such an approach, the learned reward function may be very different from the ground truth reward function or

expert. To handle noise-robustness differently, in Chapter 6, we propose a method that exploits the direct conditional probabilistic model between noise-free ground truth and noisy observations.

> robust-AIRL, a sampling based IRL method that avoids manual engineering of rewards and improves the performance of AIRL under perception noise

Via evaluation on standard Mujoco benchmark domains, we demonstrated that this method achieves a higher 'correlation with ground truth' as compared to 'SSRR noisy-AIRL'.

1.2 Thesis structure

The current chapter explained the motivation and significance of contributions made in this research. The research attempted to address the aforementioned gaps in the literature.

In Chapter 2, we give some related work and background needed to introduce the research contributions. That is followed by the introduction of new algorithms proposed in the research work.

Chapter 3 introduces I2RL framework leading the way for Chapter 4 with LME-I2RL method which can learn online under input with missing data. In Chapter 5, we show the multi-task learning method MME-MTIRL wiht its online variant. In Chapter 6, we derive two methods (RIMEO and robust-AIRL) which can learn online under noisy input.

Chapter 7 depicts the experimental evaluation of all the research contributions; and the final chapter concludes this thesis with the significance of contributions and ideas for future work.

Chapter 2

Background and related work

A significant amount of work has been accomplished in IRL over the past two decades. Our primary focus in this thesis is on exploring the existing literature that pertains to the contributions made in this thesis. However, it is worth noting that relatively little progress has been made in the area of online learning when it comes to dealing with imperfect input. If we examine each imperfection individually, such as missing data, mixed-up data from different approaches to achieving the same task, and data corrupted by learner perception errors, we can find some related methods.

In addition, we elucidate the conceptual framework derived from previous research. While these imperfections overlap on a broader level, the specific details of the related works are mostly distinct from one another. In order to gain a thorough understanding of these concepts, we will commence with some mathematical fundamentals, proceed to provide a formal background on IRL, and subsequently delve into the literature that focuses on the contributions made in this thesis.

2.1 Preliminaries

While IRL methods in literature variously ascribe models to the expert (using an MDP, hidden-parameter MDP, or a POMDP), we focus on the most popular model by far, which is the MDP.

Definition 1. [MDP] An MDP $\mathcal{M} := \langle S, A, T, R, \gamma \rangle$ models an agent's sequential decision-making process. S is a finite set of states and A is a set of actions. Mapping $T : S \times A \to \mathsf{Prob}(S)$ defines a probability distribution over the set of next states conditioned on the agent taking action a at state s; $\mathsf{Prob}(S)$ here denotes the set of all probability distributions over S. $T_{sa}(s'|s, a) \in [0, 1]$ is the probability that the system transitions to state s'. Reward function $R : S \to \mathbb{R}$ specifies the scalar reinforcement incurred for reaching state s. Another variant $R : S \times A \to \mathbb{R}$ outputs the reward received after taking action a at state s. Discount factor γ is the weight for past rewards accumulated in a trajectory, $\langle \tau = (s_0, a_0), (s_1, a_1), \ldots (s_j, a_j) \rangle$, where $s_j \in S, a_j \in A, j \in \mathbb{N}$.

A *policy* is a function mapping the current state to the next action choice(s). It can be deterministic, $\pi : S \to A$ or stochastic $\pi : S \to \mathsf{Prob}(A)$. For a policy π, value function $V^\pi : S \to \mathbb{R}$ gives the value

of a state s as the long-term expected cumulative reward incurred from the state by following π. The value of a policy π is,

$$E_{s_0}\left[V^{\pi}(s_0)\right] = E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t)|\pi\right] \tag{2.1}$$

where s_0 is the state in which the reinforcement learning agent starts executing the task.

Let $\mathbb{N}^+$ be a bounded set of natural numbers. It is pertinent to formally define a demonstration here.

Definition 2 (Set of fixed-length trajectories)**.** The set of all trajectories of finite length T from an MDP attributed to the expert E is defined as, $\mathbb{X}^T = \{X|X = (\langle s,a\rangle_1, \langle s,a\rangle_2, \ldots, \langle s,a\rangle_T), T \in \mathbb{N}^+\}, \forall s \in S_E, \forall a \in A_E\}$.

Then, the set of *all* trajectories is $\mathbb{X} = \mathbb{X}^1 \cup \mathbb{X}^2 \cup \ldots \cup \mathbb{X}^{|\mathbb{N}^+|}$. A demonstration is some finite set of trajectories of varying lengths, $\mathcal{X} = \{X^T|X^T \in \mathbb{X}^T, T \in \mathbb{N}^+\}$, and it includes the empty set. [1] Subsequently, we may define the set of demonstrations.

Definition 3 (Set of demonstrations)**.** The set of demonstrations is the set of all subsets of the set of trajectories of varying lengths. Therefore, it is the power set, $2^{\mathbb{X}} = 2^{\mathbb{X}^1 \cup \mathbb{X}^2 \cup \ldots \cup \mathbb{X}^{|\mathbb{N}^+|}}$.

IRL is the problem of inferring the reward function of an agent (also called an "expert"), given the examples (behavior) of task execution by the expert. To address the problem of IRL, Abeel and Ng [2] modeled the reward function as a linear combination of K binary features, ϕ_k, each of which maps a state and an action to a value in {0,1}. The linearized reward function is $R_{\boldsymbol{\theta}}(s,a) = \boldsymbol{\theta}^T\phi(s,a) = \sum_{k=1}^{K} \theta_k \cdot \phi_k(s,a)$, where θ_k are the *feature weights* in $\boldsymbol{\theta}$. When using a linear reward function, the learner's task is to find a $\boldsymbol{\theta}$ that makes the observed behavior optimal in the completed MDP $\langle S, A, R_{\boldsymbol{\theta}}, T\rangle$. The cumulative value of any feature ϕ_k for trajectory X is given by a function $f_k : \mathbb{X} \to \mathbb{R}$ defined as $f_k(X) = \sum_{\langle s,a\rangle_t \in X} \gamma^t \phi_k(\langle s,a\rangle_t)$, where t is the temporal index of a state-action pair in the trajectory.

Please note that Abeel and Ng [2] pointed out that, under linear reward function with weights $\boldsymbol{\theta}$, the **expected value of an agent's behavior can be presented using the expected cumulative value of features ϕ activated while simulating agent's policy**, $E_{s_0}[V^{\pi}(s_0)] = \boldsymbol{\theta} \times E_{\pi}[\sum_{t=0}^{\infty} \phi(\langle s,a\rangle_)]$. MAXENTIRL method exploits this equivalence. The term $E_{\pi}[\sum_{t=0}^{\infty} \phi(\langle s,a\rangle_)]$ is called feature-expectations.

Legacy MAXENTIRL (Maximum-Entropy Inverse Reinforcement Learning):

Many of the early methods for IRL bias their search to combat the ill-posed nature of IRL and the very large search space [2, 88]. Ziebart et al. [88], taking a contrasting perspective, seeks a distribution over all trajectories (sequences of state-action pairs) that exhibits the maximum entropy while being constrained to match the feature-expectations of learned behavior with those estimated for the observed behavior.

$$\begin{aligned} &\max_{\Delta} \left(-\textstyle\sum_{X\in\mathbb{X}} P(X)\, log\, P(X)\right) \\ &\textbf{subject to} \quad \textstyle\sum_{X\in\mathbb{X}} P(X) = 1 \\ &E_{\mathbb{X}}[\phi_k] = \hat{\phi}_k \quad \forall k \end{aligned} \tag{2.2}$$

[1] Repeated trajectories in a demonstration can usually be excluded for many methods without impacting the learning.

Here, Δ is the space of all distributions over the set $\mathbb{X}$ of all trajectories. LHS term $E_{\mathbb{X}}[\phi_k] = \sum_{X \in \mathbb{X}} P(X)\, f_k(X) = \sum_{X \in \mathbb{X}} P(X) \sum_{\langle s,a \rangle_t \in X} \gamma^t \phi_k(\langle s, a \rangle_t)$ is feature expectations derived from learned behavior and RHS $\hat{\phi}_k = \frac{1}{|\mathcal{X}|} \sum_{X \in \mathcal{X}} f_k(X) = \frac{1}{|\mathcal{X}|} \sum_{X \in \mathcal{X}} \sum_{(s,a) \in X} \phi_k(s, a)$ is an empirical estimate of the feature expectations for the observed behavior. The problem reduces to finding θ, which parameterizes the exponential distribution that exhibits the highest likelihood. As the distribution P(·) is parameterized by learned weights $\boldsymbol{\theta}$, $E_{\mathbb{X}}[\phi_k]$ represents the feature expectations $\phi_k^{\pi_E}$.

By now, the readers know the basic mathematical concepts needed to understand the math in the rest of the thesis. The remaining part of this chapter will first talk a bit about related work in online IRL and then go one by one through related work for each imperfection of training data. Please refer to Table 2.1 for a summarized version of the discussion.

Table 2.1: Related work.

My focus	**Related work**	**Analysis**
Online learning	max-margin and Bayesian IRL methods [2, 9, 21, 27, 64]	not amenable to online learning
	online learning methods using maximum entropy / deep learning [69, 46, 35, 42]	No formal analysis and no convergence properties
Missing data	LME-IRL [13, 17]	slow convergence and lack of convergence guarantees
Mixed up data	EM-MLIRL [9]	prior knowledge of the number of tasks, label bias, no online learning
	DPM-BIRL [26]	label bias, no online learning
Perception noise	Robust IRL [74], MMAPIRL[78]	prior knowledge of observation model, no online learning
	Bogert et al. [17]	technique used for inferring transition model inspired the technique for learning observation model
	deep IRL [85, 34]	inspiration for using deep learning to avoid manual feature engineering
	AIRL [35]	no noise model accommodated, performed worse in evaluations
	SSRR noisy-AIRL [25]	learned reward function may be too distant from the expert's reward function

2.2 Online Learning

Most IRL methods in literature process observations in a single pass [2, 9, 21, 27, 64], i.e. they are offline or batch methods. These methods work for cases when all the training data is available at one time,

however, as explained in Chapter 1, some real-life applications require the learner to continuously observe and process the observations.

As the field of online IRL is relatively new, the research literature is in the nascent stages of evolution. There is an early method for online IRL [46] that modifies Ng and Russell's linear program [60]. It takes as input a single trajectory (instead of the policy) and replaces the linear program with an incremental update of the reward function. In particular, for each new observation, the learner updates the reward weights every time the observed action differs from the predicted action of the expert. Apart from the algorithm for this method, the authors provide error bounds as well.

Activity forecasting has been one of the newer applications of IRL. Researchers have focused on learning human behavior models from observations, and predicting latent goals [52]. A recent method called DARKO [69] performs online IRL for activity forecasting. By tracking the short-term location goals of a person wearing a camera, DARKO uses IRL to learn a reward function based on the types of locations and objects in the environment. Based on the inferred model, it predicts a subset of possible future goals from a given finite set of goals. Herman et al. [42] present a solution to the problem of (online) learning socially acceptable navigation behavior. By using an input demonstration *labeled with acceptability*, this method incrementally adapts a learned reward function according to recent changes in the navigation behavior of the observed humans. As observed behavior can continuously change, there is no concept of a stopping criterion for the method.

There are some online learning approaches in imitation learning, however, these methods do not solve IRL problem in the original form posed by Russell [71]. The batch algorithm of maximum margin planning (MMP) [66] is a method for imitation learning, evaluated on path planning domains. In that paper, the authors also provide an extension of MMP to online MMP, and prove that the algorithm admits a sub-linear regret bound. In a separate paper [68], the same authors present an application of the technique to the problem of online structured prediction but focused primarily on classification domains. Other relevant methods admit online learning given deliberate teaching from a human [49, 4], but we focus on the class of methods with *passive observations* by the learner without any deliberation on the part of the demonstrator.

Even though these researchers made attempts to fill the gap but a focused formal analysis of online learning is missing from the literature.

2.3 Imperfect Input

As we have discussed the literature we could find about online learning, let's have a look at different data imperfections we targeted in the thesis.

2.3.1 Missing information in training data

Our motivation behind proposing 'online IRL under missing data' is an application that involves a subject robot that must observe other mobile robots from a fixed vantage point. Its local sensors allow it a limited

observation area; within this area, it can observe the other robots fully, outside this area it cannot observe at all. Previous methods [13, 17] denote this special case of partial observability where certain states are either fully observable or fully hidden as *occlusion*. Subsequently, the trajectories gathered by the learner exhibit missing data associated with time steps where the expert robot is in one of the occluded states.

Bogert and Doshi [13], while maximizing entropy over policies [21], limited the calculation of feature expectations for policies to observable states only. To ensure that the feature expectation constraint of IRL methods account for the missing data, batch method [15, 19] improves on this method by taking an expectation over the missing data conditioned on the observations.

Let's understand the mathematical formulation of batch LME-IRL method in [19]. For introducing occlusion in data, let Y be the observed portion of a trajectory observed by the learner, and Z is one way of completing the hidden portions of this trajectory, and $X = Y \cup Z$. Treating Z as a latent variable and taking its expectation gives a new definition for the expert's feature expectations:

$$\hat{\phi}^{Z|_Y}_{\boldsymbol{\theta},k} \triangleq \frac{1}{|\mathcal{Y}|} \sum_{Y \in \mathcal{Y}} \sum_{Z \in \mathbb{Z}} P(Z|Y;\boldsymbol{\theta}) \sum\nolimits_{t=1}^{T} \gamma^t \phi_k(\langle s,a \rangle_t) \tag{2.3}$$

where $\langle s,a \rangle_t \in Y \cup Z$, $\mathcal{Y}$ is the set of all observed partial trajectory Y, $\mathbb{Z}$ is the set of all possible hidden Z that can complete a trajectory Y. The program in legacy MAXENTIRL (defined in Chapter 2) is modified by replacing $\hat{\phi}_k$ with $\hat{\phi}^{Z|_Y}_{\boldsymbol{\theta},k}$, as we show below. Notice that in the case of no occlusion $\mathbb{Z}$ is empty and $\mathcal{X} = \mathcal{Y}$. Therefore $\hat{\phi}^{Z|_Y}_{\boldsymbol{\theta},k} = \hat{\phi}_k$ and this method reduces to Legacy MAXENTIRL explained before. Thus, this method generalizes the maximum entropy IRL method.

$$\begin{aligned} &\max_{\Delta} \left(- \textstyle\sum_{X \in \mathbb{X}} P(X)\, log\, P(X) \right) \\ &\textbf{subject to } \textstyle\sum_{X \in \mathbb{X}} P(X) = 1 \\ &E_{\mathbb{X}}[\phi_k] = \hat{\phi}^{Z|_Y}_{\boldsymbol{\theta},k} \quad \forall k \end{aligned} \tag{2.4}$$

However, the program in (2.4) becomes nonconvex due to the presence of $P(Z|Y)$. As such, finding its optima by Lagrangian relaxation is not trivial. Wang et al. [83] suggests a log-linear approximation to cast the problem of finding the parameters of distribution (reward weights) as the likelihood maximization that can be solved within the schema of expectation-maximization [31] (also called E-M schema).
E-step: This step involves calculating Eq. 4.1 to arrive at $\hat{\phi}^{Z|_Y,(t)}_{\boldsymbol{\theta},k}$, a conditional expectation of the K feature functions using the parameter $\boldsymbol{\theta}^{(t)}$ from the previous iteration. We may initialize the parameter vector randomly.
M-step: In this step, the modified program (2.4) is optimized by utilizing $\hat{\phi}^{Z|_Y,(t)}_{\boldsymbol{\theta},k}$ from the E-step above as the expert's constant feature expectations to obtain $\boldsymbol{\theta}^{(t+1)}$. Normalized exponentiated gradient descent [53, 77], a version of gradient descent in which the learned parameter is scaled using the exponent of the gradient, solves the program.

The next imperfection we aimed to address is the case when data from multiple ways (modeled as reward functions) to finish the same task is mixed up before showing it to the learner. The problem of learning these multiple ways is referred to as multi-task learning.

2.3.2 Demonstration using multiple preferences

The same job may be performed in one of many ways, each guided by a distinct set of preferences. An expert may switch between these varied behaviors as she performs the job, or multiple experts may interleave to perform the job. If the reward functions producing these behaviors are distinct, then the traditional IRL would yield a single reward function that cannot explain the observed trajectories accurately. However, modeling it as a multi-task problem allows the possibility of learning multiple reward functions. If the number of involved reward functions is pre-determined, we may utilize the iterative EM - MLIRL to learn distinct reward functions [9]. The method treats each unknown reward function as a generative model producing a cluster of trajectories. But, if the number of reward functions is not known apriori, multi-task IRL can be viewed as *non-parametric* mixture model clustering, which is typically anchored by a Dirichlet process (DP) [36].

Dirichlet process mixture model A DP is a stochastic process whose sample paths are drawn from functions that are distributed according to the Dirichlet distribution. DPs find application in Bayesian mixture model clustering [36] due to an interesting property exhibited by distribution $G \sim DP(\alpha, H)$, where α is the concentration parameter and H is a base distribution. Irrespective of whether H is smooth, G is a discrete distribution. Observations θ distributed according to G allow us to update the DP. The i.i.d. draws of observation θ are typically interpreted as cluster parameters. A generic DP-based Bayesian mixture model can be seen as:

$$G|\alpha, H \sim DP(\alpha, H); \quad \theta|G \sim G; \quad X_i|\theta \sim F(\theta)$$

where data X_i has distribution $F(\theta_i)$. Notice the lack of any bound on the number of mixture components. To utilize this mixture model for clustering observed data $\{X_i\}$, we must additionally assign each data point to its originating cluster, and these assignments are drawn from convex mixture weights ($\boldsymbol{\pi} = \{\pi_1, \pi_2, \ldots, \pi_D\}$) which are themselves distributed randomly. For most Dirichlet Prcoess Mixture models, the distribution G is the discrete mixture distribution parameterized by $\boldsymbol{\pi}$ as $\sum_{d=1}^{D} \pi_d \delta_{\theta_d^*}$, where θ_d^* denotes parameter value unique to mixture component indexed d. The number of components D may grow as large as needed. We may then obtain a cluster assignment c_i for data point X_i by sampling from distribution G. probabilities $\boldsymbol{\pi}$. Then, for each data point

$$\theta_d^*|H \sim H; \qquad c_i|\boldsymbol{\pi} \sim G; \quad X_i|c_i, \{\theta_d^*\} \sim F(\theta_{c_i}^*) \tag{2.5}$$

Choi and Kim [26] utilize this Bayesian mixture model application of a DP toward multi-task IRL. The data points in DPM-BIRL $\{X_i\}$ are the observed trajectories, $\{\theta_d^*\}$ parameterizes the D distinct reward functions, and $F(\theta_{c_i}^*)$ corresponds to $\frac{1}{Z(\theta_{c_i}^*)}\, e^{\sum_{t=1}^{T} Q(s_t, a_t; \theta_{c_i}^*)}$, where T is the fixed length of the trajectory, $Z(\theta_{c_i}^*)$ is the partition function, and H is taken as the Gaussian distribution. Neal [58] discusses several MCMC algorithms for posterior inference on DP-based mixture models, and Choi and Kim select the Metropolis-Hastings.

Our second contribution of 'online multi-task learning' was inspired from the realization that prevalent multi-task methods of EM-MLIRL and DPM-BIRL are offline and they suffer from label bias pointed out in Ziebart et al. [88] (label bias has been explained in Chapter 1).

The next imperfection we focused on is the presence of perception noise in demonstration data. Later, we introduce two online methods to handle the problem - the first approach infers an observation model and the second approach avoids manual feature engineering.

2.3.3 Perception noise

Prior knowledge of observation model

Observation noise is not widely addressed by the IRL literature, which tends to assume that the state-action data is accurate. However, this assumption may not hold in robotic applications where the learner is using (noisy) sensors to watch the expert (a paradigm called learn-from-observation (LfO)). Robust IRL [74] was an early technique, which estimated the ground truth underlying the noisy observations using a known observation model. But, it did not address cases where the observation model is unknown, or when the data is only available incrementally. More recently, a hierarchical Bayesian model [12] was presented to enable IRL in partially-controlled environments where elements that confound the observations of the expert may be present. The model offers a way to sample the source of each observation and a hyperprior over the observation models is defined. Bayesian IRL has also recently been extended to contexts involving both occluded and noisy observations using marginal maximum-a-posteriori inference [78]. However, the observation or sensor model is assumed to be known in order to utilize this method. In general, all current methods of learning under noisy input are not incremental and therefore can not be used in real-life applications where training data comes in installments.

We also note IRL methods that learn the stochastic transition dynamics, either by using importance sampling [20] or by decomposing each transition into its component features and learning from feature aggregates [17]. Adapting the former method for unknown observation models requires a preexisting near-perfect base observation model, which may get unrealistic. We take inspiration from the latter method, where, analogously to Bogert and Doshi [17], we decompose an observation into its component observation features followed by learning the feature probabilities, and the learner generalizes from that information. To our knowledge, there are no existing methods that have approximated the observation model for online IRL under noisy input. To that end, we introduced method RIMEO. Additionally, existing methods that learn under perception noise are not structured to work in an incremental or online fashion.

We addressed the problem by introducing an online technique that can learn using noisy input without prior knowledge of the observation model. However, the technique needs manual engineering of reward features. That led to our contribution to learning online under perception noise but without manual feature engineering. The next section explains related work for that part.

Manual engineering of features

In the realm of learning a reward function without manual engineering of reward features, some researchers have explored deep neural network architecture as approximations of complex nonlinear reward functions in large state spaces. Wulfmeier et al. [85] and Finn et al. [34] propose neural networks as function approximators that avoid the cumbersome hand-engineering of appropriate reward features. They investigated computation of the gradient of MAXENT objective function using back-propagation making IRL scalable to large spaces with nonlinear rewards. However, [85] has been limited to overly simplified applications, and [34] heavily rely on tricks like applications specific reward regularization and can not efficiently work with complete trajectories. The benefits of GANs and the limitations of previous methods motivated GAN-based online learning approaches like AIRL [35]. Learning process based on Generative adversarial network [39] can adapt the information in input data to improve the efficiency of learning. AIRL uses GAN training to train a discriminator that has the following closed-form representation that is parameterized by reward network f_θ and generator-policy network π.

$$D_\theta(s,a) = \frac{\exp f_\theta(s,a)}{\exp f_\theta(s,a) + \pi(a|s)}$$

where (s,a) is a state-action pair in the current input demonstration.

The authors proved that the objective of training the above discriminator solves the IRL problem resulting in a reward function given by $f_\theta(s,a)$. Besides the obvious benefit of not needing manual engineering of features or special regularizations, the online method of AIRL has been shown to learn a reward function that is disentangled from underlying dynamics.

However, as observed in experimental evaluations of RIMEO, AIRL, as is, does not do well under perception noise. Our method RIMEO can learn well with input comprising perception noise **but** it needs manual engineering of reward features.

Some researchers proposed IRL methods that can learn from faulty input data. The methods that use supervised learning directly necessitate either manual clustering or labeling of sub-optimal demonstrations [40, 75], which is an additional overhead. [86] extended BIRL to the case of noisy input, by using the temperature parameter of data likelihood (Boltzmann distribution) to characterize the noise level of input. However, the method has been evaluated on low-dimensional domains, it needs a significant amount of input data, and, as is, does not admit incremental learning.

Some methods target the problem of learning a behavior with returns better than the expert (demonstrator) regardless of whether the perception process is noisy or not [24, 33, 72, 44]. We, on the other hand, use the premise that input is noisy, and it is noisy due to the learner's faulty perception. Therefore, the two problems are different from each other. Moreover, the problem of learning better than the demonstrator is distant from Russell's definition of IRL [71]. Some of these methods come closer to our goal though.

Chen et al. [25] proposed a method called 'SSRR noisy-AIRL'. The method first fits input data to a noise-and-performance relationship function that can characterize noise level for trajectories and then uses regression to learn a reward function with returns higher than sub-optimal input. However, in such an approach, the learned reward function may be very different from the noise-free ground truth reward

function of the expert. And, consequently, the learned policy may be very different from the noise-free ground truth policy. In a similar direction of work, Brown et al. [23] introduced D-REX which uses behavior cloning to generate ranked training data and runs supervised learning on this ranked data to learn a reward function with average returns higher than noisy input. However, Chen et al. [25] proved that the supervised learning process of D-REX suffers from an inductive bias which leads to an inaccurate characterization of the noise-and-performance relationship for ranked trajectories.

A few shortcomings that are shared by most of the related work: it is not clear if the source of sub-optimality in data is expert or it is the learner's perception, and it is not clear how the methods can be deployed in an application that needs an incremental or online learner. To handle robustness differently, instead of learning by using trajectory labels, our approach (called robust-AIRL) exploits the conditional probabilistic relationship between noise-free ground truth and noisy observations. We chose 'SSRR noisy-AIRL' as a baseline because it outperformed D-REX. We demonstrated that our method achieves a higher correlation with ground truth as compared to 'SSRR noisy-AIRL'.

The current chapter introduced the work done in the literature about online learning and imperfections in training data. Next chapter formally established the framework of online inverse reinforcement learning so that coming chapters can use that framework for addressing the challenges explained in the first chapter.

Chapter 3

Framework of Online IRL

We introduce in this chapter a framework meant for a mathematical analysis of the problem of incremental or online learning of reward function. The basic premise behind incremental learning of reward function is that the training data (trajectories of an expert executing a task) is either available or perceived in chunks. Our framework, labeled as incremental IRL (I2RL), establishes the key components of this problem and rigorously defines the notion of an incremental variant of IRL. We formally introduce the concept of a learner building up a new inference by using the old learned information and data recently received. We explain different ways in which the value of learned information can be measured and, therefore, used for stopping the learning process. To demonstrate its usefulness, we model existing methods that perform online IRL [46, 69] using the components of this framework. This provides initial evidence that the framework is complete.

Inverse RL aims to find a reward function under which the observed behavior of the expert is optimal, with the behavior modeled as an incomplete Markov decision process (MDP) $\langle S, A, T\rangle$ [71, 60]. Per the definitions in Chapter 2, IRL ascribes an MDP without the reward function to the expert, and thus, consists of estimating the reward function, $\hat{R}_E \in \mathcal{R}$, to provide the best explanation of the learner's observations $\mathcal{X} \in 2^{\mathbb{X}}$. Therefore, we can view IRL as a function: $\zeta(MDP_{/R_E}, \mathcal{X}) = \hat{R}_E$.

3.1 Incremental IRL (I2RL) Framework

We present a framework for incremental IRL that provides a common foundation for researchers interested in developing new techniques for online IRL, and facilitates comparison between them as well as developing their theoretical properties.

To establish the definition of incremental IRL, we must first define a *session* of I2RL. Let $\hat{R}^0_E$ be an initial estimate of the expert's reward function. Our definitions reference trajectories and demonstrations, which were previously defined.

Definition 4 (Session)**.** A session of I2RL, $\zeta_i(MDP_{/R_E}, \mathcal{X}_i, \hat{R}^{i-1}_E)$, $i > 0$ and $i \in \mathbb{N}$, takes as input the expert's MDP sans the reward function, the current (i^{th}) demonstration, $\mathcal{X}_i \in 2^{\mathbb{X}}$, and the reward function estimated previously. It yields a revised estimate of the expert's reward function, $\hat{R}^i_E$.

Note that we may replace the reward function estimates with some parameter sufficiently representing it (e.g., $\boldsymbol{\theta}$). Also, for expedience in formal analysis, we may assume that the trajectories in a session $\mathcal{X}_i$ are i.i.d. from the trajectories in the previous session. When the trajectories in $\mathcal{X}_i$ are i.i.d., the demonstrations $\{\mathcal{X}_i, i \in \{1, 2, \ldots\}$ are also i.i.d. This assumption enables deriving probabilistic convergence bounds by making it possible to apply Hoeffding's inequality, as we demonstrate later in Chapters 4 and 6.

We may let the sessions run indefinitely. Alternately, we may establish some stopping criteria for incremental learning, which then offer a basis to automatically terminate the sessions once the criterion is satisfied. Let $LL(\hat{R}_E^i|\mathcal{X}_{1:i})$ be the log-likelihood of the demonstrations received up to the i^{th} session given the current estimate of the expert's reward function. We may view this likelihood as a measure of how well the learned reward function explains the observed data. In the context of I2RL, the log-likelihood must be computed without storing data from previous sessions. Here onwards, $\widehat{\mathcal{X}}$ denotes a sufficient statistic that replaces *all input trajectories from previous sessions* in the computation of log-likelihood.

Definition 5 (Stopping criterion #1)**.** Terminate the sessions of I2RL when $|LL(\hat{R}_E^i|\mathcal{X}_i, \widehat{\mathcal{X}}) - LL(\hat{R}_E^{i-1}|\mathcal{X}_{i-1}, \widehat{\mathcal{X}'})| \leqslant \rho$, where ρ is a very small positive number.

Definition 5 reflects the fact that additional sessions are not improving the learning performance significantly. On the other hand, a more effective stopping criterion is possible if we know the expert's true policy π_E^*. We can utilize the *inverse learning error* [27] in this criterion, which gives the loss of value if the learner uses the learned policy on the task instead of the expert's:

$$ILE(\pi_E^*, \pi_E) = ||V^{\pi_E^*} - V^{\pi_E}||_1$$

Here, $V^{\pi_E^*}$ is the optimal value function of E's MDP and V^{π_E} is the value function due to utilizing the learned policy π_E (obtained from solving the MDP with the learned reward function) in E's true MDP.

Let π_E^i be the optimal policy obtained from solving the expert's MDP with the reward function $\hat{R}_E^i$ learned in session i. Another stopping criterion is defined as given below.

Definition 6 (Stopping criterion #2)**.** Terminate the sessions of I2RL when $ILE(\pi_E^*, \pi_E^{i-1}) - ILE(\pi_E^*, \pi_E^i) \leqslant \rho$, where ρ is a very small positive error and is given.

Obviously, prior knowledge of the expert's policy is not common in practice. Therefore, we view this criterion as being more useful during the formative evaluation of I2RL methods.

Utilizing Defs. 4, 5, and 6, we formally define I2RL next.

Definition 7 (I2RL)**.** Incremental IRL is a sequence of learning sessions $\{\zeta_1(MDP_{/R_E}, \mathcal{X}_1, \hat{R}_E^0), \zeta_2(MDP_{/R_E}, \mathcal{X}_2, \hat{R}_E^1), \zeta_3(MDP_{/R_E}, \mathcal{X}_3, \hat{R}_E^2), \ldots, \}$, which continue infinitely or until a stopping criterion assessing convergence is met (criterion #1 or #2 depending on which one is chosen apriori).

Please note that executing a session should happen without storing any data from previous sessions. So, if needed, a session uses the sufficient statistic $\widehat{\mathcal{X}}$ as information passed on from the previous session (in addition to $\hat{R}_E^{i-1}$), $\zeta_i(MDP_{/R_E}, \mathcal{X}_i, \hat{R}_E^{i-1}, \widehat{\mathcal{X}})$. Also, reward function $\hat{R}_E^{i-1}$ can take a parametrized

form, depending on the representation being used I2RL method, e.g. weights of a linear combination of features, weights of the neural network, etc.

Regret is a common performance measure for online learning methods. It is often defined based on an expected value of cumulative loss or gain. For example, in the context of online learning in the multi-arm bandit problem [8], regret is defined as the difference between the expected value of total loss from method A, and the total loss from the best decision in hindsight.

$$\textit{Regret}_T(A) = E_{\{x_t|t\in\{1,2,\dots T\}\}}[\Sigma_{t=1}^{T} l(x_t)] - \min_{x\in\mathcal{I}} E_{\{x_t|t\in\{1,2,\dots T\}\}}[\Sigma_{t=1}^{T} l(x)]$$

where $l(\cdot)$ is a loss function, and x_t is the decision made at time t. If the distribution over $\{x_t|t \in \{1,2,\dots T\}\}$ is unknown, it may be estimated as

$$\textit{Regret}_T(A) = \frac{1}{T}[\Sigma_{t=1}^{T} l(x_t)] - \min_{x\in\mathcal{I}} \frac{1}{T}[\Sigma_{t=1}^{T} l(x)]. \tag{3.1}$$

We may model the sessions of I2RL as the iterations of an adversarial iterative game with the learner as a player, and the IRL hypotheses $\{\hat{R}_E^i\}$ as the decisions made by the player. Note that the best possible hypothesis for $\hat{R}_E^i$ is the true reward function R_E of the expert. Thus, the minimum loss achievable relative to R_E is 0; and the loss from decision $\hat{R}_E^i$ can be measured in terms of the log-likelihood difference $\left(LL(R_E|\mathcal{X}_i, \widehat{\mathcal{X}}) - LL(\hat{R}_E^{i-1}|\mathcal{X}_{i-1}, \widehat{\mathcal{X}'})\right)$. Plugging these choices into equation 3.1, we get the following definition of average regret:

Definition 8 (Average Regret in I2RL)**.** With $\left(LL(R_E|\mathcal{X}_i, \widehat{\mathcal{X}}) - LL(\hat{R}_E^{i-1}|\mathcal{X}_{i-1}, \widehat{\mathcal{X}'})\right)$ as the loss incurred in session i by the learning algorithm A_{I2RL}, average regret after T sessions is given by:

$$\textit{Regret}_T(A_{\text{I2RL}}) = \frac{1}{T}\Sigma_{i=1}^{T}\left(LL(R_E|\mathcal{X}_i, \widehat{\mathcal{X}}) - LL(\hat{R}_E^{i-1}|\mathcal{X}_{i-1}, \widehat{\mathcal{X}'})\right).$$

While somewhat straightforward, these rigorous definitions for I2RL allow us to situate the few existing online IRL techniques, and to introduce online IRL with hidden variables, as we see next.

3.2 Existing online methods formulated as I2RL

One of our contributions is to facilitate a portfolio of online methods each with its own appealing properties under the framework of I2RL. This will enable online IRL in various applications. We may easily present the method by Jin et al. [46] within the framework of I2RL. A session of this method $\zeta_i(MDP_{/R_E}, \mathcal{X}_i, \hat{R}_E^{i-1})$ is realized as follows: Each $\mathcal{X}_i$ is a single state-action pair $\langle s, a\rangle$ and initial reward function $\hat{R}_E^0 = \frac{1}{\sqrt{|S_E|}}$. For $i > 0$, $\hat{R}_E^i = \hat{R}_E^{i-1} + \alpha \cdot v_i$, where v_i is the difference in the expected value of the observed action a at state s and the (predicted) optimal action obtained by solving the MDP with

the reward function $\hat{R}_E^{i-1}$, and α is the learning rate. While no explicit stopping criterion is specified, the incremental method terminates when it runs out of observed state-action pairs.

Another I2RL method is DARKO algorithm, used for first-person activity forecasting [69]. Casting the method to the framework of I2RL, a session of this method is $\zeta_i(MDP_{/R_E}, \mathcal{X}_i, \boldsymbol{\theta}^{i-1})$, which yields $\boldsymbol{\theta}^i$. When the person wearing a camera stops the current activity, the stoppage is perceived as reaching a goal state in MDP. Input demonstration for the session, $\mathcal{X}_i$, comprises all the activity trajectories observed since the end of the previous goal until the next goal is reached. The session IRL finds the reward weights $\boldsymbol{\theta}^i$ that minimize the margin $E_{\mathbb{X}}[\phi|\pi_E^*] - \hat{\phi}$ using gradient descent. Here, the expert's policy π_E^* is obtained by using soft value iteration for solving the complete MDP that includes a reward function estimate obtained using previous weights $\boldsymbol{\theta}^{i-1}$. No stopping criterion is utilized for online learning, thereby emphasizing its continuity.

A more recent I2RL method is the deep learning (GAN) based approach of AIRL [35] (explained at the end of Chapter 2). Interpreting AIRL in I2RL framework, each iteration of GAN training can be formulated as a single session $\zeta_i(MDP_{/R_E}, \mathcal{X}_i, \boldsymbol{\theta}^{i-1}, \boldsymbol{\theta}_{\boldsymbol{gen}}^{i-1})$ taking reward-network (learned reward function) weights $\boldsymbol{\theta}$ from the previous session, generator-network (learned policy) weights $\boldsymbol{\theta}_{\boldsymbol{gen}}$ from the previous session, currently input batch $\mathcal{X}_i$ of state-action pairs as inputs. Each session starts by training the discriminator to distinguish input demonstration from the generator's output. Then, the process trains the generator to bring the generator's output closer to the input data (expert's behavior). The outputs of a session are updated weights $\boldsymbol{\theta}^i$, $\boldsymbol{\theta}_{\boldsymbol{gen}}^{\boldsymbol{i}}$ of the two networks. As no explicit stopping criterion is specified, the incremental process terminates when it runs out of batches of observed state-action pairs. A similar formulation is applicable to GAN based method called GAIL [43].

This chapter introduced the formal framework of I2RL, the stopping criteria for online learning, and how related work is captured by the framework. The next four chapters elaborate on our research contributions in the field of online learning of reward function(s) under imperfect input. These chapters go one by one through each type of imperfection. The next chapter focuses on online learning under occlusion.

Chapter 4

Missing data

The first research contribution of this research is a new method that advances the recent progress in maximum entropy IRL with a partially hidden demonstration of data [19] (LMEIRL, Chapter 2 section 'Missing information in training data') by generalizing it to an online setting. We will see later in Chapter 7 that making this offline (batch) method online boosts the convergence speed and, therefore, the success rate in time-limited application domains. In addition to making the method online, we establish key theoretical properties of this new method, which we call *online latent maximum entropy IRL* (LME-I2RL). Specifically, we show that the demonstration data likelihood increases monotonically for this method as more of the demonstration is seen. Consequently, the method shows probabilistic convergence up to a given confidence and within a desired error if a pre- computable amount of training data is available. In an experimental evaluation (Chapter 7), we exploited this convergence property to calculate how much training data must be enough for online learning. Finally, we prove that, with a high confidence, the method exhibits no-regret learning asymptotically as the number of sessions increases. In other words, the average loss across multiple sessions approaches zero with an increasing number of sessions, which is a desirable property for online learning.

4.1 Online MAXENTIRL under Occlusion

This chapter contributes a new method for online IRL, under the I2RL framework, which modifies LME IRL. It offers the capability to perform online IRL in contexts where portions of the observed trajectory may be occluded.

For differentiation, we refer to the original method as the *batch* LME IRL [19]. Recall from Chapter 2 Section 2.2.1 that the k^{th} feature expectation of the expert computed as part of the E-step.

$$\hat{\phi}^{Z|_Y}_{\boldsymbol{\theta},k} \triangleq \frac{1}{|\mathcal{Y}|} \sum_{Y \in \mathcal{Y}} \sum_{Z \in \mathbb{Z}} P(Z|Y;\boldsymbol{\theta}) \sum\nolimits_{t=1}^{T} \gamma^t \phi_k(\langle s, a \rangle_t) \tag{4.1}$$

$\hat{\phi}^{Z|_Y,i}_{\boldsymbol{\theta}^i,k}$ is the expectation of k^{th} feature for the demonstration obtained in i^{th} session, $\hat{\phi}^{Z|_Y,1:i}_{\boldsymbol{\theta}^i,k}$ is the expectation computed for all demonstrations obtained till ith session, we may rewrite Eq. 4.1 for feature k as:

$$
\begin{aligned}
\hat{\phi}^{Z|_Y,1:i}_{\boldsymbol{\theta}^i,k} &\triangleq \frac{1}{|\mathcal{Y}_{1:i}|} \sum_{Y \in \mathcal{Y}_{1:i}} \sum_{Z \in \mathbb{Z}} P(Z|Y;\boldsymbol{\theta}) \sum_{t=1}^{T} \gamma^t \phi_k(\langle s,a \rangle_t) \\
&= \frac{1}{|\mathcal{Y}_{1:i}|} \sum_{Y \in (\mathcal{Y}_{1:i-1} \cup \mathcal{Y}_i)} \sum_{Z \in \mathbb{Z}} P(Z|Y;\boldsymbol{\theta}) \sum_{t=1}^{T} \gamma^t \phi_k(\langle s,a \rangle_t) \\
&= \frac{1}{|\mathcal{Y}_{1:i}|} \Bigg(\sum_{Y \in \mathcal{Y}_{1:i-1}} \sum_{Z \in \mathbb{Z}} P(Z|Y;\boldsymbol{\theta}) \sum_{t=1}^{T} \gamma^t \phi_k(\langle s,a \rangle_t) + \\
&\quad \sum_{Y \in \mathcal{Y}_i} \sum_{Z \in \mathbb{Z}} P(Z|Y;\boldsymbol{\theta}^i) \sum_{t=1}^{T} \gamma^t \phi_k(\langle s,a \rangle_t) \Bigg) \\
&= \frac{1}{|\mathcal{Y}_{1:i-1}| + |\mathcal{Y}_i|} \left(|\mathcal{Y}_{1:i-1}| \, \hat{\phi}^{Z|_Y,1:i-1}_{\boldsymbol{\theta}^{i-1},k} + |\mathcal{Y}_i| \, \hat{\phi}^{Z|_Y,i}_{\boldsymbol{\theta}^i,k} \right) \\
&\text{(Using Eq. 4.1 and } |\mathcal{Y}_{1:i}| = |\mathcal{Y}_{1:i-1}| + |\mathcal{Y}_i|) \\
&= \frac{|\mathcal{Y}_{1:i-1}|}{|\mathcal{Y}_{1:i-1}| + |\mathcal{Y}_i|} \, \hat{\phi}^{Z|_Y,1:i-1}_{\boldsymbol{\theta},k} + \frac{|\mathcal{Y}_i|}{|\mathcal{Y}_{1:i-1}| + |\mathcal{Y}_i|} \, \hat{\phi}^{Z|_Y,i}_{\boldsymbol{\theta},k}
\end{aligned} \tag{4.2}
$$

A session of incremental LME takes as input the expert's MDP sans the reward function, the current session's trajectories, the number of trajectories observed until the previous session, the expert's empirical feature expectation and reward weights from previous session. More concisely, each session is denoted by, $\zeta_i(MDP_{/R_E}, \mathcal{Y}_i, |\mathcal{Y}_{1:i-1}|, \hat{\phi}^{Z|_Y,1:i-1}_{\boldsymbol{\theta}^{i-1}}, \boldsymbol{\theta}^{i-1})$. The sufficient statistic $\hat{X}$ for the session comprises $(|\mathcal{Y}_{1:i-1}|, \hat{\phi}^{Z|_Y,1:i-1}_{\boldsymbol{\theta}^{i-1}})$. In each session, the feature expectations using that session's observed trajectories are computed, and the output feature expectations are obtained by including these as shown above in Eq. 4.2; the latter is used in the M-step. The equation shows how computing sufficient statistic replaces the need for storing the data input in previous sessions. Of course, each session may involve several iterations of the E- and M-steps until the converged reward weights $\boldsymbol{\theta}^i$ are obtained thereby giving the corresponding reward function estimate. We refer to this method as LME I2RL. Appendix A has the link to our github codebase, which has implementation for this method.

