Inverse reinforcement learning under noisy observations

(Robust-IRL)

by

Shervin Shahryari

(Under the Direction of Prashant Doshi)

Abstract

We consider the problem of performing inverse reinforcement learning when the trajectory of the expert is not directly observable from learner's view. Instead, a noisy observation of the trajectory is provided for the learner. This problem exhibits wide-ranging applications and the specific application we consider here is the scenario in which the learner tries to penetrate a perimeter patrolled by a robot. Since the learner is hidden in a secret location, it cannot observe the patroller. Therefore it does not have access to the expert's trajectory. Instead, the learner can listen to the sound of the expert's movement and estimate its state and action using an observation model. We treat the expert's state and action as hidden data and present an algorithm based on expectation maximization and maximum entropy frameworks to solve the non-linear, non-convex problem. Previous work in this area only considers the state of the robot as hidden data and uses likelihood maximization of the observations. In contrast, our technique takes expectations over both state and action of the expert, enabling learning even in the presence of extreme noise and broader applications.

Index words: Robotics, Machine learning, Markov decision process, Expectation maximization, Optimization, Gibbs sampling

Inverse reinforcement learning under noisy observation

(Robust-IRL)

by

Shervin Shahryari

B.Sc., K.N.Toosi University of Technology, 2010

A Dissertation Submitted to the Graduate Faculty

of The University of Georgia in Partial Fulfillment

of the

Requirements for the Degree

Master of Science

Athens, Georgia

2016

Inverse reinforcement learning under noisy observation

(Robust-IRL)

by

Shervin Shahryari

Approved:

Major Professors: Prashant Doshi

Committee: Frederick Maier

Walter D. Potter

Electronic Version Approved:

Suzanne Barbour

Dean of the Graduate School

The University of Georgia

December 2016

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Definition

Robotics is a multi-disciplinary research area that is the intersection of mechanical engineering, electrical engineering, and computer science. Mechanical and electrical engineers design and manufacture the robot's structure and hardware while computer scientists bring these structures to life through programming. It is possible for an autonomous robot to find itself in a situation that is not considered by its designer. Moreover, sometimes limitation of actuators and sensors makes it difficult to prepare a robot for all possible situation. These problems are exacerbated in cases where the environment is unpredictable or dynamic, for instance, search and rescue, autonomous vehicles, or unmanned aerial vehicle. Robot learning is a branch of robotics that studies techniques allowing a robot to hone skills or adapt to the surrounding environment utilizing different learning algorithms.

An autonomous robot can operate in an environment by producing or learning an optimal policy, which is a mapping from every possible state into the action that maximizes a notion of utility. The problem of learning an optimal policy is the core of many robotics applications. Due to the volatile and unpredictable nature of most environments, the implementation

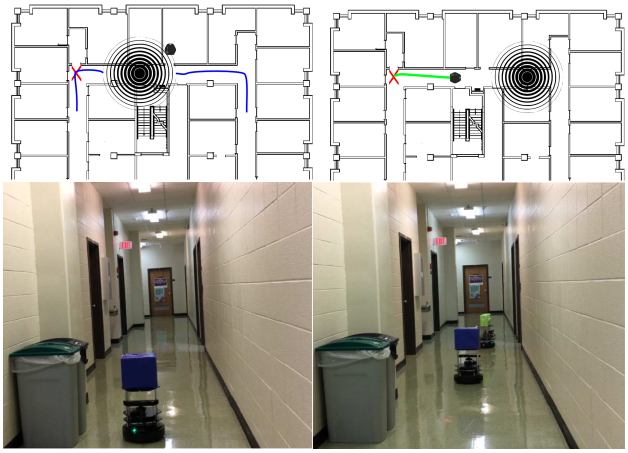


Figure 1.1: The top two images are our experiment in the simulation. We used ROS to run the simulation. The patroller securing the hallway and concentric circles around the patroller is an indication of the magnitude of the sound intensity. As the distance between the patroller and the intruder increases the intensity decreases. The lower images are from the physical experiment done with two Turtlebots. The two images on the right-hand side shows the moment that the intruder attacks and the two images on the left-hand side show the observation time.

of optimal policy by hand is often very meticulous and challenging task. Consequentially machine learning techniques have been utilized to implement the optimal policy. One way of deriving the optimal policy is first to construct or learn the reward function then use value iteration to find the best policy. Inverse reinforcement learning (IRL) technique try to exact the reward function of another robot ("expert") given an observed behavior.

Inverse reinforcement problems usually model the expert as a Markov decision process (MDP) whose solution results into the experts rewards function. Moreover, it is always assumed that the expert has full observability of the experts MDP. Therefore, it can observe experts state and action at any time. Autonomous robots are heavily dependent on their sensors and actuators to gather information about the world and make a decision. These sensors and actuators are often limited in range and quality. Sensors rarely produce noise-free information, for instance, sonar sensors cannot be used unfiltered or GPS sensors always

have a margin of error. To mitigate this problem we may assume the presence of noise in the sensory data. Consequently, we incorporate an observation model to inverse reinforcement learning formulation to help it to deal with sensory noise.

1.2 Contributions

This thesis makes the following contributions:

1. We generalize IRL to operate under a situation in which the observation of the learner from the expert has a considerable amount of noise.

2. We utilize EM to operate in the context of IRL when the true trajectory of the expert is hidden from the learner.

3. We incorporate an observation model into IRL, which enables the learner to fuse data from different sensors with different level of noise.

We perform two simulation experiments and an experiment involving two physical robots similar to what is shown in Fig. 1.1. Our comprehensive experimentation demonstrates the power of our method (Robust-IRL) in dealing with noise in observation.

1.3 Structure of the Thesis

The rest of this thesis is structured as follows. In Chapter 2 and 3 we introduce relevant background and elaborate on related works needed to understand the rest of this thesis. In Chapter 4 we introduce a novel observation model for IRL framework and generalize maximum entropy IRL in the context of noisy observation. We present our experiment settings and the results in Chapter 5. Finally, we conclude this study in Chapter 6.

Chapter 2

Background

Robust-IRL is developed based on existing state of the art maximum entropy inverse reinforcement learning technique and in this chapter, we review this precursor work as well as reinforcement learning, the Markov decision process, Gibbs sampling, and expectation maximization framework. Evaluating and competing different IRL methods require a comprehensive measurement that shows the distinction between methods under different scenarios. Therefore, we present a background on these measurements and briefly consider their capabilities and requirements.

2.1 Markov Decision Processes

Decision-making is the process of choosing actions based on the preferences of the decision-maker. A major part of this process involves analyzing a potentially infinite set of states and actions in terms of some preference principles. Considering all preference principles simultaneously the decision-maker(s) chooses an action that is most desirable for him. Due to the stochastic nature of the world, we need a framework that makes the decision-maker able to deal with situations that are out of its control or partially random. Markov decision

processes (MDPs) provide a mathematical framework for modeling situations of this kind [23]. MDPs are advantageous for investigating a wide range of optimization problems such as reinforcement learning and inverse reinforcement learning utilizing dynamic programming and different optimization techniques.

Mathematically an MDP is defined by as a 5-tuple $< S, A, T, R, \gamma >$, where:

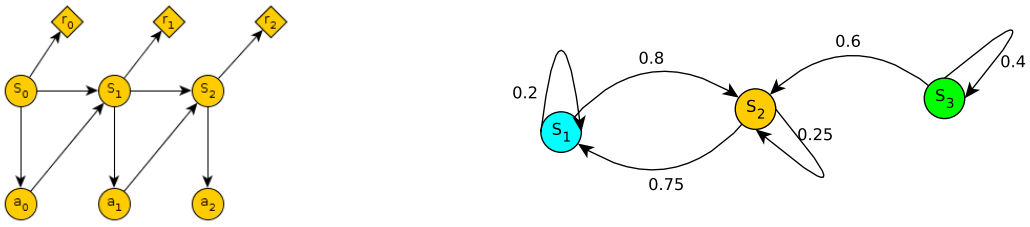
- S is a potentially infinite set of states. However, in this thesis we assume a finite set.
- A is a potentially infinite set of actions. However, in this thesis we assume a finite set.
- $T(s_{t+1} = s' | s_t = s, a_t = a)$ is the probability that executing action a at state s resulting to state s'.
- $R_a(s, s')$ is the expected immediate reward after executing action a in state s and transitioning to state s'.
- $\gamma \in [0, 1)$ is the discount factor that determines the balance between future reward and immediate reward.