Stopping Options There are multiple ways to stop the online learning process. In the experiment applications we focused on, it made sense that the learner kept learning until the demonstrations keep coming. But, for other applications, IRL practitioner can choose to stop learning earlier if the log-likelihood has stopped changing significantly, regardless of availability more input demonstrations.

Wang et al. [84] shows that if the distribution over the trajectories is log linear, then the reward function that maximizes the entropy of the distribution over trajectories also maximizes the log likelihood of the

observed portions of the trajectories. Given this linkage with log likelihood, we show an algorithm using the stopping criterion #1 as given in Chapter 3. As shown in Algorithm 2, the sessions will terminate when, $|LL(\boldsymbol{\theta}^i|\mathcal{Y}_i, |\mathcal{Y}_{1:i-1}|, \hat{\phi}_{\boldsymbol{\theta}^{i-1}}^{Z|_Y,1:i-1}, \boldsymbol{\theta}^{i-1}) - LL(\boldsymbol{\theta}^{i-1}|\mathcal{Y}_{i-1}, |\mathcal{Y}_{1:i-2}|, \hat{\phi}_{\boldsymbol{\theta}^{i-2}}^{Z|_Y,1:i-2}, \boldsymbol{\theta}^{i-2})| \leq \rho$, where $\boldsymbol{\theta}^i$ fully parameterizes the reward function estimate for the i^{th} session and ρ is a given acceptable difference.

Avoiding early stopping However, there should be additional precaution to avoid early stopping due to a temporary drop in chosen stopping metric (say livelihood). One can use an upper-bound on stddev values of log-likelihood computed over a moving window of iterations. Bigger the size of this window, less are the chances of early stopping. Precautions like these also avoid the cases like p-hacking where convergence is an artifact of the specific data, i.e. happens by chance by chance rather than merit.

Algorithm 1 Algorithm INCREMENTAL-LME($MDP_{/R_E}$,ϕ)

$i \leftarrow 1; \mathcal{Y}_{1:i-1} \leftarrow \emptyset$
$\hat{\phi}_{\boldsymbol{\theta}^{i-1},k}^{Z|_Y,1:i-1} \leftarrow 0; [\boldsymbol{\theta}^0]_k \sim \text{uniform}(0,1)$
while $|LL(\mathcal{Y}_i, |\mathcal{Y}_{i-1}|, \hat{\phi}_{\boldsymbol{\theta}^{i-1}}^{Z|_Y,1:i-1}, \boldsymbol{\theta}^i) - LL(\mathcal{Y}_{i-1}, |\mathcal{Y}_{i-2}|, \hat{\phi}_{\boldsymbol{\theta}^{i-2}}^{Z|_Y,1:i-2}, \boldsymbol{\theta}^{i-1})| > \rho$ **do**
  `/* session` $\zeta_i(M_{/R_E}, \mathcal{Y}_i, |\mathcal{Y}_{1:i-1}|, \hat{\phi}_{\boldsymbol{\theta}^{i-1}}^{Z|_Y,1:i-1}, \boldsymbol{\theta}^{i-1})$ `*/`
  repeat
    `/* E-step */`
    Use MCMC to sample trajectories from $P((Y,Z)|\boldsymbol{\theta}^{i-1})$, and compute $\hat{\phi}_{\boldsymbol{\theta}^i}^{Z|_Y,i}$ for sampled trajectories.
    `/* Updating feature expectations using sufficient statistic. */`
    Use Equation 4.2 to compute $\hat{\phi}_{\boldsymbol{\theta}^i,k}^{Z|_Y,1:i}$ for all k.
    $|\mathcal{Y}_{1:i}| \leftarrow |\mathcal{Y}_{1:i-1}| + |\mathcal{Y}_i|$
    `/* M-step */`
    $\boldsymbol{\theta}_0 \leftarrow \boldsymbol{\theta}^{i-1}, t \leftarrow 1$
    repeat
      Compute $\pi^*_{E,(t-1)}$ using $\boldsymbol{\theta}_{(t-1)}$ and $E_{\mathbb{X}}[\phi_k]$ using trajectories sampled from $\pi^*_{E,(t-1)}$.
      $z_{(t-1)} \leftarrow \hat{\phi}_{\boldsymbol{\theta}^i}^{Z|_Y,1:i} - E_{\mathbb{X}}[\phi]$ {gradient}
      $\boldsymbol{\theta}_{t,k} \leftarrow \frac{\boldsymbol{\theta}_{(t-1),k} \exp(-\eta z_{(t-1),k})}{\sum_{i=1}^{k} \boldsymbol{\theta}_{(t-1),k} \exp(-\eta z_{(t-1),k})}$
      $t \leftarrow t+1$
    until $|z| \leq \varepsilon/(1-\gamma)$
  until gradient of likelihood ≈ 0
  Compute $\hat{\pi}_i$ using learned reward $\boldsymbol{\theta}^i \leftarrow \boldsymbol{\theta}_t$.
  $i \leftarrow i+1$

4.1.1 Convergence Bounds

LME I2RL admits some significant convergence guarantees with a confidence of meeting the specified error on the demonstration likelihood. We defer the proofs of these results to Appendix B. To establish

the guarantees of LME I2RL, we first focus on the full observability setting. For a desired relaxed bound ε on the feature matching constraint (see line 18 in Algorithm 2) for session i, the confidence is bounded as follows:

Theorem 1 (Confidence for ME I2RL)**.** *Given $\mathcal{X}_{1:i}$ as the (fully observed) demonstration till session i, $\boldsymbol{\theta}_E \in [0,1]^K$ is the expert's weights, and $\boldsymbol{\theta}^i$ is the converged weight vector for session i for ME I2RL, we have,*

$$LL(\boldsymbol{\theta}_E|\mathcal{X}_{1:i}) - LL(\boldsymbol{\theta}^i|\mathcal{X}_i, |\mathcal{X}_{i-1}|, \hat{\phi}^{1:i-1}, \boldsymbol{\theta}^{i-1}) \leqslant \frac{2K\varepsilon}{(1-\gamma)}$$

with probability atleast $\max(0, 1-\delta)$*, where* $\delta = 2K\exp(-2|\mathcal{X}_{1:i}|\varepsilon^2)$.

Note that sufficient statistic $\hat{X}$ for full-observability scenario is $(|\mathcal{X}_{i-1}|, \hat{\phi}^{1:i-1})$.

Relaxing the full observability assumption, the following lemma proves that LME I2RL converges monotonically.

Lemma 1 (Monotonicity)**.** LME I2RL increases the demonstration likelihood monotonically with each new session, $LL(\boldsymbol{\theta}^i|\mathcal{Y}_i, |\mathcal{Y}_{1:i-1}|, \hat{\phi}^{Z|_Y,1:i-1}_{\boldsymbol{\theta}^{i-1}}, \boldsymbol{\theta}^{i-1}) - LL(\boldsymbol{\theta}^{i-1}|\mathcal{Y}_{i-1}, |\mathcal{Y}_{1:i-2}|, \hat{\phi}^{Z|_Y,1:i-2}_{\boldsymbol{\theta}^{i-2}}, \boldsymbol{\theta}^{i-2}) \geqslant 0$, when $|\mathcal{Y}_{1:i-1}| \gg |\mathcal{Y}_i|$.

Lemma 7 suggests that the log likelihood of the demonstration can only improve from session to session after learner has accumulated a significant amount of observations.

Following result illuminates the confidence with which I2RL under occlusion can minimize the log likelihood error. Please consider the error in approximating the feature expectations of the unobserved portions of the data, accumulated from the first to the current session of I2RL. Notice that $\hat{\phi}^{Z|_Y,1:i}_{\boldsymbol{\theta}^i,k}$ given by Eq. 4.2 is an approximation of the expectation $\hat{\phi}^{1:i}_k$ under full-observability , computed by sampling the hidden Z from $P(Z|Y,\boldsymbol{\theta}^{i-1})$ [19]. The following lemma relates the error in this sampling-based approximation to the difference between feature expectations for learned policy and that estimated for the expert's true policy.

Lemma 2 (Constraint Bounds for LME I2RL)**.** Suppose $\mathcal{X}_{1:i}$ has portions of trajectories in $\mathbb{Z}_{1:i} = \{Z|(Y,Z) \in \mathcal{X}_{1:i}\}$ occluded from the learner. Let ε_s be a bound on the error $\left|\hat{\phi}^{1:i}_k - \hat{\phi}^{Z|_Y,1:i}_{\boldsymbol{\theta}^i,k}\right|_1, k \in \{1,2\ldots K\}$ after n_s samples for approximation. Then, with probability at least $\max(0, 1-(\delta+\delta_s))$, the following holds:

$$\left|E_{\mathbb{X}}[\phi_k] - \hat{\phi}^{Z|_Y,1:i}_{\boldsymbol{\theta}^i,k}\right|_1 \leqslant \varepsilon + \varepsilon_s, k \in \{1,2\ldots K\}$$

where ε, δ are as defined in Theorem 3, and $\delta_s = 2K\exp(-2n_s\varepsilon_s^2)$.

LME I2RL computes θ^i by an optimization process using the result $\phi^{Z|_Y,i}$ of E step (sampling of occluded data) of current session along with other inputs (feature expectations and θ computed from previous session) which, in turn, depend on sampling process in previous sessions. Theorem 3 and Lemma 6 allows us to probabilistically bound the error in log likelihood for LME I2RL:

Theorem 2 (Confidence for LME I2RL)**.** *Let $\mathcal{Y}_{1:i} = \{Y|(Y,Z) \in \mathcal{X}_{1:i}\}$ be the observed portions of the demonstration until session i. ε and ε_s are inputs as defined in Lemma 6, and $\boldsymbol{\theta}^i$ is the solution of session i for LME I2RL. Then*

$$LL(\boldsymbol{\theta}_E|\mathcal{Y}_{1:i}) - LL(\boldsymbol{\theta}^i|\mathcal{Y}_i, |\mathcal{Y}_{i-1}|, \hat{\phi}^{Z|_Y,1:i-1}_{\boldsymbol{\theta}^{i-1}}, \boldsymbol{\theta}^{i-1}) \leq \frac{4K\varepsilon_l}{(1-\gamma)}$$

with confidence at least $\max(0, 1-\delta_l)$*, where* $\varepsilon_l = \frac{\varepsilon+\varepsilon_s}{2}$*, and* $\delta_l = \delta + \delta_s$*.*

Given $\varepsilon, \varepsilon_s, N$ and the total number of input partial -trajectories, $|\mathcal{Y}_{1:i}|$, Theorem 4 gives the confidence $1-\delta_l$ for I2RL under occlusion. Equivalently, $|\mathcal{Y}_{1:i}|$ can be derived using desired error bounds and confidence. As a boundary case of LME I2RL, if learner ignores occluded data (no sampling or $n_s = 0$ for E-step), the confidence for convergence becomes zero because δ_s becomes larger than 1. Please note that the above two theorems are based on **assumption** that, for each feature-pair in K reward features, the expectation of one feature computed over a trajectory is independent from the expectation of another feature over same trajectory. This is the basis of applying Hoeffding inequality in the proofs.

A desirable feature is for an online learning algorithm to be *no-regret*, i.e., where the average regret vanishes in the limit. In the context of ME I2RL, we follow our definition of average regret (Chapter 3, Definition of Regret) up to session $i = T$ as

$$\textit{Regret}_T(A_{ME\text{I2RL}}) = \frac{1}{T}\sum_{i=1}^{T} LL(\boldsymbol{\theta}_E|\mathcal{X}_{1:i}) - \frac{1}{T}\sum_{i=1}^{T} LL(\boldsymbol{\theta}^i|\mathcal{X}_i, |\mathcal{X}_{i-1}|, \hat{\phi}^{1:i-1}, \boldsymbol{\theta}^{i-1})$$

This is the regret that I2RL experiences in hindsight, in terms of log-likelihood loss, for returning $\boldsymbol{\theta}^i$ instead of $\boldsymbol{\theta}_E$ after session i, $i = 1, \ldots, T$, averaged over the T sessions. Theorem 3 implies that with a high confidence, the above expression for average regret is constant bounded, specifically by $2K\epsilon/(1-\gamma)$. However, By setting a diminishing (variable) threshold in line 18 of Algorithm 2, the total regret can be made to grow slower than T, so that the average regret vanishes in the limit (as $T \to \infty$). One such choice (by no means unique) for a variable ε for line 18 is

$$\varepsilon_i/(1-\gamma) = c/i \tag{4.3}$$

where c is some constant and i is the session index. This choice ensures that with a high confidence, $\textit{Regret}_T(A_{ME\text{I2RL}})$ is $O(\log T)/T$, which indeed vanishes in the limit. We formalize this intuition for the fully observable setting (i.e., in the context of Theorem 3; a similar result follows in the context of Theorem 4 as well) in the following theorem.

Theorem 3 (No Regret Learning)**.** *There exist choices for a variable threshold bound on the feature matching constraint, ε_i as a function of session i, such that with probability at least* $\max(0, \prod_{i=1}^{i=T}(1-\delta_i))$*, where* $\delta_i = 2K\exp(-2|\mathcal{X}_{1:i}|\varepsilon_i^2)$*,*

$$\text{Regret}_T(A_{MEI2RL}) = o(T)/T,$$

thus approaching 0 as $T \to \infty$*.*

Note that the above theorem assumes that line 18 of Alg. 2 exits in *every* session. If this does not occur in some session, for instance if the algorithm gets stuck in a local optima which becomes increasingly likely as the threshold in equation 4.3 tightens, then the theorem does not apply.

In contrast with Theorem 3, this theorem takes a long term view of the learner's performance. The theorem itself demands a gradually improving performance by this measure (due to Equation 4.3), while Theorem 3 merely improves the confidence (δ) on the learner's performance. The price for demanding greater long term accuracy is the diminishing schedule of ϵ, which may become increasingly harder to meet.

Generalization to Existing Methods: Theorem 3 holds for the online method by Rhinehart et al. [69] because it uses incremental (full-observability) maximum entropy IRL. As the latter implements online learning without an incremental update of feature expectations of the expert, thus set $\hat{\phi}^{1:i} = \hat{\phi}^{i}$, an absence of sufficient statistic, set $|\mathcal{X}_{i-1}| = 0$, and set $\hat{\phi}_k^{1:i-1} = 0, \forall k$ in Theorem 3. This demonstrates the benefit of Theorem 3 to relevant methods.

The current chapter introduced a method LME-I2RL to address the lack of online learning of reward function under demonstration with missing data. We proved that if an expert provides the learner with a pre- computable amount of partial data, LME-I2RL converges monotonically within a given error bound with a given confidence. We also showed that the method learns with 'no regret in the limit' if we set a convergence threshold value that diminishes with every subsequent session. The next chapter shows a new method 'online MME-MTIRL', which can learn online multiple reward function(s) using a training dataset comprising trajectories mixed up from a diverse set of preferences (different reward functions to finish the same task).

Chapter 5

Mixed data from different preferences

This chapter focuses on addressing the second kind of imperfection in input data used in online learning applications. Traditional methods of IRL assumed that an expert's 'preference of finishing the task' can be modeled uniquely via a single reward function and learning can be done offline. As we explained in Chapter 1, this is not always true. One task of sorting onions can be done in multiple ways depending on whether the sorter prioritizes the precision of sorting over the speed of sorting or vice-versa. The demonstration observed by the learner may have trajectories from different reward functions mixed together and the learner's goal is to infer all these reward functions distinctly, and incrementally using I2RL sessions. That motivation led to the creation of a multi-task online learning approach, online Min-Max Entropy Multi-Task IRL or online MME-MTIRL.

We present a new method for multi-task IRL that generalizes the maximum entropy IRL (MaxEntIRL) [88] to multi-task learning. Figure 5.1 shows the overview of using this method. After formulating a batch version of this method, we extend it to online learning which, in turn, helps reduce learning time in time-limited domains.

5.1 Min-Max Entropy Multi-Task IRL

Ziebart et al. [88] notes a key benefit of the MaxEnt distribution over trajectories over the distribution $F(\theta^*_{c_i})$ mentioned in Background chapter (which is the prior over trajectories utilized in the Bayesian formulation for IRL [64]), despite their initial similarities. Specifically, the latter formulation, which decomposes the trajectory into its constituent state-action pairs and obtains the probability of each state-action as proportional to the exponentiated Q-function, is vulnerable to the *label bias*. Due to the locality of the action probability computation, the distribution over trajectories is impacted by the number of action choice points (branching) encountered by a trajectory. On the other hand, the MaxEnt distribution does not suffer from this bias. A major consequence of this bias is that Bayesian methods may not assign higher likelihoods to trajectories that have higher rewards, but MaxEnt does. To illustrate this distinction,

we conducted experiments in a 10×10 Object World [55] with two desirable objects at corners, and randomly placed walls. We noted the correlation coefficients (ρ) between the variables A (log-likelihood of a trajectory as assigned by the method) and B (total reward of a trajectory) for 75 expert trajectories. While ρ_{AB} is 1 for MaxEnt due to a strictly linear relation between A and B, it is 0.1771 (p-value 0.2914) for Bayesian IRL, showing no significant correlation between A and B.

Table 5.1 shows the correlations between the log-likelihoods and the total rewards of the expert trajectories, from BIRL and MaxEnt. While the log-likelihood of a trajectory from MaxEnt is strictly linearly related to its total reward (hence the correlation coefficient of 1), there is no significant correlation between these variables for BIRL.

Table 5.1: Correlations coefficients, ρ_{AB}, between variables A= log-likelihood of a trajectory, and B= total reward of a trajectory.

ρ_{AB} for BIRL	p-value for BIRL	ρ_{AB} for MaxEnt	p-value for MaxEnt
0.1771	0.2914	1.0	0.0

This important observation motivates a new method that combines the non-parametric clustering of trajectories and the learning of multiple reward functions by finding trajectory distributions of maximum entropy. This method has the benefit of avoiding label bias.

5.1.1 Unified Optimization

A straightforward approach to the combination would be to replace the parametric distribution in MaxEntIRL with $F(\theta^*_{c_i})$ of DP-based mixture model (please refer Chapter 2 for the details of DPM), and the distribution over the trajectories with cluster assignment value c_i. Solving the nonlinear program will yield parameter $\theta^*_{c_i}$ that maximizes the entropy of $F(\theta^*_{c_i})$. Though simple, this approach is inefficient because it requires solving the MaxEnt program repeatedly – each time the DP-based mixture model is updated.

Instead, we pursue an approach that adds key elements of the DP-based mixture modeling to the nonlinear program of Max Entropy optimization. Maximum entropy optimization can learn those parameters $\{\theta_d\}$ (these are the Lagrangian multipliers) which maximize the entropy of the component (cluster) distribution $F(\theta_d)$ over the trajectories $\{X_i\}$ with cluster assignment $c_i = d$. Subsequently, each component distribution F assumes the form of an exponential-family distribution parameterized by θ_d, which is known to exhibit the maximum entropy. For our DPM model, the distribution G is the mixture $\sum_{d=1}^{D} \pi_d \delta_{\theta^*_d}$. We present distribution $F(\theta_d)$ as $P(\cdot|c_i = d)$ and we use $D = \{1, 2, \ldots D\}, K = \{1, 2, \ldots K\}$ for brevity. To the goal of maximizing the entropy of each cluster-specific distribution $P(\cdot|c_i = d)$, we add a second goal of finding component weights $\boldsymbol{\pi} = \{\pi_d\}_{d \in D}$ that exhibit a minimal entropy. The effect of this second objective is to learn a minimal number of distinct clusters. Formally, the two optimization programs are written separately as

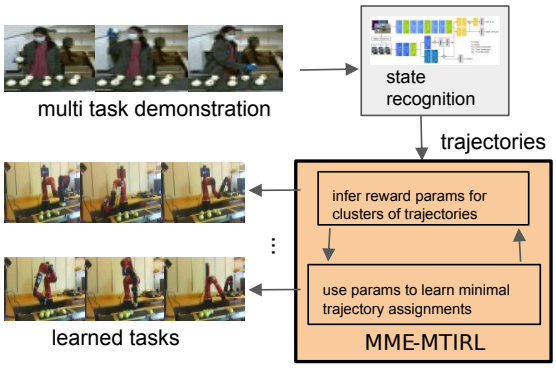


Figure 5.1: States are identified using SA-Net [76] from the human demonstration of onion sorting, and the trajectories are given as input to MME-MTIRL method. Learned tasks are demonstrated using the Sawyer cobot.

$$\max_{P(X_i|c_i)\in\Delta} -\sum_{d=1}^{D}\sum_{i=1}^{|\mathbb{X}|} P(X_i|c_i=d)\log(P(X_i|c_i=d))$$

subject to

$$\sum_{d}\sum_{X_i\in\mathbb{X}} P(X_i, c_i=d) = 1$$

$$E_{\mathbb{X}}[\phi_k|c_i=d] = \hat{\phi}_{d,k} \quad \forall d\in D, \forall k\in K$$

$$\min_{\boldsymbol{\pi}\in\Delta} \sum_{d=1}^{D} \pi_d \log(\pi_d)$$

subject to

$$\sum_{d=1}^{\mathcal{D}} \pi_d = 1$$

Here $P(X_i|c_i=d)$ can be written as $\delta_d(c_i)Pr_d(X_i)$ where $\delta_d(c_i)$ is the Kronecker delta taking a value of 1 when $c_i=d$, and 0 otherwise, and $Pr_d(X_i)$ is the distribution over all the trajectories for cluster d.

As we want to unify the problem into a single optimization program, we combine the above two programs into one program, a unified nonlinear optimization problem given by

$$\max_{Pr_d(X_i)\in\Delta^D,\boldsymbol{\pi}\in\Delta} -\sum_d \sum_{i=1}^{|\mathbb{X}|} \delta_d(c_i) Pr_d(X_i) \tag{5.1}$$
$$\log(\delta_d(c_i) Pr_d(X_i)) + \sum_d \pi_d \log(\pi_d)$$

subject to

$$\sum_d \sum_{X_i\in\mathbb{X}} P(X_i, c_i = d) = 1$$
$$E_{\mathbb{X}}[\phi_k | c_i = d] = \hat{\phi}_{d,k} \quad \forall d \in D, \forall k \in K$$
$$\sum_{d=1}^{\mathcal{D}} \pi_d = 1 \tag{5.2}$$

The first constraint above simply ensures that the joint probability distribution sums to 1. The second constraint makes the analogous constraint in MaxEntIRL more specific to matching expectations of feature functions that belong to the reward function of cluster d. Here,

$$E_{\mathbb{X}}[\phi_k | c_i = d] = \sum_{i=1}^{|\mathbb{X}|} P(X_i, c_i = d) \sum_{(s,a)\in X_i} \phi_k(s,a)$$
$$= \sum_{i=1}^{|\mathbb{X}|} P(X_i | c_i = d) P(c_i = d) \sum_{(s,a)\in X_i} \phi_k(s,a)$$
$$= \pi_d \sum_{i=1}^{|\mathbb{X}|} \delta_d(c_i) Pr_d(X_i) \sum_{(s,a)\in X_i} \phi_k(s,a),$$

Last equality follows from $P(X_i|c_i = d) = \delta_d(c_i) Pr_d(X_i)$ and $\pi_d = P(c_i = d)$ since G is a mixture distribution. The estimated feature expectation of the trajectories assigned to cluster d can be written as

$$\hat{\phi}_{d,k} = \frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} \delta_d(c_i) \sum_{(s,a)\in X_i} \phi_k(s,a).$$

Constraint 3 of the program in (5.2) ensures that the mixture weights are convex. Recall that the DP-based mixture model obtains cluster assignment c_i from mixture weights $\boldsymbol{\pi}$. We may approximate this simulation of c_i simply as $\pi_d = \frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} \delta_d(c_i)$, which is the proportion of observed trajectories currently assigned to cluster c_i. For notational convenience, let us denote $\delta_d(c_i)$ as indicator $v_{d,i}$. We may then rewrite the first constraint as

$$\sum_d \sum_{i=1}^{|\mathbb{X}|} P(X_i, c_i = d) = 1$$
$$\Leftrightarrow \sum_d \pi_d \sum_{i=1}^{|\mathbb{X}|} P(X_i | c_i = d) = 1$$
$$\Leftrightarrow \sum_d \pi_d \sum_{i=1}^{|\mathbb{X}|} \delta_d(c_i) Pr_d(X_i) = 1$$
$$\Leftrightarrow \sum_d \frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \sum_{i=1}^{|\mathbb{X}|} v_{d,i} Pr_d(X_i) = 1$$

and the second constraint is rewritten as,

$$E_{\mathbb{X},d}[\phi_k] = \frac{\sum_{i=1}^{|\mathcal{X}|} v_{d,i}}{|\mathcal{X}|} \sum_{i=1}^{|\mathbb{X}|} v_{d,i} Pr_d(X_i) \sum_{(s,a)\in X_i} \phi_k(s,a) \tag{5.3}$$

while

$$\hat{\phi}_{d,k} = \frac{1}{|\mathcal{X}|} \sum\nolimits_{i=1}^{|\mathcal{X}|} v_{d,i} \sum\nolimits_{(s,a)\in X_i} \phi_k(s,a). \tag{5.4}$$

Furthermore, we may expand the third constraint of the nonlinear program as follows:

$$\begin{aligned}
\sum\nolimits_{d=1}^{\mathcal{D}} \pi_d = 1 &\Leftrightarrow \sum\nolimits_{d=1}^{\mathcal{D}} \frac{1}{|\mathcal{X}|} \sum\nolimits_{i=1}^{|\mathcal{X}|} v_{d,i} = 1 \\
&\Leftrightarrow \sum\nolimits_{i=1}^{|\mathcal{X}|} \sum\nolimits_{d=1}^{\mathcal{D}} v_{d,i} = |\mathcal{X}| \\
&\Leftrightarrow \sum\nolimits_{d=1}^{\mathcal{D}} v_{d,i} = 1,\ \forall i
\end{aligned}$$

The last equivalence follows from the fact that every observed trajectory must belong to a single cluster, and $v_{d,i} \in \{0,1\}$. The final form of the NLP of (5.2) is as follows.

$$\begin{aligned}
&\max_{Pr_d(X_i)\in\Delta^D, \boldsymbol{v}_d\in\{0,1\}^{|\mathbb{X}|}} - \sum_d \sum_{i=1}^{|\mathbb{X}|} v_{d,i} Pr_d(X_i) \log(v_{d,i} Pr_d(X_i)) \\
&+ \frac{1}{|\mathcal{X}|} \sum_d \sum_{i=1}^{|\mathcal{X}|} v_{d,i}\ \log\left(\frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i}\right)
\end{aligned}$$

subject to

$$\begin{aligned}
&\sum\nolimits_d \frac{1}{|\mathcal{X}|} \sum\nolimits_{i=1}^{|\mathcal{X}|} v_{d,i} \sum\nolimits_{i=1}^{|\mathbb{X}|} v_{d,i} Pr_d(X_i) = 1 \\
&E_{\mathbb{X},d}[\phi_k] = \hat{\phi}_{d,k} \quad \forall d \in D, \forall k \in K \\
&\sum\nolimits_d v_{d,i} = 1, \forall i \in \{1,\ldots,|\mathbb{X}|\}
\end{aligned} \tag{5.5}$$

where $E_{\mathbb{X},d}[\phi_k]$, $\hat{\phi}_{d,k}$ are defined in Eqs. 5.3 and 5.4.

5.1.2 Gradient Descent

The Lagrangian dual for the nonlinear program in (5.5) is optimized as $\max_{Pr_d, \boldsymbol{v}_d} \min_{\eta, \boldsymbol{\theta}_d, \boldsymbol{\lambda}} \mathcal{L}$ with

$$
\begin{aligned}
\mathcal{L} = & \left(- \sum_d \sum_{i=1}^{|\mathbb{X}|} v_{d,i} Pr_d(X_i) \ \log(v_{d,i} Pr_d(X_i)) \right) \\
& + \left(\frac{1}{|\mathcal{X}|} \sum_d \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \ \log \left(\frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \right) \right) \\
& + \eta \left(\sum_d \left(\frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \right) \sum_{i=1}^{|\mathbb{X}|} v_{d,i} Pr_d(X_i) - 1 \right) \\
& + \sum_{d,k} \boldsymbol{\theta}_{d,k} \Bigg(\left(\frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \right) \sum_{i=1}^{|\mathbb{X}|} v_{d,i} Pr_d(X_i) \\
& \sum_{(s,a) \in X_i} \phi_k(s,a) - \frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \sum_{(s,a) \in X_i} \phi_k(s,a) \Bigg) \\
& + \sum_{i=1}^{|\mathcal{X}|} \lambda_i \left(\sum_d v_{d,i} - 1 \right)
\end{aligned} \tag{5.6}
$$

where the multipliers $\eta, \{\lambda_i\}_{X_i \in \mathbb{X}}$ can be substituted by using relations derived from equating the derivatives of $\mathcal{L}$ w.r.t. the variables of optimization to 0. We show the derivations of these gradients in Appendix D. The target is to learn the multipliers $\boldsymbol{\theta}_d$ (weights for the linear reward function for each learned cluster d) and the variables $v_{d,i}$ (for each trajectory $X_i \in \mathbb{X}$) that achieve $\max_{Pr_d, \boldsymbol{v}_{d,i}} \min_{\theta_{d,k}} \mathcal{L}$. We achieve the target via gradient ascent for $v_{d,i}$ and descent for $\theta_{d,k}$ using the following partial derivatives:

$$
\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \theta_{d,k}} &= E_{\mathbb{X},d}[\phi_k] - \hat{\phi}_{d,k} \\
\frac{\partial \mathcal{L}}{\partial v_{d,i}} &= \frac{\left(\sum_{i=1}^{|\mathbb{X}|} P(X_i | c_i = d) + 1 + \frac{|\mathcal{X}|}{v_{d,i}} (1 - \log Z(\theta_{d,k})) \right)}{\sum_{i=1}^{|\mathcal{X}|} v_{d,i}}
\end{aligned}
$$

where $P(X_i | c_i = d) = \frac{\exp\left(\pi_d \sum_{k=1}^{K} \theta_{d,k} \sum_{(s,a) \in X_i} \phi_k(s,a)\right)}{Z(\theta_{d,k})}$ and $Z(\theta_{d,k}) = \sum_{d \in \mathcal{D}} \pi_d \sum_{i=1}^{|\mathbb{X}|} \exp\left(\pi_d \sum_{k=1}^{K} \theta_{d,k} \sum_{(s,a) \in X_i} \phi_k(s,a)\right)$. The first derivative is the same as that used for single-task MaxEntIRL. The second derivative indicates that the chances of change in assignment is less if a cluster has many trajectories assigned to it (inversely proportional to $\sum_{i=1}^{|\mathcal{X}|} v_{d,i}$) and has a higher likelihood of generating trajectories. We approximate $E_{\mathbb{X},d}[\phi_k]$ as a running average over feature expectations of the trajectories generated by the policy computed using $\boldsymbol{\theta}_d$, which are the reward weights learned for cluster d in the current iteration of gradient descent. The code implementation for the method is available in the github link shared in Appendix A.

5.2 Extension from batch to online learning

As we are using index i for trajectories, let's use index j for I2RL session. In session j, sufficient statistic from previous session represents the information learnt until session $j-1$. This statistic includes reward weights $\theta_{d,k}^{j-1}$ for all D clusters, weights π_d^{j-1} for all clusters, and total size $|\mathcal{X}^{1:j-1}|$ of cumulative demonstration from session 1 to session $j-1$. In session j, learner computes the cluster weights and cluster specific feature expectations incrementally by using following equations

$$\begin{aligned}\pi_d^j &= \frac{1}{|\mathcal{X}^{1:j}|}\sum\nolimits_{i=1}^{|\mathcal{X}^{1:j}|} v_{d,i} \\ &= \frac{1}{|\mathcal{X}^{1:j-1}|+|\mathcal{X}^j|}\left(\sum\nolimits_{i=1}^{|\mathcal{X}^{1:j-1}|} v_{d,i} + \sum\nolimits_{i=1}^{|\mathcal{X}^j|} v_{d,i}\right) \\ &= \frac{1}{|\mathcal{X}^{1:j-1}|+|\mathcal{X}^j|}\left(\pi_d^{j-1}\times|\mathcal{X}^{1:j-1}| + \sum\nolimits_{i=1}^{|\mathcal{X}^j|} v_{d,i}\right)\end{aligned} \tag{5.7}$$

$$\begin{aligned}\hat{\phi}_{d,k}^{1:j} &= \frac{1}{|\mathcal{X}^{1:j}|}\sum\nolimits_{i=1}^{|\mathcal{X}^{1:j}|} v_{d,i}\sum\nolimits_{(s,a)\in X_i}\phi_k(s,a) \\ &= \frac{1}{|\mathcal{X}^{1:j}|}\left(\sum\nolimits_{i=1}^{|\mathcal{X}^{1:j-1}|} v_{d,i}\sum\nolimits_{(s,a)\in X_i}\phi_k(s,a) + \sum\nolimits_{i=1}^{|\mathcal{X}^j|} v_{d,i}\sum\nolimits_{(s,a)\in X_i}\phi_k(s,a)\right) \\ &= \frac{1}{|\mathcal{X}^{1:j}|}\left(|\mathcal{X}^{1:j-1}|\times\hat{\phi}_{d,k}^{1:j-1} + |\mathcal{X}^j|\times\hat{\phi}_{d,k}^{j}\right)\end{aligned} \tag{5.8}$$

Every session of online learning solves a problem of batch learning. So the expressions for computing gradients extend as is from batch learning. As information from previous sessions is already assimilated in π_d^j and $\hat{\phi}_{d,k}^{1:j}$, the gradients for session j are computed w.r.t only the trajectories in $\mathcal{X}^j$.

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\theta_{d,k}^j} &= E_{\mathbb{X},d}[\phi_k] - \hat{\phi}_{d,k}^{1:j} \\ \frac{\partial\mathcal{L}}{\partial v_{d,i}} &= \frac{\left(\sum_{i=1}^{|\mathbb{X}|} P^j(X_i|c_i=d) + 1 + \frac{|\mathcal{X}^j|}{v_{d,i}}(1-\log Z(\theta_{d,k}))\right)}{\sum_{i=1}^{|\mathcal{X}^j|} v_{d,i}}\end{aligned}$$

where $P^j(X_i|c_i=d) = \frac{\exp\left(\pi_d^j\sum_{k=1}^K\theta_{d,k}^j\sum_{(s,a)\in X_i}\phi_k(s,a)\right)}{Z(\theta_{d,k}^j)}$ for every trajectory $X_i\in\mathcal{X}^j$ and $Z(\theta_{d,k}^j) = \sum_{d\in\mathcal{D}}\pi_d^j\sum_{i=1}^{|\mathbb{X}|}\exp\left(\pi_d^j\sum_{k=1}^K\theta_{d,k}^j\sum_{(s,a)\in X_i}\phi_k(s,a)\right)$.

In a session $\zeta_j(MDP_{/R}, \mathcal{X}^j, \{\boldsymbol{\theta}_d^{j-1}\}_{\forall d\in D}, \{\boldsymbol{\pi}_d^{j-1}\}_{\forall d\in D}, |\mathcal{X}^{1:j-1}|)$ of online MME-MTIRL, for each cluster d, the learner starts reward function weights with values from the previous session ($\boldsymbol{\theta}_d^{j-1}$) and then uses gradient descent for updating the weights to θ_d^j. In parallel, the process starts cluster membership indicators ($v_{d,i}$ for each trajectory $X_i\in\mathcal{X}^j$) with random values and update them over iterations of gradient ascent process.

The current chapter introduced a new method for online learning using training data with trajectories coming from different policies corresponding to multiple reward functions. Modeling the problem of min-max entropy multi-task IRL as a unified optimization program enables the direct application of well-studied optimization algorithms (e.g. fast gradient-descent) to the problem. In particular, we derive the gradients of the Lagrangian relaxation of the nonlinear program, which then facilitates the use of

fast gradient-descent algorithm in every session of learning. The next chapter addresses the problem of learning online when learner's perception mechanism is noisy (a different kind of imperfection).

Chapter 6

Noise in the perception of learner

The last two chapters emphasized on the imperfections of missing data and mixed-up data. The motivation for this chapter is to address the imperfection of the noise caused by a faulty perception of an online learner. In real-life applications, due to hardware limitations or due to other uncontrollable external perturbations, it is common that the learner can not understand the information in a state-action pair the same way an expert demonstrated that pair. With that focus in mind, we divide this chapter into two parts. The first part introduces a method RIMEO that can learn online with noisy training data without the need for prior knowledge of an observation model (sensor model). We also derive significant convergence properties of RIMEO. However, RIMEO and the methods introduced in previous chapters rely on the manual creation of reward model features. To overcome that limitation, the second part of the chapter borrows inspiration from a deep-learning based online method AIRL and introduces online method of robust-AIRL. To explain these two methods, we must first go through some basic mathematical concepts related to observation model and robustness under noise. The next few paragraphs discuss the technical background for a noise robust batch IRL method from literature, and the relationship between the prediction scores from neural-network based perception and the observation model features.

There are multiple ways to deploy a perception mechanism for the learner. An observation can be some aspect of a state, or a complete state, or even a state-action pair in the trajectory. We use a neural network architecture–SA-Net [76]–that receives an image-stream of the expert's behavior and detects the sequence of (s, a) pairs from it. As the output of this observation process is a (s, a) pair, we consider the observation model to be a mapping that takes an observed state-action pair ($\langle s, a\rangle_o$) and a demonstrated (ground truth) state-action pair ($\langle s, a\rangle_g$) as input and gives the likelihood of observing the former pair given the latter [1].

Let $\iota_{sa} = \{\langle s, a\rangle_i \,|i \in \mathcal{N}_{sa} = \{1, 2 \ldots |S| \times |A|\}\}$ be an enumerated space for all possible (s, a) pairs. We define the observation model as a function that gives the likelihood of a perceived state-action

[1] Although we use a state-action pair as an observation, the concept of robust-IRL is generalizable to any form of observation [74].

pair $\langle s, a\rangle_o$ given the demonstrated (ground truth) state-action pair $\langle s, a\rangle_g$ [2]. That is, $O_b : |S||A| \times \iota_{sa} \rightarrow [0, 1]$ where

$$O_b(\langle s, a\rangle_o, \langle s, a\rangle_g) \triangleq P(\langle s, a\rangle_o \,|\, \langle s, a\rangle_g) \tag{6.1}$$

where $g, o \in \mathcal{N}_{sa}$. RHS is the probability of the learner observing $\langle s, a\rangle_o$ when the expert actually demonstrated $\langle s, a\rangle_g$.

In past chapters, learner directly observed the noise free data. But, that is not true for current chapter. Therefore, we distinguish ground truth from learner's observations by introducing new notations for noise-free data. Let Ξ be the space of all possible noise-free ground-truth trajectories of finite time-step length T (i.e., sequence of T $\langle s, a\rangle$ pairs) executable by the expert, and $\mathcal{X}$ be the set of (noisy) trajectories observed by the learner. Extending this observation model to trajectories, for a ground truth trajectory $\xi_g \in \Xi$, the probability of observing trajectory $\xi_o = X_o \in \mathcal{X}$ is

$$P(\xi_o|\xi_g) = \prod_{\langle s,a\rangle_o \in \xi_o, \langle s,a\rangle_g \in \xi_g} P(\langle s, a\rangle_o \,|\, \langle s, a\rangle_g). \tag{6.2}$$

The batch learning method of Robust IRL [74] uses this observation model to learn the reward function under noisy observations. In the process, it estimates the k^{th} feature expectation as

$$\hat{\phi}_{\boldsymbol{\theta},k} = \frac{1}{|\mathcal{X}|} \sum\nolimits_{\xi_o \in \mathcal{X}} \sum\nolimits_{\xi_g \in \mathbb{X}} P(\xi_g|\xi_o; \boldsymbol{\theta}) f_k(\xi_g) \tag{6.3}$$

where $P(\xi_g|\xi_o; \boldsymbol{\theta}) = \eta P(\xi_o|\xi_g) P(\xi_g; \boldsymbol{\theta})$ and $P(\xi_g; \boldsymbol{\theta})$ is the probability of the generation of a trajectory ξ_g using a policy computed with $\boldsymbol{\theta}$. Then the program of legacy MAXENTIRL (Chapter 2) becomes

$$\max_{\Delta} \left(- \sum_{\xi_o \in \mathcal{X}, \xi_g \in \mathbb{X}} P(\xi_o, \xi_g) \log P(\xi_o, \xi_g) \right) \tag{6.4}$$

subject to

$$\sum_{\xi_o \in \mathcal{X}, \xi_g \in \mathbb{X}} P(\xi_o, \xi_g) = 1$$

$$E_{\mathbb{X}}[\phi_k] = \hat{\phi}_{\boldsymbol{\theta},k} \;\; \forall k$$

where $E_{\mathbb{X}}[\phi_k] = \sum_{\xi_o \in \mathbb{X}_d} \sum_{\xi_g \in \mathbb{X}} P(\xi_o, \xi_g)\; f_k(\xi_g)$. We extend this optimization problem to the case of an unknown observation model and online learning in Sections 6.1 and 6.1.2, respectively.

[2] Although we use a state-action pair as an observation, the robust-IRL method is generalizable to any form of observations [74].

6.1 RIMEO

6.1.1 Learning distribution over observation features

Our overall approach is to approximate the unknown observation model using an underlying feature set shared among observations, and then fit it to the noisy observations via maximum entropy optimization. Let Ψ be this set of m observation features. Each feature, $\psi_j \in \Psi$, $j \in \{1, 2, \dots m\}$ is a predicate that is true if and only if a specific characteristic represented by that feature appears in the observed state-action pair under consideration. As a small example of such features, the perception of a learner observing onion sorting (task introduced in Chapter 1) can be decomposed into observation features, ψ_1: <did onion move with the hand of sorter?> and ψ_2: <is the considered onion blemished?>.

Let $\psi^{o,g}$ be the set of indicators, whose member $\psi_j^{o,g}$ assumes a value based on comparing the value of a feature on ground truth pair $\langle s, a \rangle_g$ and on observation $\langle s, a \rangle_o$: $\psi_j^{o,g} = \mathbb{1}_{\psi_j(\langle s,a \rangle_o) = \psi_j(\langle s,a \rangle_g)}$, where $\psi_j \in \Psi$ and $j \in \{1, 2, \dots, m\}$.