The core promise of MDPs is developing the optimal policy π^* for the decision-maker. π^* is a function that maps each state to an action or distribution over actions considering the fact that reward of the decision-maker must be maximized over time. To find the optimum policy one needs to solve the Bellman equation:

$$U(s) = R(s) + \gamma \max_a T(s'|s, a) U(s') \tag{2.1}$$

Eq. 2.1 can be solved utilizing dynamic programming through value iteration.

$$\begin{aligned} V_0(s) &= \max_a R(s, a) \\ V_t(s) &= \max_a R(s, a) + \gamma T(s, a, s') V_{t-1}(s') \end{aligned} \tag{2.2}$$



(a) (b)

Figure 2.1: (a) An illusteration of a MDP considering 3 horizons. s_i, a_i and r_i shows the state, action and reward of the agent at time step i. (b) A simple Markov chain with 3 state and 2 actions. The number on each edge indicates the probability of the transition.

Where t indicates the time-step (horizon). Eq. 2.2 provides the expected value for all states at the horizon t. It can be solved recursively in two manners: first, iterate for a finite number of horizons. Second, iterate until the change in the value of all states is less than some threshold ϵ, which provides an infinite horizon solution. For each MDP there are two key assumptions that must be satisfied. The environment must be fully observable for the agent. It means that at each time-step the agent should be fully aware of its own state and action. Moreover, MDPs assume that the decision-making process can be modeled as a Markov chain. Markov chain is a process that satisfies the Markov property; for predicting the future of the process, the agent only needs knowledge about the present state rather than knowing the process's full history. Eq. 2.3 shows these assumptions in a mathematical manner. Fig. 2.1a and Fig. 2.1b illustrate an MDP and a Markov chain, respectively.

$$\begin{aligned} Pr(S_{t+1}|S_{1:t}, A_{1:t}) &= Pr(S_{t+1}|S_t, A_t) \\ Pr(A_t|S_{1:t}, A_{1:t1}) &= Pr(A_t|S_t) \end{aligned} \tag{2.3}$$

2.2 Reinforcement Learning (RL)

Reinforcement learning is a sub-area of machine learning inspired by neuroscience, statistics, psychology, and computer science. It promises to find the optimal policy for an agent that maximize some notion of utility without the meticulous task of specifying the details on how to achieve the goal [13]. Because of its generality, this technique has been utilized in many domains such as game theory, control theory, multi-agent systems, finance, simulation-based optimization, entertainment, robotics and much more. Game theory utilizes reinforcement learning to explain how equilibrium may arise under bounded rationality. In finance, reinforcement learning may be used to develop an autonomous agent that can efficiently trade stocks with minimal human intervention [16]. Recently, the state of the art algorithm deep inverse reinforcement learning developed an agent that can play console games without explaining the details of the game to the agent. The developed agent easily outperformed human experts in some games [19].

In reinforcement learning the agent interacts with its environment in discrete time-steps (decision epochs). At each decision epoch t, the agent receives an observation and the immediate reward r_t. The agent then executes an action a. The environment evolves to a new state s_{t+1}, and this cycle continues. The agent's goal is to maximize the reward received. The agent can choose its action according to the history or it can randomize its action selection. Fig. 2.2 illustrates the idea of reinforcement learning.

As we stated in Sec.2.1, if the 5-tuple $< S, A, T, R, \gamma >$ is known to the agent, the agent can find the optimal policy through value iteration. However, when we target large MDPs or if the probabilities or rewards are unknown to the agent, it is not possible to use value iteration in the aforementioned manner to find the optimal policy. For example, if the space of the states and actions are large enough, the value iteration may not converge in a resealable time. Instead, we can use RL to find the optimal policy. There are lots of methods to solve

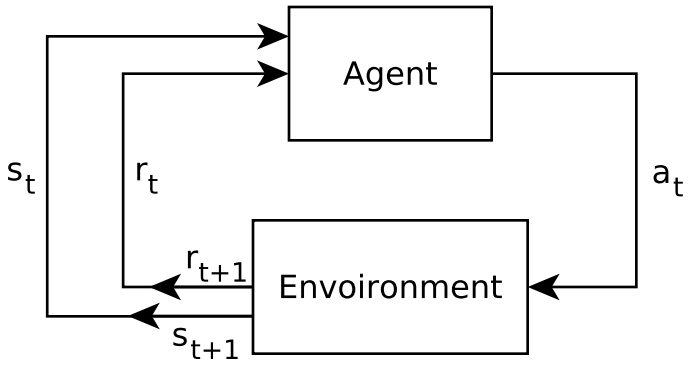


Figure 2.2: The agent execute action a_t according to current state s_t and current reward r_t. after that the environment transition and provides the agent with the new state s_{t+1} and the new reward r_{t+1}.

reinforcement learning problems. One way of solving these type of problems is to define a function which corresponds to taking the action a at state s and then continue either according to current policy (exploitation) or randomly (exploration).

$$Q(s,a) = \sum_{s'} T(s'|s,a)(R_a(s,s') + \gamma V(s')) \tag{2.4}$$

Where γ is a real number between 0 and 1. The function in Eq. 2.4 is known as Q-function. Although this function is also unknown, it can be learned. One can put the following interpretation on Eq. 2.4: "I was at state s and choose to take action a, then the outcome was state s' with reward $R_a(s,s')$". Therefore the agent can use the experience to update an array of Q values. This method of solving reinforcement learning problems is known as Q-learning [28]. Reinforcement learning differs from standard supervised learning in the sense that there is no correct input and output pair presented to the agent.

2.3 Inverse Reinforcement Learning (IRL)

As the name suggests, inverse reinforcement learning (IRL) problem is the inverse of the RL problem. In IRL an agent (learner) seeks to learn the motivation of another agent (expert) which is assumed to be modeled as an MDP and it performs the optimal policy. IRL is a branch of Learning from Demonstration (LfD) or imitation learning, where the reward function of an expert is learned through examples, and the objective of the learner is to reproduce the demonstrated behavior [8] [2] [21]. Current IRL methods assume the presence of a single expert that solves an MDP. Moreover, they assume except the reward function the MDP is fully known and observable by the learner [8]. In other words the learner has access to an incomplete MDP of the expert ($< S, A, T, \gamma >$) that misses the reward function $R_a(s, s')$ [21].

There is an another critical assumption. Due to the large size of the space of the possible reward function, it is common practice to represent the reward function as a linear combination of $K > 0$ binary features. $R_E(s, a) = \Sigma_1^K \theta_k \phi_k(s, a)$. θ_k are weights and $\phi_k(s, a) \rightarrow \{0, 1\}$ is a binary feature function that maps a pair of state and action to either 0 (not activated) or 1 (activated) [1]. The expert solves the MDP given its reward function and derives the optimal policy, then it executes that policy several times. The learner observes the expert and constrains itself to find a reward function that matches the feature expectation of the behavior shown by the expert [1].

However, unfortunately, IRL is an ill-posed problem [21]. there is more than one reward function that can explain the expert's behavior. All proposed IRL methods try to mitigate this issue. The first solution was proposed by Ng and Russell [21]. They proposed to formulate the problem as a linear program which maximizes the margin of value between the expert's action in each state and all other possible actions in that state. They also suggested a penalty term which encourages the use of smaller magnitude rewards. Then we can solve this inverse

reinforcement learning formulation as a linear program and find the reward function.

Later works on IRL utilizes other approaches to mitigate this ill-posed problem [20] [24] [5] [25] [17]. The state of the art IRL algorithm Max-Ent IRL make use of maximum entropy principle [31].

2.4 Max-Ent IRL

As we stated in sec. 2.3 inverse reinforcement learning is an ill-posed problem; it means that there are more than one reward function that can make sense of the observed behavior of the expert. State of the art algorithm max-Ent IRL, proposed by Ziebart *et al.* [31], makes use of maximum entropy principle to mitigate this ill-posed problem. It chooses a distribution over trajectories which has the maximum entropy. Doing so results in recovering a reward function with the least extra assumptions made.

Let the trajectory T with an arbitrary length of L be a sequence of state and action pairs, $T = (< s, a >^0, < s, a >^1, ..., < s, a >^L)$, where s and a belong to sets of possible states and actions of the expert, respectively, and T belongs to a set of observed trajectories τ. Also let $\mathcal{T}$ be the finite set of all possible trajectories of length L; $\tau \subseteq \mathcal{T}$.

In addition to matching feature expectation of the expert, Max-Ent IRL consider the distribution over trajectories that has the maximum entropy among all other distributions. Mathematically this problem can be formulated as a convex nonlinear optimization problem [31].

$$
\begin{aligned}
&\max_{\Delta}(-\sum_{T\in\mathcal{T}} Pr(T)\log Pr(T)) \\
&\textit{subjected to} \\
&\sum_{T\in\mathcal{T}} Pr(T) = 1 \\
&\sum_{T\in\mathcal{T}} Pr(T) \sum_{<s,a>\in T} \phi_k(s,a) = \hat{\phi}_k
\end{aligned}
\tag{2.5}
$$

Where Δ is the space of all possible $Pr(T)$. In order to solve this optimization problem we can apply Lagrangian relaxation to bring both constrains into the objective function and then solve the dual utilizing exponentiated gradient descent.

$$
\begin{aligned}
\mathcal{L}(Pr(T),\theta,\eta) = &-\sum_{T\in\mathcal{T}} Pr(T)\log Pr(T) + \sum_k \theta_k \\
&(\sum_{T\in\mathcal{T}} Pr(T) \sum_{<s,a>\in T} \phi_k(s,a) - \hat{\phi}_k) \\
&+ \eta(\sum_{T\in\mathcal{T}} Pr(T) - 1)
\end{aligned}
\tag{2.6}
$$

Now we take the partial derivative with respect to $Pr(T)$ and set it to zero to find the optimal value:

$$
\frac{\partial\mathcal{L}}{\partial Pr(T)} = -\log Pr(T) - 1 + \sum_k \theta_k \sum_{<s,a>\in T} \phi_k(s,a) + \eta = 0 \tag{2.7}
$$

Solving Eq. 2.7 for $Pr(T)$ we have:

$$
Pr(T) = \frac{e^{\sum_k \theta_k \sum_{<s,a>\in T} \phi_k(s,a)}}{n(\theta)} \tag{2.8}
$$

Where $n(\theta)$ is the normalizing factor. By plugging Eq. 2.8 into Eq. 2.6 we arrive at Eq. 2.9:

$$
\mathcal{L}^{dual}(\theta) = \log(n(\theta)) - \sum_k \theta_k \hat{\phi}_k \tag{2.9}
$$

Eq. 2.9 is the dual program, which can be solved by using the exponentiated gradient descent to find the optimal values of θ. Eq. 2.10 shows the gradient.

$$\nabla \mathcal{L}^{dual}(\theta) = \sum_{T \in \mathcal{T}} Pr(T) \sum_{<s,a> \in T} \phi_k(s,a) - \hat{\phi}_k \tag{2.10}$$

As it shown above the calculation of the gradient involves summing over the set of all possible trajectories that may be intractable in most of problems. However Ziebart *et al.* proposed an efficient approach that calculates the expected edge frequency (state visitation frequency) [31].

2.5 Gibbs Sampling

Gibbs sampling was established by Stuart Geman and Donald Geman in 1984 and named after the physicist Josiah Willard Gibbs [9]. Gibbs sampling [12] is a Markov chain Monte Carlo (MCMC) [10] algorithm that approximates a probability distribution. Gibbs sampling is useful when we cannot calculate the probability distribution exactly. For instance Gibbs sampling can approximate a joint distribution, a marginal distribution, a summation, expected value of a variable and more.

When calculating the joint distribution is difficult but it is easier to calculate the conditional distribution of each random variable, we can use the Gibbs sampling. The Gibbs sampling algorithm approximate a joint distribution by drawing samples from the distribution of each variable conditioned on the current values of the other random variables. In other words, in Gibbs sampling we need to calculate the conditional distributions and sample them exactly. However, when it is not possible to do so we can utilize other variation of the Gibbs sampling [18] [11].

2.6 Expectation Maximization

The Expectation maximization (EM) framework exists in various forms since at least the early 1960s. However, it was formalized by Arthur Dempster, Nan Laird, and Donald Rubin in 1977 [7]. They generalized the method and proposed a convergence condition for a wider class of problems. Despite the fact that their paper established the EM framework as an important tool of statistical analysis to deal with hidden data the convergence condition was flawed. However, Wu published a correct convergence condition in 1983 [29].

EM algorithm is an iterative algorithm that finds maximum likelihood estimation of parameters for statistical models when the problem cannot be solved directly due to the presence of hidden data. The EM algorithm alternates between two steps. First, an expectation step that called E-step. In this step the EM algorithm uses the current estimation of the parameters and calculates expectation of the log-likelihood. After that it performs the maximization step called M-step. In the M-step, the EM algorithm updates the parameters such that they maximize the expected log-likelihood found on the E-step, and construct the distribution over the latent variables for the next E-step. Fig. 2.3 illustrate the EM algorithm.

Algorithm 1: EM algorithm

1: Initialize parameter θ randomly
2: **while** not converged **do**
3: Compute the best value for the hidden data Z given the current parameter values θ_t.
4: Use values of hidden date Z from previous step to update parameters value θ_{t+1}
5: **end while**
6: **return** θ

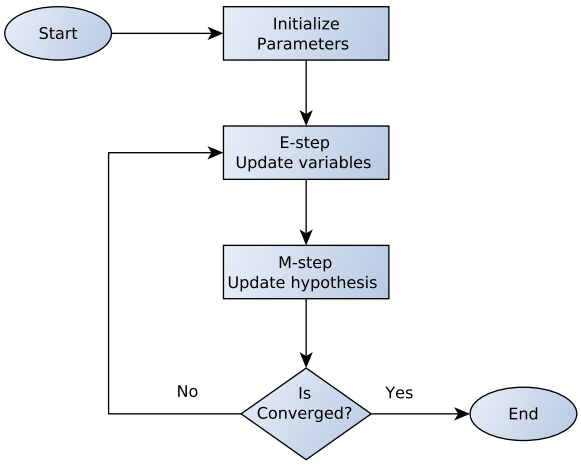


Figure 2.3: This flowchart illusterates the EM framework

2.7 Evaluation of IRL methods

Given different algorithms to solve inverse reinforcement learning problems we need an evaluation method to compare these techniques. Since the goal of inverse reinforcement learning is to recover the reward function of the expert, one might be tempted to directly compare reward functions. However, it is possible that two drastically different reward functions return similar optimum policies. Instead, Choi and Kim [6] propose to solve the experts MDP with both the true reward function and the learned reward function separately then calculate the difference between the value functions of obtained optimal policies; this difference is known as inverse learned error (ILE). Let π^L be the policy produced by the learned reward function, and π^E be the optimal policy produced by the true reward function. Then one can calculate the ILE by using Eq. 2.11

$$ILE = ||V^{\pi^L} - V^{\pi^E}|| \tag{2.11}$$

where V^{π^L} is the value function calculated by utilizing policy π^L on the expert's MDP, and V^{π^E} is the value function of the expert's MDP, when we are utilizing policy π^E .

Chapter 3

Related Work

In this chapter, we elaborate on more advanced and generalized IRL techniques, which are not covered in previous chapters.