We assume that the learner knows the feature set and that the features are independent. For simplicity of notations, we can drop superscript o, g for features $\psi_j^{o,g}$ without sacrificing comprehension. For each $\langle s, a \rangle_o \in \mathcal{X}$, $\langle s, a \rangle_g \in \Xi$, we approximate the observation probability (Eq. 6.1) to be the product of (unknown) probabilities of the features ψ_j according to the values taken by the corresponding indicators $\psi_j^{o,g}$,

$$O_b(\langle s, a \rangle_o, \langle s, a \rangle_g) \approx \prod_{\psi_j \in \Psi, s.t. \psi_j^{o,g}=1} P(\psi_j) \cdot \prod_{\psi_j \in \Psi, s.t. \psi_j^{o,g}=0} P(\bar{\psi}_j) \tag{6.5}$$

Here $P(\psi_j)$ can be interpreted as the the probability of the jth feature being non-noisy, and $P(\bar{\psi}_j) = 1 - P(\psi_j)$. Though the learner knows the features, it cannot see the noise-free ground truth state-action pairs and those pairs not included in the demonstration. Therefore, obtaining a *complete* observation model requires inferring a distribution over observation features for all state-action pairs, regardless of whether they are observed or not.

Scores for observations Most robotics applications use some classification or regression architectures (typically a neural network) for perceiving the current configuration of the environment. In a neural network, a linear layer before the output layer predicts the activation values $\mathbf{z} = \eta^T \mathbf{h} + \mathbf{b}$, where $\mathbf{h}$ is the output from the hidden layers and $\mathbf{b}$ is the bias. As a specific example, let $\{1, 2 \dots, n\}$ be the set of indices mapping to the set of labels, where n is the total number of output nodes of the network. Then, the output of o-th node in the final layer is a **score** $c(o) = P(y_x = o|x) = \frac{\exp \mathbf{z}_o}{\sum_{j \in \{1,2\dots,n\}} \exp \mathbf{z}_j}$, $o \in \{1, 2 \dots, n\}$

We focus on the online IRL problem with the learner's perception pipeline using such architecture for identifying the state-action pairs (an example of such architecture is [76]). Using the above interpretation, the score at the oth node of the output layer, $c(\langle s, a \rangle_o)$, is the likelihood that the observation is $\langle s, a \rangle_o$ when the ground truth pair is $\langle s, a \rangle_g$. The proposed method is based on the knowledge that the scores are available from the perception neural network of the learner, and that the ground truth is unavailable.

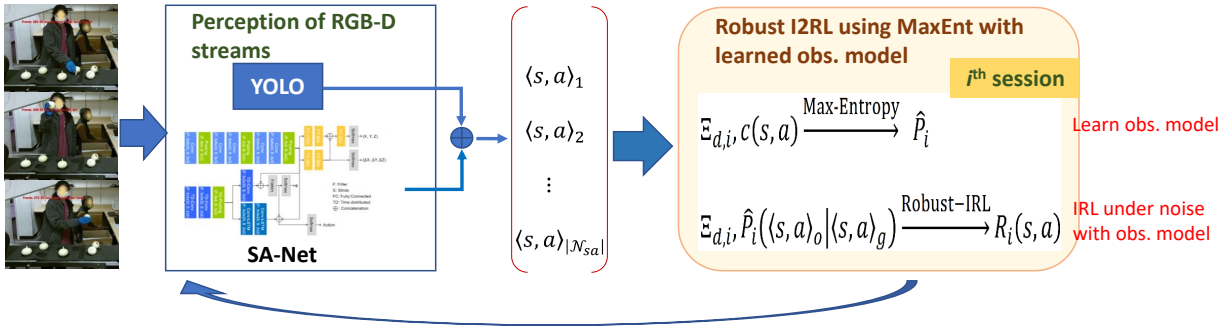


Figure 6.1: An overview of RIMEO. On the left is an image stream depicting the hidden ground truth, $\langle s,a\rangle_g$, of the expert's behavior. As the perception process is not perfect, the input to the learning algorithm is noisy. This input is an array of possibly incorrect observations and corresponding (classification) scores, $c(\langle s,a\rangle_o)$, that measure possible match with unknown $\langle s,a\rangle_g$. Not knowing the ground truth, the learner uses the scores as the likelihood of the similarity between ground truth and observation. In every session of online learning, the learner uses maximum entropy optimization with observations $\mathcal{X}$ and corresponding scores as inputs to learn conditional probabilities in the observation model, which in turn, are used to learn the reward function.

Maximum-Entropy Observation Model The likelihood of an observation $\langle s,a\rangle_o$ given any ground truth $\langle s,a\rangle_g$ can be empirically estimated as a running average of the confidence scores from observed trajectories:

$$\hat{O}(\langle s,a\rangle_o, \cdot) \triangleq \frac{\sum_{\xi_o \in \mathcal{X}} \sum_{\langle s,a\rangle \in \xi_o} c(\langle s,a\rangle) \mathbb{1}_{\langle s,a\rangle_o = \langle s,a\rangle}}{\sum_{\xi_o \in \mathcal{X}} \sum_{\langle s,a\rangle \in \xi_o} \mathbb{1}_{\langle s,a\rangle_o = \langle s,a\rangle}}. \tag{6.6}$$

Utilizing the principle of maximum entropy (a form of least commitment), the learner infers a unique distribution over the observation features Ψ which has the maximum entropy, while satisfying the constraint imposed by Eqs. 6.5 and 6.6: that the aggregated observation feature probabilities match the corresponding empirical estimates of observation likelihoods. As the ground truth is not available to the learner, it may instead consider the state-action pairs that could have been in the input, $\langle s,a\rangle_{\hat{g}}$ where $\hat{g}$ is the estimated ground truth from the expert, rather than considering all possible state-action pairs.

We simplify the notation here by using ψ_j and $\bar{\psi}_j$ for cases $\psi_j^{o,\hat{g}} = 1$ and $\psi_j^{o,\hat{g}} = 0$. The optimization problem is:

$$\max_{\Delta_\psi} -\sum\nolimits_{\psi_j \in \Psi} \left(P(\psi_j)\log P(\psi_j) + P(\bar{\psi}_j)\log P(\bar{\psi}_j)\right) \quad \textbf{subject to}$$

$$\prod_{\psi_j^{o,\hat{g}}=1} P(\psi_j) \prod_{\psi_j^{o,\hat{g}}=0} P(\bar{\psi}_j) = \hat{O}(\langle s,a\rangle_o, \cdot), \;\; \forall \langle s,a\rangle_o \in \Xi_d$$

$$P(\psi_j) + P(\bar{\psi}_j) = 1 \;\; \forall \psi_j \in \Psi.$$

The Lagrangian relaxation of the above program is:

$$\mathcal{L}(\Psi, \mathbf{v}, \boldsymbol{\lambda}) = -\left(\sum\nolimits_{\psi_j \in \Psi} P(\psi_j)\log P(\psi_j) + P(\bar{\psi}_j)\log P(\bar{\psi}_j)\right) + \sum_{\langle s,a\rangle_{\hat{g}} \in \mathbb{X}, \langle s,a\rangle_o \in \mathcal{X}} v^{o,\hat{g}} \left(\prod_{\psi_j^{o,\hat{g}}=1} P(\psi_j) \prod_{\psi_j^{o,\hat{g}}=0} P(\bar{\psi}_j) - \hat{O}\right) + \sum_{\psi_j \in \Psi} \lambda_{\psi_j}\left(\left(P(\psi_j) + P(\bar{\psi}_j)\right) - 1\right) \tag{6.7}$$

where $\mathbf{v}$, $\boldsymbol{\lambda}$ are vectors of multipliers $v^{o,\hat{g}}$ and λ_{ψ_j}.

Notice that the same feature ψ_j could be shared by multiple, distinct $\langle s,a\rangle_{\hat{g}}$ and observation $\langle s,a\rangle_o$ pairs, which then activate $\psi_j^{o,\hat{g}}$. Several of these active indicators for various j can be present in the LHS of the first constraint and these receive a probability in aggregate from $\hat{O}$. Bard [11] was the first to note that a maximum entropy approach is beneficial when just the aggregate probabilities are available. The critical points of a Lagrangian often occur at saddle points rather than at local maxima or minima [82], and numerical optimization solvers may not find the saddle points. Therefore, we modify the Lagrangian objective to make critical points occur at local optima, $\mathcal{L}' = \sqrt{\left(\frac{\partial \mathcal{L}}{\partial P(\psi_j)}\right)^2 + \left(\frac{\partial \mathcal{L}}{\partial v^{o,\hat{g}}}\right)^2 + \left(\frac{\partial \mathcal{L}}{\partial \lambda_{\psi_j}}\right)^2}$, and use limited memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) [56, 62] unconstrained optimization solver to learn the maximum-entropy distribution P^*. Equation 6.5 then obtains the observation model from this distribution, which paves the way to obtain $P(\xi_o|\xi_g)$ using Eq. 6.2 making it possible to apply the previous Robust IRL technique.

6.1.2 Session of RIMEO

In session i of online process, learner receives demonstration $\Xi_{d,i}$ and solves above optimization problem to infer P_i^*. The learner uses distribution P_i^* to estimate the feature expectations:

$$\hat{\phi}^i_{\boldsymbol{\theta}^i,k} \triangleq \frac{1}{|\Xi_{d,i}|} \sum_{\xi_o \in \Xi_{d,i}} \sum_{\xi_g \in \mathbb{X}} P_i^*(\xi_g|\xi_o;\boldsymbol{\theta}) f_k(\xi_g) = \frac{1}{|\Xi_{d,i}|} \sum_{\xi_o \in \Xi_{d,i}} \sum_{\xi_g \in \mathbb{X}} \eta P_i^*(\xi_o|\xi_g) P(\xi_g;\boldsymbol{\theta}) f_k(\xi_g). \tag{6.8}$$

Let $\Xi_{d,1:i}$ be the cumulative demonstration from ıst up to i^{th} session, and $\hat{\phi}^{1:i}_{\boldsymbol{\theta}^i,k}$ be the corresponding incremental feature expectations. The learner updates the latter from session to session as

$$\hat{\phi}^{1:i}_{\boldsymbol{\theta}^i,k} \triangleq \frac{1}{|\Xi_{d,1:i}|} \sum_{\xi_o \in \Xi_{d,1:i}} \sum_{\xi_g \in \mathbb{X}} P^*_{1:i}(\xi_g|\xi_o;\boldsymbol{\theta}) f_k(\xi_g) = \frac{1}{|\Xi_{d,1:i}|}\Bigg(\sum_{\xi_o \in \Xi_{d,1:i-1}} \sum_{\xi_g \in \mathbb{X}} P^*_{1:i-1}(\xi_g|\xi_o;\boldsymbol{\theta})\, f_k(\xi_g) + \sum_{\xi_o \in \Xi_{d,i}} \sum_{\xi_g \in \mathbb{X}} P_i^*(\xi_g|\xi_o;\boldsymbol{\theta}) f_k(\xi_g)\Bigg) = \frac{1}{|\Xi_{d,1:i-1}| + |\Xi_{d,i}|}\left(|\Xi_{d,1:i-1}|\, \hat{\phi}^{1:i-1}_{\boldsymbol{\theta}^{i-1},k} + |\Xi_{d,i}|\, \hat{\phi}^{i}_{\boldsymbol{\theta}^i,k}\right). \tag{6.9}$$

Algorithm 2 RIMEO

WS (window size) $\leftarrow$ 5; max-restarts $\leftarrow$ 5; $i \leftarrow 1$; $\Xi_{d,1:i-1} \leftarrow \emptyset$; $\hat{\phi}^{1:i-1}_{\boldsymbol{\theta}^{i-1},k} \leftarrow 0$; $[\boldsymbol{\theta}^0]_k \sim$ uniform$(0,1)$;$P^*_{1:i-1}(\psi) \sim$ uniform$(0,1)$
while $std\_dev\_z > \rho$ **do**
  $P^*_{1:i} \leftarrow P^*_{1:i-1}$
  Compute $\hat{O}_o$ using scores for $\Xi_{d,i}$ from Eq. 8.
  repeat
    Compute $\mathcal{L}', \nabla\mathcal{L}'$ using $P^*_{1:i}(\psi)$ and $\Xi_{d,i}$.
    $P^*_{1:i}(\psi) \leftarrow$ update-step-LBFGS($\mathcal{L}', \nabla\mathcal{L}'$)
  until $||\nabla\mathcal{L}'||_1 \approx 0$
  Update learned observation model using $P^*_{1:i}$ in Eq. 7.
  repeat
    compute $\hat{\phi}^i_{\boldsymbol{\theta}^i}$ and $\hat{\phi}^{1:i}_{\boldsymbol{\theta}^i,k}$ using Eqs. 10, 11.
    $|\Xi_{d,1:i}| \leftarrow |\Xi_{d,1:i-1}| + |\Xi_{d,i}|$
    $\boldsymbol{\theta}_0 \leftarrow \boldsymbol{\theta}^{i-1}, t \leftarrow 1$
    repeat
      Compute $\pi^*_{E,(t-1)}$ using $\boldsymbol{\theta}_{(t-1)}$ and $E_\Xi[\phi_k]$ using trajectories sampled from $\pi^*_{E,(t-1)}$.
      $z_{(t-1)} \leftarrow \hat{\phi}^{1:i}_{\boldsymbol{\theta}^i} - E_\Xi[\phi]$ {gradient}
      $\boldsymbol{\theta}_{t,k} \leftarrow \frac{\boldsymbol{\theta}_{(t-1),k} \exp(-\eta z_{(t-1),k})}{\sum_{k=1}^{K} \boldsymbol{\theta}_{(t-1),k} \exp(-\eta z_{(t-1),k})}$
      $t \leftarrow t+1$
    until $|z_t| \leq \varepsilon_r/(1-\gamma)$
    $j \leftarrow j+1$
  until $j >$ max-restarts
  Compute $\hat{\pi}_i$ using learned reward $\boldsymbol{\theta}^i \leftarrow \boldsymbol{\theta}_t$.
  $i \leftarrow i+1$ {next session}
  $z_i \leftarrow z_t$; mov-window-z $\leftarrow [z_{i-WS}, \ldots, z_i]$
  $std\_dev\_z \leftarrow$ std-dev(mov-window-z)

Following the concept of a session in I2RL, a session of RIMEO takes as input the expert's MDP sans the reward function, the current session's trajectories, the number of trajectories observed until the previous session, the observation feature distribution learned in the previous session, the expert's empirical feature expectations from the previous session, and the reward weights from the previous session. It is $\zeta_i(MDP_{/R}, \Xi_{d,i}, \alpha_{1:i-1}, \boldsymbol{\theta}^{i-1})$ where sufficient statistic $\alpha_{1:i-1}$ contains the summary up to the last session $(i-1)$, viz., cumulative size of the demonstration set ($|\Xi_{d,1:i-1}|$), the learned P^* values from the optimization in Eq. 6.7, and the cumulative feature expectations ($\hat{\phi}^{1:i-1}_{\boldsymbol{\theta}^{i-1}}$). Specifically, $\alpha_{1:i-1} = (|\Xi_{d,1:i-1}|, P^*_{1:i-1}, \hat{\phi}^{1:i-1}_{\boldsymbol{\theta}^{i-1}})$. This information, together with $\Xi_{d,i}$ and $\boldsymbol{\theta}^{i-1}$, is sufficient for Eq. 6.8 and thence Eq. 6.9.

For demonstration $\Xi_{d,i}$ of the current session i, the scores $c(\langle s, a\rangle)$ for input (s, a) pairs are available from the perception pipeline. The process starts by computing $\hat{O}$ using the scores and Eq. 6.6. The resulting $\hat{O}$ is plugged into Eq. 6.7 to get $\nabla\mathcal{L}'$, using which $P^*_{1:i}$ is updated by the L-BFGS gradient ascent starting with the initial value of $P^*_{1:i-1}(\psi)$. The learned $P^*_{1:i}$ are then used to compute the updated observation model probabilities $P(\langle s, a\rangle_o \mid \langle s, a\rangle_g)$ (Eqs. 6.1, 6.5). This extends to a corresponding update for full trajectories, $P_{1:i}(\xi_o|\xi_g, \boldsymbol{\theta})$ (Eq. 6.2).

After getting $P^*_{1:i}$, the learner uses MCMC to sample ground truth trajectories from $P^*_{1:i}(\xi_g|\xi_o, \boldsymbol{\theta})$ for computing the feature expectations $\hat{\phi}^i_{\boldsymbol{\theta}^i}$ (Eq. 6.8), followed by computation of $\hat{\phi}^{1:i}_{\boldsymbol{\theta}^i}$ via incremental update (Eq. 6.9). Finally, the latter is used for the main Robust IRL optimization (Eq. **??**) via exponential gradient descent to learn the reward weights ($\boldsymbol{\theta}^i$) and corresponding policy π_i. The link to our GitHub codebase implementing RIMEO is in Appendix A. [3]. One common problem faced in incremental learning is abnormally high jumps in the metric used for stopping, often leading to *an early termination or p-hacking bias*. To avoid that, we use a threshold over the standard deviation of a moving window over the gradients for IRL. When the deviation in gradient is low, then there are no frequent spikes in the learned observation model and the learned weights.

6.1.3 Convergence Properties of RIMEO

While some of the results in this section are analogous to those for LME - I2RL [5] (Chapter 4), Lemma 2 is the main original result of this paper since it incorporates the additional uncertainty due to noisy observations and an unknown observation model. We present proofs for all convergence properties in Appendix E. The demonstration log-likelihood for session i is given by $LL(\boldsymbol{\theta}^i|\Xi_{d,i}, \alpha_{1:i-1}, \boldsymbol{\theta}^{i-1})$.

Lemma 3 (Monotonicity)**.** The demonstration likelihood increases monotonically with each new session, $LL(\boldsymbol{\theta}^i|\Xi_{d,i}, \alpha_{1:i-1}, \boldsymbol{\theta}^{i-1}) - LL(\boldsymbol{\theta}^{i-1}|\Xi_{d,i-1}, \alpha_{1:i-2}, \boldsymbol{\theta}^{i-2}) \geqslant 0$, when $|\Xi_{d,1:i-1}| \gg |\Xi_{d,i}|$.

Note that feature expectations estimated under noisy perception with an estimated observation model, $\hat{\phi}^{1:i}_{\boldsymbol{\theta}^i,k}$ from Eq. 6.9, is an approximation of the expectations when the observation model is known accurately. $\hat{\phi}^{1:i}_{\boldsymbol{\theta}^i,k}$ is computed by sampling the hidden ground truth ξ_g from $P(\xi_g|\xi_o, \boldsymbol{\theta}^{i-1})$. This, in turn, is an approximation of the expectations estimated under noise-free perception, $\hat{\phi}^{1:i}_k$. As in I2RL [5], we assume that the latter estimation using n_s samples obeys Hoeffding bounds, i.e., $P(|\hat{\phi}^{1:i}_k - \hat{\phi}^{1:i}_{\boldsymbol{\theta}^i,k}| \leq \varepsilon_s) \geq 1 - \delta_s$, where $\delta_s = 2K\exp(-2n_s\varepsilon_s^2)$. Additionally, we assume that our observation model estimation with n_o samples also obeys Hoeffding bounds as $P(\max_{\langle s,a\rangle_o}|O - \hat{O}| \leq \varepsilon_o) \geq 1 - \delta_o/K$, where O is the true observation model (of which $\hat{O}$ is an estimate), and $\delta_o = 2K|\Psi|\exp(-2n_o\varepsilon_o^2)$. Note that the max operator in this assumption is only for *observed* $\langle s, a\rangle_o$, not all possible $\langle s, a\rangle_o$.

We also assume that all features in Ψ are observed in the very first session. Then the following lemma relates the error due to the sampling-based approximation, i.e. $\left|\hat{\phi}^{1:i}_k - \hat{\phi}^{1:i}_{\boldsymbol{\theta}^i,k}\right|$, to the cumulative difference w.r.t. feature expectations for the expert's true policy, $E_{\mathbb{X}}[\phi_k]$.

[3] For the gradient descent part, we use exponentiated gradient descent, but any solver should work.

Lemma 4 (Constraint Bound)**.** Under the assumptions stated above, with probability at least $\max(0, 1-\delta_r)$: $\left|(1-\gamma)\left(E_{\mathbb{X}}[\phi_k] - \hat{\phi}^{1:i}_{\boldsymbol{\theta}^i,k}\right)\right|_1 \leqslant \varepsilon_r, \forall k \in \{1, 2 \ldots K\}$, where $\delta_r = \delta+\delta_s+\delta_o, \varepsilon_r = \varepsilon+\varepsilon_s+L|\Psi|\varepsilon_o$, L is the longest trajectory length, and ε, δ are as defined in Theorem 1 in [5].

Lemma 8 allows us to probabilistically bound the error in the log likelihood for RIMEO, leading to the main result below specifying the confidence in its convergence.

THEOREM 1 (CONFIDENCE). *Let ε_r, δ_r be as defined in Lemma 8, and $\boldsymbol{\theta}^i$ be the solution of session i for RIMEO. Then $LL(\boldsymbol{\theta}_E|\Xi_{d,1:i}) - LL(\boldsymbol{\theta}^i|\Xi_{d,i}, \alpha_{1:i-1}, \boldsymbol{\theta}^{i-1}) \leq \frac{2K\varepsilon_r}{(1-\gamma)}$, with confidence at least $\max(0, 1-\delta_r)$, where $\boldsymbol{\theta}_E$ are the true weights of the expert.*

Given $\varepsilon, \varepsilon_s, L, |\Psi|, \varepsilon_o$ and the total number of noisy observed trajectories ($|\xi_{1:i}|$), Theorem 6 gives the confidence $1-\delta_r$ for convergence of RIMEO. Equivalently, given desired error bounds and confidence, the size of data required $|\xi_{1:i}|$ can be derived.

Extending the definition of average regret (Chapter 3) up to session $i = T$ to RI2RL-MEOM, we get

$$Reg_T(A_{RI2RL-MEOM}) = \frac{1}{T}\sum_{i=1}^{T} LL(\boldsymbol{\theta}_E|\Xi_{d,1:i}) - \frac{1}{T}\sum_{i=1}^{T} LL(\boldsymbol{\theta}^i|\Xi_{d,i}, \hat{\xi}_{i-1}, \boldsymbol{\theta}^{i-1}).$$

Then Theorem 6 implies no-regret learning, following the same arguments presented in [5]. The formal result is:

THEOREM 2 (NO REGRET LEARNING). *There exist choices for a variable threshold bound on the feature matching constraint, e.g., $\varepsilon_{r,i} = const/i$ as a function of session i, such that with probability at least $\max(0, \prod_{i=1}^{i=T}(1-\delta_r))$, where δ_r is as defined in Lemma 8 ,*

$$\text{Reg}_T(A_{RI2RL-MEOM}) = o(T)/T,$$

thus approaching 0 as $T \to \infty$.

Please note that these convergence properties have similarities with those for LME-I2RL in Chapter 4. That happens because both the methods follow same I2RL framework for online learning of reward function.Theorem 7 takes a long term view of the learner's performance where convergence demands a gradually improving performance through a tighter threshold, whereas the Theorem 6 merely improves the confidence (δ) on the learner's performance with accumulating sessions without similarly demanding an improved performance (e.g., by reducing the log likelihood loss). The price for demanding greater long term accuracy is the diminishing ϵ, which may become increasingly harder to meet.

Note that the above theorem assumes that descent of Fig. 2 exits in *every* session. If this does not occur in some session, for instance if the algorithm gets stuck in a local optima which becomes increasingly likely as the threshold in equation 4.3 tightens, then the theorem does not apply.

6.1.4 Computational complexity

We enunciate the complexity of the major steps of Algorithm 2 below, and then combine them into the overall complexity.

Computation of $\hat{O}$: O(total number of observed state action pairs) + constant = $O(|\Xi_d|L)$.

Complexity of BFGS update: $2 \times O(m^2)$ where m is the number of observation features.

#Updates for convergence in gradient of Lagrangian: n_{bfgs}.

Complexity of updating observation model O_b using Eq 6.5: $O(m|\Xi_d|L\mathcal{N}_{sa})$.

#Iterations for forward RL: n_{RL}.

Feature expectations E: (number of trajectories simulated to approx trajectory space)$\times$ complexity for one trajectory = $O(LK)$, when the number of simulated trajectories is a constant.

Complexity of step updating θ: $O(K)$.

#Descent iterations for reaching $z < \varepsilon$: n_{desc}.

Therefore, the complexity of each iteration of each session is $O(m|\Xi_d|L\mathcal{N}_{sa} + LK + m^2 n_{bfgs} + \eta)$, where $\eta = n_{RL} + n_{desc}$. The quantities of η as well as n_{bfgs} are difficult to bound and in practice are often set to some predetermined constants, making them independent of m or K. As a result, the method scales gracefully w.r.t the number of observation features m and reward function features K, because the complexity is polynomial w.r.t both.

While conducting research on RIMEO, we identified similar gaps that spurred further investigation. Firstly, noise-robust methods such as RIMEO necessitate manual engineering of reward features, whereas AIRL eliminates the need for such prior engineering. Secondly, when confronted with noisy input, AIRL (employed as a baseline for RIMEO) exhibits inferior performance compared to alternative methods [7], potentially because the existing structure of AIRL lacks the explicit capacity to account for input data noise.

To address both of these issues, we devised a method to enhance the performance of AIRL under perception noise. This improvement, in turn, eliminates the requirement for manual feature engineering in online learning scenarios affected by perception noise.

6.2 robust-AIRL

This section begins by introducing the concepts behind the robust-AIRL method, which involves modifying the GAN-training objective for AIRL. Subsequently, it delves into the implementation details of the algorithm for proposed method.

6.2.1 GAN-training objective

We can logically write the GAN training objective given in the first part in Algorithm 1 in paper [39] as following

$$\max_{\theta} \left(E_{(x^i)\sim p_x} \left[\log D(x^i) \right] + E_{(z^i)\sim p_g} \left[\log[1 - D(G(z^i))] \right] \right)$$

or equivalently

$$\max_{\theta} \sum_{i=1}^{m} \left(p_x(x^i) \log D(x^i) + p_g(z^i) \log[1 - D(G(z^i))] \right)$$

where θ are weights of the discriminator network, x is input data, m are number of input data points, p_x is distribution used for sampling input data, D is output of discriminator network, z is input noise for generator, p_g is noise prior for generator input, G is generator network.

If we can use above representation for GAN training objective, then the objective of discriminator training in AIRL [35] is same as

$$\max_{\theta} \left(\sum_{\xi \in \Xi} P_{exp}(\xi) \sum_{(s,a,s') \in \xi} \log D(s,a,s') + \sum_{\xi \in X_{gen}} P(\xi|\theta_{gen}) \sum_{(s,a,s') \in \xi} \log[1 - D(s,a,s')] \right)$$

where Ξ is set of noise free input trajectories, P_{exp} is true trajectory distribution of expert (can be derived from expert's policy), X_{gen} is set of trajectories generated by generator, θ_{gen} are weights of generator network.

A discriminator in AIRL computes a divergence between the input demonstration and the generator's output. If input itself is noisy, is AIRL by design capable of learning efficiently? Instead of using noisy input directly in computations, the discriminator can sample hidden noise-free ground truth and average the divergence over multiple samples. That way, instead of using noisy input, the divergence is computed by using an approximation of ground truth. If input trajectories ξ_o in set Ξ_d are noisy, then we can use Gibbs sampling with transition dynamics and observation model to sample ground truth trajectory $\xi_g \in \mathbb{X}$. Then the objective of discriminator becomes:

$$\max_{\theta} \left(\sum_{\xi_o \in \Xi_d} P(\xi_o) \sum_{\xi_g \in \mathbb{X}} P(\xi_g|\xi_o) \sum_{(s,a,s') \in \xi_g} \log D(s,a,s') + \sum_{\xi \in X_{gen}} P(\xi|\theta_{gen}) \sum_{(s,a,s') \in \xi} \log[1 - D(s,a,s')] \right)$$

Here $P(\xi_o)$ is the empirical distribution governing noisy input data and $P(\xi_g|\xi_o)$ is the distribution over noise-free ground truth conditioned on noisy input. $P(\xi_g|\xi_o)$ can be derived from an observation model for the noisy perception mechanism of the learner. To address the problem of avoiding manual engineering of the reward function, we assume that the observation model is available.

6.2.2 Session of robust-AIRL

Most of the current online methods need manual engineering of rewards ([73] and [7]). Using AIRL as an online method will help avoid that limitation. Therefore, we use 'AIRL as is' in an online fashion by processing each batch of training samples as 'an individual demonstration for a session' of I2RL. With this motivation in mind, we will analyze robust-AIRL as an online method that admits the incremental learning framework introduced in the third chapter. That means, here onwards, robust-AIRL is a sequence of sessions with $\boldsymbol{\theta}^i$ as the i^{th} session's estimate of the expert's reward function.

Algorithm

Each session ζ_i uses the weights $\boldsymbol{\theta}^{i-1}$ of reward network f_θ^{i-1} and the weights $\boldsymbol{\theta}_{gen}^{i-1}$ of generator-policy network π^{i-1} from the previous session $i-1$ as the sufficient statistic. This sufficient statistic contains summarized information learned from all previous sessions. Along with the weights and sufficient statistics, each session needs the following as inputs: observation model $P(\langle s,a\rangle_o \mid \langle s,a\rangle_g)$, state transition model $P(S_{g,t}|s_{g,t-1},a_{g,t-1})$, currently observed noisy demonstration $\Xi_{d,i}$, Gibbs sampling convergence threshold δ_G. Learner simulates an assumed ground truth trajectory from policy π^{i-1} learned in the previous session. In every iteration used for computing discriminator logits, the process computes Gibbs sampling probability distributions ($P(s_{g,t}|\text{MB}(s_{g,t}))$ and $P(a_{g,t}|\text{MB}(a_{g,t}))$) for each time step t in input batch of state-action pairs. Then learner samples ground truth from Gibbs sampler and uses that to compute a running average of logits. The cycle keeps going until logits converge to stable values, at which point the regular steps of AIRL (train discriminator D, reward net f_θ, and generator π) are followed.

6.2.3 Gibbs samplers

This section explains the computation of Gibbs sampling distributions used in the algorithm. We analyze the perception and decision-making process of the learner as a dynamic Bayesian network and found the Markov blankets for noise free ground truth state and action at time step t (Figure 6.2). With observation model $P(s_{o,t},a_{o,t}|s_{g,t},a_{g,t})$ and transition dynamics (either approximated or exact) available, one can derive a Gibbs sampling distribution (referred as Gibbs sampler here on-wards) [41] for ground truth as following:

$$\begin{aligned} P(s_{g,t}|\text{MB}(s_{g,t})) &= \alpha \times T_1 \times T_2 \times T_3 \times T_4 \times T_5 \\ P(a_{g,t}|\text{MB}(a_{g,t})) &= \alpha \times T_2 \times T_3 \times T_5 \end{aligned} \tag{6.10}$$

where α is a normalization constant and

$T_1 : P(s_{g,t}|s_{g,t-1},a_{g,t-1})$ is the probability of ground truth current timestep state given previous timestep state and action.

$T_2 : \pi(a_{g,t}|s_{g,t})$ is the probability of ground truth action being a_g given ground truth state.

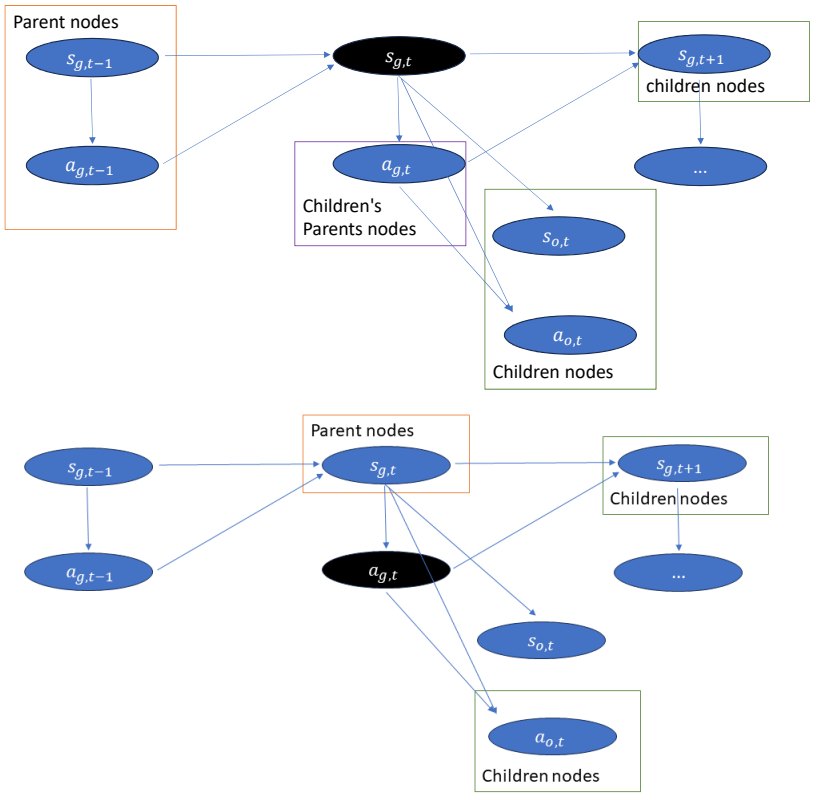


Figure 6.2: **top** DBN shows the markov blanket for ground truth noise free state at time step t and **bottom** DBN show Markov blanket for ground truth action at time step t.

Algorithm 3 Session $\zeta_i(MDP_{/R}, \Xi_{d,i}, \boldsymbol{\theta}^{i-1}, \boldsymbol{\theta}^{i-1}_{gen})$ of robust-AIRL

concatenate $\xi_o \in \Xi_{d,i}$ to create batches of s_o, a_o
for each batch of s_o, a_o pairs **do**
  simulate a same sized batch of s_g, a_g using π^{i-1}
  `/* "potential" ground truth needed to create Gibbs sampler */`
  $i \leftarrow 0, \boldsymbol{l}_{D,avg} \leftarrow 0$
  repeat
    Compute distribution $P(s_{g,t}|\text{MB}(s_{g,t}))$ and $P(a_{g,t}|\text{MB}(a_{g,t}))$ for each timestep s_o, a_o
    Use them to sample a new batch of $\hat{s}_g, \hat{a}_g$
    $(s_g, a_g) \leftarrow \hat{s}_g, \hat{a}_g$
    `/* update potential ground truth with sampled data */`
    compute discriminator logits $\boldsymbol{l}_D$ using s_g, a_g
    $i \leftarrow i + 1$
    $\boldsymbol{l}_{D,avg} \leftarrow \frac{\boldsymbol{l}_{D,avg}(i-1)+\boldsymbol{l}_D}{i}$
    last_avg $\leftarrow \boldsymbol{l}_{D,avg}$
  until $|\boldsymbol{l}_{D,avg} - last\_avg|_1 \leq \delta_G$
  use $\boldsymbol{l}_{D,avg}$ to train discriminator D
  use trained D to update reward network weights
  train generator-policy network π^{i-1} updating its weights

T_3 : $P(s_{g,t+1}|s_{g,t}, a_{g,t})$ is the probability of $s_{g,t+1}$ as the next timestep state given ground truth current timestep state and action

T_4 : $P(s_{o,t}|s_{g,t})$ is the probability of observing state s_o given ground truth state

T_5 : $P(a_{o,t}|s_{g,t}, a_{g,t})$ is the probability of observing a_0 given ground truth state and action

To derive the product distributions in Eq 6.10, **for every time step t of observed trajectory**, one will need following inputs:

- prev step $s_{g,t-1}, a_{g,t-1}$ from ground truth trajectory
- next step $s_{g,t+1}$ from ground truth trajectory
- observed state s_o
- observed action a_o

6.2.4 Continuous state domains

For continuous state domains, we have tried two techniques to compute Gibbs samplers. First one discretizes the state space but doesn't scale well with the number of dimensions of state space. To address that issue, the second approach uses a product of Gaussians.

Most continuous state gym environments have deterministic transition dynamics,. Let $s_{in,s_{t-1},a_{t-1}} = f(s_{g,t-1}, a_{g,t-1})$ be deterministic next state we can derive from transition dynamics. This knowledge is useful in both techniques.

Technique 1: Discretization of state space

For simplicity of analysis, let's consider the case of continuous states and discrete actions. Let's say we have divided each dimension of continuous state into Q parts. For n_s dimensional state space, there are total Q^{n_s} partitions (multi-dimensional intervals) of state space S. For sampling a continuous state $s_{g,t}$, the learner will have to compute the Gibbs sampling distribution for ground truth state-space partition at timestep t. The learner will first draw a sample of state-space partition S_g and then she will sample randomly from S_g partition to get $s_{g,t}$. In DBNs (Fig 6.2), The state-space partition nodes replace the state nodes, and that leads to following Gibbs distribution expressions:

$$\begin{aligned} P(S_{g,t}|\text{MB}(S_{g,t})) &= \alpha \times T_6 \times T_7 \times T_8 \times T_9 \times T_{10} \\ P(a_{g,t}|\text{MB}(a_{g,t})) &= \alpha \times T_7 \times T_8 \times T_{10} \end{aligned} \tag{6.11}$$

As S_g is a continuum, we need integrals to compute terms used in above equations.

$T_6 : P(S_g|s_{g,t-1}, a_{g,t-1}) = \left(\int_{s_{g,t} \in S_g} P(s_{g,t}|s_{g,t-1}, a_{g,t-1}) d(s_{g,t})\right)$ is the probability of ground truth current timestep state being in partition S_g given previous timestep state and action.

$T_7 : \pi(a_{g,t} = a_g|S_g) = \left(\int_{s_{g,t} \in S_g} \pi(a_{g,t} = a_g|s_{g,t})\right)$ is the probability of ground truth (discrete) action being a_g given ground truth state being in partition S_g.

$T_8 : P(s_{g,t+1}|S_g, a_{g,t} = a_g) = \left(\int_{s_{g,t} \in S_g} P(s_{g,t+1}|s_{g,t}, a_{g,t} = a_g) d(s_{g,t})\right)$ is the probability of $s_{g,t+1}$ as next timestep state given ground truth previous timestep state in partition S_g and current timestep discrete action is a_g.

T_9 and T_{10}: $P(s_o|S_g)\ P(a_{o,t} = a_o|s_{g,t} \in S_g, a_{g,t} = a_g) = \left(\int_{s_{g,t} \in S_g} P(s_o|s_{g,t}) d(s_{g,t})\right) P(a_{o,t} = a_o|s_{g,t} \in S_g, a_{g,t} = a_g)$ is the probability of observing state s_o and discrete a_0 given ground truth state is in partition S_g and ground truth action is a_g.

One can use the knowledge of deterministic next state $s_{in,s_{t-1},a_{t-1}}$ to make the computation for terms T_6 and T_8 simpler

$$P(s_{g,t}|s_{g,t-1}, a_{g,t-1}) = \begin{cases} 1 & \text{if } s_{g,t} = s_{in,s_{t-1},a_{t-1}} \\ 0 & \text{otherwise} \end{cases}$$

Scalability issue For every time step t of an observed trajectory, the algorithm will need a Gibbs sampling probability measure over sample space comprising all values of S_g and action a_g. The computation of gibbs sampling probability has to be done for every combination of a 'state space partition' and an action. For n_s number of dimensions, the total number of partitions are Q^{n_s}. Since the latter increases exponentially with the number of dimensions, the complexity for computing gibbs samplers does not scale gracefully [4]. The same problem extends to environments with continuous states and continuous actions.

Technique 2: Product of multi-variate Gaussians

Instead of discretizing the state space, one can approximate the terms T_{1-5} in Eq 6.10 as multi-variate Gaussian distributions with the following mean and co-variances:

T_1 and T_3: As dynamics are deterministic for many continuous state continuous action Gym environments, the mean tensor is the deterministic next state $s_{in,s_{t-1},a_{t-1}}$, and covariance tensor can be a matrix with assumed small values on its diagonal. This way we model transition dynamics such that a given action in a given state leads to the deterministic next state with close to highest probability.

T_2: Learner samples a large set of actions predicted by policy and then uses the mean and the covariance of that sample set. The result is an approximation of generator's policy as a Gaussian distribution.

T_4 and T_5: The mean is noise-free ground truth and covariance can be a diagonal matrix with its values depending on the level of noise wanted in the perception process of observing the expert's behavior. Higher the covariance, more are the chances of having observation further away from the ground truth.

Please note that here we approximate learned policy (term $T_2 = P(a|s)$) to a be a multi variate Gaussian. The level of accuracy of this approximation is dependent on upon how to close to a uni-modal $\{P(a|s), \forall s \in S\}$ distributions actually are. The approximation accuracy will impact the relevance of MCMC samples, and thereby the accuracy of robust-AIRL process.

The mean and covariance for two product distributions in Eq 6.10 can be computed using standard formulae for the product of multi-variate Gaussians (Section 8.1.8 in [63]). The rest of the steps of robust-AIRL are same as the algorithm shown before.

Chapters 4 to 6 introduced the methods to address different kinds of imperfection (missing data, mixing up of data, noisy perception, etc.) in training data for online learning. The next chapter has the results of an evaluation of proposed four methods using simulated and real-life challenging applications.

[4] Please note that this limitation does not hold in discrete state discrete action domains.

Chapter 7

Experimental Evaluation

To comprehensively evaluate the performance of our research contributions, we used a diversity of challenging application domains and IRL metrics. We have organized this chapter in the following manner so that the readers can easily understand the process of evaluation and can reproduce it if needed. It starts with an explanation of application domains used for experimentation, followed by details about the metrics used for measuring performance. After that, we talk a little bit about the baselines used for a comparison against the performances of the proposed methods. The discussion leads to an explanation about how each experiment was setup and what results we got. We explain the results in the same order as we introduced the research contributions in previous chapters: LME-I2RL, online MME-MTIRL, RIMEO, robust-AIRL.

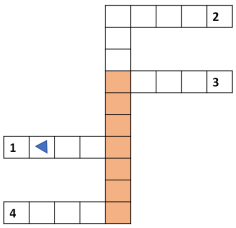
7.1 Evaluation domains

We used following application domains for the evaluation:

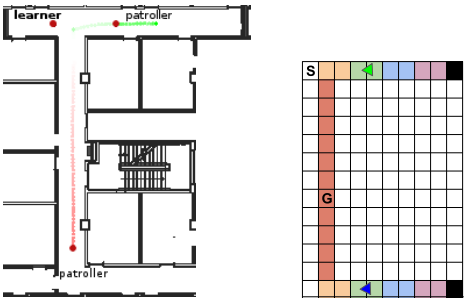
7.1.1 Observing and Penetrating Cyclic Patrols

Bogert and Doshi [13] introduced the domain of robot patrolling for evaluating IRL under occlusion and simulated it in ROS-based Player Stage [37]. It involves a robotic learner observing patrollers from a vantage point with a limited view continuously navigate some hallway in cyclic trajectories. The learner is tasked with reaching a goal location without being spotted by any of the patrollers. Each patroller can see up to three grid cells in front.