3.1 IRL, Multiple Experts and Partial Observability

As we mentioned in Sec. 2.3 inverse reinforcement learning originally was introduced as a framework for single agent settings. However, Bogert and Doshi [3] extended the inverse reinforcement learning framework to suit multi-agent settings. They assumed the presence of multiple experts instead of a single expert [3]. One solution would be to model multiple experts with a joint MDP, where joint state includes the local state of each expert. The action space should be defined as the Cartesian product of the set of actions of each expert. The transition function of this joint MDP gives a distribution over the next joint state given the current joint state and the current joint action. Also, the reward function of this joint MDP gives the immediate reward given the current joint state and the current joint action. Although this straightforward approach considers all interactions between experts, it suffers from a major drawback which prevents it from being practical [3].

As one may expect, in most problems the joint state space and joint action space are large. Consequentially the transition function becomes large as well, which significantly increases the size of the MDP and the time needed for solving it exactly.

Instead of modeling the experts with a joint MDP, Bogert and Doshi modeled the experts with separate MDP. This helps them solve problems with large state and action space but leaves out the interaction between experts. To overcome this problem they propose a novel approach to model the interaction between experts through a game [3].

Moreover, they assumed that some significant portion of the experts' trajectory is occluded from the learner. This is a realistic assumption, especially in robotic domains, when the learner has minimal or no control over the environment. To find the experts' reward function in the presence of occlusion, they limited the optimization over the observable portion of the trajectory and project feature expectations from the observable portion of the trajectory to occluded portions [3].

On the other hand, Choi and Kim [6] also try to relax the assumption of the full observability of the environment;they did this from the expert's perspective [6]. They assumed that the expert cannot be modeled as an MDP due to the partial observability of the environment. They proposed a novel approach to reformulate IRL problem while modeling the expert as a POMDP instead of an MDP.

3.2 Hidden Data IRL

As we mentioned Bogert and Doshi proposed a novel algorithm to deal with partially occluded trajectories. However, their method is not capable of handling dynamic occlusion. Later they proposed to treat the occluded portion of the trajectory as hidden data and the remaining as observed data [4]. Then they proposed a reformulation of the Max-Ent IRL in an expectation maximization frameworks. This new technique recovers a distribution over the hidden data,

which makes the learner enable to learn in presence of dynamic occlusion [4].

This generalized IRL technique that operates in presence of hidden data is suitable for a variety of robotics domains especially when the learner has little control over the environment. In this new technique, they recover a distribution over hidden features from the portions of the trajectory that are visible.

To evaluate their method they test it on a ball sorting domain, in which a robot is tasked with learning to sort 6 different types of balls by observing a human expert.

Despite the presence of a camera system that observes all the movements, a key factor in performing the task successfully remains unobserved. This is the amount of pressure that is applied by the expert to each ball, which is enough to not damage the ball and not cause the ball to drop. The proposed method by Bogert and Doshi successfully learned the task and it had a performance comparable to the scenario in which the amount of pressure is also provided for the learner.

3.3 Related Studies

Although all the previous studies mentioned in Sec. 3.1 and Sec. 3.2 make IRL more general and extend it to situations where occlusion is inevitable, these techniques are not able to deal with situations when the observation of the learner from the expert is noisy. However, Kitani *et al.* [14] proposed a new technique which can learn human activity by receiving noisy information from a tracking algorithm. This technique is able to make use of noisy observations where observations only depend on the state of the expert.

Chapter 4

Robust Inverse Reinforcement Learning

In this chapter, we discuss our novel approach for solving inverse reinforcement learning problems when the learner's observation is noisy. First we start by adopting the hidden Markov decision process (hMDP) introduced by Kitani *at el.* [14]. Then in Sec. 4.2, we introduce a novel observation model to handle noise. Moreover, we reformulate the maximum entropy inverse reinforcement learning technique for our setting in Sec. 4.3. After that, in Sec. 4.4 we incorporate Expectation maximization framework to maximum entropy principle to solve the program. At the end, we go over the algorithm of the robust-IRL.

4.1 Hidden Markov Decision Process

Motivated by the application of utilizing noisy sensory data, we consider a setting where the learner receives the sound intensity of the expert's movement sound as observations instead of receiving expert's state and action.

As we mentioned above the expert solves a Markov decision process to construct its

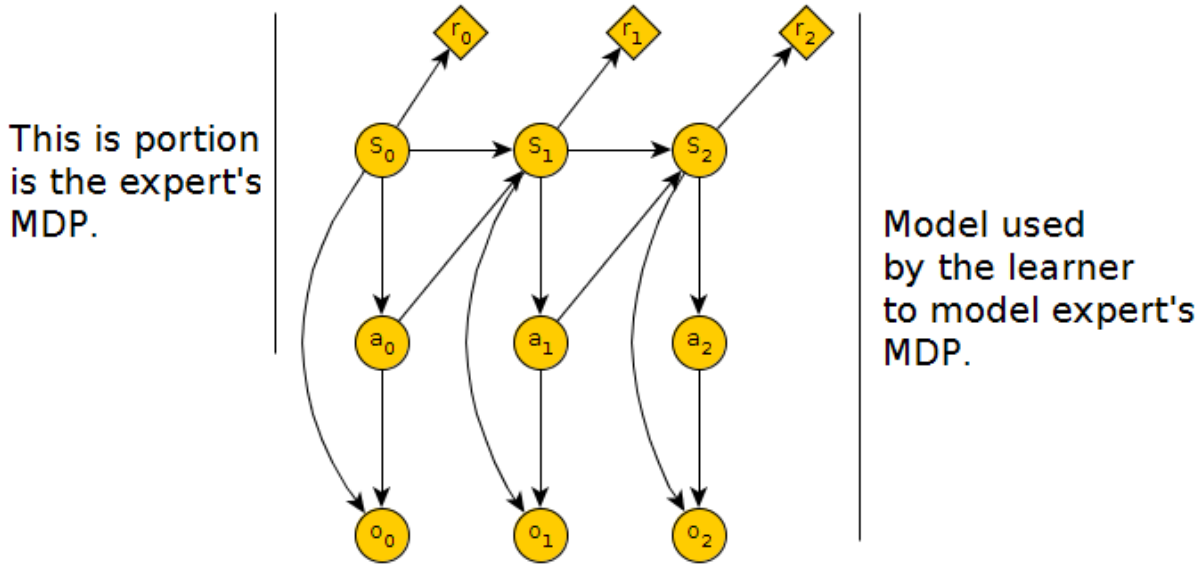


Figure 4.1: In hMDP the state and action are hidden from the learner but an observation of state and action is provided at each time step

policy. However, since the learner cannot observe the expert's state and action, it cannot model the expert as an MDP. Kitani *et al.* propose a hidden Markov decision processes framework to resolve this issue [14]. We adopt this framework to model the expert from learner's perspective. Unlike the hMDP proposed by Kitani *et al.* our adoption of hMDP incorporates actions into the observation model. Figure 4.1 illustrates our proposed hMDP.

4.2 Observation Model

As one may notice the observation model plays a crucial role in robust inverse reinforcement learning. Incorporating actions into the observation model introduces some challenges, which arises from considering observations in discrete time. Using sound intensity makes it easy

to infer the state, however, we must have two consecutive sound intensity in order to infer what the action was. Therefore the Markovian assumption would not be true anymore. This problem arises from the fact that observations are considered in discrete time steps. We can handle this issue in the following manner.

Sound intensity is inversely proportional to the square distance of the sound source to the listener's location , $I = \frac{k}{R^2}$, where k is a constant. In the proposed hMDP, state and action happen in discrete time steps (decision epochs), however observation evolves in continuous time within each decision epoch. Since the observation evolves in continuous time, we can represent an observation at each decision epoch by a function $f(t)$, where t represents continuous time.