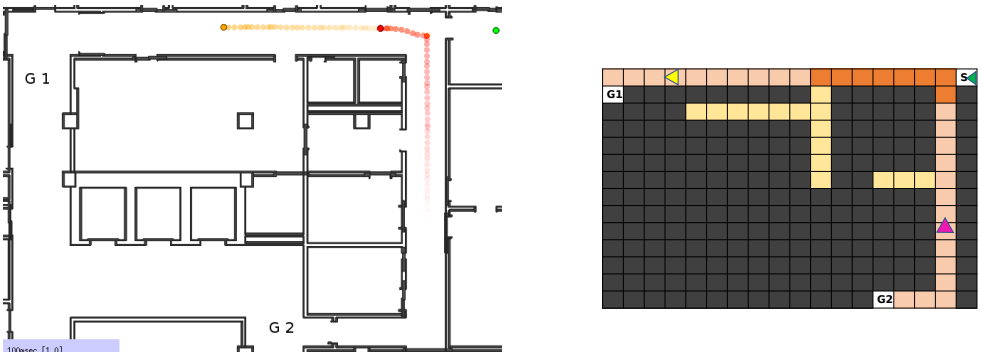
We evaluate the performance of LME I2RL on three instances of the patrolling domain. The first instance, shown in Fig. 7.1(a), involves a single patroller covering fully four hallways protruding from a common corridor in a loop. The learner is able to see a portion of just the corridor (shown shaded) from its vantage point leading to about 70% of the patroller's trajectory being occluded from its view. We utilize this first instance to evaluate the accuracy of learning only. Figure 7.1(b) shows a second instance of the patrolling domain, also utilized by Bogert and Doshi previously. The map for this instance pertains to a



(a) A single patroller denoted by the triangle moves clockwise to the ends of four hallways in the numbered order, and only the shaded area is visible to the learner.



(b) Two patrollers on the right navigate the hallways in an environment, which simulates that of a real building [13]. The color shaded regions (long hallway, turning points and three small divisions in small hallways on both sides) are the five regions defining movement and turn-around features. S and G are start and goal locations for the learner.



(c) A larger instance of the multi-robot patrolling domain. The two patrollers are navigating the hallway while the learner penetrates the patrol after learning has finished. The patrollers and their corresponding trails are shown in yellow and light pink, the learner is shown in green. The adjacent grid indicates navigable cells of hallways in orange and those of rooms in yellow. We show the two goal locations in the grid. S is the start location, G1 and G2 are the two goal locations. The area visible to the learner is shown in dark orange at the top right region of the grid.

Figure 7.1: We use three differently-sized instances of the patrolling domain [13] for evaluating the performance by LME I2RL. The learner is unaware of where each patroller turns around, their speed or navigation capabilities.

portion of the fifth floor of the Boyd building on the University of Georgia campus. Two patrollers execute, independently for the most part, a cyclic trajectory with the learner able to view just 32% of the trajectory from its vantage point in the physical instance. The patrollers engage in coordinated motion when they pass by each other to avoid collisions. The learner is tasked with reaching the cell location marked 'G' without being spotted by any of the patrollers. Consequently, this instance requires the learner to utilize the learned behaviors of the patrollers in its own planning problem and execute its plan. The final instance of the domain, shown in Fig. 7.1(c), involves two patrollers executing cyclic trajectories in a significantly

larger space (also on Boyd's fifth floor) with the learner located in a room and viewing outside. Both patrollers and the learner can also enter two rooms whose entrances lie in the two hallways. The learner is able to view just 18% of the patrolling trajectory from its vantage point, and is tasked with reaching one of two possible goal locations without being detected by any patroller. Consequently, this instance uses a map that is significantly larger than the previous ones, potentially more complex trajectories and planning, and significantly greater occlusion. Second and third instances are more complex than first instance, because the learner needs to solve its own distinct decision-making problem (modeled as another MDP) to reach the goal location(s). The solution to learner's problem depends on correctly predicting the patrols. The latter can be estimated from inferring each patroller's preferences given that the learner knows their dynamics.

MDP Setup

The learner ascribes an MDP sans the reward function to model the expert's task behavior. In all instances of the domain, each patroller's behavior is modeled using an MDP. The state of each patroller in the MDP has three dimensions $\langle x, y, \theta \rangle$, which gives the x and y coordinates of the cell decomposition of the corridors and hallways, and θ is the orientation of the patroller. Each patroller executes one of four possible actions: move-forward, turn left or turn right 90 degrees, and stop. The motion model of the patroller, modeled as the transition function in the MDP, is stochastic. The MDP for instance 1 of the domain has 192 states, instance 2 has 124 states while the MDP for the third instance has 184 states. Please refer Appendix C for the reward features and expert's weights used in three patrolling domains.

For the second and third instances of domains, after learning the reward function, the learner computes the policies for both patrollers. It uses the policies and recently observed locations to predict the future locations of the two patrollers. The learner's own MDP has the same state space as the patroller's and additionally includes a discrete timestep as a fourth dimension of its state. As the patrollers are constantly moving and the learner incurs a penalty for being spotted, the timestep allows the learner to represent the map with the patrollers' current location included. The learner solves its MDP over a finite horizon to obtain a policy that guides its actions to reach the goal location. On seeing both patrollers, it waits until a state with a positive value occurs before moving.

7.1.2 Robotic Sorting of Onions

This proof of concept domain was inspired from the long term vision to deploy robotic arms on complex processing lines involving manipulation tasks, using IRL. The setup involves a learner robot observing an expert sort onions in a post-harvest processing facility. The expert aims to identify and remove onions with blemishes from the collection of onions present on a static conveyor belt. Blemished onions are dropped in a bin while others are left on the table.

MDP Setup

The state of a sorter is perfectly observed and composed of four factors: *onion* and *gripper location*, *quality prediction*, and *multiple predictions*. Here, an onion's location can be on the sorting table, picked up, under inspection (involves taking it closer to the head), inside the blemished-onion bin, or the onion has been returned to the table post inspection. *Gripper location* is similar but does not include the return back to the table. *Quality prediction* of the onion can be blemished, unblemished, or unknown. The simultaneous predictions for multiple onions is either available or not.

The expert's actions involve focusing attention on a new onion on the table at random, picking it up, bringing the grasped onion closer and inspecting it, placing it in the bin, placing it back on the table, roll its gripper over the onions, and attend to the next onion among those whose quality has been predicted. Please refer Appendix F for reward features used for this MDP.

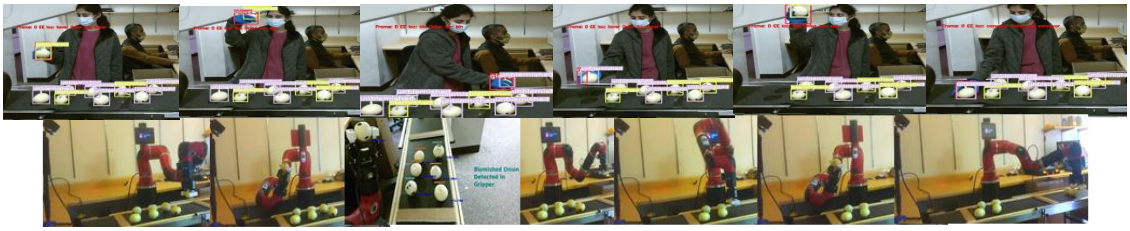
In a visit to a real-world onion processing line attached to a farm, we observed that two distinct sorting techniques were in common use and the human sorters frequently switched from one technique to another. For evaluating MME-MTIRL (multi task learning), we model the expert as acting according to the output of two MDPs both of which share the state and action sets, the transition function and the reward features. They differ in the weights assigned to the features, which yields different behaviors (ways of sorting). When used two distinct vectors of real-valued weights on these feature functions yield two distinct reward functions for the two behaviors:

- Pick-Inspect-Place sorting: The MDP with one of these solves to obtain a policy that makes the expert randomly pick an onion from the table, inspect it closely, and place it in the bin if it appears blemished, otherwise place it back on the table.

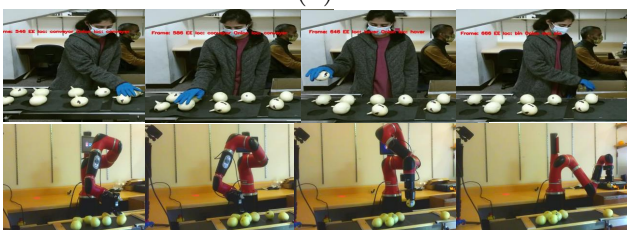
- Roll-Pick-Place sorting: The second reward function yields a policy that has the expert robot roll its gripper over the onions, quickly identify blemished onions, pick only those and place them in the bin.

In the MDP, we model the classification of blemished and unblemished onions by using a distribution over prediction values. For careful inspection, our distribution assigns higher mass to the correct prediction. For rolling, this mass is lower than that of inspection. This makes the accuracy of careful inspection higher. Both these sorting techniques are illustrated in Fig. 7.2.

For evaluating RIMEO, we made some changes to this domain because the method needs only one reward function for onion-sorting. We used only pick-inspect-place sorting trajectories in training data. We also reduced the state space by removing *multiple predictions* part of state. A state of this task is composed of *onion location* and *end-effector location* {on table, picked up in hover region, at eye level (under inspection), or near the bin for blemished onion}, and *quality prediction* {blemished, unblemished, or unknown}, resulting in 22 states. The actions involve attending to a new onion on the table, picking it up, bringing the grasped onion closer to eye and inspecting it, placing it in the bin, and placing it back on the table. We utilize $K = 11$ predicates as reward features, $\phi_k(s, a)$, listed in Appendix F. To learn the observation model, we use 8 binary predicates ψ_j as observation features, also listed in Appendix F.



(a)



(b)

Figure 7.2: Human demonstration and learned behaviors executed by Sawyer of the two sorting techniques: (a) pick-inspect-place (picks each onion, inspect it closely), (b) roll-pick-place (roll them, expose hidden surfaces, to classify many onions simultaneously). The process of identification of states in demonstration involves tracking the locations of claimed onion and hand (referred as EE), prediction of claimed onion, and list of blemished onions. First two can be at four locations: conveyor, hover, front of eyes, and bin; and last two are derived directly from YOLO's output.

We used Pearson product-moment correlation (0.03 with p-value 0.04, averaged over all pairs of features from demonstration) to confirm that these observation features are sufficiently independent.

7.1.3 OpenAI Gym Domains

For evaluating robust-AIRL, we used standard benchmark RL domains from OpenAI (Refer `https://gymnasium.farama.org/environments` for OpenAI domains).

Mountain car

Mountain car (Figure 7.11) is a continuous state reinforcement learning domain with deterministic transition dynamics [57]. The car starts from a location stochastically chosen in the valley. The goal of agent is to derive best strategy to accelerate the car making it reach the goal state located at the top of right hill. A state of Mountain Car is a combination of two elements: Position along x-axus and Velocity. The valid ranges for position and velocity are [-1.2, 0.6] and [-0.07, 0.07]. An action decides the direction of application of force: value 0 accelerates car to left, 1 means do not accelerate, and value 2 accelerates car to right. As the goal is to reach right side top as soon as possible, reward function penalizes agent for every time step. Please refer `https://gymnasium.farama.org/environments/classic_control/mountain_car/` if interested in further details on the environment.

Some of the standard high-dimensional RL environments often used in literature are Half Cheetah, Hopper, and Walker2D [22].

Hopper

The hopper is a figure with one leg, existing in two dimensions, comprised of an upper torso, a middle thigh, a lower leg, and a solitary foot supporting the entire body. The objective is to generate forward (rightward) movement by exerting torques on the three hinges that connect the four body parts, enabling hopping motions. The domain has 3 dimensional action space $[-1, 1]^3$ and 11 dimensional state space comprising positions and velocities of different links. We used the default reward function of this environment to generate expert's behavior, which incentivizes forward movement and penalizes large torque values. An episode ends when an upper bound of 1000 steps is reached or state elements go outside a predefined healthy range of values.

Walker 2D

The walker is a figure with two legs, composed of a singular upper torso, with the legs branching out from it. The objective is to synchronize the movements of both legs to achieve forward (rightward) motion. This is accomplished by exerting torques on the six hinges that connect the six body parts. An action is a 6-dimensional vector of torque values applied to the joints, with each torque value limited to the interval $[-1, 1]$. A state is a 17-dimensional vector comprising a) positional values of links, b) velocity of those parts. The reward function and the episode ending conditions are same as Hopper.

Half Cheetah

Half Cheetah is a cat-like robot consisting of 9 links connected by 8 joints. The goal of environment is to find torque values that will make the cheetah run forward as fast as possible, toward the right direction. The domain has 6 dimensional actions and 17 dimensional states.

For inserting noise in the behavior of expert, we use high covariance value of 0.05 in Gaussian distributions representing the observation model terms used in Gibbs samplers (Chapter 6, second part). Readers interested in more details about any of three environments can refer `https://gymnasium.farama.org/environments/mujoco/`.

7.2 Metrics

We measured the efficacy of the methods in learning a reward function by using the following *metrics*:

- *Learned behavior accuracy (LBA)* is the proportion of all states at which the actions prescribed by the inversely learned policy is same as the action prescribed by demonstrator's policy;
- *Inverse learning error (ILE)*, as previously defined in Chapter 3, loss in value incurred of one uses the learned policy in the demonstrator's MDP; and

- *Expected Value Difference (EVD)* (common in multi-taskIRL [27]) is the loss of value, averaged over the trajectories, if the learner uses the policy obtained by solving the expert's MDP with the learned reward function (parameterized by $\boldsymbol{\theta}^L$) instead of the expert's true policy obtained by solving its MDP with its actual reward function.

- *Correlation w.r.t. ground truth* is the correlation between ground truth reward values and learned reward values for same state-actions pairs, thereby, measuring the degree of closeness between two.

Note that when the learned behavior accuracy is high, the ILE is expected to be low. However, as MDPs admit multiple optimal policies, a low ILE need not translate into a high behavior accuracy. As such, these two metrics are not strictly correlated.

As explained in Chapter 3, ILE is calculated as $ILE(\pi^*_E, \pi_E) = ||V^{\pi^*_E} - V^{\pi_E}||_1$. Here, $V^{\pi^*_E}$ is the optimal value function of E's (expert's) MDP and V^{π_E} is the value function due to utilizing the policy π_E (obtained from solving the MDP with the learned reward function) in E's MDP.

We used EVD in evaluating performance of multi-task learning. Since the learning process involves cluster assignments for input trajectories y_i, the metric EVD is computed as

$$EVD = \frac{1}{|\mathcal{Y}|} \sum\nolimits_{i=1}^{|\mathcal{Y}|} \sum\nolimits_{(s,a)\in y_i} \|V^{\sigma^{\boldsymbol{\theta}^*_{c_i}}}(s) - V^{\sigma^{\boldsymbol{\theta}^L_{c_i}}}(s)\|_1.$$

$\boldsymbol{\theta}^*_{c_i}$ and $\boldsymbol{\theta}^L_{c_i}$ are the true and learned reward weights (component parameters) for the cluster c_i assigned to the observed trajectory $y_i \in \mathcal{Y}$, and $\sigma^{\{\cdot\}}$ denotes the corresponding policy.

Since above definition of LBA works only for discrete states, we modified it slightly to measure LBA for OpenAI gym domains.

LBA for continuous state discrete action domain For input state s, $a_{g|s}$ is deterministically sampled action from ground truth (expert's) policy and $a_{l|s}$ is deterministically sampled action from the learned policy. Indicator function $1_{a_{g|s}=a_{l|s}}$ takes the value 1 if the two actions are same for a state. With that structure, LBA is the approximated integral of indicator values over state space S.

$$\frac{100}{|S|} \int_{s\in S} 1_{a_{g|s}=a_{l|s}} = \frac{100}{|Q|} \left(\sum_{S_i, i\in\{1,2\ldots Q\}} \int_{s\in S_i} 1_{a_{g|s}=a_{l|s}} \right)$$

where S_i are state space partitions.

LBA for continuous state continuous action domains For extending the previous expression of LBA to continuous actions, we need a metric to calculate the similarity between two action vectors. Let $D_a(a_{g|s}, a_{l|s})$ be the distance between $a_{g|s}$ and $a_{l|s}$. We used D_a to be L2-norm but it can be any distance metric of user's choice (Manhattan distance, max of absolute values, etc.). Let D_m be the maximum D_a

distance possible between two actions. As almost every gym domain defines max and min values possible for the dimensions of action space, for L2-norm, D_m is the distance between minimum valued and maximum valued action. The metric of similarity between two actions $a_{g|s}$ and $a_{l|s}$ is $(1 - D_a(a_{g|s}, a_{l|s})/D_m)$. Then LBA is the Monte carlo approximation of integral over the similarity w.r.t. each of N states sampled from S.

$$\frac{100}{|N|}\left(\sum_{i\in\{1,2...N\}}(1 - D_a(a_{g|s}, a_{l|s})/D_m)\right)$$

Some metrics are **specific to the application domains**.

For 2nd and 3rd instances of *Perimeter Patrol* application, we used *Success rate* and *Timeout rate*. *Success rate* is the percentage of runs where the learner reaches the goal state undetected by the patrollers, which is the culmination of learning the patrollers' trajectories accurately, planning, and navigating to the goal location. The latter metric calculates the percentage of runs in which IRL fails to converge to a reward estimate in a reasonable amount of time. We set the threshold for a timeout as the total time taken for the 'perception of trajectories in demonstration, learning, and two rounds of patrolling', averaged over many trials. This gives the learner at least two chances for penetrating the patrol.

For *Onion Sorting* application, we used a pair of metrics to measure the performance of Sawyer on the sorting task using the learned reward functions. *Precision* is the ratio of the number of onions placed in the bin that are actually blemished to the total number of onions placed in the bin. *Recall* is the ratio of the number of onions placed in the bin that are actually blemished to the number of onions that are actually blemished (including both in bin and on table).

For three OpenAI continuous state continuous action environments, we used the **averaged return of trajectory** because it is a commonly used metric in literature for high-dimensional OpenAI domains, and using that will help put the results in the same perspective as the rest of the literature. In these three environments, due to the structure of reward function, a longer trajectory have a default tendency for larger return. Therefore, to make this metric fair, we fix the episode length and then calculate return averaged over multiple episodes.

We should recall the goals of contributions so that there is more clarity about why we chose some of the metrics specifically to evaluate these contributions.

- LME-I2RL learns a reward function online using demonstration with occluded (missing) information, and it learns faster than batch method LME-IRL. To visualize the impact of improvement in the learning speed, we used the metrics of 'learning duration', *Success rate*, and *Timeout rate* in perimeter patrol domain.

- online MME-MTIRL learns reward function(s) when input has data generated using one reward function mixed up that from other reward function(s). EVD is a commonly used metric in such

multi-task learning methods because it allows accumulating the information from all learned clusters in one metric.

- RIMEO learns observation features distribution and learns reward function under perception noise. We use KLD (KL-divergence) metric because that metric can compare learned feature distribution with ground truth distribution available in simulated domains.

- robust-AIRL learns under perception noise without manual engineering of reward features. For evaluating GAN-based methods in OpenAI gym domains, metric reward correlation is common because it enables measuring the closeness of learned information and expert without relying on the distance based metrics. Also, this metric was precise enough to capture the impact of perception noise.

7.3 Baselines

For online learning under missing data (LME-I2RL), we used two baselines: the offline or batch version of learning under missing data (LMEIRL), an online variant of imitation learning method GAIL [43]. The reason for choosing LMEIRL is to show the impact of modifying batch learning to online learning. We choose GAIL as a baseline for comparison against state-of-the-art. An online learning process must use the information passed from an I2RL session to its subsequent session. To dive deeper into the impact of this transfer of knowledge, we created a custom baseline (LME I2RL *with random weights*) that uses sessions but without passing information.

Since online MME-MTIRL is multi task learning method, we choose state-of-the-art multi task learning methods as baselines: DPM-BIRL [29] and EM-MLIRL [9]. Using these baselines helped brought out the impact of removing label bias via proposed method. We also used these batch learning baselines to depict the impact of learning faster using online learning.

We use MLIRL [81] and AIRL [35] as baselines for the proposed method RIMEO. Especially, since AIRL is online learning method, it served as a good choice for a baseline. To our knowledge, no other IRL method infers an observation model. Therefore, we create a custom baseline using the frequentist approach, which uses robust I2RL with a frequentist estimation of the observation model (RIFOM). While RIMEO solves an optimization problem to learn P_i^*, RIFOM assumes that the ground truth is the same as the observation and learns P_i^* as the running average of normalized frequencies of observation features activated in the demonstration.

Since SSRR noisy-AIRL [25] is a state-of-the-art method to learn well despite noisy input and without need for manual engineering of reward features, it fits well as a baseline for robust-AIRL. As robust-AIRL is derived from AIRL, AIRL (both under noise-free and noisy input data) became an obvious choice for another baseline.

7.4 Evaluation Results

In this section, we will show evaluation results for the research contributions addressing the issue of online learning under certain imperfections in input data. We categorize the sub-sections the same way we categorized the last few chapters, by type of imperfection in data: missing information, data mixed up from other preferences for finishing a task, and noise coming from the faulty perception of the learner.

7.4.1 Missing data: LME I2RL

The performance of LME-I2RL is compared with the previous batch LME to evaluate the advantage of online IRL in the above mentioned domain in both simulations and physical experiments. As the learner's vantage point in perimeter patrol domain limits its observability, the patrolling domain requires IRL but under occlusion. Please note that if we say that the domain has 30% occlusion, that means 70% (100 -30) part of the information about patrolling trajectories is missing from demonstration data. Our evaluation shows that the incremental method learns the patrolling behaviors faster as compared to the batch method, which leads to a higher success rate. It suffers from far fewer timeouts as compared to the batch method where timeouts result due to failures to complete the inverse learning and forward planning within a specific time limit. Furthermore, LME-I2RL learns behavior that is also significantly more accurate as compared to online GAIL.

Experiment Setup

In our simulation experiments of perimeter patrol domain, we vary the degree of occlusion, the number of states in the domain, as well as the number of patrollers. Our experiments with physical TurtleBots in two environments confirm the performance obtained in simulation with the learner capable of observing just about 30% and 18% of the patrollers' trajectories, respectively. For scalability, later on, we show a comparison on physical robots on two instances of the domain, one of which is significantly larger than the other.

The same data was input to both proposed method and baselines in order to achieve a fair comparison. Each data point is the mean of 100 trials for a fixed degree of observability and a fixed number of trajectories in a demonstration $\mathcal{X}$. While the entire demonstration was given as input to the batch variant, the $\mathcal{X}_i$ for each session of I2RL had one trajectory composed of five state-action pairs. As such, the incremental learning stops when there are no more trajectories remaining to be processed.

Theorem 'Confidence for LME I2RL' from Chapter 4 allows us to derive an upper bound on the number of state-action pairs needed across all sessions to meet the given log likelihood error, which signals convergence. Table 7.1(a) shows this relation between the acceptable error ε_l, which is a function of ε (error bound for full observability) and ε_s (error bound for approximating feature expectations in Lemma 'Constraint Bounds for LME I2RL'), and the number of trajectories for a 80% confidence level. In our simulations, we choose $\varepsilon = 0.05$ and $\varepsilon_s = 0.05$, which yields $\varepsilon_l = \frac{\varepsilon+\varepsilon_s}{2} = 0.075$. Setting $\varepsilon_s = 0.05$ yields the maximum number of MCMC samples required in each E-step as $N = -\frac{1}{2\varepsilon_s^2} \ln \frac{\delta_s}{2K} = 957$.

Table 7.1: (a) Number of trajectories required for ε_l convergence in the second patrolling domain ($K = 6, \gamma = 0.99$) with confidence $1 - \delta_l = 1 - (\delta + \delta_s) = 1 - (0.1 + 0.1) = 0.8$. We use $\varepsilon_s = 0.075$ for both 30% and 70% observability. (b) Confidence of convergence increases with more trajectories (from more sessions) with $\varepsilon_l = 0.075$.

ε_l $(\varepsilon, \varepsilon_s)$	$\lvert\mathcal{Y}_{1:i}\rvert$
0.125 (0.2, 0.05)	60
0.075 (0.1, 0.05)	**239**
0.05 (0.05, 0.05)	957
0.045 (0.04, 0.05)	1496

(a)

$\lvert\mathcal{Y}_{1:i}\rvert$	$\max_{(0,\, 1-\delta_l)}$
115	0
135	0.19
200	0.78
375	0.99

(b)

For a ε_l of 0.075 for our experiments, Table 7.1(a) shows that at most 239 trajectories would be needed. Table 7.1(b) shows that, for the chosen value of ε_l, the confidence of convergence increases with more sessions as we should expect.

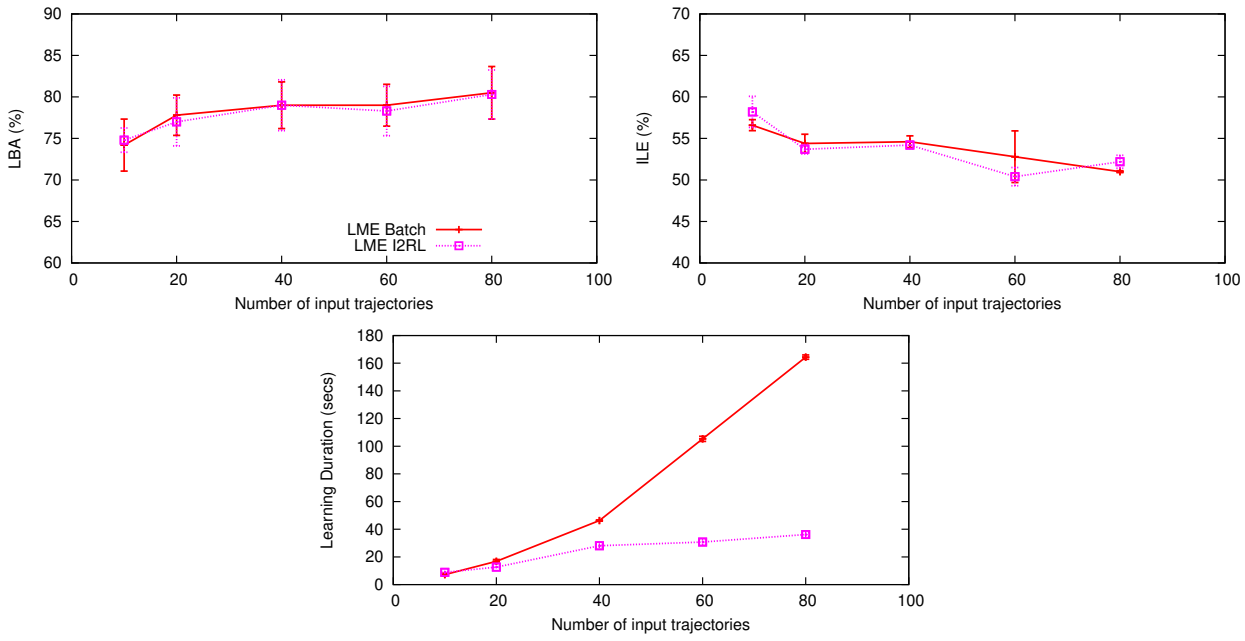
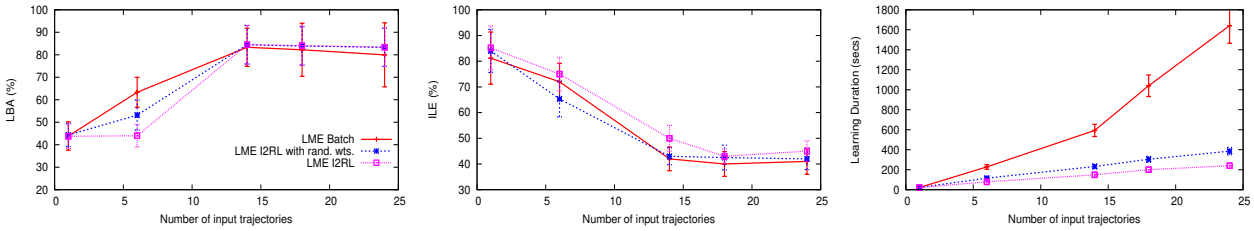


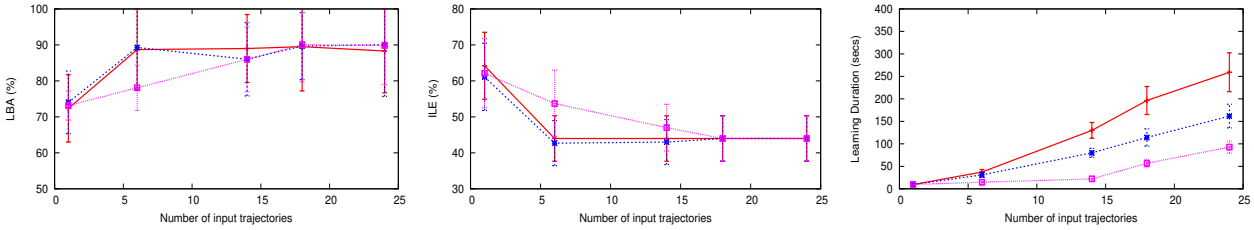
Figure 7.3: Performances of batch and incremental LME in the first domain instance of Figure 7.1(a). All simulations were run on a Ubuntu 14 LTS system with an Intel i5 2.8GHz CPU core and 8GB RAM.

Performance Results in Simulation

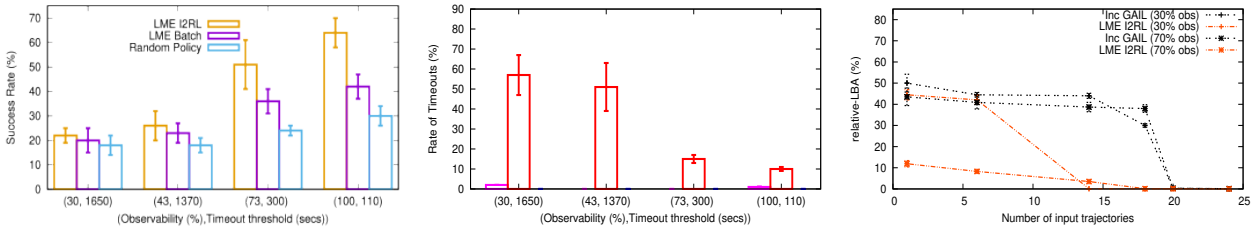
In the single-patroller instance of patrolling domain, LME I2RL shows a learning accuracy that remains close to that of batch LME with the accuracy increasing as the number of observed trajectories increase Figs. 7.3). Importantly, LME I2RL processes the trajectories and achieves the eventual learning accuracy in significantly less time. Furthermore, it shows slow growth in its cumulative learning duration as we provide more trajectories.



(a) Learned behavior accuracy, ILE, and learning duration under a 30% degree of observability.



(b) Learned behavior accuracy, ILE, and learning duration under a 70% degree of observability.



(c) Success rates and timeouts under 30%, 70%, and full observability. Performance when the learner employs a random policy is shown as well. This baseline method does not engage in IRL and picks a random set of reward weights for computing the patroller's policy. Rightmost chart shows the relative difference computed as (LBA for full observability - LBA under occlusion)/(LBA for full observability) for both 30% and 70% observability. We expect this ratio to reduce as the observability increases.

Figure 7.4: Various metrics for comparing the performances of batch and incremental LME on the second instance of the patrolling domain (Fig. 7.1(b)).

Next, we report the evaluation results for the second instance of patrolling domain (involving two patrollers). As these experiments are simulations, we may vary the learner's observability, and Fig. 7.4(a) shows the results under a 30% degree of observability while Fig. 7.4(b) is for 70% degree of observability. To better understand the differentiations in performance, we introduce a third variant that implements each session as, $\zeta_i(MDP_{/R_E}, \mathcal{Y}_i, |\mathcal{Y}_{i:i-1}|, \hat{\phi}^{\hat{Z}|_Y,1:i-1}_{\boldsymbol{\theta}^i})$. Notice that this incremental variant does not utilize

the previous session's reward weights; instead, it initializes them randomly in each session. We label it as LME I2RL *with random weights*.

We empirically verify that convergence is indeed achieved within 239 sessions (each having one trajectory). As the size of demonstration increases, both batch and incremental variants exhibit a similar quality of learning although initially the incremental performs slightly worse. Importantly, LME I2RL achieves these learning accuracies in significantly less time compared to the batch method, with the speed up ratio increasing to four as $|\mathcal{X}|$ grows. On the other hand, the batch method generally fails to converge in the total time taken by the incremental variant. Between the two degrees of observability, less observability exhibits a longer learning duration because of the need for increased inference that is time consuming. Notice that a random initialization of weights in each session, performed in LME I2RL with random weights, leads to higher learning durations as expected. The link to video of a simulation is available in Appendix A.

Is there a benefit due to the reduced learning time? Figure 7.4(c) shows the success rates of the learner when each of the three methods are utilized for IRL. LME I2RL begins to demonstrate comparatively better success rates under 30% observability, which further improves when the observability is at 70%, as we should expect. While the batch LME's success rate does not exceed 40%, the incremental variant succeeds in reaching the goal location undetected in about 65% of the runs under full observability (the last data point). A deeper analysis in order to understand these differences in success rates between batch and the incremental generalization of LME reveals that batch LME suffers from a large percentage of *timeouts* while incremental LME suffers from very few timeouts. Notice that as the perception and learning times increase with the size of input data, so does the corresponding timeout threshold. The batch method shows a small 10% timeout rate for the full observability case which increases to more than a 50% timeout rate for low observability; whereas, the rate for incremental method stays below 4% throughout. LME with low observability requires more time due to the larger portion of the trajectory being hidden, which requires sampling a larger trajectory for computing the expectation. Of course, other factors play secondary roles in the success rate as well.

We compared the performance of LME I2RL with that of an online version of GAIL [43], a state-of-the-art policy learning method cast in the schema of generative adversarial networks. We experimented with various simulation settings for GAIL and eventually settled on one that seemed most appropriate for our domain (500 iterations of TRPO with an adversary-batch-size of 1,000, 2 hidden-layer [64×8] network for both generator and discriminator, adversary-epochs = 5, and generator-batch-size = 150). We obtained a maximum LBA of 52% for the fully observable simulations (note that fully observable trajectories still may not yield all possible state-action pairs). As this absolute performance was rather low, we analyzed the relative impact of occlusion in our scenario on the performance of GAIL. The rightmost chart in Figure 7.4(c) shows that while both LME I2RL and online GAIL demonstrate the same relative difference initially, GAIL requires significantly more trajectories before it catches up with its full-observability performance, for both the 30% and 70% observability cases. As such, online GAIL appears to be far more impacted by occlusion than LME I2RL.

Performance Results on Physical Robots We used Physical TurtleBots to set up the *second* and *third* instances of the domain both of which exhibit less than 30% observability (> 70% part of patrolling trajectories is missing from observation). The TurtleBots, acting as both the patrollers and the learner, are each equipped with a Kinect-360 RGB-D camera and an ASUS laptop. Differently colored boxes are placed on top of each as markers (see Fig. 7.5). The learner uses the ROS stacks for TurtleBot and and CMVision to perceive the centroid, the width, and the height of the colored boxes on the two patrollers to track them. Utilizing this data about the patrollers, the dimensions of the known floor map, and the learner's own position retrieved via Monte Carlo localization, the learner tracks the state (x,y, orientation) from its observations of each of the two patrollers. We utilize aggregated observations over a duration of 2 seconds to recognize the state. Having recognized the states in this way, the action of a patroller is inferred by using the states before and after its motion for 2 seconds. The computation of the timeout threshold is same as that for simulations.

Figure 7.5: **(top-left)** Patrollers (in pink and red) in the longer hallway of instance 2 of the domain. **(top-right)** Learner's (green) perspective as it observes the patrollers from its vantage point. **(bottom)** Learner penetrating the patrol to reach the goal undetected (first door on its right). The learner perceives the location and the orientation of each patroller by using the depth and the bounding box for color blob detected via CMVision.

For each data point in the physical experiments, we conducted 50 trials, with the number of trajectories and the number of state-action pairs per trajectory being the same as those in simulations. As the degree of observability cannot be changed in our physical setup, we varied the number of input trajectories in order to observe the change in success rate and timeout rate. Figure 7.6 shows the results of a comparison between the performances of batch LME and LME I2RL deployed on the TurtleBot. Despite the low observability, the success rate for LME I2RL is consistently higher than that for batch LME (reaching close to 40%), thereby showing consistency with the results in simulations. The timeout rate while higher than those in simulations remains much lower for LME I2RL compared to its batch counterpart.

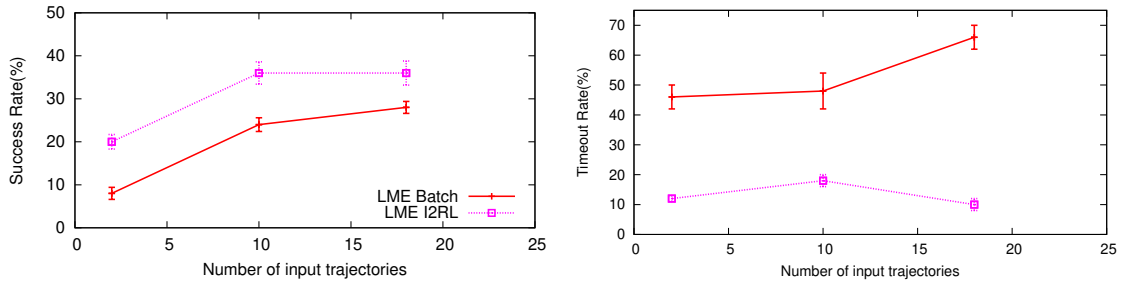


Figure 7.6: The success and timeout rates achieved in the physical experiments for the second instance of the domain. The learner has less than 30% observability.

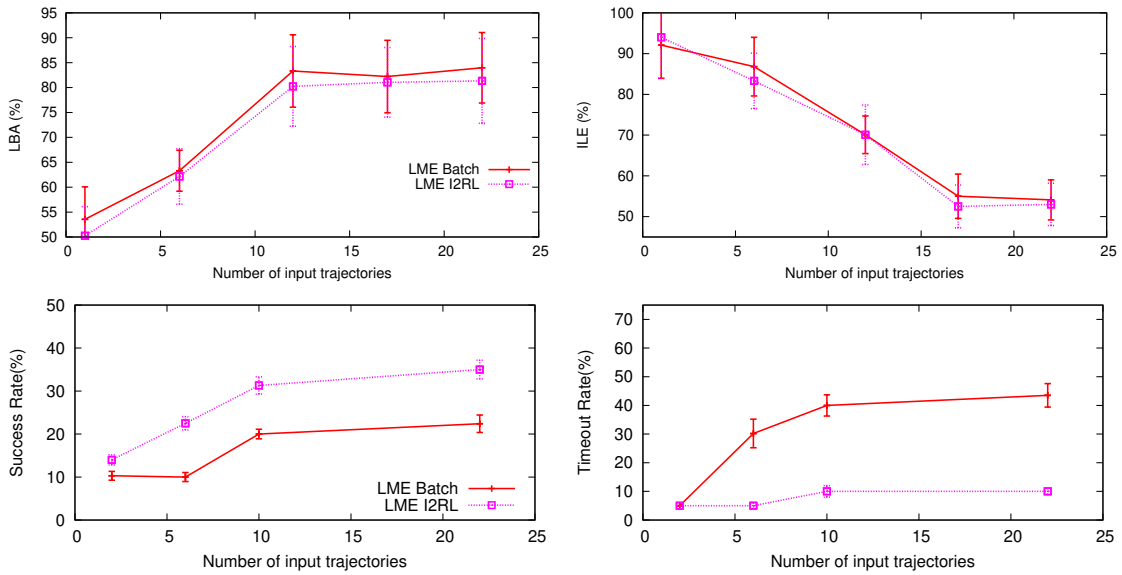


Figure 7.7: Comparative performance of batch LME and LME I2RL in the third instance of the patrolling domain using physical robots. We assumed knowledge of the patrollers' true polices in order to obtain the LBA and ILE metrics but note that such information is typically unavailable to the learner in practice.

We ran physical robot experiments on the larger third instance of the patrolling domain whose map is shown in Fig. 7.1(c) (some visuals of this set up are available in Appendix C.) Note that the theoretical convergence analysis performed previously in Section 7.1.3 for the second domain instance also applies to the third instance because the number of MCMC samples in the E-step remain the same except that $K = 5$ for the third domain. Following the same sample complexity calculations as before, LME I2RL must converge to $\varepsilon_l = 0.075$ with probability $(1 - \delta_l) = 0.8$ in about 230 input trajectories. Indeed, it achieves convergence within this prescribed size of the input data.

Despite very low observability and a larger state space, I2RL has a success rate going up to 35% as shown in Fig. 7.7, which is 10% higher than that for the batch method. Interestingly, that success rate is reached by observing about 22 trajectories only, i.e. 110 state-action pairs. The timeout rate for I2RL stays below 10% while that of batch LME crosses 40%. Clearly, faster convergence helps the attacker to succeed more often. Observe that for the initial data points, despite both methods exhibiting a similar LBA, the success rate of I2RL is increasing but that of batch LME stays constant followed by a slight increase. This occurs in part because I2RL shows less timeouts for those data points and also learns patroller behaviors whose accuracy improves with more data.

To summarize the results for LME-I2RL, experiments conducted on multiple configurations of the patrolling domain in both simulation and on physical robots demonstrate the significant improvement in task performance brought about by online IRL. This is predominantly due to the incremental learning that leverages the learned outcomes from previous sessions while keeping the learning problem in each session small. Next, we discuss the results of online learning under second imperfection in training data.

7.4.2 Multiple tasks interleaving: MME-MTIRL

We evaluate the performance of online MME-MTIRL in comparison with two previous multi-task IRL techniques and batch MME-MTIRL, on the problem of sorting out blemished onions from unblemished. Expert demonstration data is a mixture of two ways of sorting, and learner is supposed to infer reward functions for both ways individually. We show that online MME-MTIRL improves on both baselines and learns the reward functions to a high level of accuracy, which allows the collaborative robot Sawyer to observe and reproduce both ways of sorting the onions while making few mistakes. We evaluate the performance of this method in comparison with the previous techniques (baselines) on a simulated onion sorting problem. In particular, one baseline (DPM-BIRL) sometimes learns more clusters than in the ground truth, while the other baseline (EM-MLIRL, which knows the true number of clusters) shows low accuracy, compared to our method.

Experiment Setup

We test the learning performance of different methods in onion-sorting, by using simulated trajectory data as input, and we compare their sorting performances using real-life (human) demonstration as input. For the former, we generate the two weight vectors of expert as follows: we collect expert trajectories by executing the behaviors from multiple start states, run an IRL algorithm on the two sets of trajectories to generate the respective weights, and finally verify that using these weights indeed yields the desired behaviors.