Theorem 1. *Let $k,\ a,\ b$ and c be constants then the structure of $f(t)$ is as follows:*

$$f(t) = \frac{k}{at^2 + bt + c} \tag{4.1}$$

Proof. Suppose that the expert moves from one point to another point. If:

First point coordinate = (x_0, y_0)

Second point coordinate = (x, y)

Velocity along the x axis = v_x

Velocity along the y axis = v_y

Time at the first point = t_0

Time at the second point = t

Then:

$$x = v_x(t - t_0) + x_0 = v_x t + (x_0 - v_x t_0)\ y = v_y(t - t_0) + y_0 = v_y t + (y_0 - v_y t_0)$$

$$r^2 = (x - x_0)^2 + (y - y_0)^2\ r^2 = (v_x t + (x_0 - v_x t_0) - x_0)^2 + (v_y t + (y_0 - v_y t_0) - y_0)^2$$

$$r^2 = (v_x t - v_x t_0)^2 + (v_y t - v_y t_0)^2\ r^2 = v_x^2 t^2 + v_x^2 t_0^2 - 2v_x^2 t_0 t + v_y^2 t^2 + v_y^2 t_0^2 - 2v_y^2 t_0 t$$

$$r^2 = (v_x^2 + v_y^2)t^2 + (-2v_x^2 t_0 - 2v_y^2 t_0)t + (v_x^2 t_0^2 + v_y^2 t_0^2)\ r^2 = at^2 + bt + c \text{ where: } a = v_x^2 + v_y^2$$

$$b = -2v_x^2 t_0 - 2v_y^2 t_0 \; c = v_x^2 t_0^2 + v_y^2 t_0^2 \; I = \frac{k}{r^2} \; I = \frac{k}{at^2+bt+c}$$

□

Modeling observation in continuous time has advantages over other approaches. Imagine a scenario in which the time of the observation is stochastic; in this scenario, the length of the decision epoch serves as bound for the observation time. The learner receives some samples in the decision epoch and uses regression to find the function that explains observation over continuous time. Consequentially, this approach recovers information about the time that there is no sample provided for the learner.

Figure 4.2 illustrates how we can recover information by extrapolating or interpolating when there is not much sample to cover the entire decision epoch.

4.3 Robust IRL Program

In inverse reinforcement learning the true trajectory of length L is $T = (< s, a >^0, < s, a >^1 , ..., < s, a >^L)$. We consider cases where this trajectory is not provided for the learner; instead, a sequence of observation is provided from the learner. Lets $\vec{\omega}$ be a sequence of observations of length M, $\vec{\omega} = (o^0, o^1, ..., o^M)$, and lets consider the sequence of observations $\vec{\omega}$ as Y, the observed data, and the true trajectory T as Z, hidden data. In other words, $X = (Y \cup Z)$, where X is the total data.

One can simply utilize the observation model $Pr(o^i| < s, a >)$ to calculate the most likely state action pair at time step i. However, this approach totally disregards the effect of the transition function and the policy of the expert in constructing the expert's trajectory. In contrast, we propose a revised formulation of maximum entropy inverse reinforcement learning that allows an expectation over trajectories (hidden data) given a sequence of observations. This method allows considering the effect of the transition function and expert's policy in

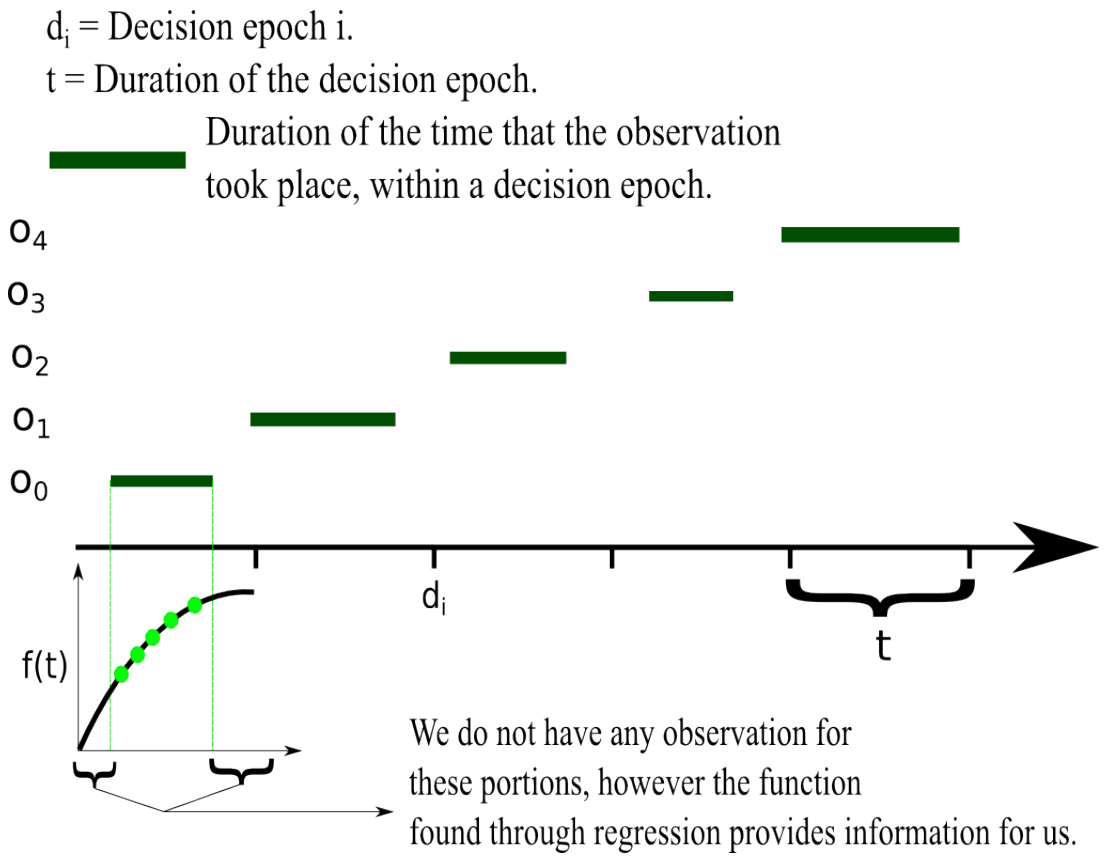


Figure 4.2: Illustration of f(t). d_i shows the end of each decision epoch i. t is the continues time and each green bar shows potion of the decision epoch which the sound intensity samples are provided.

constructing a distribution over possible trajectories.

$$\begin{aligned} &\max_{\Delta}(-\sum_{\vec{\omega},T} Pr(\vec{\omega},T)log(Pr(\vec{\omega},T)) \\ &subjected\ to \\ &\sum_{\vec{\omega},T} Pr(\vec{\omega},T) = 1 \\ &\sum_{\vec{\omega}\in\Omega}\sum_{T\in\tau} Pr(\vec{\omega},T) \sum_{(s,a)\in T} \phi_k(s,a) = \hat{\phi}_k \end{aligned} \tag{4.2}$$

where:

$$Pr(\vec{\omega}, T) = Pr(\vec{\omega}|T)Pr(T) \tag{4.3}$$

$$Pr(T) = Pr(s_0)\prod_{i=1}^{n-1} Pr(s_{i+1}|s_i, a_i)Pr(a_i|s_i) \tag{4.4}$$

$$\hat{\phi_k} = \frac{1}{|\tilde{\Omega}|}\sum_{\vec{\omega}\in\tilde{\Omega}}\sum_{T} Pr(T|\vec{\omega}) \sum_{(s,a)\in T} \phi_k(s,a) \tag{4.5}$$

$$Pr(T|\vec{\omega}) = \eta Pr(\vec{\omega}|T)Pr(T) \tag{4.6}$$

Here, Δ is the space of all distributions $Pr(\vec{\omega}, T)$ and Ω is the set of sequence of observations from the learner's perspective. In other words, $\vec{\omega} \in \Omega$.