As careful inspection tends to be more accurate than simply rolling over the onions, we expect the behavior of pick-inspect-place (refer Section 7.1.2) to exhibit a higher precision compared to the roll-pick-place. On the other hand, it is slower compared to rolling and placing, hence its recall is expected to be lower for pick-inspect-place.

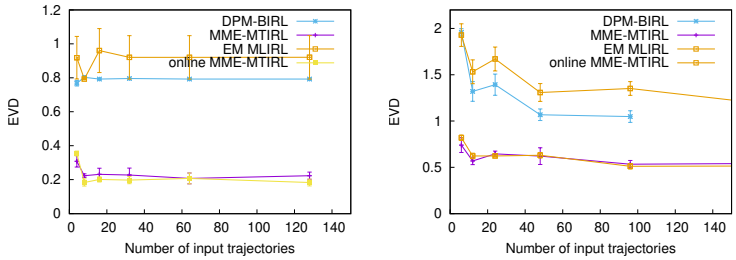


Figure 7.8: Average EVDs of MME-MTIRL, DPM-BIRL and EM-MLIRL as the number of trajectories increases, with two demonstrated behaviors (or tasks) **(top)** and three demonstrated behaviors **(bottom)**. Vertical bars are the standard deviations. Note that DPM-BIRL did not successfully terminate for the last data point.

Performance Results Using Simulated Data

We use the metric of EVD as defined previously to measure the performance of MME-MTIRL. Figure 7.8 (top) shows EVD decreased as the number of input trajectories increased for MME-MTIRL method as well as existing baselines, DPM-BIRL [29] and EM-MLIRL [9]. Each data point is the average of 10 runs on the set of trajectories generated as described previously. MME-MTIRL methods correctly learned two clusters (starting with initial D of 4) in most runs, though a few yielded just one cluster. On the other hand, DPM-BIRL mostly learned three clusters while EM-MLIRL learned two predominantly (because EM-MLIRL takes the number of clusters as a prior knowledge/ input). For a preliminary demonstration of scalability of MME-MTIRL in the number of learned tasks, we add a (meaningless) third behavior where the sorter keeps considering new onions but does not inspect them, preferring to do nothing. Figure 7.8 (bottom) shows the average EVDs for learning three tasks. Notice that the MaxEnt based MME-MTIRL exhibits EVDs that are consistently and significantly lower than those of the Bayesian DPM-BIRL. The former correctly learns three clusters in two-thirds of the trajectory sets (otherwise four), while DPM-BIRL varies between three and four. Unlike the two DP methods, EM-MLIRL performs worse throughout, converging to incorrect reward functions that are likely local optima. In terms of speed of learning, DPM-BIRL is faster than MME-MTIRL, and EM-MLIRL is significantly slower than both. For 96 input trajectories, on average, they take 11.94 seconds, 49.06 seconds, and 265.43 seconds respectively. To address the issue of speed of learning, we extend batch MME-MTIRL to online setting and computed EVD for the same. An I2RL session of online MME-MTIRL has the same amount data as the difference between two consecutive x-values for data points in Figure 7.8. For the same amount of total input trajectories, the average compute time for online MME-MTIRL is up to 75% less than batch MME-MTIRL, whereas learning performance (EVD value) remains the same (Figure 7.8). This is the result we expected for online learning using multi task input.

Performance Results Using Real-life Data

Next, we evaluate the performance of the these methods using human demonstration data. We simulate the onion sorting domain in our laboratory, and use as training data the videos of both sorting behaviors executed by human sorters (Fig. 7.2). We utilize SA-Net [76] to process the human demonstrations by identifying the sequences of states from the image frames. The network identifies sequences of states with prediction and listStatus received directly from the object recognition network YOLO [47]. We derive the actions in a trajectory from the state sequence. We use the Sawyer robot, a cobot arm with 7 degrees of freedom and a range of about 1.25m, as the learner executing learned policies. We partially simulate a moving conveyor belt by repeatedly making a collection of onions – some of these are blemished – appear on the table for a fixed amount of time **after which the onions disappear**. We task Sawyer with sorting as many onions as possible from each collection before it disappears. We utilize the MoveIt motion planner to plan Sawyer's actions.

Table 7.2: P-I-P and R-P-P stand for Pick-inspect-place and Roll-pick-place resp. Column labels TP denotes true positive (# blemished onions in bin), FP denotes false positive (# good onions in bin), TN denotes true negatives (# good onions remaining on table), and FN denotes false negatives (# blemished onions remaining on table). P and R denote precision (= TP/(TP+FP)) and recall (= TP/(TP+FN)) in %, respectively.

		(TP,FP,TN,FN)	**P%, R%**
Expert	P-I-P	(7,0,12,5)	100.00, 58.33
	R-P-P	(9,4,8,3)	69.23, 75.00
Learned (MME-MTIRL)	P-I-P	(7,1,11,5)	87.50, 58.33
	R-P-P	(9,4,8,3)	69.23, 75.00
Learned (DPM-BIRL)	P-I-P	(7,3,9,5)	70.00, 58.33
	R-P-P	(9,4,8,3)	69.23, 75.00
Learned (EM-MLIRL)	P-I-P	(6,3,9,6)	66.67, 50.00
	R-P-P	(6,4,8,6)	60.00, 50.00
Learned (Online MME-MTIRL)	P-I-P	(8,0,12,4)	100.00, 66.67
	R-P-P	(10,4,8,2)	71.43, 83.33

Does the improvement in learning translate to improved performance in the sorting task? In Table 7.2, we show the average precision and recall of the expert engaged in using the two sorting techniques and the analogous metrics for the learned behaviors using all three IRL approaches. For each of the three IRL methods, we compute the learned policies for two behaviors by using the feature weights averaged over 10 runs of learning. Then we execute each policy in the physical domain for three moving sets of eight onions per set – four blemished and four good – giving as output the precision and the recall corresponding to each learned behavior. The performance of the behaviors learned by MME-MTIRL is closer to that of the expert's than those learned by the two baseline methods. The roll-pick-place behavior is learned satisfactorily and exhibits precision and recall close to those of the true behavior. Finally, we also show in Fig. 7.2 Sawyer executing both of the onion-sorting behaviors autonomously. It uses Kinect-v2 point-cloud to track the onion locations and YOLO to classify onions held in the gripper. We used same human

demonstration data to evaluate online MME-MTIRL, dividing the set of demonstration trajectories to serve as a sequence of input demonstrations over multiple I2RL sessions. As explained in Section 7.4.2, onions disappear after a constant time limit is crossed. A learner keeps observing training until the learning has converged and then it starts sorting. Faster the learning process finish, more time a learner has to go through onions. Since online MME-MTIRL converged faster than batch MME-MTIRL, former method had time to pick more blemished onions from table and therefore has higher recall than latter method.

This evaluation proved that proposed method not only address the gap of lack of online multi task learning without label bias (for explanation of label bias see Chapter 1 section on multi task learning) but also learns quicker than baseline. Next section discuss the evaluation results for online learning under third type of imperfection in data.

7.4.3 Noisy perception of learner

We contributed two methods that address online learning under perception noise. Therefore, this section has two parts, showing evaluation results for RIMEO and robust-AIRL respectively.

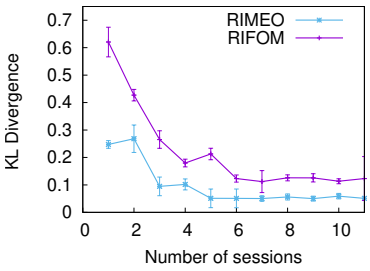
No prior knowledge of observation model: RIMEO

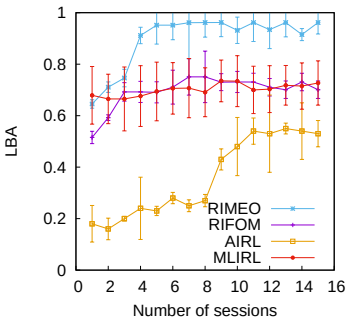
As evaluating RIMEO does not need multiple reward functions, we used only pick-inspect-place behaviour of sorting onion domain (i.e. demonstration did not have trajectories from roll-pick-place behaviour). The expert aims to identify blemished onions from a collection of onions on a table, and drop blemished ones in a bin while allowing others to pass.

Experiment Setup We use videos of a human sorting onions as the training demonstration. We utilize SA-Net [76] and YOLO [47] for identifying the sequences of states in the videos. The perception pipeline assigns prediction score $c(s, a)$ to each observed state-action of the trajectory perceived by learner. This noisy trajectory and the confidence scores are input to I2RL sessions of all methods. The learned policy is used in a sorting domain simulated in Gazebo 7 (ROS Melodic) with the cobot Sawyer as the learner. The experiments were run on a desktop with Intel Xeon CPU (E5-1603, 2.8GHz) and 16GB RAM. We introduce noise as $P(\psi_j(\langle s, a\rangle_o) \neq \psi_j(\langle s, a\rangle_g) = 0.3$ for $j =$ *BlemishedOnion, OnionToBin, OnionToTable* (please refer observation features of onion sorting domain in Appendix F), and no noise in the remaining observation features.

Performance Results: We evaluate the divergence of the learned $P^*(\psi)$ from a *simulated* dataset for which the true distribution $P(\psi)$ is known. Figure 7.9 shows the KL divergences between the distributions $P(\psi)$ and $P_i^*(\psi)$ with sessions i. Each data point is averaged over 50 runs, and each session has 2 trajectories of size 3 as input. The divergence for RIMEO is significantly lower than that of RIFOM (custom baselines explained before in Section 7.3) for all the sessions, indicating that RIMEO learns a more accurate P_i^*. Observe that the divergence starts stabilizing after the sixth session. As AIRL or MLIRL do not directly learn an observation model, we do not report their divergence measures.

Figure 7.9 shows the average LBA for our *human demonstration data set* as the number of sessions increases, for all three methods. While AIRL's LBA improves initially (until session 11), it flattens out





Method	TP,FP,FN,TN	Precision%, Recall%
Expert	7, 0, 12, 5	100.00, 58.33
RIMEO	8, 1, 11, 4	88.89, 66.67
RIFOM	8, 3, 9, 4	72.73, 66.67
AIRL	5, 4, 8, 7	55.56, 41.67
MLIRL	7, 3, 9, 5	70.00, 58.33

Figure 7.9: (**left top**) Divergence between $P(\psi)$ and $\hat{P}(\psi)$ for simulated onion sorting data. (**right top**) LBA of the learned policy on the human demonstration data. (**bottom**) Column label TP denotes true positive (# blemished onions in bin), FP is false positive (# good onions in bin), TN true negative (# good onions remaining on table), and FN denotes false negatives (# blemished onions remaining on table).

to a value (maximum 0.48) that is considerably lower than the other two methods. On average, AIRL is also slower to converge (about 8 minutes) as compared to RIMEO (96.69 seconds). This performance is likely due to the need of neural network based adversarial techniques for more training data and that AIRL suffers when the training data (that is used by the discriminator) is noisy. Furthermore, RIFOM's LBA flattens out around 23% lower than RIMEO's. This is because the maximum-entropy optimization of the underlying observation features in RIMEO generalizes the learned information across (s, a) pairs not present in the demonstration, but the frequentist baseline fails to generalize. RIMEO eventually reaches an LBA of 96% whereas RIFOM's stays around 70%. However, as RIMEO solves an additional optimization problem for learning $\hat{P}$, it takes longer to converge (96.69 seconds, on average) compared to RIFOM's (85.99 seconds). MLIRL's LBA is similar to RIFOM's, but its high variance and inability to improve its performance can be attributed to observation noise. As MLIRL is not incremental (it was run with accumulating data in batch mode), runtime comparison is not meaningful and hence not reported. [1]

To evaluate sorting performance, Sawyer uses the learned policies to sort a fixed number of onions. The table in Fig. 7.9(right) shows the average precision and recall of the expert and the 4 learning methods. As the expert aims to inspect every onion whose blemish is not immediately visible, which is slow, it exhibits high precision but low recall. RIMEO's precision is closest to the expert's due to lower false positives. AIRL performs worse in both precision and recall having higher false positives and true negatives. This is

[1]For interested readers, one way to investigate AIRL's performance further is to apply AIRL with the features used in linear reward function used by RIMEO.

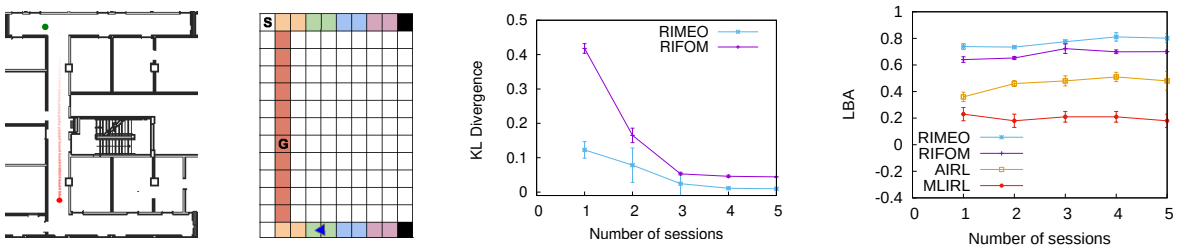


Figure 7.10: (**left**) Patroller (red) navigates hallways [13], and learner (green) aims to pass undetected. The 5 colored regions (long hallway, turning points, and three small divisions in small hallways on both sides) define movement and turn-around features. S and G are start and goal locations for the learner. (**middle**) Divergence between $P(\psi)$ and $\hat{P}(\psi)$, and (**right**) LBA of the learned policies on Gazebo simulation data of learner.

consistent with its low LBA, indicating its reduced ability to successfully sort onions under noise. Among the two robust I2RL methods, as the steps after learning the observation model are the same, we attribute the better precision of RIMEO to a more accurate observation model.

To establish the generality of RIMEO, we evaluate it in a second domain: a simplified version of the second instance of perimeter patrol domain. It involves one patroller continuously navigates a hallway in cyclic trajectories; the other is a learner tasked with reaching a goal location without being spotted by the patroller (Fig. 7.11). The state space, action space, reward features are same as explained in beginning of this chapter. The observation feature set Ψ contains 4 binary predicates described in Appendix F.

We introduce noise as $P(\psi_j(\langle s,a\rangle_o) \neq \psi_j(\langle s,a\rangle_g) = 0.3$ for $j =$ *MoveForward, TurnRight*, and 0 for other features. Average of pairwise feature correlation from the patroller's demonstration is -0.14 (p-value 0.06), indicating that the features are reasonably independent. The two figures (right) in Fig. 7.11 show the performance of both robust I2RL methods in perimeter patrol. As in onion-sorting, KL divergence and LBA for RIMEO is better than RIFOM, AIRL and MLIRL. These results in two unrelated tasks indicate that the presented method is sufficiently general for broad applicability.

We noticed two things that motivated further research: RIMEO needs manual engineering of reward features and AIRL does not need that engineering; and AIRL didn't perform well under perception noise. So if we find a way to improve AIRL performance under noise, then that should help avoid the need for manual feature engineering in online learning under perception noise.

No engineering of features: robust-AIRL

We evaluate the performance of AIRL, robust-AIRL, and the baseline SSRR noisy-AIRL under noisy input in various application domains. For each application domain, we first show how the task is modeled as an MDP, followed by results we got and insights we noticed. The first domain is a discrete state discrete action perimeter patrol domain, the second one is a continuous state discrete action OpenAI Gym domain of mountain car, and lastly, we use a set of continuous state continuous action high dimensional OpenAI Gym domains.

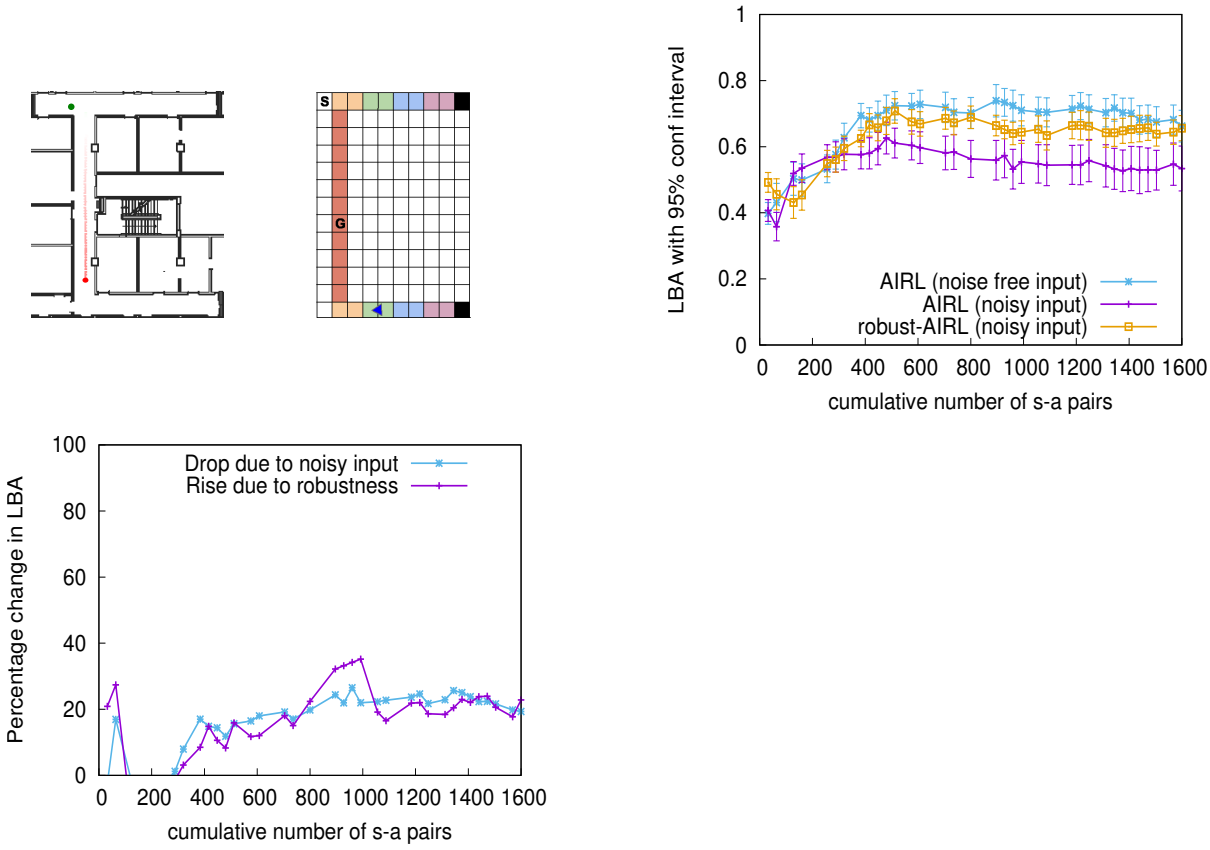


Figure 7.11: (**left** top) Patroller (red) navigates hallways [13], and learner (green) aims to pass undetected. The 5 colored regions (long hallway, turning points, and three small divisions in small hallways on both sides) define movement and turn-around features. S and G are start and goal locations for the learner. (**right** top) The third figure shows LBA values for three methods. (**left** bottom) The fourth figure shows how much LBA value changed from the introduction of noise in perception, and the introduction of Gibbs sampling in learning process. 'Drop due to noisy input' is computed as ((LBA AIRL under noisy input) - (LBA AIRL under noise-free input))/(LBA AIRL under noise-free input). 'Rise due to robustness' is computed as ((LBA robust-AIRL under noisy input) - (LBA AIRL under noisy input))/(LBA AIRL under noisy input).

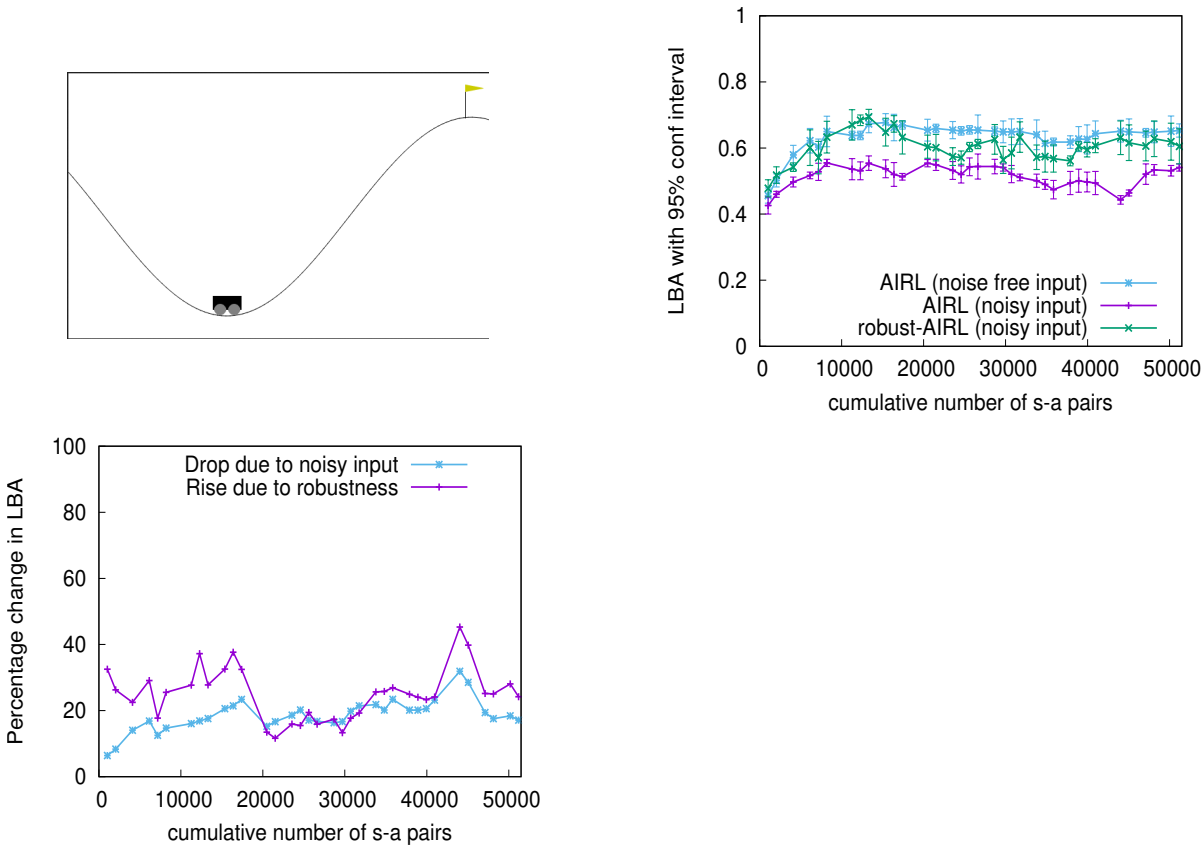


Figure 7.12: In Mountain Car environment, the goal of a learner is to find a policy that accelerates car to top right hill. The second figure shows LBA values for three methods and the Third figure shows how much LBA value changed from the introduction of noise and then the introduction of Gibbs sampling.

As first application domain, we use the same perimeter patrol domain as one used for RIMEO in previous section. For each data point of LBA, we conducted 50 trials and averaged the output. For success rate, we created 10 blocks of 10 runs, we computed percentage of success of reach block, and then averaged results over blocks. The learner can view (state , action) pair at every time step of patroller's trajectory but the perception mechanism of the learner is noisy. With a probability of 5%, the learner perceives move-forward action as turn-left action, and turn-right action as move-forward. Analyzing further 11.5% of total possible state-action pairs were noisy. Appendix G shows the hyper-parameter values used in GAN training. X-axis in the performance plots (7.11) is the cumulative number of state-action pairs seen so far by the learner. We used around 35 sessions so as to ensure stabilization of LBA values (although it somewhat settled much sooner). The size of demonstration varies for each I2RL session. As shown in 7.11, adding noise to input demonstration reduces LBA values. Averaging the percentage-change in LBA over the last few readings (15 sessions), noisy input dropped LBA by close to 20% and introducing robustness brought learning accuracy back up significantly. For each method, the learner in perimeter patrol used the policy learned at the end of last I2RL session. We average the success rate results over 20 attempts to breach. Under noisy input, the perimeter patrol success-rate of AIRL policy is within 45% - 53% range with 95% confidence and that for robust-AIRL lay within the range 59% - 66% . Our evaluation shows that the robust-AIRL method learned the patrolling behavior more accurately and faster as compared to the AIRL method, which led to a higher success rate.

To apply robust-AIRL in Mountain Car domain, we apply Technique 1 of computing Gibbs samplers (refer Chapter 6) , i.e. discretizing the continuous state space. We divided each dimension in $Q = 10$ parts, which created $Q^{n_s} = 10^2$ partitions of state space. At each time step of input batch of observations, we use $n_a = 10000$ discrete sample states from state partition to approximate [2] the value of integrals needed to compute Gibbs sampler probability values. A state has two parts: position and velocity. The size of position dimension of state space is $|X| = (0.6 - (-1.2)) = 1.8$. The cart position span of each state partition is $\frac{|X|}{Q}$. For insertion of visibly explicit noise, we increase the position part of the current state by V=10% of $\frac{|X|}{Q}$ in the direction of ground truth action (left for action 0 and 1, right for action 2), with probability 0.25. Then the observation model or noise insertion model is,

$$P(s_o|s_{g,t} \in S_g, a_{g,t} = a_g) = \begin{cases} 0.25 \text{ if } (s_o = s_{g,t} - V\frac{|X|}{D} \text{ and action is 0 or 1) OR} \\ (s_o = s_{g,t} + V\frac{|X|}{D} \text{ and action is 2)} \\ 0.75 \text{ otherwise} \end{cases}$$

The introduction of noise in the input data dropped LBA value by about 22% w.r.t AIRL under noise free input (Figure 7.12). AIRL couldn't accommodate that noise but robust-AIRL managed to increase performance back up closer to LBA under noise free input. However, as explained in Chapter 6 about 'Technique 1', the state space discretization technique suffers from a lack of scalability with the number of dimensions in a state. That's why, in the next set of OpenAI application environments, we used Technique 2 of computing Gibbs samplers, the product of Gaussians.

[2] numerical Monte carlo approximation

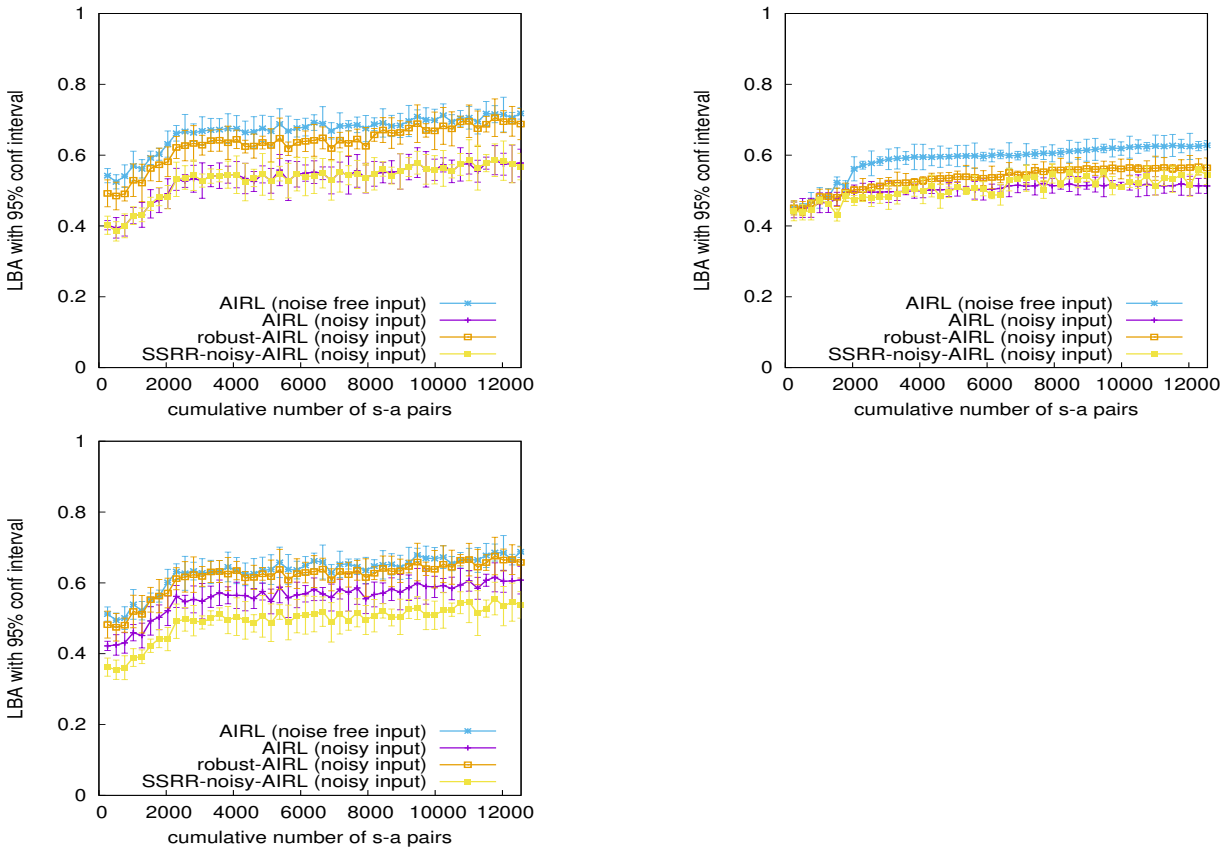


Figure 7.13: In three continuous state continuous action environments (Hopper, Walker2D, HalfCheetah), the goal of a learner is to find a policy that moves robot towards right side as fast as possible. These plots show the LBA values achieved by four methods.

Table 7.3: Averaged episode return for Mujoco domains

	Hopper	Walker2D	Half Cheetah
Expert	2810	2700	2507
AIRL (noise free input)	2130	2204	2251
AIRL (noisy input)	702	848	750
robust-AIRL (noisy input)	1810	1390	1662
SSRR + noisy-AIRL (noisy input)	2050	1932	1555

Table 7.4: Pearson correlation coeff between learned and ground truth rewards for Mujoco domains. Second part of each tuple is corresponding p-value.

	Hopper	Walker2D	Half Cheetah
AIRL (noise free input)	(0.95, 0.02)	(0.94, 0.01)	(0.96, 0.02)
AIRL (noisy input)	(0.87, 0.04)	(0.85, 0.02)	(0.86, 0.03)
robust-AIRL (noisy input)	(0.96, 0.04)	(0.93, 0.03)	(0.94, 0.04)
SSRR noisy-AIRL (noisy input)	(0.81, 0.02)	(0.90, 0.03)	(0.87, 0.02)

We evaluated the proposed method and baseline (SSRR noisy-AIRL) on three continuous state continuous action environments using three metrics: correlation with ground truth rewards, average return per trajectory, and LBA. [3].

In each of three environments, the introduction of noise in the input demonstration corrupted the motion of the expert and the performance of the learner dropped (Table 7.4.3 row 2 and plots in Figure 7.13). Consequently, the motion of learner was sluggish and noisy. Robust-AIRL was able to improve trajectory return by more than 40% in most of them. As we explained Chapter 2, SSRR noisy-AIRL keeps looking for reward function with better trajectory returns than input, whereas robust-AIRL is looking for a reward function that is closer to the expert's (ground truth). As a result, SSRR noisy-AIRL gave higher returns for Hopper and Walker, however, the correlation (Table 7.4.3) and LBA (Figure 7.13) values were higher for robust-AIRL. For Hopper environment, reward values for robust-AIRL got slightly closer to ground truth than those for AIRL without noisy input. This probably happened because the artificially added noise made the hopper hop higher than the behavior learned by generator under noise free input.

[3]we noticed that LBA is less precise than correlation metric, due to inability of exactly measuring the similarity between two continuous actions w.r.t tasks at hand.

Figure 7.14: Images from a video of a human sorting onions. Such images were used to record the continuous states and actions used for learning in continuous space version of onion sorting domain.

Table 7.5: Results on real life data-set for onion sorting domain.

	$V^f_{\Xi_d}$	$V^f_{\pi_{gen}}$	ILE_{emp}
AIRL (noise free input)	0.907	0.726	0.199
AIRL (noisy input)	0.889	0.625	0.297
robust-AIRL (noisy input)	0.843	0.637	0.243

Evaluation using real life data

To make evaluation domain closer to real life, we evaluated Robust-AIRL on a dataset collected by observing a human sorting onions (Figure 7.14) with artificially introduced noise. MDP for this domain has 64 dimensional state space with each state comprising the information about x-y plane locations of demonstrator's arm, onions, and bin. Two dimensional action space models the movement of demonstrator's arm. An action includes change is hand position along x axis and change in hand position along y axis. On comparing state before action and state after action, if the change in location of an onion synchronizes with the change in location of hand, then it is assumed that onion is in the hand.

Please note that correlation metric and LBA metric for Mujoco environments (Hopper, Walker, HalfCheetah) were computed using the expert's reward function, which is available in OpenAI Gym domains. However, the expert's reward function is not available in the real-life dataset. Therefore, we compute the error metric of empirical ILE which is the relative difference between *averaged return of noise-free demonstration* and *averaged return of learned policy*, both computed using the learned reward function.

$$ILE_{emp} = \frac{V^f_{\pi_{gen}} - V^f_{\Xi_d}}{V^f_{\pi_{gen}}}$$

where

$$V^f_{\pi_{gen}} = \frac{1}{|\mathcal{X}_{gen}|} \sum_{X \in \mathcal{X}_{gen}} \sum_{<s,a> \in X} f(\boldsymbol{\theta}, s, a)$$

$$V^f_{\Xi_d} = \frac{1}{|\Xi_d|} \sum_{\xi \in \Xi_d} \sum_{<s,a> \in \xi} f(\boldsymbol{\theta}, s, a)$$

where Ξ_d are ground truth training data before noise insertion, f is reward function neural network with weights $\boldsymbol{\theta}$ learnt in robust-AIRL algorithm, $\mathcal{X}_{gen}$ is a set of trajectories simulated from generator's policy π_{gen}.

AIRL mostly uses batch normalization as the final layer of reward neural network. For bounding ILE within [0,1], we made the output of this last layer pass through Sigmoid layer. We introduced simulated perception noise in data by using a covariance of 0.01 in the Gaussian for observation model (terms T_{4-5} in Gibbs samplers, Chapter 6). As a result, ILE increased by about 10% (shown in Table 7.4.3). To accommodate learning under this noise, we used Robust-AIRL in continuous state and action spaces. As expected, the proposed method enhanced learning accuracy, bringing the error closer to that obtained under noise-free input. The ILE improved by almost 6% due to the robustness introduced by the MCMC sampling process in the algorithm.

Robust AIRL performed better than the baselines in both discrete and continuous space domains. It also learned well in the extremely challenging real-life continuous space domain of onion sorting.

Current chapter comprises the discussion about how all proposed I2RL (online IRL) methods performed admirably against baselines. Next and final chapter concludes this thesis with insights and future work.

Chapter 8

Conclusion

Researchers have recently started paying attention to the field of online IRL under imperfect input. Real world time-limited IRL applications typically demand learning algorithms that can build up learned reward function incrementally as more training demonstration becomes available. Such applications also call for robustness against imperfections in training data. The data imperfections can be of different types and we focused on these types:

- some information missing from the demonstration
- data from other tasks getting mixed up in the demonstration
- perception of learner-robot getting faulty due to hardware inaccuracies, etc.

My thesis proposes I2RL methods that address these gaps and we proved that methods perform as well (if not better) than state-of-the-art baselines in challenging application domains. The current chapter shares some final remarks about these methods in the same order in which they were introduced in the thesis. After concluding remarks, as requested by the dissertation committee, we discuss what challenges I faced while implementing these online learning methods. We conclude this chapter with some future work research ideas that can build work on top of the research in the thesis.

8.1 Final Remarks

To analyze the class of methods for online IRL mathematically, my research group introduced a formal framework called I2RL in Chapter 3. We used the framework for analyzing state-of-the-art online IRLliterature [46, 69, 35] and we used this framework to introduce new online learning methods (LME-I2RL, online MME-MT IRL, RIMEO, robust-AIRL. I2RL facilitates comparing various online IRL techniques, and facilitates establishing the theoretical properties of online methods. In particular, it provides a commonly usable definition of sessions and some reasonable evaluation metrics for researchers interested in developing techniques for online IRL. Table 8.1 shows such a comparison of online learning methods we discussed in thesis.

Table 8.1: Using I2RL framework to analyze and compare the online IRL methods in current literature and similar methods in this thesis. Second column shows the form used for reward function and last column shows the sufficient statistic passed from a session to subsequent session. This thesis contributed the last four methods.

Method	$\hat{R}_E^i$	statistic $\widehat{\mathcal{X}}$
Jin et al. [46]	$\hat{R}_E^{i-1}$	None
DARKO [69]	$\boldsymbol{\theta}^{i-1}$	None
AIRL [35]	$\boldsymbol{\theta}^{i-1}$	$\boldsymbol{\theta}_{\boldsymbol{gen}}^{i-1}$
LME-I2RL [5]	$\boldsymbol{\theta}^{i-1}$	$\lvert\mathcal{Y}_{1:i-1}\rvert, \hat{\phi}_{\boldsymbol{\theta}^{i-1}}^{Z\lvert Y,1:i-1}$
online version of MME-MTIRL [6]	$\{\boldsymbol{\theta}_d^{j-1}\}_{\forall d\in D}$	$\{\boldsymbol{\pi}_d^{j-1}\}_{\forall d\in D}, \lvert\mathcal{Y}^{1:j-1}\rvert$
RIMEO [7]	$\boldsymbol{\theta}^{i-1}$	$\lvert\Xi_{d,1:i-1}\rvert, P_{1:i-1}^*, \hat{\phi}_{\boldsymbol{\theta}^{i-1}}^{1:i-1}$
robust-AIRL	$\boldsymbol{\theta}^{i-1}$	$\boldsymbol{\theta}_{\boldsymbol{gen}}^{i-1}$

Next, we go through concluding remarks for each of the four I2RL methods we contributed in this thesis (LME-I2RL, online MME-MTIRL, RIMEO, robust-AIRL), one by one.

In Chapter 4, we presented a new method, LME-I2RL within the I2RL framework that generalizes 'maximum entropy IRL under occlusion' to online settings. Casting this method to the context of I2RL allowed us to establish key theoretical properties of maximum entropy I2RL and LME I2RL by ensuring the desired monotonic progress in learning toward convergence with a given confidence. With more training trajectories or better observability, likelihood loss $LL(\boldsymbol{\theta}_E|\mathcal{Y}_{1:i}) - LL(\boldsymbol{\theta}^i|\mathcal{Y}_i)$ for I2RL gets smaller, and the learned weights $\boldsymbol{\theta}^i$ get closer to the true weights or their equivalent. The confidence of convergence $1-\delta$ increases with more training data (greater $|\mathcal{Y}_{1:i}|$), more room for error (higher ε and ε_s), and fewer reward features (lower K). For LME, one way to look at the theoretical results is that error bound Lemma uses ε and ε_s to limit the absolute value of the gradient ($\hat{\phi} - E_X[\phi]$) used in a likelihood-maximization process. We showed in the Chapter 2 that feature-expectations represent the value of corresponding policy. Therefore bounding the gradient restricts the difference between the value of a learned policy and the value of the expert's policy. If two policies are within pre-specified closeness, then so should the likelihoods computed using those two policies. Exploiting that information from Lemma, Theorem bounds the probability of error in maximization. To complete the formal analysis of our online method, we introduced the notion of regret for I2RL and we proved that the average regret for LME I2RL approaches zero as the number of learning sessions increase.

Our comprehensive experiments show that the new I2RL method is better than the state-of-the-art batch method in time-limited domains; it generally reproduces the batch method's accuracy but in significantly less time. In particular, we showed that given the practical constraints on computation time exhibited by an online IRL application, the new method is able to solve the problem with a higher success rate. This IRL generalization also suffers less from occlusion than imitation learning methods that directly learn the policy or behavior. We showed that LME I2RL continues to perform better than

batch IRL on a larger state space and under lower observability. This offers evidence that LME- I2RL method is scalable to more complex domains.

Working on LME-I2RL led to the idea to extend I2RL to the case of learning multiple reward functions in a real-life agriculture-inspired application. The next contribution of online MME-MTIRL followed this idea and enables multi-task online learning. Here, we give some concluding remarks for that method as well.

Online multi-task learning has been motivated by the fact that humans may exhibit multiple distinct ways (called behaviors) of solving the same problem each of which optimizes a different reward function while still sharing the features. For IRL to remain useful in this context, it must be generalized to not only learn how many reward functions are being utilized in the demonstration, but also to learn the parameters underlying each task's model. Existing state-of-the-art multi task IRL methods are not online and they suffer from label bias (please refer to Chapter 1 for the meaning of label bias). We presented a new multi-task IRL method that combines MaxEnt IRL – a key IRL technique – with elements of the DP-based Bayesian mixture model. While minimizing the number of behaviors learned to explain the observations, it leverages the advantages of legacy MaxEnt IRL and facilitates solving the generalization as a single unified optimization problem. In the real-world inspired domain of onion sorting, we showed that both batch MME-MTIRL and online MME-MTIRL improves on previous multi-task IRL methods. The behaviors induced by the learned reward functions imitated the observed ones for the most part.

We expected the online method to give better results than batch methods because the learning converges faster in online method. Faster learning helped online learner to process more onions before they disappear, which in turn led to the sorting performance of online MME-MTIRL better than that for MME-MTIRL.

The next two contributions (RIMEOand robust-AIRL) extended online learning to inference under perception noise in training data. Next, we share some final comments about what we achieved in online learning under perception noise.

We introduced an online IRL method RIMEO that accommodates noisy observations and infers the unknown observation model by maximum entropy optimization. Not only does this method fill a gap in the prior work on online IRL but it also makes online learning more pragmatic, as evaluated in two robotics tasks of post-harvest onion sorting and breaching a patrol. We evaluated the proposed method against a state-of-the-art baseline on a video data set capturing post-harvest processing of onions in a laboratory, and showed that the method performs significantly better than the baseline. Furthermore, we formally proved that the method converges monotonically and admits a probabilistic upper bound on sample complexity that takes into account the accuracy of the learned observation model. The method also displays no-regret online learning.

After introducing RIMEO, we focused on exploiting deep learning based methods to avoid the need for manual feature engineering. Unlike previous online IRL techniques (LME-I2RL, RIMEO), both baseline 'SSRR noisy-AIRL' and proposed method robust-AIRL do not need manual engineering of reward features. robust-AIRL enables learning under perception noise in challenging high-dimensional continuous spaces. We showed that the proposed method of robust-AIRL can perform well under a

significant degree of perception noise in the demonstration. Unlike AIRL, the proposed method was able to learn a reward function with a reasonable level of trajectory returns. Also, the correlation with ground truth is higher for robust-AIRL than SSRR noisy-AIRL.

General Comments One of the advantages of online learning is to have a way to stop processing further input if learning has converged. The batch methods, on the other hand, have to finish processing all the data before stopping. Such benefits of online learning for LME-I2RL and multi-task I2RL performance are specific to time-limited tasks like permeter patrol and onion sorting because the learning there focuses on a faster convergence. However, the online learning advantages in methods RIMEO and robust-AIRL are more generic as the latter focus on both faster convergence and noisy perception.

8.2 Challenges Encountered

As requested by the committee, I have included this section to explain all challenges I encountered while working on online learning using imperfect input and how I address them.

- It was difficult to find the best way to stop online learning. Tuning of convergence thresholds takes a lot of time and I reduced some of that time by using parallelization in code.
- In I2RL, if the sufficient statistic (Chapter 3) is additive in nature, then it will explode in size and memory when sessions of learning progress. In online MME-MTIRL, even though trajectory indicators $v_{d,i}$ are optimization variables, they should not be used as sufficient statistic because of this explosion problem. I addressed this issue by using the running average of each mixture-component weight (π_d^j in Chapter 5) as the statistic.
- Derivation of convergence guarantees for LMEI2RL and RIMEO was not straightforward because there is no precedence for something like that in IRL literature.
- For online ME-MTIRL, the optimization problem involves a mixture of discrete and continuous variables. It took a lot of iterations to understand what part of the derivation steps can be relaxed to derive at gradient expressions that perform well in practice and are easily implementable in code.
- Designing of reward model features and observation model features for applications took a lot of trial and error. That need for reward designing was avoided by using deep learning in the final contributions (robust - AIRL).
- While conducting experiments, getting precise movements in the navigation robots and in the robotic arm was challenging because of laser scanning issues (bad localization) and trajectory planner issues (low precision).

8.3 Future Work

While the method LME-I2RL has shown advances in the area of online IRL, there remain multiple avenues for future research. One path is to focus on understanding how methods in the I2RL framework can learn without prior knowledge of the dynamics of experts. Another avenue is to explore the usage of LMEI2RL in other applications: for example, Rhinehart et. al.'s domain of activity forecasting.

Convergence guarantees for LME-I2RL do not distinguish between the quality of information available in different images (or different states in input). In future work, if a researcher is interested in studying that aspect of problem, then there is a need of deriving (or manually providing) usefulness ranking of images in data. After having such a ranking system in place, the rank information has to be incorporated in optimization constraints of LME-I2RL (may be parametrizing the weights in linear reward function). If that much is achieved, the derivation of new sample complexity bound should follow the steps similar to those for LME-I2RL.

All the current evaluations of online MME-MTIRL are on discrete low-dimensional state space. Therefore, for bringing this work further close to addressing real life challenges, one possible future work direction for online multi-task learning can be exploiting deep learning to accommodate multi-task learning in high-dimensional continuous state / action spaces.

Method RIMEO has some considerable limitations. The current set up restricts the observation features to binary functions. The assumption of independence of observation model features and the approximation of observation probability mass may not be easily satisfied in some applications, and could have contributed to the learned observation model not being accurate. To meet the assumption that all ψ are observed in the first session of online learning, it may need to contain more demonstrations. Therefore, future research can target accommodating covariance of features within optimization problem and accounting for activation of some but not all features in each session. Also, future work should try accommodating deep learning in RIMEO so as to enable learning with noise and without observation model in high-dimensional continuous spaces.

robust-AIRL has a shortcoming in that it needs a prior knowledge of an observation model to create Gibbs samplers. A future research direction can be to borrow the inference of observation model from RIMEO and then use it for robust-AIRL. Another limitation is that the Technique 2 of computing Gibbs samplers approximates learned policy as a Gaussian distribution, which may get inaccurate for some environments where policy distributions $T_2 = P(\cdot|s)$ are multi modal. One future work idea to address that is to approximate $T_2 = P(\cdot|s)$ using mixture models and integrate that into Technique 2 for computing Gibbs samplers.

While working on online IRL under imperfect input, we contributed a new framework to formally analyze online methods and we contributed online methods that learn under imperfections like training data with missing information, data generated with multiple preferences, and data with perception noise from learner side. We also showed evidences that the proposed methods learn better than baselines on challenging application domains. We concluded this thesis by summarizing the potential future work ideas to overcome the limitations of proposed methods.

Publication List:

- S. Arora and P. Doshi, "A Survey of Inverse Reinforcement Learning: Challenges, Methods and Progress." Artificial Intelligence journal, Vol 297, August 2021
- S. Arora, P. Doshi, and B. Banerjee. "Online Inverse Reinforcement Learning Under Occlusion", Journal of Autonomous Agents and Multi-agent Systems (JAAMAS), Vol 35, 4, 2021
- S. Arora, P. Doshi and B. Banerjee, "Min-Max Entropy Inverse RL of Multiple Tasks," 2021 IEEE International Conference on Robotics and Automation (ICRA), Xi'an, China, 2021, pp. 12639-12645
- S. Arora, P. Doshi, and B. Banerjee. "Online Inverse Reinforcement Learning Under Occlusion", In Proceedings of the 18th International Conference on Autonomous Agents and Multi Agent Systems (AAMAS 2019), pp 1170-1178
- S. Arora, P. Doshi, and B. Banerjee. "Online Inverse Reinforcement Learning with Learned Observation Model." In Proceedings of The 6th Conference on Robot Learning 2023 (Proceedings of Machine Learning Research, Vol. 205), pp 1468-1477

Appendix A

Appendix A

A.1 Links to Open-Source Code and experiment videos

MDP definitions and frontend python code for our two physical experimental domains: `https://github.com/thinclab/sorting_patrol_MDP_irl`.

Backend solvers of LME-I2RL, MME-MTIRL, RIMEO in language D: `https://github.com/thinclab/irld`.

Robust-AIRL implementation: `https://github.com/thinclab/imitation/tree/robust_online_airl`

Codebase for baseline for Robust-AIRL: `https://github.com/CORE-Robotics-Lab/SSRR/`

The Gazebo simulation with Sawyer is available at: `https://github.com/thinclab/sawyer_irl_project`.

Video for LME-I2RL evaluation in simulation is available at `https://youtu.be/B3wA6z111ws`.

Video for LME-I2RL evaluation in physical robots is available here: `https://drive.google.com/file/d/1CRddwh0JMckUsJZSU4SXhrsthoEnM_cB/view?usp=sharing`.

Appendix B

Appendix B

B.1 Proofs for LME I2RL

Although a trajectory in a non-terminating MDP can be infinitely long, we derive range of f_k first for bounded-length trajectories and extend it later by applying infinity limit. Let T_{max} be the maximum length of any trajectory, $0 \leq |X| \leq T_{max}$.

Then,

$$\sum_{t=0}^{0} \gamma^t 0 \leq \sum_{\langle s,a\rangle_t \in X} \gamma^t \phi_k(\langle s,a\rangle_t) \leq \sum_{t=0}^{T_{max}} \gamma^t$$
$$0 \leq f_k(X) \leq (1-\gamma^{T_{max}})/(1-\gamma)$$

Applying limit $T_{max} \to \infty$ gives us

$$0 \leq f_k(X) \leq \frac{1}{1-\gamma} \tag{B.1}$$

Extending the definition to all k features, we introduce function $f : \mathbb{X} \to \mathbb{R}^k$ as $f(X) = \sum_{\langle s,a\rangle_t \in X} \gamma^t \phi(\langle s,a\rangle_t)$.

Note that learned feature expectations can be expressed in terms of f_k as

$$E_{\mathbb{X}}[\phi_k] \triangleq \sum\nolimits_{X \in \mathbb{X}} Pr(X)\ f_k(X),\ k = 1 \ldots K$$

The sessions for latent and full-observation MAXENTIRL updates estimated feature expectations of expert as follows.

$$\hat{\phi}^{Z|_Y,1:i}_{\boldsymbol{\theta}^i,k} \triangleq \frac{1}{|\mathcal{Y}_{1:i}|}\sum_{Y\in\mathcal{Y}_{1:i}}\sum_{Z\in\mathbb{Z}} Pr(Z|Y;\boldsymbol{\theta}) \sum_{\langle s,a\rangle_t\in Y\cup Z}\gamma^t\phi_k(\langle s,a\rangle_t)$$
$$= \frac{|\mathcal{Y}_{1:i-1}|}{|\mathcal{Y}_{1:i-1}|+|\mathcal{Y}_i|}\,\hat{\phi}^{Z|_Y,1:i-1}_{\boldsymbol{\theta}^{i-1},k} + \frac{|\mathcal{Y}_i|}{|\mathcal{Y}_{1:i-1}|+|\mathcal{Y}_i|}\,\hat{\phi}^{Z|_Y,i}_{\boldsymbol{\theta}^i,k} \tag{B.2}$$

$$\hat{\phi}^{1:i}_k \triangleq \frac{1}{|\mathcal{Y}_{1:i}|}\sum_{Y\in\mathcal{Y}_{1:i}}\sum_{Z\in\mathbb{Z}} Pr(Z|Y;\boldsymbol{\theta}) \sum_{\langle s,a\rangle_t\in Y\cup Z}\gamma^t\phi_k(\langle s,a\rangle_t)$$
$$= \frac{|\mathcal{Y}_{1:i-1}|}{|\mathcal{Y}_{1:i-1}|+|\mathcal{Y}_i|}\,\hat{\phi}^{1:i-1}_k + \frac{|\mathcal{Y}_i|}{|\mathcal{Y}_{1:i-1}|+|\mathcal{Y}_i|}\,\hat{\phi}^{i}_k \tag{B.3}$$

From definitions of feature-expectations and Equation B.1, $E_{\mathbb{X}}[\phi_k], \hat{\phi}^{1:i}_k, \hat{\phi}^{Z|_Y,1:i}_{\boldsymbol{\theta}^i,k} \in \left[0, \frac{1}{(1-\gamma)}\right]$.

Theorem 4 (Confidence for maximum entropy IRL)**.** *Given $\mathcal{X}_{1:i}$ as the (fully observed) demonstration till session i, $\boldsymbol{\theta}_E \in [0,1]^K$ is the expert's weights, and $\boldsymbol{\theta}^i$ is the solution of session i for incremental maximum entropy IRL, we have*

$$LL(\boldsymbol{\theta}_E|\mathcal{X}_{1:i}) - LL(\boldsymbol{\theta}^i|\mathcal{X}_{1:i}) \leq \varepsilon$$

with probability at least $1-\delta = 1 - 2K\exp\left[-\frac{|\mathcal{X}_{1:i}|\varepsilon^2(1-\gamma)^2}{2K^2}\right]$.

Theorem 3 (Confidence for ME I2RL). *Given $\mathcal{X}_{1:i}$ as the (fully observed) demonstration till session i, $\boldsymbol{\theta}_E \in [0,1]^K$ is the expert's weights, and $\boldsymbol{\theta}^i$ is the converged weight vector for session i for ME I2RL, we have,*

$$LL(\boldsymbol{\theta}_E|\mathcal{X}_{1:i}) - LL(\boldsymbol{\theta}^i|\mathcal{X}_i, |\mathcal{X}_{i-1}|, \hat{\phi}^{1:i-1}, \boldsymbol{\theta}^{i-1}) \leqslant \frac{2K\varepsilon}{(1-\gamma)}$$

with probability atleast $\max(0, 1-\delta)$*, where* $\delta = 2K\exp(-2|\mathcal{X}_{1:i}|\varepsilon^2)$.

Proof of Theorem 1. We use the notation:

$$E_{\mathbb{X}}[\phi_k] \triangleq \sum_{X\in\mathbb{X}} Pr(X) \sum_{\langle s,a\rangle\in X}\phi_k(s,a),\ k = 1\ldots K$$

By allowing a relaxation in the constraints of maximum entropy estimation problem, [32] derived sample complexity bounds for the problem.

$$\max_{\Delta} \left(-\sum_{X\in\mathbb{X}} Pr(X)\, log\, Pr(X)\right)$$

$$\textbf{subject to} \quad \sum_{X\in\mathbb{X}} Pr(X) = 1 \qquad \left|E_{\mathbb{X}}[\phi_k] - \hat{\phi}_k^{1:i}\right| \leq \beta_k^{full} \quad \forall k \in \{1\dots K\} \tag{B.4}$$

Here $\beta^{full} \in \mathbb{R}^K$ is a vector of upper bounds on the differences between $E_{\mathbb{X}}[\phi_k]$ and $\hat{\phi}_k^{1:i}$.

Following proofs by Dudik et al.[32], relaxed constraints maximum entropy IRL problem is same as $\min_{\boldsymbol{\theta}}(-\sum_{X\in\mathcal{X}_{1:i}} \tilde{Pr}(X) \quad \log Pr(X|\boldsymbol{\theta}) + \sum_k \beta_k^{full}|\theta_k|) = \min_{\boldsymbol{\theta}}(-LL(\boldsymbol{\theta}|\ \mathcal{X}_i, |\mathcal{X}_{i-1}|, \hat{\phi}^{1:i-1}, \boldsymbol{\theta}^{i-1}) + \sum_k \beta_k^{full}|\theta_k|) = \min_{\boldsymbol{\theta}} NLL_{\beta^{full}}(\boldsymbol{\theta}|\mathcal{X}_i, |\mathcal{X}_{i-1}|, \hat{\phi}^{1:i-1} \ , \boldsymbol{\theta}^{i-1})$ (say).

The proof here is partially inspired from Corollary 1 in [32]. Let $\beta_k^{full} = \beta_c^{full} = \varepsilon/(1-\gamma)$ for all $k \in \{1\dots K\}$, where β_c^{full} is constant because ε is fixed input. For normalized exponentiated gradient descent used here for computing maximum, $\sum_1^K |\theta_k| = 1$. Then, $NLL_{\beta^{full}}(\boldsymbol{\theta}|\mathcal{X}_i, |\mathcal{X}_{i-1}|, \hat{\phi}^{1:i-1}, \boldsymbol{\theta}^{i-1}) = (-LL(\boldsymbol{\theta}|\mathcal{X}_i,\ |\mathcal{X}_{i-1}|, \hat{\phi}^{1:i-1}, \boldsymbol{\theta}^{i-1}) + \beta_c^{full} \sum_1^k |\theta_k|) = (-LL(\boldsymbol{\theta}|\mathcal{X}_i, |\mathcal{X}_{i-1}|,\ \hat{\phi}^{1:i-1}, \boldsymbol{\theta}^{i-1}) + \beta_c^{full})$. Assume that $\boldsymbol{\theta}^i$ minimizes $NLL_{\beta^{full}}(\boldsymbol{\theta}| \quad \mathcal{X}_i, |\mathcal{X}_{i-1}|, \hat{\phi}^{1:i-1}, \boldsymbol{\theta}^{i-1})$, a solution maximizing $LL(\boldsymbol{\theta}|\mathcal{X}_i, |\mathcal{X}_{i-1}|, \quad \hat{\phi}^{1:i-1}, \boldsymbol{\theta}^{i-1})$.

Since $E_{\mathbb{X}}[\phi_k] \in \left[0, \frac{1}{(1-\gamma)}\right]$, we get $(1-\gamma)E_{\mathbb{X}}[\phi_k] \in [0,1]$. We multiply the relaxed constraint with $(1-\gamma)$ and define the negation of constraint as following event: $A : \left|(1-\gamma)E_{\mathbb{X}}[\phi_k] - (1-\gamma)\hat{\phi}_k^{1:i}\right| > (1-\gamma)\beta_c^{full} = \varepsilon$ for some $k \in \{1\dots K\}$. A can be decomposed into following feature specific events

$$A_k : (1-\gamma)\left|E_{\mathbb{X}}[\phi_k] - \hat{\phi}_k^{1:i}\right| > \varepsilon,$$

where $k \in \{1, 2\dots K\}$. We divide this constraint on absolute value further in two signed events:

$$(A_k)_1 : (1-\gamma)E_{\mathbb{X}}[\phi_k] - (1-\gamma)\hat{\phi}_k^{1:i} > \varepsilon$$

$$(A_k)_2 : -(1-\gamma)E_{\mathbb{X}}[\phi_k] + (1-\gamma)\hat{\phi}_k^{1:i} > \varepsilon$$

Then event A is same as logical disjunction $(A_1)_1 \vee (A_1)_2 \vee (A_2)_1 \dots$.

Applying Hoeffding's inequality, the upper bound of probability of each signed event is given by: $P((A_k)_1) \leq \exp(-2\varepsilon^2\ |\mathcal{X}_{1:i}|) = \frac{\delta}{2K}(say), P((A_k)_2) \leq \frac{\delta}{2K}$. Applying aforesaid bounds to events for each of the K features, we get $2K$ events with exactly same upper bound $\frac{\delta}{2K}$ on their respective probabilities. We use Fretchet's inequality to derive an upper bound for the disjunction:

$P(A) = P((A_1)_1 \vee (A_1)_2 \vee (A_2)_1 \dots) \leq \min(1, P((A_1)_1) + P((A_1)_2) + P((A_2)_1) \dots)$

As each of the probabilities in RHS are bounded from above by $\frac{\delta}{2K}$, their sum is bounded as:

$P(A) \leq \min(1, \sum_1^{2K} \frac{\delta}{2K}) = \min(1, \delta)$

Reverting to the negation of A, the probability that $\left|(1-\gamma)E_{\mathbb{X}}[\phi_k] - (1-\gamma)\hat{\phi}_k^{1:i}\right| \quad \leq \varepsilon \quad \forall k \in \{1\dots K\}$ is at least $1 - \min(1,\delta) = \max(0, 1-\delta)$.

To keep reward value bounded, IRL assumes $||\boldsymbol{\theta}^*||_1 \leq 1$ for all $\boldsymbol{\theta}^*$. Using the assumption and Theorem 1 in [32], we get error bound:

For every $\boldsymbol{\theta}^* \in [0,1]^K$, $NLL_{\beta full}(\boldsymbol{\theta}^i|\mathcal{X}_i, |\mathcal{X}_{i-1}|, \hat{\phi}^{1:i-1}, \boldsymbol{\theta}^{i-1}) \ -NLL_{\beta full}(\boldsymbol{\theta}^*|\mathcal{X}_i, |\mathcal{X}_{i-1}|, \hat{\phi}^{1:i-1}, \boldsymbol{\theta}^{i-1}) \leq 2\sum_1^K \beta_c^{full} = 2K \quad \beta_c^{full} = \frac{2K\varepsilon}{(1-\gamma)}$
with probability at least $\max(0, 1-\delta)$, where $\delta = \ 2K \exp\big(-2\,\varepsilon^2|\mathcal{X}_{1:i}|\big)$.

We modify the bound in the form of positive log-likelihood of expert's policy, by using relation $NLL_{\beta full}(\boldsymbol{\theta}^*|\mathcal{X}_{1:i}) = (-LL(\boldsymbol{\theta}^*| \quad \mathcal{X}_{1:i}) + \sum_1^K \beta_k^{full}|\theta_k|)$ and $\boldsymbol{\theta}^* = \boldsymbol{\theta}_E$.

Then, with $\mathcal{X}_{1:i}$ as input, with probability at least $\max(0, 1-\delta)$,

$$\begin{aligned} &NLL_{\beta full}(\boldsymbol{\theta}^i|\mathcal{X}_i, |\mathcal{X}_{i-1}|, \hat{\phi}^{1:i-1}, \boldsymbol{\theta}^{i-1}) - NLL_{\beta full}(\boldsymbol{\theta}_E|\mathcal{X}_{1:i}) \\ &= LL(\boldsymbol{\theta}_E|\mathcal{X}_{1:i}) - LL(\boldsymbol{\theta}^i|\mathcal{X}_i, |\mathcal{X}_{i-1}|, \hat{\phi}^{1:i-1}, \boldsymbol{\theta}^{i-1}) \leq \frac{2K\varepsilon}{(1-\gamma)} \end{aligned}$$

Insight: Likelihood-loss $LL(\boldsymbol{\theta}_E|\mathcal{X}_{1:i}) - LL(\boldsymbol{\theta}^i|\mathcal{X}_i, |\mathcal{X}_{i-1}|, \quad \hat{\phi}^{1:i-1}, \boldsymbol{\theta}^{i-1})$ for MAXENTIRL gets smaller and the learned weights $\boldsymbol{\theta}^i$ is getting closer to the best weights possible (expert's weights θ_E) with more training trajectories or higher $|\mathcal{X}_{1:i}|$. The confidence of convergence $1-\delta$ increases with more training (higher $|\mathcal{X}_{1:i}|$), more room for error (higher ε) and less features (lower K).

□

Lemma 5 (Monotonicity)**.** LME I2RL increases the demonstration likelihood monotonically with each new session, $LL(\boldsymbol{\theta}^i|\mathcal{Y}_i, |\mathcal{Y}_{1:i-1}|, \hat{\phi}^{Z|_Y,1:i-1}_{\boldsymbol{\theta}^{i-1}}, \boldsymbol{\theta}^{i-1}) - LL(\boldsymbol{\theta}^{i-1}|\mathcal{Y}_{i-1}, |\mathcal{Y}_{1:i-2}|, \hat{\phi}^{Z|_Y,1:i-2}_{\boldsymbol{\theta}^{i-2}}, \boldsymbol{\theta}^{i-2}) \geqslant 0$, when $|\mathcal{Y}_{1:i-1}| \gg |\mathcal{Y}_i|$.

Proof of Lemma 1. Log-likelihood of demonstrated behavior can be split as

$$\begin{aligned} &LL(\boldsymbol{\theta}^i|\mathcal{Y}_{1:i}) = LL(\boldsymbol{\theta}^i|\mathcal{Y}_i, |\mathcal{Y}_{i-1}|, \hat{\phi}^{Z|_Y,1:i-1}_{\boldsymbol{\theta}^{i-1}}, \boldsymbol{\theta}^{i-1}) \\ &= \sum_{Y\in\mathcal{Y}_{1:i}} \tilde{Pr}(Y)\ log\ Pr(Y;\boldsymbol{\theta}) \\ &= \sum_{Y\in\mathcal{Y}_{1:i}} \tilde{Pr}(Y) \sum_{Z\in\mathbb{Z}} Pr(Z|Y;\boldsymbol{\theta}^i)\ log\ Pr(Y,Z;\boldsymbol{\theta}) + \\ &(-\sum_{Y\in\mathcal{Y}_{1:i}} \tilde{Pr}(Y) \sum_{Z\in\mathbb{Z}} Pr(Z|Y;\boldsymbol{\theta}^i)\ log\ Pr(Z|Y;\boldsymbol{\theta})) \\ &= Q(\mathcal{Y}_i, |\mathcal{Y}_{i-1}|, \hat{\phi}^{Z|_Y,1:i-1}_{\boldsymbol{\theta}^{i-1}}, \boldsymbol{\theta}, \boldsymbol{\theta}^{i-1}) + C(\mathcal{Y}_{1:i}, \boldsymbol{\theta}, \boldsymbol{\theta}^i) \end{aligned}$$

Here $\tilde{Pr}$ is distribution of trajectories in observed training data ($\sum_{X\in\mathcal{X}} \tilde{Pr}(X)[\cdot]$ and $\frac{1}{|\mathcal{X}|}\sum_{X\in\mathcal{X}}[\cdot]$ can be used interchangeably). EM method maximizes the log-likelihood by maximizing only Q value over $\boldsymbol{\theta}$; and $\boldsymbol{\theta} = \boldsymbol{\theta}^i$ maximizes

$Q(\mathcal{Y}_i, |\mathcal{Y}_{i-1}|, \hat{\phi}^{Z|_Y,1:i-1}_{\boldsymbol{\theta}^{i-1}}, \boldsymbol{\theta}, \boldsymbol{\theta}^{i-1})$ ([83]). After all the EM iterations for current session i, the final Q value is $Q(\mathcal{Y}_i, |\mathcal{Y}_{i-1}|, \hat{\phi}^{Z|_Y,1:i-1}_{\boldsymbol{\theta}^{i-1}}, \boldsymbol{\theta}^i, \boldsymbol{\theta}^i)$. Therefore, the difference in the likelihoods achieved by weights learned in consecutive sessions can be expressed as a difference in Q values. Note that LME IRL learns

reward weights by inferring the maximum entropy distribution $Pr(X;\boldsymbol{\theta}) = \frac{\exp(\sum_k \theta_k f_k(X))}{\Omega^X_{\boldsymbol{\theta}}}$, where $\Omega^X_{\boldsymbol{\theta}} = \sum_{X\in\mathbb{X}} \exp(\sum_k \theta_k f_k(X))$ and $X = (Y,Z)$. Expand Q value as $Q(\mathcal{Y}_i, |\mathcal{Y}_{i-1}|, \hat{\phi}^{Z|_Y,1:i-1}_{\boldsymbol{\theta}^{i-1}}, \boldsymbol{\theta}^i, \boldsymbol{\theta}^i) = \sum_{Y\in\mathcal{Y}_{1:i}} \tilde{Pr}(Y) \sum_{Z\in\mathbb{Z}} Pr(Z|Y;\boldsymbol{\theta}^i) \log\left(\frac{\exp(\sum_k \theta^i_k f_k((Y,Z)))}{\Omega^{(Y,Z)}_{\boldsymbol{\theta}^i}}\right) = \sum_k \theta^i_k \cdot \sum_{Y\in\mathcal{Y}_{1:i}} \tilde{Pr}(Y) \sum_{Z\in\mathbb{Z}} Pr(Z|Y;\boldsymbol{\theta}^i) f_k((Y,Z)) - \log\,\Omega^{(Y,Z)}_{\boldsymbol{\theta}^i} = \sum_k \theta^i_k \cdot \hat{\phi}^{Z|_Y,1:i}_{\boldsymbol{\theta}^i,k} - \log\,\Omega^{(Y,Z)}_{\boldsymbol{\theta}^i}$.

Therefore the improvement in log likelihood over session i is

$$
\begin{aligned}
&LL(\boldsymbol{\theta}^i|\mathcal{Y}_i, |\mathcal{Y}_{i-1}|, \hat{\phi}^{Z|_Y,1:i-1}_{\boldsymbol{\theta}^{i-1}}, \boldsymbol{\theta}^{i-1}) - LL(\boldsymbol{\theta}^{i-1}|\mathcal{Y}_{i-1}, |\\
&\mathcal{Y}_{i-2}|, \hat{\phi}^{Z|_Y,1:i-2}_{\boldsymbol{\theta}^{i-2}}, \boldsymbol{\theta}^{i-2})\\
&= Q(\mathcal{Y}_i, |\mathcal{Y}_{i-1}|, \hat{\phi}^{Z|_Y,1:i-1}_{\boldsymbol{\theta}^{i-1}}, \boldsymbol{\theta}^i, \boldsymbol{\theta}^i) - Q(\mathcal{Y}_{i-1}, |\\
&\mathcal{Y}_{i-2}|, \hat{\phi}^{Z|_Y,1:i-2}_{\boldsymbol{\theta}^{i-2}}, \boldsymbol{\theta}^{i-1}, \boldsymbol{\theta}^{i-1})\\
&= \sum_k \theta^i_k \hat{\phi}^{Z|_Y,1:i}_{\boldsymbol{\theta}^i,k} - \log\,\Omega^{(Y,Z)}_{\boldsymbol{\theta}^i} - \sum_k \theta^{i-1}_k \hat{\phi}^{Z|_Y,1:i-1}_{\boldsymbol{\theta}^{i-1},k} +\\
&\log\,\Omega^{(Y,Z)}_{\boldsymbol{\theta}^{i-1}}\\
&= \log\frac{\Omega^{(Y,Z)}_{\boldsymbol{\theta}^{i-1}}}{\Omega^{(Y,Z)}_{\boldsymbol{\theta}^i}} + \sum_k \left(\theta^i_k \frac{|\mathcal{Y}_{1:i-1}|}{|\mathcal{Y}_i| + |\mathcal{Y}_{1:i-1}|} - \theta^{i-1}_k\right) \hat{\phi}^{Z|_Y,1:i-1}_{\boldsymbol{\theta}^{i-1},k}\\
&+ \sum_k \left(\theta^i_k \frac{1}{|\mathcal{Y}_i| + |\mathcal{Y}_{1:i-1}|} \hat{\phi}^{Z|_Y,i}_{\boldsymbol{\theta}^i,k}\right)\\
&\text{(substitute } \hat{\phi}^{Z|_Y,1:i}_{\boldsymbol{\theta}^i,k} \text{ using Eq. 4.2 and simplifying)}
\end{aligned}
$$

The final expression is minimized only for $\boldsymbol{\theta}^i = \boldsymbol{\theta}^{i-1}$ when $|\mathcal{Y}_{1:i-1}| \gg |\mathcal{Y}_i|$, i.e., when a significant amount of training data has been accumulated. The expression is also concave in parameter $\boldsymbol{\theta}^i$. Therefore, $LL(\boldsymbol{\theta}^i|\mathcal{Y}_i, |\mathcal{Y}_{i-1}|, \hat{\phi}^{Z|_Y,1:i-1}_{\boldsymbol{\theta}^{i-1}}, \boldsymbol{\theta}^{i-1}) - LL(\boldsymbol{\theta}^{i-1}|\mathcal{Y}_{i-1}, |\mathcal{Y}_{i-2}|, \hat{\phi}^{Z|_Y,1:i-2}_{\boldsymbol{\theta}^{i-2}}, \boldsymbol{\theta}^{i-2}) \geq 0$ for consecutive sessions thereafter. Hence, the LME I2RL is proved to converge over sequence of sessions, yielding a feasible log-linear solution to latent-MAXENT and corresponding weights solving IRL.

□

Lemma 6 (Constraint Bounds for LME I2RL)**.** Suppose $\mathcal{X}_{1:i}$ has portions of trajectories in $\mathbb{Z}_{1:i} = \{Z|(Y,Z) \in \mathcal{X}_{1:i}\}$ occluded from the learner. Let ε_s be a bound on the error $\left|\hat{\phi}^{1:i}_k - \hat{\phi}^{Z|_Y,1:i}_{\boldsymbol{\theta}^i,k}\right|_1, k \in \{1,2\ldots K\}$ after n_s samples for approximation. Then, with probability at least $\max(0, 1-(\delta+\delta_s))$, the following holds:

$$\left|(1-\gamma)\big(E_{\mathbb{X}}[\phi_k] - \hat{\phi}^{Z|_Y,1:i}_{\boldsymbol{\theta}^i,k}\big)\right|_1 \leqslant \varepsilon + \varepsilon_s, k \in \{1,2\ldots K\}$$

where ε, δ are as defined in Theorem 3, and $\delta_s = 2K\exp(-2n_s\varepsilon_s^2)$.

Proof of Lemma 2. We define the event

$A_k : (1-\gamma)\left|E_{\mathbb{X}}[\phi_k] - \hat{\phi}^{1:i}_k\right| > \varepsilon, k \in \{1,2\ldots K\}$.

Applying Hoeffding's inequality for A_k, we get

$P(A_k) \leq 2\exp(-2\varepsilon^2|\mathcal{X}_{1:i}|) \leq \frac{\delta}{K}$ for any $k \in \{1, 2 \dots K\}$, and for the same ε, δ as in Theorem 1. Similarly, for partial observation, given ε_s as the bound on the error in sampling based approximation of $\hat{\phi}_l^{1:i}$ as $\hat{\phi}_{\boldsymbol{\theta}^i,l}^{Z|_Y,1:i}$, and n_s samples, let us define the event

$B_l : (1-\gamma)\left|\hat{\phi}_l^{1:i} - \hat{\phi}_{\boldsymbol{\theta}^i,l}^{Z|_Y,1:i}\right| > \varepsilon_s, l \in \{1, 2 \dots K\}$.

Similar to procedure for $P(A_k)$, applying Hoeffding bound gives us $P(B_l) < \frac{\delta_s}{K}$, $\delta_s = 2K\exp(-2(\varepsilon_s)^2 n_s)$.

Applying Fretchets inequality over both sets A and B of events gives us:

$P\left((\cup_k A_k) \vee (\cup_l B_l)\right) < \min(1, \sum_{k=1}^{K}\frac{\delta}{K} + \sum_{l=1}^{K}\frac{\delta_s}{K}) = \min(1, \delta + \delta_s)$.

That is, $P\left(\exists k, l, s.t. A_k \vee B_l\right) < \min(1, \delta + \delta_s)$. Taking complement, $P\left(\forall k, l, \overline{A}_k \wedge \overline{B}_l\right) \geq \max(0, 1 - \delta - \delta_s)$. But $\forall k, l, \overline{A}_k \wedge \overline{B}_l$ implies that $\forall k$:

$(1-\gamma)(\left|E_{\mathbb{X}}[\phi_k] - \hat{\phi}_k^{1:i}\right| + \left|\hat{\phi}_k^{1:i} - \hat{\phi}_{\boldsymbol{\theta}^i,k}^{Z|_Y,1:i}\right|) \leq \varepsilon + \varepsilon_s$

Calling $(\varepsilon + \varepsilon_s) = 2\varepsilon_l$, and $(\delta + \delta_s) = \delta_l$ we get

$P\big(\forall k, (1-\gamma)(\left|E_{\mathbb{X}}[\phi_k] - \hat{\phi}_k^{1:i}\right| + \left|\hat{\phi}_k^{1:i} - \hat{\phi}_{\boldsymbol{\theta}^i,k}^{Z|_Y,1:i}\right|) \leq$

$2\varepsilon_l\big) \geq \max(0, 1 - \delta_l)$.

Using inequality $\left|E_{\mathbb{X}}[\phi_k] - \hat{\phi}_{\boldsymbol{\theta}^i,k}^{Z|_Y,1:i}\right| \leq \left|E_{\mathbb{X}}[\phi_k] - \hat{\phi}_k^{1:i}\right| + \left|\hat{\phi}_k^{1:i} - \hat{\phi}_{\boldsymbol{\theta}^i,k}^{Z|_Y,1:i}\right|$, we get:

$$P\left(\forall k, (1-\gamma)(\left|E_{\mathbb{X}}[\phi_k] - \hat{\phi}_{\boldsymbol{\theta}^i,k}^{Z|_Y,1:i}\right|) \leq 2\varepsilon_l\right) \geq \max(0, 1 - \delta_l).$$

□

THEOREM 4 (CONFIDENCE FOR LME I2RL). *Let $\mathcal{Y}_{1:i} = \{Y|(Y,Z) \in \mathcal{X}_{1:i}\}$ be the observed portions of the demonstration until session i. ε and ε_s are inputs as defined in Lemma 6, and $\boldsymbol{\theta}^i$ is the solution of session i for LME I2RL. Then*

$$LL(\boldsymbol{\theta}_E|\mathcal{Y}_{1:i}) - LL(\boldsymbol{\theta}^i|\mathcal{Y}_i, |\mathcal{Y}_{i-1}|, \hat{\phi}_{\boldsymbol{\theta}^{i-1}}^{Z|_Y,1:i-1}, \boldsymbol{\theta}^{i-1}) \leq \frac{4K\varepsilon_l}{(1-\gamma)}$$

with confidence at least $\max(0, 1 - \delta_l)$*, where* $\varepsilon_l = \frac{\varepsilon+\varepsilon_s}{2}$*, and* $\delta_l = \delta + \delta_s$*.*

Proof of Theorem 2. Latent maximum entropy IRL problem is equivalent to $\max_{\boldsymbol{\theta}} \sum_{Y \in \mathcal{Y}_{1:i}} \tilde{Pr}(Y) \log Pr(Y|\boldsymbol{\theta})$ (Section 3.3, [19]) or $\max_{\boldsymbol{\theta}} LL(\boldsymbol{\theta}^i|\mathcal{Y}_i, |\mathcal{Y}_{i-1}|, \hat{\phi}_{\boldsymbol{\theta}^{i-1}}^{Z|_Y,1:i-1}, \boldsymbol{\theta}^{i-1})$.

Relaxed constraint latent maximum entropy IRL is:

$$\begin{aligned} &\max_{\Delta}\left(-\textstyle\sum_{X \in \mathbb{X}} Pr(X)\, log\, Pr(X)\right) \\ &\textbf{subject to} \quad \textstyle\sum_{X \in \mathbb{X}} Pr(X) = 1 \\ &\left|E_{\mathbb{X}}[\phi_k] - \hat{\phi}_{\boldsymbol{\theta}^i,k}^{Z|_Y,1:i}\right| \leq \beta_k \quad \forall k \in \{1 \dots K\} \end{aligned} \tag{B.5}$$

Here $\beta \in \mathbb{R}^K$ is an estimate of vector of upper bounds on the differences between $E_{\mathbb{X}}[\phi_k]$ and $\hat{\phi}_{\boldsymbol{\theta}^i,k}^{Z|_Y,1:i}$.

The form of relaxed latent maximum entropy problem and the likelihood for that problem is no different than those for relaxed maximum entropy. Starting from results in Lemma 2, assuming $\beta_k = \beta_c = 2\varepsilon_l/(1-\gamma)$ for all $k \in \{1 \dots K\}$ and using steps similar to the proof of Theorem 1, we get $LL(\boldsymbol{\theta}_E|\mathcal{Y}_{1:i}) - LL(\boldsymbol{\theta}^i|\mathcal{Y}_i, |\mathcal{Y}_{i-1}|, \hat{\phi}^{Z|_{Y},1:i-1}_{\boldsymbol{\theta}^{i-1}}, \boldsymbol{\theta}^{i-1}) \leq \frac{4K\varepsilon_l}{(1-\gamma)}$ with probbility at least $\max(0, 1-\delta_l)$. □

THEOREM 5 (NO REGRET LEARNING). *There exist choices for a variable threshold bound on the feature matching constraint, ε_i as a function of session i, such that with probability at least* $\max(0, \prod_{i=1}^{i=T}(1-\delta_i))$, *where* $\delta_i = 2K\exp(-2|\mathcal{X}_{1:i}|\varepsilon_i^2)$,

$$\text{Regret}_T(A_{MEI_2RL}) = o(T)/T,$$

thus approaching 0 as $T \to \infty$.

Proof of Sub-Linear Regret. The log-loss after ith session is $LL(\boldsymbol{\theta}_E|\mathcal{X}_{1:i}) - LL(\boldsymbol{\theta}^i|\mathcal{X}_i, |\mathcal{X}_{i-1}|, \hat{\phi}^{1:i-1}, \boldsymbol{\theta}^{i-1})$. Let the regret after $i = T$ be $\frac{1}{T}\sum_{i=1}^{T} LL(\boldsymbol{\theta}_E|\mathcal{X}_{1:i}) - \frac{1}{T}\sum_{i=1}^{T} LL(\boldsymbol{\theta}^i|\mathcal{X}_i, |\mathcal{X}_{i-1}|, \hat{\phi}^{1:i-1}, \boldsymbol{\theta}^{i-1})$. According to Theorem 3, $LL(\boldsymbol{\theta}_E|\mathcal{X}_{1:i}) - LL(\boldsymbol{\theta}^i|\mathcal{X}_i, |\mathcal{X}_{i-1}|, \hat{\phi}^{1:i-1}, \boldsymbol{\theta}^{i-1}) \leq \beta \cdot \varepsilon_l$ with probbility at least $\max(0, 1-\delta)$, where $\beta = \frac{2K}{(1-\gamma)}$. As ε is user specified, let $\varepsilon = \frac{c}{i}$. Then, the inequality becomes $LL(\boldsymbol{\theta}_E|\mathcal{X}_{1:i}) - LL(\boldsymbol{\theta}^i|\mathcal{X}_i, |\mathcal{X}_{i-1}|, \hat{\phi}^{1:i-1}, \boldsymbol{\theta}^{i-1}) \leq \beta\frac{c}{i}$. Summing the result over $i \in \{1, 2, \dots T\}$ and dividing by T, we get

$$\frac{1}{T}\sum_{i=1}^{T} LL(\boldsymbol{\theta}_E|\mathcal{X}_{1:i}) - \frac{1}{T}\sum_{i=1}^{T} LL(\boldsymbol{\theta}^i|\mathcal{X}_i, |\mathcal{X}_{i-1}|, \hat{\phi}^{1:i-1}, \boldsymbol{\theta}^{i-1}) \leq \beta\frac{1}{T}\sum_{i=1}^{T}\frac{c}{i}$$

The RHS above is bounded as $\beta\frac{1}{T}\sum_{i=1}^{T}\frac{c}{i} \leq \beta\frac{1}{T}c\log T = \beta c\frac{\log T}{T}$. As $T \to \infty$, $\beta c\frac{\log T}{T} \to 0$. Therefore, as sessions progress, regret is guaranteed to vanish. □

Appendix C

Appendix C

C.1 LME I2RL application domains

C.1.1 Features for three instances of perimeter patrol

All reward functions are sufficiently modeled as a linear weighted combination of pre-determined feature functions where the weights are unknown. The reward function of the single patroller in first instance of perimeter patrol utilizes *four* feature functions. Each function activates when the patroller reaches the end of the previously numbered hallway and remains activated until the end of the target hallway is reached, thereby enabling clockwise patrolling. To continue moving out of the hallways and in the vertical corridor (not a part of any hallway), the state in this domain instance additionally includes the recently visited hallway. A hallway is deemed to have been visited when the navigating agent has reached its end. The four binary feature functions are:

- *SwitchToHallway1(s,a)* returns 1 when action a in state s makes the patroller move from hallway 4 to hallway 1 in s', otherwise 0;
- *SwitchToHallway2(s,a)* returns 1 when action a in state s makes the patroller move from hallway 1 to 2;
- *SwitchToHallway3(s,a)* returns 1 when action a in state s makes the patroller move from hallway 2 to 3; and
- *SwitchToHallway4(s,a)* returns 1 when action a in state s makes the patroller move from hallway 3 to 4.

If equal weights are given to each of the above features, the MDP policy then guides the patroller to cycle through the four hallways in a clockwise manner.

The reward function for each patroller in the second instance of the domain utilizes *six* binary state-action feature functions, which divide the grid broadly into five regions as shown.

- *HasMoved(s,a)* returns 1 if action a at state s makes the patroller change its grid cell, 0 otherwise;
- *Turn1(s,a)* returns 1 if action a in state s makes the patroller turn (left or right) in the region of the hallway shaded orange in Fig 7.1(b);
- *Turn2(s,a)* returns 1 if action a in state s makes the patroller turn in the region of the hallway shaded yellow in Fig 7.1(b);
- *Turn3(s,a)* returns 1 if action a in state s makes the patroller turn in the region of the hallway shaded green in Fig 7.1(b);
- *Turn4(s,a)* returns 1 if action a in state s makes the patroller turn in the region of the hallway shaded blue in Fig 7.1(b);
- *Turn5(s,a)* returns 1 if action a in state s makes the patroller turn in the region of the hallway shaded magenta in Fig 7.1(b).