Given above, we can apply Lagrangian relaxation to bring both constraints into the objective function, however, because of the presence of conditional probability in the Lagrangian $\mathcal{L}(Pr(\vec{\omega}, T); \theta; \eta)$, the relaxed objective function is non-convex.

$$\begin{aligned}
&\frac{\partial \mathcal{L}(Pr(X), \theta)}{\partial Pr(X)} = -log(Pr(X)) - 1 \\
&+ \sum_{k}\theta_k \sum_{(s,a)}\phi_i(s,a) + \sum_{k=1}^{K}\theta_k\Bigg(\sum_{Y\in\Omega}\tilde{Pr}(Y) \\
&\frac{\sum\limits_{Z'\in\mathcal{Z}}\Bigg[\sum\limits_{<s,a>\in X'}\phi_k(s,a) - \sum\limits_{<s,a>\in X}\phi_k(s,a)\Bigg]Pr(X')}{Pr(Y)^2})\Bigg) \\
&+ \eta
\end{aligned} \tag{4.7}$$

where $Y = T$, $Z = \vec{\omega}$, $X = (Y \cup Z)$ and $X' = (Y \cup Z')$. As one can see, there is no closed form solution for Eq. 4.7. However, Wang *et al.* [27] proposed an approximation with the

following form:

$$\frac{\partial \mathcal{L}(Pr(\vec{\omega}, T), \theta)}{\partial Pr(\vec{\omega}, T)} \approx -log(Pr(\vec{\omega}, T)) - 1 + \sum_k \theta_k \sum_{(s,a)} \phi_i(s, a) + \eta \tag{4.8}$$

Setting Eq. 4.8 to zero:

$$Pr((\vec{\omega}, T), \theta) \approx \frac{e^{\sum_k \theta \sum_{(s,a)} \phi_i(s,a)}}{n(\theta)} \tag{4.9}$$

Where $n(\theta)$ is the normalizer constant.

Now we can approximately calculate the optimal value of $Pr(\vec{\omega}, T)$, therefore we can calculate the value of Lagrangian parameters. By plugging the above equation to $\mathcal{L}(Pr(\vec{\omega}, T); \theta; \eta)$ we arrive at:

$$\mathcal{L}^{(Dual)}(\theta) = log(n(\theta)) - \sum_k \theta_k \sum_{\vec{\omega} \in \tilde{\Omega}} \frac{1}{|\tilde{\Omega}|} \sum_T Pr(T|\vec{\omega}) \sum_{(s,a)} \phi(s, a) \tag{4.10}$$

4.4 Expectation Maximization

Because of the presence of $Pr(\vec{\omega}|T)$ in the Eq. 4.10 we cannot use exponentiated gradient descent to obtain the optimal value of parameter vector. However, Wang *et al.* [27] proposed an iterative EM approach that we incorporated into our Maximum entropy model to find the optimum value of vector parameter.

Likelihood of Lagrangian parameter can be defined as follow:

$$\begin{aligned} LL(\theta|\vec{\omega}) &= \log \prod_{\vec{\omega} \in \Omega} Pr(\vec{\omega}; \theta)^{\tilde{Pr}(\vec{\omega})} \\ &= \sum_{\vec{\omega} \in \Omega} \tilde{Pr}(\vec{\omega}) \log Pr(\vec{\omega}; \theta) \sum_{T \in \mathcal{T}} Pr(T|\vec{\omega}; \theta) \\ &= \sum_{\vec{\omega} \in \Omega} \tilde{Pr}(\vec{\omega}) \sum_{T \in \mathcal{T}} Pr(T|\vec{\omega}; \theta) \log Pr(\vec{\omega}; \theta) \end{aligned} \tag{4.11}$$

Rewriting $Pr(\vec{\omega}|\theta)$ as $\frac{Pr(\vec{\omega},T;\theta)}{Pr(T|\vec{\omega},\theta)}$ in Eq. 4.11:

$$\begin{aligned} LL(\theta|\vec{\omega}) &= \sum_{\vec{\omega}\in\Omega} \tilde{Pr}(\vec{\omega}) \sum_{T\in\mathcal{T}} Pr(T|\vec{\omega};\theta) \log \frac{Pr(\vec{\omega},T;\theta)}{Pr(T|\vec{\omega},\theta)} \\ &= \sum_{\vec{\omega}\in\Omega} \tilde{Pr}(\vec{\omega}) \sum_{T\in\mathcal{T}} Pr(T|\vec{\omega};\theta)(\log(Pr(\vec{\omega},T;\theta)) \\ &\quad - \log(Pr(T|\vec{\omega},\theta))) \end{aligned} \tag{4.12}$$

Now we may use EM to improve the likelihood in Eq. 4.12 iteratively. We can reformulate the likelihood as $Q(\theta,\theta^t) + C(\theta,\theta^t)$ where:

$$Q(\theta,\theta^t) = \sum_{\vec{\omega}\in\Omega} \tilde{Pr}(\vec{\omega}) \sum_{T\in\mathcal{T}} Pr(T|\vec{\omega};\theta^t) \log(Pr(\vec{\omega},T;\theta)) \tag{4.13}$$

$$C(\theta,\theta^t) = \sum_{\vec{\omega}\in\Omega} \tilde{Pr}(\vec{\omega}) \sum_{T\in\mathcal{T}} Pr(T|\vec{\omega};\theta^t) \log(Pr(T|\vec{\omega},\theta)) \tag{4.14}$$

Replacing $Pr(\vec{\omega},T;\theta)$ in Eq. 4.13 with Eq.4.3:

$$Q(\theta,\theta^t) = -(log(n(\theta)) - \sum_k \theta_k \sum_{\vec{\omega}\in\tilde{\Omega}} \frac{1}{|\tilde{\Omega}|} \sum_T Pr(T|\vec{\omega}) \sum_{(s,a)} \phi(s,a)) \tag{4.15}$$

One may notice that Q function is the negative of the dual presented in Eq. 4.10. Therefore maximizing the Q function is equal to minimizing the dual. Using these facts we may reformulate the original problem stated in Eq. 4.2 as follow.

E-step

In the E-step we use the parameter θ^t from the previous iteration to calculate the future expectation of the expert.

$$\hat{\phi}_k^{T|\vec{\omega},t} = \sum_{\vec{\omega}\in\Omega} \tilde{Pr}(\vec{\omega}) \sum_{T\in\mathcal{T}} Pr(T|\vec{\omega};\theta^t) \sum_{<s,a>\in T} \phi_k(s,a) \tag{4.16}$$

To calculate $Pr(T|\vec{\omega})$ we may use Bayes rule.

$$Pr(T|\vec{\omega}) = \eta Pr(\vec{\omega}|T)Pr(T) \tag{4.17}$$

Where:

$$Pr(T) = Pr(s_0)\prod_{i=1}^{n-1} Pr(s_{i+1}|s_i, a_i)Pr(a_i|s_i) \tag{4.18}$$

$$Pr(\vec{\omega}|T) = \prod_{i=1}^{n} Pr(o_i|s_i, a_i) \tag{4.19}$$

In Eq. 4.18 $Pr(a_i|s_i)$ is the expert's policy give θ^t and in Eq. 4.19 $Pr(o_i|s_i, a_i)$ is the observation model.

M-step

In the M-Step we utilize the feature expectation that has been calculated in the E-Step to obtain the θ.

$$\begin{aligned}
&\max_{\Delta}(-\sum_{\vec{\omega},T} Pr(\vec{\omega}, T)log(Pr(\vec{\omega}, T)) \\
&\textit{subjected to} \\
&\sum_{\vec{\omega},T} Pr(\vec{\omega}, T) = 1 \\
&\sum_{\vec{\omega}\in\Omega}\sum_{T\in\tau} Pr(\vec{\omega}, T) \sum_{(s,a)\in T} \phi_k(s, a) = \hat{\phi}_k^{T|\vec{\omega},t}
\end{aligned} \tag{4.20}$$

As it shown in Eq. 4.16 calculating E-Step involves a summation over all possible trajectories. Calculating this summation may not be feasible in real domain problem. we utilize Gibbs sampling to approximate this summation.

4.5 Approximating the E-step

In the E-step, we calculate the feature exception of the reward function under consideration. However, as one may notice in Eq. 4.16 the second summation ($\sum_{T \in \mathcal{T}}$) is over the set of all possible trajectories of a given length. This is problematic in problems with slightly big state and action space. Consider a scenario in which the size of the state and action space is 5 and 2 respectively, and let's assume that the length of the trajectory is 12. Therefore, the size of $\mathcal{T}$ is 10^{12}. Obviously, we cannot calculate the feature expectation exactly even for such a small problem.

Although we cannot calculate the Eq. 4.16 exactly, we can approximate it by using Gibbs sampling. First, we must initialize a trajectory according to the transition function, observation, and current policy. After that, we choose a node (variable) randomly and sample that according to its Markov blanket [26]. As we mentioned earlier Gibbs sampling approximate a distribution by sampling random variables of that distribution conditioned on all other random variables. However, we sample each node (variable) conditioned on its Markov blanket; we are doing this because each node is conditionally independent of all other nodes given its Markov blanket. Markov blanket of a given node is the parents, children, and parents of children of that node. Figure 4.3 illustrates an example of Markov blanket in a Bayesian network.

Figure 4.4 illustrates one step of the Gibbs sampling for a trajectory with length 3. Considering the trajectory from the previous step, we choose a node at random. It can be either an action node a_i or a state node s_i. The top portion of the image shows the Markov blanket and the conditional distribution we need to use for sampling if the node that is chosen is an action node. Moreover, the bottom portion of the image shows the Markov blanket and the conditional distribution we need to use for sampling if the node that is chosen is a state node.

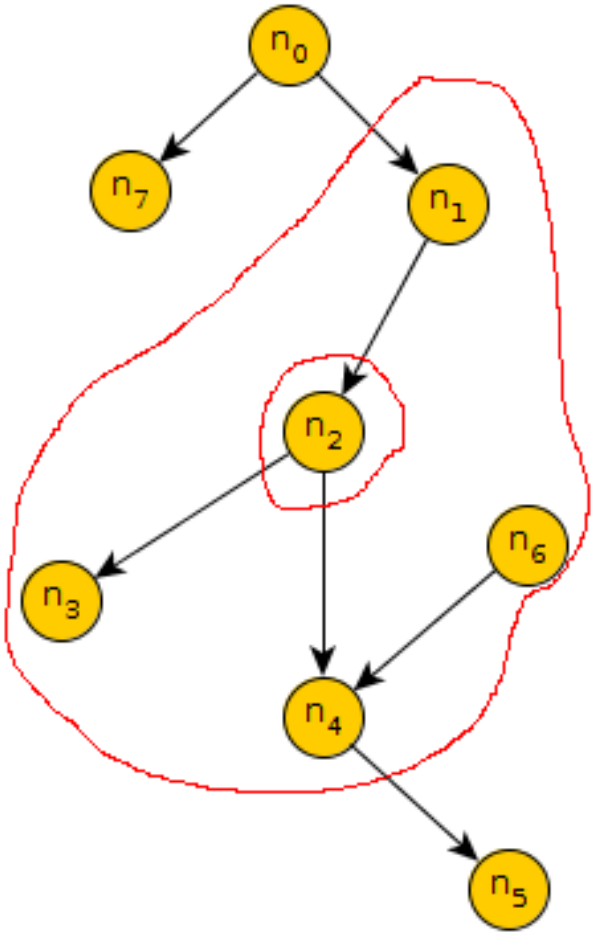


Figure 4.3: All the nodes between the two red boundaries make the Markov blanket for n_2.

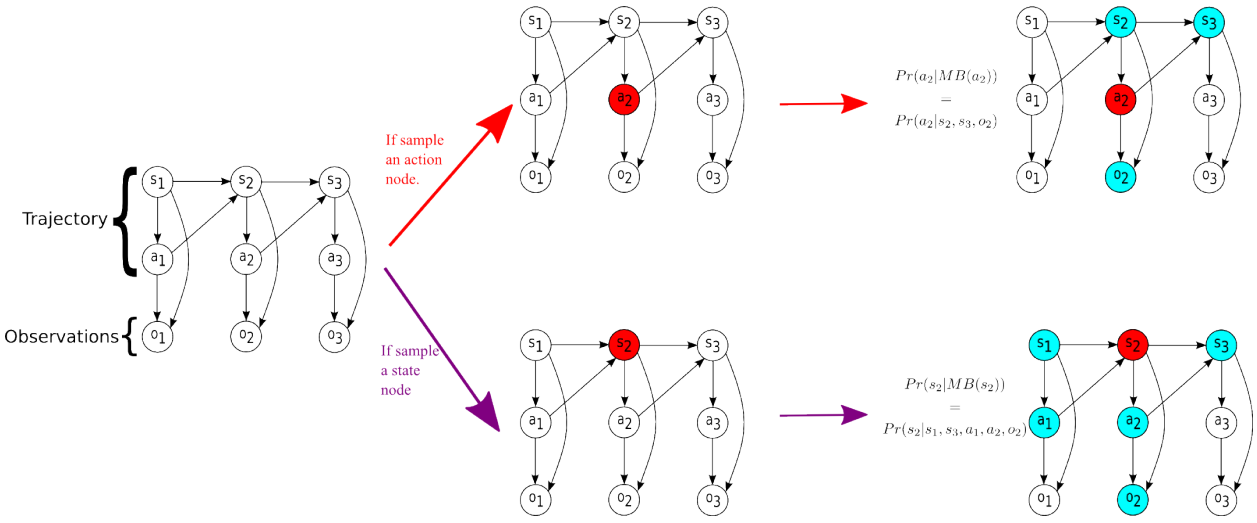


Figure 4.4: This image illustrates each step of the Gibbs sampling.

4.6 Algorithm

Following is the complete algorithm of robust-IRL. Algorithm 2 shows the overall steps in robust-IRL. Algorithm 3 and 4 show a detailed description of E-step and Gibbs sampling needed for E-step respectively.

In algorithm 2, first, we initialize the reward function randomly and then we construct the optimal policy accordingly. Then we do E-step and M-Step repeatedly till convergence.

Algorithm 3 shows the exact solution for the E-step. We calculate the probability of each trajectory given the transition function, observation model, and current policy. Then we multiply this probability by the feature count of the trajectory to calculate the feature expectation. As we mentioned before this might become infeasible in domains with large state and action spaces. Algorithm 4 shows how to approximate the feature expectation using Gibbs sampling.

Algorithm 2 Robust inverse reinforcement learning

$RewardWeights \leftarrow$ Initialize
$Policy \leftarrow$ Initialize
while FeatureException not converged **do**
    E-step:
    $FeatureException \leftarrow \sum_{\vec{\omega} \in \tilde{\Omega}} \frac{1}{|\tilde{\Omega}|} \sum_{T} Pr(T|\vec{\omega}, \theta^{(t)}) \sum_{(s,a)} \phi(s,a)$
    M-step:
    $RewardWeights \leftarrow log(n(\theta)) - \sum_{k} \theta_k \sum_{\vec{\omega} \in \tilde{\Omega}} \frac{1}{|\tilde{\Omega}|} \sum_{T} Pr(T|\vec{\omega}) \sum_{(s,a)} \phi(s,a)$

    update Reward Function
    update Policy
end while

Algorithm 3 E-step

$FeatureException \leftarrow$ Initialize to all zero
for all $T \in \tau$ **do**
    $Pr(T) = Pr(s_0) \prod_{i=1}^{n-1} Pr(s_{i+1}|s_i, a_i) Pr(a_i|s_i)$
    $Pr(T|\vec{\omega}) = \eta Pr(\vec{\omega}|T) Pr(T)$
    $FeatureException = FeatureException + \sum_{\vec{\omega} \in \tilde{\Omega}} \frac{1}{|\tilde{\Omega}|} Pr(T|\vec{\omega}, \theta^{(t)}) \sum_{(s,a)} \phi(s,a)$
end for

Algorithm 4 E-step Gibbs Sampling

$T \leftarrow$ Initialize using Observation and current Policy
$FeatureExpectationVector \leftarrow$ Initialize to all zero
while $FeatureExpectationVector$ not converged **do**
    for all $numberofsamplingsteps$ **do**
        Sample one Node in T at random according to its Markov blanket using observations, transition function, observation model and current policy
        Update $FeatureExpectationVector$
    end for
end while

Chapter 5

Experiments and Results

In this chapter, we elaborate on two domains that we used to evaluate the Robust IRL method. Also, we explain another algorithm that we used as a baseline.