A weight vector $\boldsymbol{\theta}_E$ for these features such as $\langle .57, 0, 0, 0, .43, 0\rangle$ makes each of the two patrollers constantly execute cyclic trajectory that involves turning around in the region shaded blue of the top and bottom hallway.

For the third instance of the domain , the reward function is composed of the following *five* feature functions.

- *HasMoved(s,a)* returns 1 if action a at state s makes the patroller change its grid cell, 0 otherwise;
- *InRoom(s,a)* returns 1 if the patroller in state s is inside a room, 0 otherwise;
- *Turn(s,a)* returns 1 if action a in state s makes the patroller turn in a hallway;
- *EnterRoom(s,a)* returns 1 if action a at state s makes the patroller enter a room from one of the hallways;
- *LeaveRoom(s,a)* returns 1 if action a at state s makes the patroller enter a hallway from one of the rooms.

A weight vector of $\langle 1, -1, .1, 0, 0\rangle$ gives the highest preference to constantly moving, some preference to turning around in the hallways, and the least preference to being in a room. No reward is given for entering or leaving the rooms. As a result, the patrollers keep patrolling the hallways and turn around just before the hallways end.

C.1.2 Snapshot for robotics experiments of third instance

For the physical set up of third instance of perimeter patrol domain, shows the location of the attacker, the patrollers, and the hallways navigated by the patrollers.

Figure C.1: **(top)** The learner (green) observes the two patroller (pink and yellow). **(bottom-right)** Patrollers navigating the hallway at the top of the map. **(bottom-left)** The learner penetrates the patrol moving through the hallway on the right side of the map (reaching the goal marked G2 in the grid). colors in the carpet.

APPENDIX D

APPENDIX D

D.1 Derivation of gradients for MME-MTIRL

Our non linear optimization program after unification of two objectives is:

$$\max_{Pr_d(X_i)\in\Delta^D, \boldsymbol{v}_{d,i}\in\{0,1\}} -\sum_{d=1}^{D}\sum_{i=1}^{|\mathbb{X}|} v_{d,i}Pr_d(X_i)\ \log v_{d,i}Pr_d(X_i) + \frac{1}{|\mathcal{X}|}\sum_{d=1}^{D}\sum_{i=1}^{|\mathcal{X}|} v_{d,i}\ \log \frac{1}{|\mathcal{X}|}\sum_{i=1}^{|\mathcal{X}|} v_{d,i}$$

subject to

$$\sum_{d=1}^{D}\frac{1}{|\mathcal{X}|}\sum_{i=1}^{|\mathcal{X}|} v_{d,i}\sum_{i=1}^{|\mathbb{X}|} v_{d,i}Pr_d(X_i) = 1$$
$$E_{\mathbb{X}}[\phi_{d,k}] = \hat{\phi}_{d,k} \quad \forall d \in D, \forall k \in K$$
$$\sum_{d=1}^{D} v_{d,i} = 1, \forall X_i \in \mathcal{X} \tag{D.1}$$

where $E_{\mathbb{X}}[\phi_{d,k}] = \frac{1}{|\mathcal{X}|}\sum_{i=1}^{|\mathcal{X}|} v_{d,i}\sum_{i=1}^{|\mathbb{X}|} v_{d,i}Pr_d(X_i)\ \sum_{(s,a)\in X_i}\phi_{d,k}(s,a)$

and $\hat{\phi}_{d,k} = \frac{1}{|\mathcal{X}|}\sum_{i=1}^{|\mathcal{X}|} v_{d,i}\sum_{(s,a)\in X_i}\phi_k(s,a)$.

Lagrangian relaxation of the optimization problem is

$$\mathcal{L} \tag{D.2}$$
$$= \left(- \sum_{d=1}^{D} \sum_{i=1}^{|\mathbb{X}|} v_{d,i} Pr_d(X_i) \log v_{d,i} Pr_d(X_i) \right)$$
$$+ \left(\frac{1}{|\mathcal{X}|} \sum_{d=1}^{D} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \log \frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \right)$$
$$+ \eta \left(\sum_{d=1}^{D} \left(\frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \right) \sum_{i=1}^{|\mathbb{X}|} v_{d,i} Pr_d(X_i) - 1 \right)$$
$$+ \sum_{d \in \mathcal{D}} \sum_{k \in \mathcal{K}} \boldsymbol{\theta}_{d,k} \Bigg(\left(\frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \right) \sum_{i=1}^{|\mathbb{X}|} v_{d,i} Pr_d(X_i) \sum_{(s,a) \in X_i} \phi_k(s,a)$$
$$- \frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \sum_{(s,a) \in X_i} \phi_k(s,a) \Bigg)$$
$$+ \sum_{i=1}^{|\mathcal{X}|} \lambda_i \left(\sum_{d \in \mathcal{D}} v_{d,i} - 1 \right) \tag{D.3}$$

Maximization happens w.r.t. $Pr_d(X_i)$. Therefore, we take derivative w.r.t that.

$$\frac{\partial \mathcal{L}}{\partial Pr_d(X_i)} = 0$$
$$- \sum_{d=1}^{D} v_{d,i} \log(v_{d,i} Pr_d(X_i)) - \sum_{d=1}^{D} \left(v_{d,i} Pr_d(X_i) \frac{1}{(v_{d,i} Pr_d(X_i))} \right) v_{d,i}$$
$$+ \eta \sum_{d=1}^{D} v_{d,i} \left(\frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \right) + \sum_{d=1}^{D} v_{d,i} \left(\frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \right) \sum_{k \in \mathcal{K}} \boldsymbol{\theta}_{d,k} \sum_{(s,a) \in X_i} \phi_k(s,a) = 0$$
$$- \sum_{d=1}^{D} v_{d,i} \log(v_{d,i} Pr_d(X_i)) - \sum_{d=1}^{D} v_{d,i} + \eta \sum_{d=1}^{D} v_{d,i} \left(\frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \right)$$
$$+ \sum_{d=1}^{D} v_{d,i} \left(\frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \right) \sum_{k \in \mathcal{K}} \boldsymbol{\theta}_{d,k} \sum_{(s,a) \in X_i} \phi_k(s,a) = 0 \tag{D.4}$$

Take $\sum_{d=1}^{D} v_{d,i}$ common from all terms to cancel it out.

$$\sum_{d=1}^{D} v_{d,i}\Bigg(- \log(v_{d,i} Pr_d(X_i)) - \left(v_{d,i} Pr_d(X_i) \, \frac{1}{(v_{d,i} Pr_d(X_i))} \right)$$
$$+\eta \left(\frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \right) + \sum_{k \in \mathcal{K}} \boldsymbol{\theta}_{d,k} \left(\frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \right) \sum_{(s,a) \in X_i} \phi_k(s,a) \Bigg) = 0$$
$$\log \left(v_{d,i} Pr_d(X_i) \right) = -1 + \eta \left(\frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \right) + \sum_{k \in \mathcal{K}} \boldsymbol{\theta}_{d,k} \left(\frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \right) \sum_{(s,a) \in X_i} \phi_k(s,a) \tag{D.5}$$

Let partition function $\Omega = \sum_{d \in \mathcal{D}} (\frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i}) \sum_{i=1}^{|\mathbb{X}|} \exp \left(\sum_{k \in \mathcal{K}} \boldsymbol{\theta}_{d,k} \left(\frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \right) \sum_{(s,a) \in X_i} \phi_k(s,a) \right)$ be related to $\eta' = \eta \left(\frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \right)$ via following

$$\begin{aligned} 1 - \eta' &= \log \Omega \\ \eta' &= 1 - \log \Omega \\ \eta &= \frac{|\mathcal{X}|}{\sum_{i=1}^{|\mathcal{X}|} v_{d,i}} \Big(1 - \log \Omega \Big) \end{aligned} \tag{D.6}$$

Taking exponential on both sides on Eq D.5 and bringing $\eta' = \eta \left(\frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \right)$ to denominator in RHS.

$$v_{d,i} Pr_d(X_i) = \frac{\exp \left(\sum_{k \in \mathcal{K}} \boldsymbol{\theta}_{d,k} \left(\frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \right) \sum_{(s,a) \in X_i} \phi_k(s,a) \right)}{\exp(1 - \eta')}$$
$$= \frac{\exp \left(\sum_{k \in \mathcal{K}} \boldsymbol{\theta}_{d,k} \left(\frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \right) \sum_{(s,a) \in X_i} \phi_k(s,a) \right)}{\sum_{d \in \mathcal{D}} (\frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i}) \sum_{i=1}^{|\mathbb{X}|} \exp \left(\sum_{k \in \mathcal{K}} \boldsymbol{\theta}_{d,k} \left(\frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \right) \sum_{(s,a) \in X_i} \phi_k(s,a) \right)} \tag{D.7}$$
$$= \frac{\exp \left(\sum_{k \in \mathcal{K}} \boldsymbol{\theta}_{d,k} \left(\frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \right) \sum_{(s,a) \in X_i} \phi_k(s,a) \right)}{\Omega} \tag{D.8}$$
$$\tag{D.9}$$

We will need following two partial derivatives of $\log \Omega$ for coming steps

$$\frac{\partial}{\partial \boldsymbol{\theta}_{d,k}} \log \Omega = \frac{1}{\left(\sum_{d \in \mathcal{D}} (\frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i}) \sum_{i=1}^{|\mathbb{X}|} \exp\left(\sum_{k \in \mathcal{K}} \boldsymbol{\theta}_{d,k} \left(\frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \right) \sum_{(s,a) \in X_i} \phi_k(s,a) \right) \right)}$$

$$(\frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i}) \sum_{i=1}^{|\mathbb{X}|} \exp\left(\sum_{k \in \mathcal{K}} \boldsymbol{\theta}_{d,k} \left(\frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \right) \sum_{(s,a) \in X_i} \phi_k(s,a) \right)$$

$$\left(\frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \right) \sum_{(s,a) \in X_i} \phi_k(s,a)$$

$$= (\frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i})^2 \sum_{i=1}^{|\mathbb{X}|} \frac{\exp\left(\sum_{k \in \mathcal{K}} \boldsymbol{\theta}_{d,k} \left(\frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \right) \sum_{(s,a) \in X_i} \phi_k(s,a) \right)}{\sum_{d \in \mathcal{D}} (\frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i}) \sum_{i=1}^{|\mathbb{X}|} \exp\left(\sum_{k \in \mathcal{K}} \boldsymbol{\theta}_{d,k} \left(\frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \right) \sum_{(s,a) \in X_i} \phi_k(s,a) \right)}$$

$$\sum_{(s,a) \in X_i} \phi_k(s,a)$$

$$= (\frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i})^2 \sum_{i=1}^{|\mathbb{X}|} Pr(X_i | c_i = d) \sum_{(s,a) \in X_i} \phi_k(s,a)$$

$$= \left(\frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \right) E_{\mathbb{X},d}[\phi_k] \tag{D.10}$$

If $X_i \in \mathcal{X}$,

$$\frac{\partial}{\partial v_{d,i}} \log \Omega = \frac{1}{\left(\sum_{d \in \mathcal{D}} (\frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i}) \sum_{i=1}^{|\mathbb{X}|} \exp\left(\sum_{k \in \mathcal{K}} \boldsymbol{\theta}_{d,k} \left(\frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \right) \sum_{(s,a) \in X_i} \phi_k(s,a) \right) \right)}$$

$$\left((\frac{1}{|\mathcal{X}|}) \sum_{i=1}^{|\mathbb{X}|} \exp\left(\sum_{k \in \mathcal{K}} \boldsymbol{\theta}_{d,k} \left(\frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \right) \sum_{(s,a) \in X_i} \phi_k(s,a) \right) + \left(\frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \right) \right.$$

$$\left. \sum_{i=1}^{|\mathbb{X}|} \left(\sum_{k \in \mathcal{K}} \boldsymbol{\theta}_{d,k} \left(\frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \right) \sum_{(s,a) \in X_i} \phi_k(s,a) \right) \left(\sum_{k \in \mathcal{K}} \boldsymbol{\theta}_{d,k} \sum_{(s,a) \in X_i} \phi_k(s,a) \right) \right)$$

(ignoring higher order term)

$$= (\frac{1}{|\mathcal{X}|}) \sum_{i=1}^{|\mathbb{X}|} \frac{\exp\left(\sum_{k \in \mathcal{K}} \boldsymbol{\theta}_{d,k} \left(\frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \right) \sum_{(s,a) \in X_i} \phi_k(s,a) \right)}{\sum_{d \in \mathcal{D}} (\frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i}) \sum_{i=1}^{|\mathbb{X}|} \exp\left(\sum_{k \in \mathcal{K}} \boldsymbol{\theta}_{d,k} \left(\frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \right) \sum_{(s,a) \in X_i} \phi_k(s,a) \right)}$$

$$= \frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathbb{X}|} \frac{\exp\left(\sum_{k \in \mathcal{K}} \boldsymbol{\theta}_{d,k} \left(\frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \right) \sum_{(s,a) \in X_i} \phi_k(s,a) \right)}{\Omega} \tag{D.11}$$

In EqD.3, we substitute the value of $\log\left(v_{d,i}Pr_d(X_i)\right)$ from Eq D.5

$$
\begin{aligned}
&\mathcal{L} \\
&= \Bigg(-\sum_{d=1}^{D}\sum_{i=1}^{|\mathbb{X}|} v_{d,i}Pr_d(X_i)\left(-1 + \eta\left(\frac{1}{|\mathcal{X}|}\sum_{i=1}^{|\mathcal{X}|} v_{d,i}\right) + \sum_{k\in\mathcal{K}} \boldsymbol{\theta}_{d,k}\left(\frac{1}{|\mathcal{X}|}\sum_{i=1}^{|\mathcal{X}|} v_{d,i}\right)\right.\right. \\
&\left.\left.\sum_{(s,a)\in X_i}\phi_k(s,a)\right)\right) \\
&+ \left(\frac{1}{|\mathcal{X}|}\sum_{d=1}^{D}\sum_{i=1}^{|\mathcal{X}|} v_{d,i}\,\log\frac{1}{|\mathcal{X}|}\sum_{i=1}^{|\mathcal{X}|} v_{d,i}\right) \\
&+ \eta\left(\sum_{d=1}^{D}\frac{1}{|\mathcal{X}|}\sum_{i=1}^{|\mathcal{X}|} v_{d,i}\sum_{i=1}^{|\mathbb{X}|} v_{d,i}Pr_d(X_i) - 1\right) \\
&+ \sum_{d\in\mathcal{D}}\sum_{k\in\mathcal{K}}\boldsymbol{\theta}_{d,k}\Bigg(\left(\frac{1}{|\mathcal{X}|}\sum_{i=1}^{|\mathcal{X}|} v_{d,i}\right)\sum_{i=1}^{|\mathbb{X}|} v_{d,i}Pr_d(X_i)\sum_{(s,a)\in X_i}\phi_{d,k}(s,a) \\
&- \frac{1}{|\mathcal{X}|}\sum_{i=1}^{|\mathcal{X}|} v_{d,i}\sum_{(s,a)\in X_i}\phi_k(s,a)\Bigg) + \sum_{i=1}^{|\mathcal{X}|}\lambda_i\left(\sum_{d\in\mathcal{D}} v_{d,i} - 1\right)
\end{aligned}
\tag{D.12}
$$

$$
\begin{aligned}
&= \sum_{d=1}^{D} \sum_{i=1}^{|\mathbb{X}|} v_{d,i} Pr_d(X_i) \; - \eta \sum_{d=1}^{D} \left(\frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \right) \sum_{i=1}^{|\mathbb{X}|} v_{d,i} Pr_d(X_i) \\
&- \sum_{d=1}^{D} \sum_{i=1}^{|\mathbb{X}|} v_{d,i} Pr_d(X_i) \sum_{k \in \mathcal{K}} \boldsymbol{\theta}_{d,k} \left(\frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \right) \sum_{(s,a) \in X_i} \phi_k(s,a) \\
&+ \left(\frac{1}{|\mathcal{X}|} \sum_{d=1}^{D} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \; \log \frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \right) + \eta \sum_{d=1}^{D} \frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \sum_{i=1}^{|\mathbb{X}|} v_{d,i} Pr_d(X_i) \\
&- \eta + \sum_{d \in \mathcal{D}} \sum_{k \in \mathcal{K}} \boldsymbol{\theta}_{d,k} \left(\frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \right) \sum_{i=1}^{|\mathbb{X}|} v_{d,i} Pr_d(X_i) \sum_{(s,a) \in X_i} \phi_{d,k}(s,a) \\
&- \sum_{d \in \mathcal{D}} \sum_{k \in \mathcal{K}} \boldsymbol{\theta}_{d,k} \frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \sum_{(s,a) \in X_i} \phi_k(s,a) + \sum_{i=1}^{|\mathcal{X}|} \lambda_i \left(\sum_{d \in \mathcal{D}} v_{d,i} - 1 \right) \\
&= \sum_{d=1}^{D} \sum_{i=1}^{|\mathbb{X}|} v_{d,i} Pr_d(X_i) - 0 - 0 \;\; \text{(2nd term cancels 5th term, 3rd cancels 7th)} \\
&+ \left(\frac{1}{|\mathcal{X}|} \sum_{d=1}^{D} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \; \log \frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \right) + 0 - \eta \\
&+ 0 - \sum_{d \in \mathcal{D}} \sum_{k \in \mathcal{K}} \boldsymbol{\theta}_{d,k} \frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \sum_{(s,a) \in X_i} \phi_k(s,a) + \sum_{i=1}^{|\mathcal{X}|} \lambda_i \left(\sum_{d \in \mathcal{D}} v_{d,i} - 1 \right)
\end{aligned}
\tag{D.13}
$$

Using Eq D.6 and substitute η, we get

$$
\begin{aligned}
&\mathcal{L} \\
&= (\sum_{d=1}^{D} \sum_{i=1}^{|\mathbb{X}|} v_{d,i} Pr_d(X_i) - \eta) - \sum_{d \in \mathcal{D}} \sum_{k \in \mathcal{K}} \boldsymbol{\theta}_{d,k} \frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \sum_{(s,a) \in X_i} \phi_k(s,a) \\
&+ \left(\frac{1}{|\mathcal{X}|} \sum_{d=1}^{D} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \; \log \frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \right) + \sum_{i=1}^{|\mathcal{X}|} \lambda_i \left(\sum_{d \in \mathcal{D}} v_{d,i} - 1 \right) \\
&= (\sum_{d=1}^{D} \sum_{i=1}^{|\mathbb{X}|} v_{d,i} Pr_d(X_i) - \frac{|\mathcal{X}|}{\sum_{i=1}^{|\mathcal{X}|} v_{d,i}} + \frac{|\mathcal{X}|}{\sum_{i=1}^{|\mathcal{X}|} v_{d,i}} \log \Omega) \\
&- \sum_{d \in \mathcal{D}} \sum_{k \in \mathcal{K}} \boldsymbol{\theta}_{d,k} \frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \sum_{(s,a) \in X_i} \phi_k(s,a) \\
&+ \left(\frac{1}{|\mathcal{X}|} \sum_{d=1}^{D} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \; \log \frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \right) + \sum_{i=1}^{|\mathcal{X}|} \lambda_i \left(\sum_{d \in \mathcal{D}} v_{d,i} - 1 \right)
\end{aligned}
\tag{D.14}
$$

For deriving Lagrange multiplier λ_i, take derivative w.r.t. $v_{d,i}$ and make it equal to zero.

$$
\begin{aligned}
&\frac{\partial \mathcal{L}}{\partial v_{d,i}} = P_d(X_i) - \frac{|\mathcal{X}|}{(\sum_{i=1}^{|\mathcal{X}|} v_{d,i})^2} + \frac{|\mathcal{X}|}{(\sum_{i=1}^{|\mathcal{X}|} v_{d,i})} \frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathbb{X}|} v_{d,i} P_d(X_i) + \frac{|\mathcal{X}|}{(\sum_{i=1}^{|\mathcal{X}|} v_{d,i})^2} \log \Omega \\
&- \sum\nolimits_{k \in \mathcal{K}} \boldsymbol{\theta}_{d,k} \frac{1}{|\mathcal{X}|} \sum\nolimits_{(s,a) \in X_i} \phi_k(s,a) + \Bigg(\log \frac{1}{|\mathcal{X}|} \sum\nolimits_{i=1}^{|\mathcal{X}|} v_{d,i} \\
&+ \frac{1}{\frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i}} (\frac{1}{|\mathcal{X}|} \sum\nolimits_{i=1}^{|\mathcal{X}|} v_{d,i}) \Bigg) \frac{\partial (\frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i})}{\partial v_{d,i}} + \lambda_i = 0 \\
&Pr_d(X_i) + \frac{|\mathcal{X}|}{(\sum_{i=1}^{|\mathcal{X}|} v_{d,i})^2} \big(-1 + \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathbb{X}|} v_{d,i} P_d(X_i) + \log \Omega \big) - \sum\nolimits_{k \in \mathcal{K}} \boldsymbol{\theta}_{d,k} \frac{1}{|\mathcal{X}|} \\
&\sum\nolimits_{(s,a) \in X_i} \phi_k(s,a) + \left(\log \frac{1}{|\mathcal{X}|} \sum\nolimits_{i=1}^{|\mathcal{X}|} v_{d,i} + 1 \right) \frac{1}{|\mathcal{X}|} + \lambda_i = 0 \\
&\lambda_i = \sum\nolimits_{k \in \mathcal{K}} \boldsymbol{\theta}_{d,k} \frac{1}{|\mathcal{X}|} \sum\nolimits_{(s,a) \in X_i} \phi_k(s,a) - \left(\log \frac{1}{|\mathcal{X}|} \sum\nolimits_{i=1}^{|\mathcal{X}|} v_{d,i} + 1 \right) \frac{1}{|\mathcal{X}|} - Pr_d(X_i) \\
&+ \frac{|\mathcal{X}|}{(\sum_{i=1}^{|\mathcal{X}|} v_{d,i})^2} \big(1 - \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathbb{X}|} v_{d,i} P_d(X_i) - \log \Omega \big)
\end{aligned}
\tag{D.15}
$$

Substitution of above λ_i value in Eq D.14 gives

$$
\begin{aligned}
L = &(\sum\nolimits_{d=1}^{D} \sum\nolimits_{i=1}^{|\mathbb{X}|} v_{d,i} Pr_d(X_i) - \frac{|\mathcal{X}|}{\sum_{i=1}^{|\mathcal{X}|} v_{d,i}} + \frac{|\mathcal{X}|}{\sum_{i=1}^{|\mathcal{X}|} v_{d,i}} \log \Omega) - \sum\nolimits_{d \in \mathcal{D}} \sum\nolimits_{k \in \mathcal{K}} \boldsymbol{\theta}_{d,k} \frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \\
&\sum\nolimits_{(s,a) \in X_i} \phi_k(s,a) + \left(\frac{1}{|\mathcal{X}|} \sum\nolimits_{d=1}^{D} \sum\nolimits_{i=1}^{|\mathcal{X}|} v_{d,i} \log \frac{1}{|\mathcal{X}|} \sum\nolimits_{i=1}^{|\mathcal{X}|} v_{d,i} \right) \\
&+ \sum_{i=1}^{|\mathcal{X}|} \Bigg(\sum\nolimits_{k \in \mathcal{K}} \boldsymbol{\theta}_{d,k} \frac{1}{|\mathcal{X}|} \sum\nolimits_{(s,a) \in X_i} \phi_k(s,a) - \left(\log \frac{1}{|\mathcal{X}|} \sum\nolimits_{i=1}^{|\mathcal{X}|} v_{d,i} + 1 \right) \frac{1}{|\mathcal{X}|} - Pr_d(X_i) \\
&+ \frac{|\mathcal{X}|}{(\sum_{i=1}^{|\mathcal{X}|} v_{d,i})^2} \big(1 - \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathbb{X}|} v_{d,i} P_d(X_i) - \log \Omega \big) \Bigg) \left(\sum\nolimits_{d \in \mathcal{D}} v_{d,i} - 1 \right)
\end{aligned}
$$

$$
\begin{aligned}
&= (\sum_{d=1}^{D}\sum_{i=1}^{|\mathbb{X}|} v_{d,i} Pr_d(X_i) - \frac{|\mathcal{X}|}{\sum_{i=1}^{|\mathcal{X}|} v_{d,i}} + \frac{|\mathcal{X}|}{\sum_{i=1}^{|\mathcal{X}|} v_{d,i}} \log\Omega) \\
&- \sum_{d\in\mathcal{D}}\sum_{k\in\mathcal{K}} \boldsymbol{\theta}_{d,k} \frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \sum_{(s,a)\in X_i} \phi_k(s,a) \\
&+ \left(\frac{1}{|\mathcal{X}|} \sum_{d=1}^{D}\sum_{i=1}^{|\mathcal{X}|} v_{d,i} \log \frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \right) \\
&+ \sum_{i=1}^{|\mathcal{X}|} \sum_{k\in\mathcal{K}} \left(\sum_{d\in\mathcal{D}} v_{d,i} - 1 \right) \boldsymbol{\theta}_{d,k} \frac{1}{|\mathcal{X}|} \sum_{(s,a)\in X_i} \phi_k(s,a) \\
&- \sum_{i=1}^{|\mathcal{X}|} \left(\log \frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} + 1 \right) \frac{1}{|\mathcal{X}|} \left(\sum_{d\in\mathcal{D}} v_{d,i} - 1 \right) - \sum_{i=1}^{|\mathcal{X}|} Pr_d(X_i) \left(\sum_{d\in\mathcal{D}} v_{d,i} - 1 \right) \\
&+ \sum_{i=1}^{|\mathcal{X}|} \frac{|\mathcal{X}|}{(\sum_{i=1}^{|\mathcal{X}|} v_{d,i})^2} (1 - \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathbb{X}|} v_{d,i} P_d(X_i) - \log\Omega) \left(\sum_{d\in\mathcal{D}} v_{d,i} - 1 \right) \\
&= (\sum_{d=1}^{D}\sum_{i=1}^{|\mathbb{X}|} v_{d,i} Pr_d(X_i) - \frac{|\mathcal{X}|}{\sum_{i=1}^{|\mathcal{X}|} v_{d,i}} + \frac{|\mathcal{X}|}{\sum_{i=1}^{|\mathcal{X}|} v_{d,i}} \log\Omega) \\
&- \sum_{d\in\mathcal{D}}\sum_{k\in\mathcal{K}} \boldsymbol{\theta}_{d,k} \frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \sum_{(s,a)\in X_i} \phi_k(s,a) \\
&+ \left(\frac{1}{|\mathcal{X}|} \sum_{d=1}^{D}\sum_{i=1}^{|\mathcal{X}|} v_{d,i} \log \frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \right) + \sum_{i=1}^{|\mathcal{X}|} \sum_{k\in\mathcal{K}} \left(\sum_{d\in\mathcal{D}} v_{d,i} - 1 \right) \boldsymbol{\theta}_{d,k} \frac{1}{|\mathcal{X}|} \sum_{(s,a)\in X_i} \phi_k(s,a) \\
&- \sum_{i=1}^{|\mathcal{X}|} \left(\frac{1}{|\mathcal{X}|} \sum_{d\in\mathcal{D}} v_{d,i} \log \frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} + \frac{1}{|\mathcal{X}|} \sum_{d\in\mathcal{D}} v_{d,i} \right) + \sum_{i=1}^{|\mathcal{X}|} \frac{1}{|\mathcal{X}|} \left(\log \frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} + 1 \right) \\
&- \sum_{i=1}^{|\mathcal{X}|} \sum_{d\in\mathcal{D}} v_{d,i} Pr_d(X_i) + \sum_{i=1}^{|\mathcal{X}|} Pr_d(X_i) + \frac{|\mathcal{X}|}{\sum_{i=1}^{|\mathcal{X}|} v_{d,i}}
\end{aligned}
$$

(ignoring higher order terms)

(D.16)

Some terms (first and last) cancel out and we get

$$
\begin{aligned}
&= (\sum_{d=1}^{D} \sum_{i=1}^{|\mathbb{X}|} v_{d,i} Pr_d(X_i) + \frac{|\mathcal{X}|}{\sum_{i=1}^{|\mathcal{X}|} v_{d,i}} \log \Omega) \\
&- \sum_{d\in\mathcal{D}} \sum_{k\in\mathcal{K}} \boldsymbol{\theta}_{d,k} \frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \sum_{(s,a)\in X_i} \phi_k(s,a) \\
&+ \sum_{i=1}^{|\mathcal{X}|} \sum_{k\in\mathcal{K}} \left(\sum_{d\in\mathcal{D}} v_{d,i} - 1 \right) \boldsymbol{\theta}_{d,k} \frac{1}{|\mathcal{X}|} \sum_{(s,a)\in X_i} \phi_k(s,a) \\
&- \sum_{i=1}^{|\mathcal{X}|} \frac{1}{|\mathcal{X}|} \sum_{d\in\mathcal{D}} v_{d,i} + \sum_{i=1}^{|\mathcal{X}|} \frac{1}{|\mathcal{X}|} \log \frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} + \sum_{i=1}^{|\mathcal{X}|} \frac{1}{|\mathcal{X}|} - \sum_{i=1}^{|\mathcal{X}|} \sum_{d\in\mathcal{D}} v_{d,i} Pr_d(X_i) + \sum_{i=1}^{|\mathcal{X}|} Pr_d(X_i) \\
&= \frac{|\mathcal{X}|}{\sum_{i=1}^{|\mathcal{X}|} v_{d,i}} \log \Omega - \sum_{d\in\mathcal{D}} \sum_{k\in\mathcal{K}} \boldsymbol{\theta}_{d,k} \frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \sum_{(s,a)\in X_i} \phi_k(s,a) \\
&+ \sum_{i=1}^{|\mathcal{X}|} \sum_{k\in\mathcal{K}} \left(\sum_{d\in\mathcal{D}} v_{d,i} - 1 \right) \boldsymbol{\theta}_{d,k} \frac{1}{|\mathcal{X}|} \sum_{(s,a)\in X_i} \phi_k(s,a) \\
&- 1 + \sum_{i=1}^{|\mathcal{X}|} \frac{1}{|\mathcal{X}|} \log \frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} + 1 + \sum_{d=1}^{D} Pr_d(X_i) (\sum_{i=1}^{|\mathbb{X}|} v_{d,i} - \sum_{i=1}^{|\mathcal{X}|} v_{d,i}) + \sum_{i=1}^{|\mathcal{X}|} Pr_d(X_i) \\
&= \frac{|\mathcal{X}|}{\sum_{i=1}^{|\mathcal{X}|} v_{d,i}} \log \Omega - \sum_{d\in\mathcal{D}} \sum_{k\in\mathcal{K}} \boldsymbol{\theta}_{d,k} \frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \sum_{(s,a)\in X_i} \phi_k(s,a) + \sum_{i=1}^{|\mathcal{X}|} \frac{1}{|\mathcal{X}|} \log \frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \\
&+ \sum_{d=1}^{D} Pr_d(X_i) (\sum_{i=1}^{|\mathbb{X}|} v_{d,i} - \sum_{i=1}^{|\mathcal{X}|} v_{d,i}) + \sum_{i=1}^{|\mathcal{X}|} Pr_d(X_i) \\
&+ \sum_{i=1}^{|\mathcal{X}|} \sum_{k\in\mathcal{K}} \left(\sum_{d\in\mathcal{D}} v_{d,i} - 1 \right) \boldsymbol{\theta}_{d,k} \frac{1}{|\mathcal{X}|} \sum_{(s,a)\in X_i} \phi_k(s,a) \\
&= \frac{|\mathcal{X}|}{\sum_{i=1}^{|\mathcal{X}|} v_{d,i}} \log \Omega - \sum_{d\in\mathcal{D}} \sum_{k\in\mathcal{K}} \boldsymbol{\theta}_{d,k} \frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \sum_{(s,a)\in X_i} \phi_k(s,a) + \sum_{i=1}^{|\mathcal{X}|} \frac{1}{|\mathcal{X}|} \log \frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} v_{d,i} \\
&+ \sum_{d=1}^{D} Pr_d(X_i) (\sum_{i=1}^{|\mathbb{X}|} v_{d,i} - \sum_{i=1}^{|\mathcal{X}|} v_{d,i}) + \sum_{i=1}^{|\mathcal{X}|} Pr_d(X_i) \\
&\text{(last term became o because } \sum_{d\in\mathcal{D}} v_{d,i} = 1 \text{ for each } i) \qquad \text{(D.17)}
\end{aligned}
$$

With $\boldsymbol{v}_i = \begin{bmatrix} v_{1,i} \\ v_{2,i} \\ \vdots \\ v_{D,i} \end{bmatrix}$ be a vector of assignment indicators for trajectory X_i. With $\mathcal{L}$ as Lagrangian dual for problem, the gradients w.r.t $\boldsymbol{v}_i$ will be $\begin{bmatrix} \frac{\partial}{\partial v_{1,i}}\mathcal{L} \\ \frac{\partial}{\partial v_{d,i}}\mathcal{L} \\ \vdots \end{bmatrix}$. The gradients w.r.t optimization variables $v_{1,i}$ and $\boldsymbol{\theta}_{d,k}$ are

$$\begin{aligned}
\frac{\partial}{\partial v_{d,i}}\mathcal{L} &= \frac{|\mathcal{X}|}{\sum_{i=1}^{|\mathcal{X}|} v_{d,i}}\Big(\frac{1}{|\mathcal{X}|}\sum_{i=1}^{|\mathbb{X}|}\frac{\exp\left(\sum_{k\in\mathcal{K}}\boldsymbol{\theta}_{d,k}\left(\frac{1}{|\mathcal{X}|}\sum_{i=1}^{|\mathcal{X}|} v_{d,i}\right)\sum_{(s,a)\in X_i}\phi_k(s,a)\right)}{\Omega}\Big) \\
&\quad \text{(using Eq D.11)} \\
&\quad + \sum_{i=1}^{|\mathcal{X}|}\frac{1}{|\mathcal{X}|}\left(\frac{1}{\frac{1}{|\mathcal{X}|}\sum_{i=1}^{|\mathcal{X}|} v_{d,i}}\right)\frac{1}{|\mathcal{X}|} + P_d(X_i) \qquad \text{(last term is } -P_d(X_i) \text{ for } X_i \text{ in demonstration)} \\
&\quad = \Big(\sum_{i=1}^{|\mathbb{X}|}\frac{\exp\left(\sum_{k\in\mathcal{K}}\boldsymbol{\theta}_{d,k}\left(\frac{1}{|\mathcal{X}|}\sum_{i=1}^{|\mathcal{X}|} v_{d,i}\right)\sum_{(s,a)\in X_i}\phi_k(s,a)\right)}{\sum_{i=1}^{|\mathcal{X}|} v_{d,i}\Omega}\Big) + \left(\frac{1}{\sum_{i=1}^{|\mathcal{X}|} v_{d,i}}\right) \\
&\quad + \frac{|\mathcal{X}|}{v_{d,i}\sum_{i=1}^{|\mathcal{X}|} v_{d,i}}\Big(1 - \log\Omega\Big) \\
\frac{\partial}{\partial v_{d,i}}\mathcal{L} &= \left(\frac{1}{\sum_{i=1}^{|\mathcal{X}|} v_{d,i}}\right)\Big(\sum_{i=1}^{|\mathbb{X}|}\frac{\exp\left(\sum_{k\in\mathcal{K}}\boldsymbol{\theta}_{d,k}\left(\frac{1}{|\mathcal{X}|}\sum_{i=1}^{|\mathcal{X}|} v_{d,i}\right)\sum_{(s,a)\in X_i}\phi_k(s,a)\right)}{\Omega}\Big) \\
&\quad + 1 + \frac{|\mathcal{X}|}{v_{d,i}}(1 - \log\Omega)\Big) \qquad \text{(D.18)}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial}{\partial \boldsymbol{\theta}_{d,k}}\mathcal{L} &= \frac{|\mathcal{X}|}{\sum_{i=1}^{|\mathcal{X}|} v_{d,i}}\left(\left(\frac{1}{|\mathcal{X}|}\sum_{i=1}^{|\mathcal{X}|} v_{d,i}\right)E_{\mathbb{X},d}[\phi_k]\right) \qquad \text{(Using Eq D.10)} \\
&\quad - \frac{1}{|\mathcal{X}|}\sum_{i=1}^{|\mathcal{X}|} v_{d,i}\sum_{(s,a)\in X_i}\phi_k(s,a) \\
&\quad = E_{\mathbb{X},d}[\phi_k] - \hat{\phi}_{d,k}
\end{aligned}$$

Appendix E

Appendix E

E.1 Proofs for RIMEO

Lemma 7 (Monotonicity)**.** The demonstration likelihood increases monotonically with each new session, $LL(\boldsymbol{\theta}^i|\Xi_{d,i}, \alpha_{1:i-1}, \boldsymbol{\theta}^{i-1}) - LL(\boldsymbol{\theta}^{i-1}|\Xi_{d,i-1}, \alpha_{1:i-2}, \boldsymbol{\theta}^{i-2}) \geqslant 0$, when $|\Xi_{d,1:i-1}| \gg |\Xi_{d,i}|$.

Proof: Log-likelihood of demonstrated behavior can be split as

$$
\begin{aligned}
&LL(\boldsymbol{\theta}^i|\Xi_{d,i}, \alpha_{1:i-1}, \boldsymbol{\theta}^{i-1}) \\
&= \sum_{\xi' \in \Xi_{d,1:i}} \tilde{P}(\xi') \log P(\xi'; \boldsymbol{\theta}) \\
&= \sum_{\xi' \in \Xi_{d,1:i}} \tilde{P}(\xi') \sum_{\xi \in \Xi} P(\xi|\xi'; \boldsymbol{\theta}^i) \log P(\xi, \xi'; \boldsymbol{\theta}) + (- \sum_{\xi' \in \Xi_{d,1:i}} \tilde{P}(\xi) \sum_{\xi \in \Xi} P(\xi|\xi'; \boldsymbol{\theta}^i) \log P(\xi|\xi'; \boldsymbol{\theta})) \\
&= Q(\Xi_{d,1:i}, \boldsymbol{\theta}^i) + C(\Xi_{d,1:i}, \boldsymbol{\theta}^i)
\end{aligned}
$$

Here $\tilde{P}$ is distribution of trajectories in observed training data ($\sum_{\xi' \in \Xi_{d,1:i}} \tilde{P}(\xi')[\cdot]$ and $\frac{1}{|\Xi_{d,1:i}|} \sum_{\xi' \in \Xi_{d,1:i}} [\cdot]$ can be used interchangeably). The EM method maximizes the log-likelihood by maximizing only Q value over $\boldsymbol{\theta}$; and $\boldsymbol{\theta} = \boldsymbol{\theta}^i$ maximizes $Q(\Xi_{d,1:i}, \boldsymbol{\theta}^i)$ ([83]). After all the EM iterations for current session i, the final Q value is $Q(\Xi_{d,1:i}, \boldsymbol{\theta}^i)$. Therefore, the difference in the likelihoods achieved by weights learned in consecutive sessions can be expressed as a difference in Q values. Note that Robust IRL learns reward weights by inferring the maximum entropy distribution $P(\xi, \xi'; \boldsymbol{\theta}) = \frac{\exp(\sum_k \theta_k f_k(\xi))}{\Omega^{\Xi}_{\boldsymbol{\theta}}}$ (Equation 15 in [74]), where $\Omega^{\Xi}_{\boldsymbol{\theta}} = \sum_{\xi \in \Xi} \exp(\sum_k \theta_k f_k(\xi))$. Expand Q value as $Q(\Xi_{d,1:i}, \boldsymbol{\theta}^i) = \sum_{\xi' \in \Xi_{d,1:i}} \tilde{P}(\xi') \sum_{\xi \in \mathbb{X}} P(\xi|\xi'; \boldsymbol{\theta}^i) \log \left(\frac{\exp(\sum_k \theta^i_k f_k(\xi))}{\Omega^{\Xi}_{\boldsymbol{\theta}^i}} \right)$ $= \sum_k \theta^i_k \cdot \sum_{\xi' \in \Xi_{d,1:i}} \tilde{P}(\xi') \sum_{\xi \in \Xi} P(\xi|\, \xi'; \boldsymbol{\theta}^i) f_k(\xi) - \log \Omega^{\Xi}_{\boldsymbol{\theta}^i} = \sum_k \theta^i_k \cdot \hat{\phi}^{1:i}_{\boldsymbol{\theta}^i, k} - \log \Omega^{\Xi}_{\boldsymbol{\theta}^i}$.

Therefore the improvement in log likelihood over session i is

$$
\begin{aligned}
&LL(\boldsymbol{\theta}^i|\Xi_{d,i},\alpha_{1:i-1},\boldsymbol{\theta}^{i-1}) - LL(\boldsymbol{\theta}^{i-1}|\Xi_{d,i-1},\alpha_{1:i-2},\boldsymbol{\theta}^{i-2}) \\
&= Q(\Xi_{d,1:i},\boldsymbol{\theta}^i) - Q(\Xi_{d,1:i-1},\boldsymbol{\theta}^{i-1}) \\
&= \sum_k \theta_k^i \hat{\phi}^{1:i}_{\boldsymbol{\theta}^i,k} - \log\Omega^{\Xi}_{\boldsymbol{\theta}^i} - \sum_k \theta_k^{i-1}\hat{\phi}^{1:i-1}_{\boldsymbol{\theta}^{i-1},k} + \log\Omega^{\Xi}_{\boldsymbol{\theta}^{i-1}} \\
&= \log\frac{\Omega^{\Xi}_{\boldsymbol{\theta}^{i-1}}}{\Omega^{\Xi}_{\boldsymbol{\theta}^i}} + \sum_k \left(\theta_k^i \frac{|\Xi_{d,1:i-1}|}{|\Xi_{d,i}|+|\Xi_{d,1:i-1}|} - \theta_k^{i-1}\right)\hat{\phi}^{1:i-1}_{\boldsymbol{\theta}^{i-1},k} + \sum_k \left(\theta_k^i \frac{1}{|\Xi_{d,i}|+|\Xi_{d,1:i-1}|}\hat{\phi}^{i}_{\boldsymbol{\theta}^{i},k}\right)
\end{aligned}
$$

(substitute $\hat{\phi}^{1:i}_{\boldsymbol{\theta}^i,k}$ using Eq. 11 from main paper and simplifying)

The final expression is minimized only for $\boldsymbol{\theta}^i = \boldsymbol{\theta}^{i-1}$ when $|\Xi_{d,1:i-1}| \gg |\Xi_{d,i}|$, i.e., when a significant amount of training data has been accumulated. The expression is also concave in parameter $\boldsymbol{\theta}^i$. Therefore, $LL(\boldsymbol{\theta}^i|\Xi_{d,i},\alpha_{1:i-1},\boldsymbol{\theta}^{i-1}) - LL(\boldsymbol{\theta}^{i-1}|\Xi_{d,i-1},\alpha_{1:i-2},\boldsymbol{\theta}^{i-2}) \geq 0$ for consecutive sessions thereafter. □

Lemma 8 (Constraint Bound)**.** Under the assumptions stated in Sec. 5.2 (main paper), the following holds with probability at least $\max(0, 1-\delta_r)$:

$$\left|(1-\gamma)\left(E_\Xi[\phi_k] - \hat{\phi}^{1:i}_{\boldsymbol{\theta}^i,k}\right)\right|_1 \leqslant \varepsilon_r, k \in \{1,2\ldots K\}$$

where L is the maximum length of any trajectory, $\delta_r = \delta + \delta_s + \delta_o$ and $\varepsilon_r = \varepsilon + \varepsilon_s + L|\Psi|\varepsilon_o$, and ε, δ are as defined in Theorem 1 in [5].

Proof: Suppose the true (unknown) observation model $\forall o, g$ is $O^*_{o,g}$. Solving the NLP with the true observation model gives the true $P(\psi)$, since the constraint below is satisfied.

$$\prod_{\psi^{o,g}=1} P(\psi) \prod_{\psi^{o,g}=0} (1-P(\psi)) = O^*_{o,g} \tag{E.1}$$

Using these true $P(\psi)$ instead of $P^*(\psi)$, we can generate a version of Eq. 11 (main paper):

$$\phi^{1:i}_{\boldsymbol{\theta},k} = \frac{1}{|\Xi_{d,1:i}|}\sum_{\xi'\in\Xi_{d,1:i}}\sum_{\xi\in\Xi}\eta P(\xi'|\xi)P(\xi;\boldsymbol{\theta})f_k(\xi)$$

From the accumulated sessions, we get estimates of $O^*_{o,g}$, call it $\hat{O}_{o,g}$ (Eq. 8 in the main paper). We assume that this estimate satisfies Hoeffding bounds for *the observed state-action pairs*, viz., $P(|O^*_{o,g} - \hat{O}_{o,g}| \leq \epsilon_o) \geq 1 - \frac{\delta_o}{K}$, where $\delta_o = 2K|\Psi|\exp(-2\epsilon_o^2 n_o)$, n_o being the number of samples used to construct $\hat{O}_{o,g}$. The key issue is that this estimate may not be available yet for the $\langle s,a\rangle_o$ pairs that were not observed. Regardless, we assume that all features in Ψ are observed in the very first session. Hence, after solving the NLP, we obtain $\hat{O}_{o,g}$ for *all* o, g, using the $P^*(\psi)$ from observed $\langle s,a\rangle_o$s and

$$\hat{O}_{o,g} = \prod_{\psi^{o,g}=1} P^*(\psi) \prod_{\psi^{o,g}=0} (1-P^*(\psi)) \tag{E.2}$$

Under the assumptions above, with probability $\geq 1-\delta_o$, $\max_{\langle s,a\rangle_o} |O^*_{o,g} - \hat{O}_{o,g}| \leq \epsilon_o$, but only for the observed $\langle s,a\rangle_o$. Since $\max_{any\ \psi} |P(\psi) - P^*(\psi)| \leq 1$, in turn this yields $\max_{any \langle s,a\rangle_o} |O^*_{o,g} - \hat{O}_{o,g}| \leq |\Psi|\epsilon_o$. Consequently, if the length of trajectories is bounded by L, then with probability $\geq 1-\frac{\delta_o}{K}$ we have $\forall k$

$$
\begin{aligned}
|\phi^{1:i}_{\boldsymbol{\theta}^i,k} - \hat{\phi}^{1:i}_{\boldsymbol{\theta}^i,k}| &= \frac{1}{|\Xi_{d,1:i}|} \sum_{\xi' \in \Xi_{d,1:i}} \sum_{\xi \in \Xi} f_k(\xi) \eta P(\xi;\boldsymbol{\theta}) |(P(\xi'|\xi) - P^*(\xi'|\xi))| \\
&= \frac{1}{|\Xi_{d,1:i}|} \sum_{\xi' \in \Xi_{d,1:i}} \sum_{\xi \in \Xi} f_k(\xi) \eta P(\xi;\boldsymbol{\theta}) |(\prod_{o,g} O^*_{o,g} - \prod_{o,g} \hat{O}_{o,g})| \\
&\leq \frac{1}{|\Xi_{d,1:i}|} \sum_{\xi' \in \Xi_{d,1:i}} \sum_{\xi \in \Xi} f_k(\xi) \eta P(\xi;\boldsymbol{\theta}) L \max_{any\ o} |O^*_{o,g} - \hat{O}_{o,g}| \\
&\leq L|\Psi|\epsilon_o/(1-\gamma)
\end{aligned}
$$

The rest of the proof follows similar steps as in [5]. We define the events A_k, B_l, C_j as:

$A_k : (1-\gamma)\left|E_\Xi[\phi_k] - \hat{\phi}^{1:i}_k\right| > \varepsilon, k \in \{1, 2 \dots K\}$.

Applying Hoeffding's inequality for A_k, we get $P(A_k) \leq 2\exp(-2\varepsilon^2 |\Xi_{d,1:i}|) \leq \frac{\delta}{K}$ for any $k \in \{1, 2 \dots K\}$, and for the same ε, δ as in Theorem 1. Similarly, for noisy observation, given ε_s as the bound on the error in sampling based approximation of $\hat{\phi}^{1:i}_l$ as $\phi^{1:i}_{\boldsymbol{\theta}^i,l}$, and n_s samples, let us define the event

$B_l : (1-\gamma)\left|\hat{\phi}^{1:i}_l - \phi^{1:i}_{\boldsymbol{\theta}^i,l}\right| > \varepsilon_s, l \in \{1, 2 \dots K\}$.

Similar to procedure for $P(A_k)$, applying Hoeffding bound gives us $P(B_l) < \frac{\delta_s}{K}$, $\delta_s = 2K\exp(-2\varepsilon_s^2 n_s)$. Finally,

$C_j : (1-\gamma)\left|\phi^{1:i}_{\boldsymbol{\theta}^i,j} - \hat{\phi}^{1:i}_{\boldsymbol{\theta}^i,j}\right| > L|\Psi|\varepsilon_o, j \in \{1, 2 \dots K\}$. Then following the argument above, $P(C_j) < \frac{\delta_o}{K}$.

Applying Fretchets inequality over the sets A, B, and C of events gives us:

$P\left((\cup_k A_k) \vee (\cup_l B_l) \vee (\cup_j C_j)\right) < \min(1, \sum_{k=1}^K \frac{\delta}{K} + \sum_{l=1}^K \frac{\delta_s}{K} + \sum_{j=1}^K \frac{\delta_o}{K}) = \min(1, \delta + \delta_s + \delta_o)$.

That is, $P\left(\exists k, l, j s.t. A_k \vee B_l \vee C_j\right) < \min(1, \delta + \delta_s + \delta_o)$. Taking complement, $P\left(\forall k, l, j, \overline{A}_k \wedge \overline{B}_l \wedge \overline{C}_j\right) \geq \max(0, 1 - \delta - \delta_s - \delta_o)$. But $\forall k, l, j, \overline{A}_k \wedge \overline{B}_l \wedge \overline{C}_j$ implies that $\forall k$:

$(1-\gamma)(\left|E_\Xi[\phi_k] - \hat{\phi}^{1:i}_k\right| + \left|\hat{\phi}^{1:i}_k - \phi^{1:i}_{\boldsymbol{\theta}^i,k}\right| + \left|\phi^{1:i}_{\boldsymbol{\theta}^i,k} - \hat{\phi}^{1:i}_{\boldsymbol{\theta}^i,k}\right|) \leq \varepsilon + \varepsilon_s + L|\Psi|\varepsilon_o$.

Hence $P\big(\forall k, (1-\gamma)(\left|E_\Xi[\phi_k] - \hat{\phi}^{1:i}_k\right| + \left|\hat{\phi}^{1:i}_k - \phi^{1:i}_{\boldsymbol{\theta}^i,k}\right| + \left|\phi^{1:i}_{\boldsymbol{\theta}^i,k} - \hat{\phi}^{1:i}_{\boldsymbol{\theta}^i,k}\right|) \leq \varepsilon + \varepsilon_s + L|\Psi|\varepsilon_o\big) \geq \max(0, 1 - \delta - \delta_s - \delta_o)$.

Using $\left|E_\Xi[\phi_k] - \hat{\phi}^{1:i}_{\boldsymbol{\theta}^i,k}\right| \leq \left|E_\Xi[\phi_k] - \hat{\phi}^{1:i}_k\right| + \left|\hat{\phi}^{1:i}_k - \phi^{1:i}_{\boldsymbol{\theta}^i,k}\right| + \left|\phi^{1:i}_{\boldsymbol{\theta}^i,k} - \hat{\phi}^{1:i}_{\boldsymbol{\theta}^i,k}\right|$, $\delta_r = \delta + \delta_s + \delta_o$, and $\varepsilon_r = \varepsilon + \varepsilon_s + L|\Psi|\varepsilon_o$, we get:

$$P\left(\forall k, (1-\gamma)(\left|E_\Xi[\phi_k] - \hat{\phi}^{1:i}_{\boldsymbol{\theta}^i,k}\right|) \leq \varepsilon_r\right) \geq \max(0, 1 - \delta_r). \qquad \square$$

THEOREM 6 (CONFIDENCE). *Let ε_r, δ_r be as defined in Lemma 8, and $\boldsymbol{\theta}^i$ be the solution of session i for RI2RL-MEOM. Then*

$$LL(\boldsymbol{\theta}_E|\Xi_{d,1:i}) - LL(\boldsymbol{\theta}^i|\Xi_{d,i}, \alpha_{1:i-1}, \boldsymbol{\theta}^{i-1}) \leq \frac{2K\varepsilon_r}{(1-\gamma)},$$

with confidence at least $\max(0, 1-\delta_r)$*, where $\boldsymbol{\theta}_E$ are the true weights of the expert.*

Proof: Each session of RI2RL-MEOM solves a maximum entropy estimation problem for Robust IRL. By allowing a relaxation in the constraints for a session, we get

$$\begin{array}{l} \max\limits_{\Delta} \left(-\sum_{\xi' \in \Xi_{d,1:i}, \xi \in \Xi} P(\xi', \xi) \log P(\xi', \xi) \right) \\ \textbf{subject to} \quad \sum_{\xi' \in \Xi_{d,1:i}, \xi \in \Xi} P(\xi', \xi) = 1 \\ \left| E_\Xi[\phi_k] - \hat{\phi}^{1:i}_{\boldsymbol{\theta}^i, k} \right| \leq \beta_k \quad \forall k \end{array} \tag{E.3}$$

where

$$E_\Xi[\phi_k] \triangleq \sum\nolimits_{\xi \in \Xi, \xi' \in \Xi_{d,1:i}} P(\xi, \xi')\, f_k(\xi),\ k = 1 \ldots K \tag{E.4}$$

Here $\beta \in \mathbb{R}^K$ is a vector of upper bounds on the differences between feature expectations. Following the proofs by Dudik et al. [32], the above relaxed constraints problem is the same as $\min_{\boldsymbol{\theta}}(-\sum_{\xi \in \Xi_{d,1:i}} \tilde{P}(\xi) \ \log P(\xi|\boldsymbol{\theta}) + \sum_k \beta_k|\theta_k|) = \min_{\boldsymbol{\theta}}(-LL(\boldsymbol{\theta}|\, \Xi_{d,i}, \alpha_{1:i-1}, \boldsymbol{\theta}^{i-1}) + \sum_k \beta_k|\theta_k|) = \min_{\boldsymbol{\theta}} NLL_\beta(\boldsymbol{\theta}|\Xi_{d,i}, \alpha_{1:i-1}, \boldsymbol{\theta}^{i-1})$ (say). Here $NLL =$ negative log likelihood.

The proof here is partially inspired from Corollary 1 in [32]. Let $\beta_k = \beta_c = \varepsilon/(1-\gamma)$ for all $k \in \{1 \ldots K\}$, where β_c is a constant because ε is a fixed input. For normalized exponentiated gradient descent used in reward-learning part of RI2RL session, $\sum_1^K |\theta_k| = 1$. Then, $NLL_\beta(\boldsymbol{\theta}|\Xi_{d,i}, \alpha_{1:i-1}, \boldsymbol{\theta}^{i-1}) = (-LL(\boldsymbol{\theta}|\Xi_{d,i}, \alpha_{1:i-1}, \boldsymbol{\theta}^{i-1}) + \beta_c \sum_1^k |\theta_k|) = (-LL(\boldsymbol{\theta}|\Xi_{d,i}, \alpha_{1:i-1}, \boldsymbol{\theta}^{i-1}) + \beta_c)$. Assume that $\boldsymbol{\theta}^i$ minimizes $NLL_\beta(\boldsymbol{\theta}|\, \Xi_{d,i}, \alpha_{1:i-1}, \boldsymbol{\theta}^{i-1})$, a solution maximizing $LL(\boldsymbol{\theta}|\Xi_{d,i}, \alpha_{1:i-1}, \boldsymbol{\theta}^{i-1})$.

Since $E_\Xi[\phi_k] \in \left[0, \frac{1}{(1-\gamma)}\right]$, we get $(1-\gamma)E_\Xi[\phi_k] \in [0, 1]$. Using the result from the previous Lemma, the probability that $\left|(1-\gamma)E_\Xi[\phi_k] - (1-\gamma)\hat{\phi}^{1:i}_{\boldsymbol{\theta}^i, k}\right| \leq \varepsilon_r\ \forall k \in \{1 \ldots K\}$ is at least $\max(0, 1-\delta_r)$. To keep the reward value bounded, IRL assumes $||\boldsymbol{\theta}^*||_1 \leq 1$ for all $\boldsymbol{\theta}^*$. Using the assumption and Theorem 1 in [32], we get the following error bound:

For every $\boldsymbol{\theta}^* \in [0, 1]^K$, $NLL_\beta(\boldsymbol{\theta}^i|\Xi_{d,i}, \alpha_{1:i-1}, \boldsymbol{\theta}^{i-1}) - NLL_\beta(\boldsymbol{\theta}^*|\Xi_{d,i}, \alpha_{1:i-1}, \boldsymbol{\theta}^{i-1}) \leq 2\sum_1^K \beta_c = 2K\,\beta_c = \frac{2K\varepsilon_r}{(1-\gamma)}$, with probability at least $\max(0, 1-\delta_r)$.

We modify the bound in the form of positive log-likelihood of expert's policy, by using the relation $NLL_\beta(\boldsymbol{\theta}^*|\Xi_{d,1:i}) = (-LL(\boldsymbol{\theta}^*|\, \Xi_{d,1:i}) + \sum_1^K \beta_k|\theta_k|)$ and $\boldsymbol{\theta}^* = \boldsymbol{\theta}_E$.

Then, with $\Xi_{d,1:i}$ as input, with probability at least $\max(0, 1-\delta_r)$,

$$\begin{aligned} &NLL_\beta(\boldsymbol{\theta}^i|\Xi_{d,i}, \alpha_{1:i-1}, \boldsymbol{\theta}^{i-1}) - NLL_\beta(\boldsymbol{\theta}_E|\Xi_{d,1:i}) \\ &= LL(\boldsymbol{\theta}_E|\Xi_{d,1:i}) - LL(\boldsymbol{\theta}^i|\Xi_{d,i}, \alpha_{1:i-1}, \boldsymbol{\theta}^{i-1}) \leq \frac{2K\varepsilon_r}{(1-\gamma)}. \end{aligned}$$

□

THEOREM 7 (NO REGRET LEARNING). *There exist choices for a variable threshold bound on the feature matching constraint, $\varepsilon_{r,i}$ as a function of session i, such that with probability at least* $\max(0, \prod_{i=1}^{i=T}(1-\delta_{r,i}))$*, where $\delta_{r,i}$ is as defined in Lemma 8 ,*

$$\text{Reg}_T(A_{RI_2RL}) = o(T)/T,$$

thus approaching o as $T \to \infty$.

Proof of Sub-Linear Regret. The log-loss after ith session is $LL(\boldsymbol{\theta}_E|\Xi_{d,1:i}) - LL(\boldsymbol{\theta}^i|\Xi_{d,i}, \hat{\xi}_{i-1}, \boldsymbol{\theta}^{i-1})$. Let the regret after $i = T$ be $\frac{1}{T}\sum_{i=1}^{T} LL(\boldsymbol{\theta}_E|\Xi_{d,1:i}) - \frac{1}{T}\sum_{i=1}^{T} LL(\boldsymbol{\theta}^i|\Xi_{d,i}, \hat{\xi}_{i-1}, \boldsymbol{\theta}^{i-1})$. According to Theorem 6, $LL(\boldsymbol{\theta}_E|\Xi_{d,1:i}) - LL(\boldsymbol{\theta}^i|\Xi_{d,i}, \hat{\xi}_{i-1}, \boldsymbol{\theta}^{i-1}) \leq \alpha \cdot \varepsilon_r$ with probbility at least $\max(0, 1-\delta_r)$, where $\alpha = \frac{2K}{(1-\gamma)}$. As ε_r is user specified, let $\varepsilon_{r,i} = \frac{c}{i}$. Then, the inequality becomes $LL(\boldsymbol{\theta}_E|\Xi_{d,1:i}) - LL(\boldsymbol{\theta}^i|\Xi_{d,i}, \hat{\xi}_{i-1}, \boldsymbol{\theta}^{i-1}) \leq \alpha\frac{c}{i}$. Summing the result over $i \in \{1, 2, \ldots T\}$ and dividing by T, we get

$$\frac{1}{T}\sum_{i=1}^{T} LL(\boldsymbol{\theta}_E|\Xi_{d,1:i}) - \frac{1}{T}\sum_{i=1}^{T} LL(\boldsymbol{\theta}^i|\Xi_{d,i}, \hat{\xi}_{i-1}, \boldsymbol{\theta}^{i-1}) \leq \alpha\frac{1}{T}\sum_{i=1}^{T}\frac{c}{i}$$

The RHS above is bounded as $\alpha\frac{1}{T}\sum_{i=1}^{T}\frac{c}{i} \leq \alpha\frac{1}{T}c\log T = \alpha c\frac{\log T}{T}$. As $T \to \infty$, $\alpha c\frac{\log T}{T} \to 0$. Therefore, as sessions progress, regret is guaranteed to vanish.

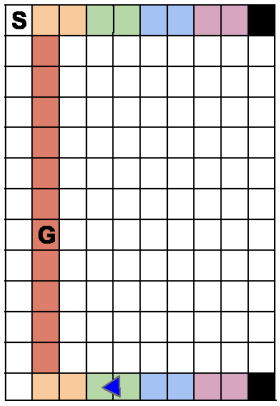
□

Appendix F

Appendix F

F.1 Features of Perimeter Patrol

Reward Features



The 6 reward features $\phi_k(s, a)$, in the context of the above figure, are:

- *HasMoved*(s, a): true iff a in s makes the patroller change its grid cell;
- *Turn1*(s, a): true iff a in s makes the patroller turn (left or right) in the orange part of the hallway;
- *Turn2*(s, a): true iff a in s makes the patroller turn in the yellow part of hallway;
- *Turn3*(s, a): true iff a in s makes the patroller turn in the green part of hallway;
- *Turn4*(s, a): true iff a in s makes the patroller turn in the blue part of hallway;
- *Turn5*(s, a): true iff a in s makes the patroller turn in the magenta part of hallway.

A weight vector $\boldsymbol{\theta}_E$ for these features such as $\langle .57, 0, 0, 0, .43, 0\rangle$ makes the patroller constantly execute a cyclic trajectory.

Observation Features

The observation feature set Ψ contains the following 4 binary predicates:

- *MoveForward:* patroller is moving forward;
- *TurnLeft:* patroller is turning left;
- *y is o:* patroller location has $y = 0$;
- *TurnRight:* patroller is turning right;

Average of pairwise feature correlation from the patroller's demonstration is -0.14 (p-value 0.06), indicating that the features are reasonably independent.

F.2 Features of Onion Sorting

Reward Features

The 11 reward features $\phi_k(s, a)$ are:

- *CreateList*(s,a): Roll all onions and create a list of predictions (blemished/unblemished/unknown);
- *ClaimNewOnion*(s,a): considers a new onion on table;
- *PickUnknown*(s,a): is 1 when onion with unknown prediction is picked;
- *AvoidNoOp*(s,a): the action a changes the state;
- *InspectNewOnion*(s,a): is 1 when an onion is inspected for the first time and a prediction is made for it;
- *GoodOnTable*(s,a): considered onion is unblemished and is placed on the table;
- *BlemishedNotOnTable*(s,a): onion is blemished and is not placed on the table;
- *GoodNotInBin*(s,a): onion is unblemished and is not placed in the bin;
- *BlemishedInBin*(s,a): onion is blemished and is placed in the bin;
- *PickBlemished*(s,a): onion with prediction blemished is picked;
- *EmptyList*(s,a): finish sorting bad onions out of the conveyor.

Observation Features used for RIMEO

The 8 observation features, $\psi_j, j = 1, \ldots, 8$, are listed below. Each indicator $\psi_j^{o,g}$ takes the value 1 iff the predicate value is the same for both $\langle s, a \rangle_g$ and $\langle s, a \rangle_o$.

- *BlemishedOnion*: considered onion is blemished;
- *MoveWithHand*: onion moves with the hand;
- *StartFromConv*: onion was on the table before action;
- *LeavingAtEye*: onion leaves atEye location;

- *OnionToBin*: onion moves to the bin;
- *HandToBin*: hand moves to the bin;
- *OnionToTable*: onion moves to the table;
- *HandToTable*: hand moves to the table;

Appendix G

Appendix G

G.1 Hyper-parameters used for AIRL methods

Here is the key explaining meaning of hyperparmeters used in GAN training:

Batch size gen : batch size for generator (number of (s,a) pairs or transitions used to train the generator policy in each AIRL iteration)

Training time steps : training time step bound (An upper bound on the total number of (s,a,s') transitions to sample from the environment during AIRL training, throughout multiple iterations of generator training)

rollout size : size of input demonstration

batch size for discriminator : Twice the number of (s,a) samples in each batch of expert data because each discriminator batch contains a generator sample for every input sample

num updates disc : number of discriminator updates after each round of generator update = 4

num envs : number of Gym environments used in parallel

RL algo gen : RL algorithm used from stable baselines library to compute generator's policy. Hyper-parameters for PPO are default recommended onse (learning rate 0.0003 , batch size 64 , number of epochs 10 , entropy coefficient 0).

len epi : Length of one episode

δ_G : Gibbs sampling threshold on the normed change in the running averaged of logits

Table G.1: Values of parameters used for both application domains

	Perim Patrol	Mount Car	HalfCheetah/Hopper /Walker	SortOnion (continuous)
Training time steps	7500	2048	7500	384000
batch size disc (AIRL)	32 to 128	1024 to 4096	512	128
num updates disc	4	4	4	4
num envs	8	4	4	64
RL algo gen	PPO	PPO	PPO	PPO
len epi	8	Not Applicable	<=1000	Not applicable
δ_G	0.0375	0.1	(0.1 or 20 iterations max)	0.02
rollout size (AIRL)	32 to 128	1024 to 4096	256	32
Batch size gen	64	2048	1024	128

Bibliography

[1] Pieter Abbeel and Andrew Y. Ng. "Apprenticeship Learning via Inverse Reinforcement Learning". In: *Proceedings of the Twenty-first International Conference on Machine Learning*. ICML '04. Banff, Alberta, Canada: ACM, 2004, pp. 1–8. ISBN: 1-58113-838-5.

[2] Pieter Abbeel and Andrew Y. Ng. "Apprenticeship Learning via Inverse Reinforcement Learning". In: *Twenty-first International Conference on Machine Learning (ICML)*. 2004, pp. 1–8.

[3] Pieter Abbeel et al. "An Application of Reinforcement Learning to Aerobatic Helicopter Flight". In: *Proceedings of the 19th International Conference on Neural Information Processing Systems*. NIPS'06. Canada: MIT Press, 2006, pp. 1–8.

[4] Kareem Amin, Nan Jiang, and Satinder Singh. "Repeated inverse reinforcement learning". In: *Advances in Neural Information Processing Systems*. 2017, pp. 1815–1824.

[5] Saurabh Arora, Prashant Doshi, and Bikramjit Banerjee. "I2RL: online inverse reinforcement learning under occlusion". In: *Autonomous Agents and Multi-Agent Systems* 35.1 (Nov. 2020), p. 4. ISSN: 1573-7454.

[6] Saurabh Arora, Prashant Doshi, and Bikramjit Banerjee. "Min-Max Entropy Inverse RL of Multiple Tasks". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 12639–12645.

[7] Saurabh Arora, Prashant Doshi, and Bikramjit Banerjee. "Online Inverse Reinforcement Learning with Learned Observation Model". In: *Proceedings of The 6th Conference on Robot Learning*. Ed. by Karen Liu, Dana Kulic, and Jeff Ichnowski. Vol. 205. Proceedings of Machine Learning Research. PMLR, 14–18 Dec 2023, pp. 1468–1477.

[8] Peter Auer et al. "Gambling in a rigged casino: The adversarial multi-armed bandit problem". In: *Electronic Colloquium on Computational Complexity (ECCC)* 7.68 (2000).

[9] Monica Babes-Vroman et al. "Apprenticeship learning about multiple intentions". In: *28th International Conference on Machine Learning (ICML)*. 2011, pp. 897–904.

[10] Chris L. Baker, Rebecca Saxe, and Joshua B. Tenenbaum. "Action understanding as inverse planning". In: *Cognition* 113.3 (2009). Reinforcement learning and higher cognition, pp. 329–349. ISSN: 0010-0277.

[11] Y. Bard. "Estimation of State Probabilities Using the Maximum Entropy Principle". In: *IBM Journal of Research and Development* 24.5 (1980), pp. 563–569.

[12] Kenneth Bogert and Prashant Doshi. "A Hierarchical Bayesian Process for Inverse RL in Partially-Controlled Environments". In: 2022, pp. 145–153.

[13] Kenneth Bogert and Prashant Doshi. "Multi-robot Inverse Reinforcement Learning Under Occlusion with Interactions". In: *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems*. AAMAS '14. 2014, pp. 173–180.

[14] Kenneth Bogert and Prashant Doshi. "Multi-Robot Inverse Reinforcement Learning Under Occlusion with State Transition Estimation". In: *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*. AAMAS '15. Istanbul, Turkey: International Foundation for Autonomous Agents and Multiagent Systems, 2015, pp. 1837–1838. ISBN: 978-1-4503-3413-6.

[15] Kenneth Bogert and Prashant Doshi. "Scaling Expectation-Maximization for Inverse Reinforcement Learning to Multiple Robots Under Occlusion". In: *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*. AAMAS '17. 2017, pp. 522–529.

[16] Kenneth Bogert and Prashant Doshi. "Scaling Expectation-Maximization for Inverse Reinforcement Learning to Multiple Robots under Occlusion". In: *16th Conference on Autonomous Agents and MultiAgent Systems*. 2017, pp. 522–529.

[17] Kenneth Bogert and Prashant Doshi. "Toward Estimating Others' Transition Models Under Occlusion for Multi-robot IRL". In: *24th International Joint Conference on Artificial Intelligence (IJCAI)*. 2015, pp. 1867–1873.

[18] Kenneth Bogert et al. "Expectation-Maximization for Inverse Reinforcement Learning with Hidden Data". In: *Proceedings of the 2016 International Conference on Autonomous Agents and Multiagent Systems*. AAMAS '16. International Foundation for Autonomous Agents and Multiagent Systems, 2016, pp. 1034–1042.

[19] Kenneth Bogert et al. "Expectation-Maximization for Inverse Reinforcement Learning with Hidden Data". In: *2016 International Conference on Autonomous Agents and Multiagent Systems*. 2016, pp. 1034–1042.

[20] Abdeslam Boularias, Jens Kober, and Jan Peters. "Relative Entropy Inverse Reinforcement Learning". In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*. 2011, pp. 182–189.

[21] Abdeslam Boularias, Oliver Krömer, and Jan Peters. "Structured Apprenticeship Learning". In: *European Conference on Machine Learning and Knowledge Discovery in Databases, Part II*. 2012, pp. 227–242.

[22] Greg Brockman et al. "OpenAI Gym". In: *CoRR* abs/1606.01540 (2016). arXiv: 1606.01540. URL: http://arxiv.org/abs/1606.01540.

[23] Daniel S Brown, Wonjoon Goo, and Scott Niekum. "Better-than-demonstrator imitation learning via automatically-ranked demonstrations". In: *Conference on robot learning*. PMLR. 2020, pp. 330–359.

[24] Daniel Brown et al. "Extrapolating Beyond Suboptimal Demonstrations via Inverse Reinforcement Learning from Observations". In: *Proceedings of the 36th International Conference on Machine Learning*. Vol. 97. Proceedings of Machine Learning Research. 2019, pp. 783–792.

[25] Letian Chen, Rohan Paleja, and Matthew Gombolay. "Learning from Suboptimal Demonstration via Self-Supervised Reward Regression". In: *Proceedings of the 2020 Conference on Robot Learning*. Ed. by Jens Kober, Fabio Ramos, and Claire Tomlin. Vol. 155. Proceedings of Machine Learning Research. PMLR, 16–18 Nov 2021, pp. 1262–1277.

[26] Jaedeug Choi and Kee-Eung Kim. "Bayesian Nonparametric Feature Construction for Inverse Reinforcement Learning". In: *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*. IJCAI '13. Beijing, China: AAAI Press, 2013, pp. 1287–1293. ISBN: 978-1-57735-633-2.

[27] Jaedeug Choi and Kee-Eung Kim. "Inverse Reinforcement Learning in Partially Observable Environments". In: *J. Mach. Learn. Res.* 12 (2011), pp. 691–730.

[28] Jaedeug Choi and Kee-Eung Kim. "Inverse Reinforcement Learning in Partially Observable Environments". In: *J. Mach. Learn. Res.* 12 (2011), pp. 691–730. ISSN: 1532-4435.

[29] Jaedeug Choi and Kee-Eung Kim. "Nonparametric Bayesian Inverse Reinforcement Learning for Multiple Reward Functions". In: *25th International Conference on Neural Information Processing Systems (NIPS)*. 2012, pp. 305–313.

[30] Anthony Stentz David Silver James Bagnell. "High Performance Outdoor Navigation from Overhead Data using Imitation Learning". In: *Proceedings of Robotics: Science and Systems IV*. Zurich, Switzerland, June 2008.

[31] A. P. Dempster, N. M. Laird, and D. B. Rubin. "Maximum likelihood from incomplete data via the EM algorithm". In: *Journal of the Royal Statistical Society, Series B (Methodological)* 39 (1 1977), pp. 1–38.

[32] Miroslav Dudík, Steven J. Phillips, and Robert E. Schapire. "Performance Guarantees for Regularized Maximum Entropy Density Estimation". In: *Learning Theory*. Ed. by John Shawe-Taylor and Yoram Singer. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 472–486. ISBN: 978-3-540-27819-1.

[33] Brochu Eric, Nando Freitas, and Abhijeet Ghosh. "Active Preference Learning with Discrete Choice Data". In: *Advances in Neural Information Processing Systems*. Ed. by J. Platt et al. Vol. 20. Curran Associates, Inc., 2007. URL: https://proceedings.neurips.cc/paper_files/paper/2007/file/b6a1085a27ab7bff7550f8a3bd017df8-Paper.pdf.

[34] Chelsea Finn, Sergey Levine, and Pieter Abbeel. "Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization". In: *arXiv preprint* arXiv:1603.00448 (2016).

[35] Justin Fu, Katie Luo, and Sergey Levine. "Learning Robust Rewards with Adverserial Inverse Reinforcement Learning". In: *Proceedings of the 6th International Conference on Learning Representations (ICLR)*. 2018. URL: https://openreview.net/forum?id=rkHywl-A-.

[36] Andrew Gelman et al. *Bayesian Data Analysis*. 3rd. CRC Press, 2013.

[37] Brian P. Gerkey, Richard T. Vaughan, and Andrew Howard. "The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems". In: *In Proceedings of the 11th International Conference on Advanced Robotics*. 2003, pp. 317–323.

[38] A. Gleave and O. Habryka. "Multi-task maximum entropy inverse reinforcement learning". In: *arXiv preprint* arXiv:1805.08882 (2018).

[39] Ian Goodfellow et al. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani et al. Vol. 27. Curran Associates, Inc., 2014. URL: https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.

[40] Daniel H Grollman and Aude Billard. "Donut as I do: Learning from failed demonstrations". In: *2011 IEEE International Conference on Robotics and Automation*. 2011, pp. 3804–3809.

[41] W. K. Hastings. "Monte Carlo sampling methods using Markov chains and their applications". In: *Biometrika* 57.1 (Apr. 1970), pp. 97–109.

[42] Michael Herman et al. "Inverse reinforcement learning of behavioral models for online-adapting navigation strategies". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2015, pp. 3215–3222.

[43] Jonathan Ho and Stefano Ermon. "Generative Adversarial Imitation Learning". In: *Advances in Neural Information Processing Systems (NIPS) 29*. 2016, pp. 4565–4573.

[44] Borja Ibarz et al. "Reward learning from human preferences and demonstrations in atari". In: *Advances in neural information processing systems* 31 (2018).

[45] E. T. Jaynes. "Information Theory and Statistical Mechanics". In: *Phys. Rev.* 106 (4 May 1957), pp. 620–630.

[46] Zhuo-jun Jin et al. "Convergence Analysis of an Incremental Approach to Online Inverse Reinforcement Learning". In: *Journal of Zhejiang University - Science C* 12.1 (2010), pp. 17–24.

[47] Glenn Jocher et al. *ultralytics/yolov5: v3.1*. Version v3.1. Oct. 2020.

[48] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. "Reinforcement Learning: A Survey". In: *J. Artif. Int. Res.* 4.1 (May 1996), pp. 237–285. ISSN: 1076-9757.

[49] Parameswaran Kamalaruban et al. "Interactive teaching algorithms for inverse reinforcement learning". In: *arXiv preprint arXiv:1905.11867* (2019).

[50] Beomjoon Kim and Joelle Pineau. "Socially Adaptive Path Planning in Human Environments Using Inverse Reinforcement Learning". In: *International Journal of Social Robotics* 8.1 (2016), pp. 51–66. ISSN: 1875-4805.

[51] Kris M. Kitani et al. "Activity Forecasting". In: *Proceedings of the 12th European Conference on Computer Vision - Volume Part IV*. ECCV'12. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 201–214.

[52] Kris M. Kitani et al. "Activity Forecasting". In: *12th European Conference on Computer Vision - Volume Part IV*. 2012, pp. 201–214.

[53] Jyrki Kivinen and Manfred K Warmuth. "Exponentiated gradient versus gradient descent for linear predictors". In: *information and computation* 132.1 (1997), pp. 1–63.

[54] Henrik Kretzschmar et al. "Socially compliant mobile robot navigation via inverse reinforcement learning". In: *The International Journal of Robotics Research* 35.11 (2016), pp. 1289–1307. ISSN: 0278-3649, 1741-3176. DOI: 10.1177/0278364915619772.

[55] Sergey Levine, Zoran Popović, and Vladlen Koltun. "Nonlinear Inverse Reinforcement Learning with Gaussian Processes". In: *24th International Conference on Neural Information Processing Systems (NIPS)*. 2011, pp. 19–27.

[56] Dong C Liu and Jorge Nocedal. "On the limited memory BFGS method for large scale optimization". In: *Mathematical programming* 45.1 (1989), pp. 503–528.

[57] Andrew William Moore. *Efficient Memory-based Learning for Robot Control*. Tech. rep. University of Cambridge, 1990.

[58] Radford Neal. "Markov Chain Sampling Methods for Dirichlet Process Mixture Models". In: *Journal of Computational and Graphical Statistics* 9.2 (2000).

[59] Gergely Neu and Csaba Szepesvári. "Apprenticeship Learning using Inverse Reinforcement Learning and Gradient Methods". In: *Twenty-Third Conference on Uncertainty in Artificial Intelligence* (2007), pp. 295–302. eprint: 1206.5264.

[60] Andrew Ng and Stuart Russell. "Algorithms for inverse reinforcement learning". In: *Seventeenth International Conference on Machine Learning*. 2000, pp. 663–670.

[61] Andrew Ng and Stuart Russell. "Algorithms for inverse reinforcement learning". In: *Proceedings of the Seventeenth International Conference on Machine Learning* 0 (2000), pp. 663–670. ISSN: 00029645.

[62] Jorge Nocedal and Naoaki Okazaki. *libLBFGS: a library of Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS)*. https://github.com/chokkan/liblbfgs. 2013.

[63] K. B. Petersen and M. S. Pedersen. *The Matrix Cookbook*. Version 20121115. Nov. 2012.

[64] Deepak Ramachandran and Eyal Amir. "Bayesian Inverse Reinforcement Learning". In: *20th International Joint Conference on Artifical Intelligence (IJCAI)*. 2007, pp. 2586–2591.

[65] Nathan D. Ratliff, J. Andrew Bagnell, and Martin A. Zinkevich. "Maximum Margin Planning". In: *Proceedings of the 23rd International Conference on Machine Learning*. ICML '06. Pittsburgh, Pennsylvania, USA: ACM, 2006, pp. 729–736. ISBN: 1-59593-383-2.

[66] Nathan D. Ratliff, J. Andrew Bagnell, and Martin A. Zinkevich. "Maximum Margin Planning". In: *23rd International Conference on Machine Learning*, pp. 729–736.

[67] Nathan D. Ratliff, David Silver, and J. Andrew Bagnell. "Learning to Search: Functional Gradient Techniques for Imitation Learning". In: *Auton. Robots* 27.1 (July 2009), pp. 25–53. ISSN: 0929-5593.

[68] Nathan Ratliff, J. Bagnell, and Martin Zinkevich. "(Online) Subgradient Methods for Structured Prediction". In: *Journal of Machine Learning Research - Proceedings Track* 2 (Jan. 2007), pp. 380–387.

[69] Nicholas Rhinehart and Kris M. Kitani. "First-Person Activity Forecasting with Online Inverse Reinforcement Learning". In: *International Conference on Computer Vision (ICCV)*. 2017.

[70] Stuart Russell. "Learning Agents for Uncertain Environments (Extended Abstract)". In: *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*. COLT' 98. Madison, Wisconsin, USA: ACM, 1998, pp. 101–103. ISBN: 1-58113-057-0.

[71] Stuart Russell. "Learning Agents for Uncertain Environments (Extended Abstract)". In: *Eleventh Annual Conference on Computational Learning Theory*. 1998, pp. 101–103.

[72] Dorsa Sadigh et al. "Active Preference-Based Learning of Reward Functions". In: *Robotics: Science and Systems*. 2017.

[73] Arora Saurabh, Prashant Doshi, and Bikramjit Banerjee. "A Framework and Method for Online Inverse Reinforcement Learning". In: *arXiv preprint* arXiv:1805.07871v1 (2018).

[74] Shervin Shahryari and Prashant Doshi. "Inverse Reinforcement Learning Under Noisy Observations". In: AAMAS '17 (2017).

[75] Kyriacos Shiarlis, Joao Messias, and Shimon Whiteson. "Inverse Reinforcement Learning from Failure". In: *Proceedings of the 2016 International Conference on Autonomous Agents and Multiagent Systems*. AAMAS '16. Singapore, Singapore: International Foundation for Autonomous Agents and Multiagent Systems, 2016, pp. 1060–1068. ISBN: 978-1-4503-4239-1.

[76] Nihal Soans et al. "SA-Net: Robust State-Action Recognition for Learning from Observations". In: *2020 IEEE International Conference on Robotics and Automation, ICRA*. 2020, pp. 2153–2159.

[77] Jacob Steinhardt and Percy Liang. "Adaptivity and Optimism: An Improved Exponentiated Gradient Algorithm". In: *31st International Conference on Machine Learning*. 2014, pp. 1593–1601.

[78] Prasanth Sengadu Suresh and Prashant Doshi. "Marginal MAP estimation for inverse RL under occlusion with observer noise". In: *Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence*. Vol. 180. 2022, pp. 1907–1916.

[79] Tomer D. Ullman et al. "Help or Hinder: Bayesian Models of Social Goal Inference". In: *Proceedings of the 22Nd International Conference on Neural Information Processing Systems*. NIPS'09. Vancouver, British Columbia, Canada: Curran Associates Inc., 2009, pp. 1874–1882. ISBN: 978-1-61567-911-9.

[80] Adam Vogel et al. “Improving Hybrid Vehicle Fuel Efficiency Using Inverse Reinforcement Learning”. In: 2012.

[81] Monica C Vroman. “MAXIMUM LIKELIHOOD INVERSE REINFORCEMENT LEARNING”. PhD thesis. Rutgers, The State University of New Jersey, 2014.

[82] G. R. (Gordon Raymond) Walsh. *Methods of optimization / G. R. Walsh.* eng. London ; Wiley, 1975, pp. 39–44. ISBN: 0471919225.

[83] Shaojun Wang and Dale Schuurmans Yunxin Zhao. “The Latent Maximum Entropy Principle”. In: *ACM Transactions on Knowledge Discovery from Data* 6.8 (2012).

[84] Shaojun Wang et al. “The Latent Maximum Entropy Principle”. In: *IEEE International Symposium on Information Theory*. 2002, pp. 131–131.

[85] Markus Wulfmeier and Ingmar Posner. “Maximum Entropy Deep Inverse Reinforcement Learning”. In: *arXiv preprint* (2015).

[86] Jiangchuan Zheng, Siyuan Liu, and Lionel M. Ni. “Robust Bayesian Inverse Reinforcement Learning with Sparse Behavior Noise”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 28.1 (June 2014).

[87] Brian D. Ziebart et al. “Maximum Entropy Inverse Reinforcement Learning”. In: *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*. AAAI’08. Chicago, Illinois: AAAI Press, 2008, pp. 1433–1438. ISBN: 978-1-57735-368-3.

[88] Brian D. Ziebart et al. “Maximum Entropy Inverse Reinforcement Learning”. In: *23rd National Conference on Artificial Intelligence - Volume 3*. 2008, pp. 1433–1438.

[89] Brian D. Ziebart et al. “Navigate Like a Cabbie: Probabilistic Reasoning from Observed Context-aware Behavior”. In: *Proceedings of the 10th International Conference on Ubiquitous Computing*. UbiComp ’08. Seoul, Korea: ACM, 2008, pp. 322–331. ISBN: 978-1-60558-136-1.

[90] Brian D. Ziebart et al. “Planning-based Prediction for Pedestrians”. In: *Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IROS’09. St. Louis, MO, USA: IEEE Press, 2009, pp. 3931–3936. ISBN: 978-1-4244-3803-7.