5.1 Baseline

We propose a method as a baseline for comparison, which we call it most likely trajectory method. One way of dealing with noise is to consider a distribution over all possible trajectories given the observation. Since the true trajectories are not available for the learner, treat them as hidden data and combine EM framework with maximum entropy principle to solve the problem. On the other hand, there is an alternative approach. At each time-step t after receiving the observation o_t the learner can calculate $Pr(o_t|s,a)$ for $\forall s \in S$ and $\forall a \in A$ and then choose the (s,a) for the time-step t with the highest probability. After constructing the trajectory the learner could use this trajectory and learn the reward function. As expected this method is faster than robust IRL method because it avoids the expectation maximization error, however, it is not robust to the presence of noise.

5.2 Learning Drone Reconnaissance Routine

The first domain is a simulation-only domain in which a robot is tasked with learning the policy of a drone that protects a corridor. In this domain, the learner is hidden from the drone's sight. An important challenge in this domain is that the only observation available to the learner is the sound from drone's propellers. The drone follows the optimal policy according to its MDP, however, the robot models the drone as an hMDP because the true state and action of the expert are hidden fro the learner. The state of the MDP is the location and orientation of the drone in the corridor. The drone has 3 actions, going forward, turn around, hover. We model the transition function as executing the intended action with the probability of 0.9 percent and the remaining probability is uniformly distributed between two remaining actions. Moreover, we model the reward function as a linear combination of binary following features.

- Moved forward: it returns 1 if the drone moves forward, otherwise 0.
- Turned around: it return 1 if the drone make a U-turn at state s, otherwise 0.

The observation in the hMDP are the parameters of the function $f(t)$ described in the section 4.2.Since in this domain, the learner only receives the expert's movement sound the observation model is solely constructed based on the sound intensity. Figure 5.1 is an illustration of the problem domain.

Figure 5.2 shows the results that we obtained for our simulation. As the level of noise increases, both methods show lower performance. However, robust-IRL is less affected. Moreover, as the level of noise decreases the observation model becomes more deterministic. Consequentially, the must likely trajectory method become as accurate as robust-IRL. Because when the observation model is near deterministic the constructing of the true trajectory becomes easier.

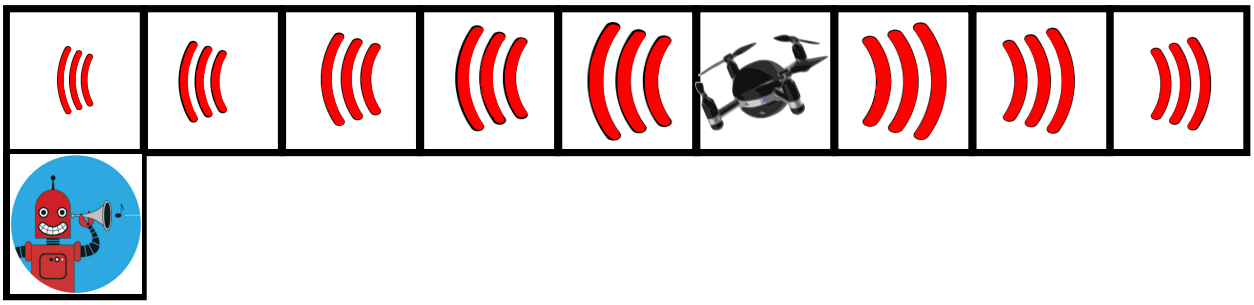


Figure 5.1: Learning drone reconnaissance routine: The robot disguises in the lower left corner and listen to sound from drone's propellers and learns its routine.

5.3 Penetrating a Patrol

This domain is done in simulation and physical robot experiment. In this domain, a Turtlebot (intruder) is tasked with learning the policy of another Turtlebot (patroller) that is patrolling an area. The intruder must learn the patroller's policy and reach to the goal state without being seen by the patroller. Both robots in our simulation and physical experiments are Turtlebot equipped with Microsoft XBox 360 Kinect, which provides a camera and a ranging sensor. Moreover, the learner equipped with a microphone that enables it to listen to expert's foot step. In this domain, the MDP's state is consists of the patroller's location and orientation and the actions are, turning 90 degree, moving forward, no operation. Patroller's reward function modeled as a linear combination of binary following features.

- Moved forward: it returns 1 if the patroller moves forward, otherwise 0.
- Turned around: it return 1 if the patroller makes a U-turn at state s, otherwise 0.

In this domain, the observation for the hMDP is consists of sensory data from the Kinect's range finder and sound intensity from the microphone. When the expert is in sight the learner considers the range finder data as observation and when the expert is not in sight the sound intensity considered as the observation. We call this observation model hybrid

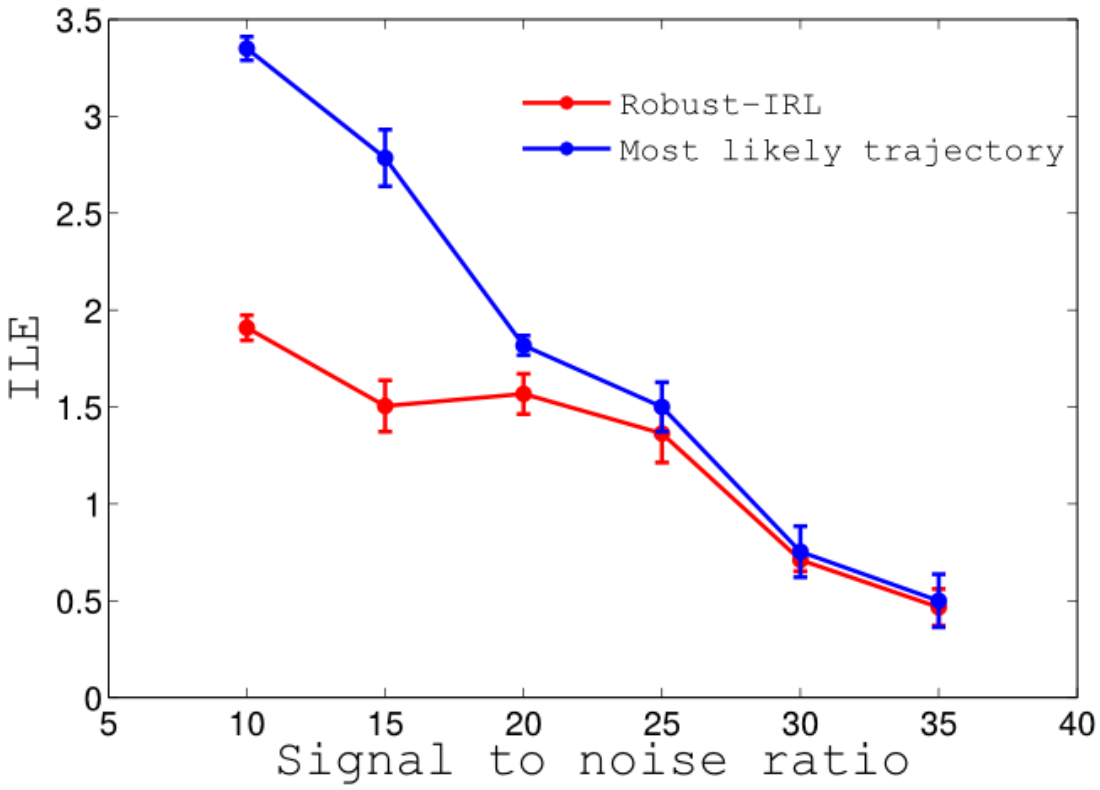


Figure 5.2: Performance evaluation of two methods for the learning drone reconnaissance routine simulation. ILE performance for the robust-IRL method and using most likely trajectory method. The horizontal axis shows the amount of noise added to the observation. As expected robust-IRL method has a better performance compare to the other method as the level of noise increases.

observation model. Moreover, we consider the case where we disregard the sound from the expert and only use the visual information. In this observation model when the expert is not in sight, we consider a uniform distribution over possible state and actions. We call this observation model non-hybrid observation model. As expected utilizing the information from the microphone helps the learner. Figure 5.3 illustrates *penetrating a patrol* problem domain.

Figure 5.4 shows the number of successful runs versus the amount of noise added to the sound of patroller's movement. Each run considered to be successful if the learner can

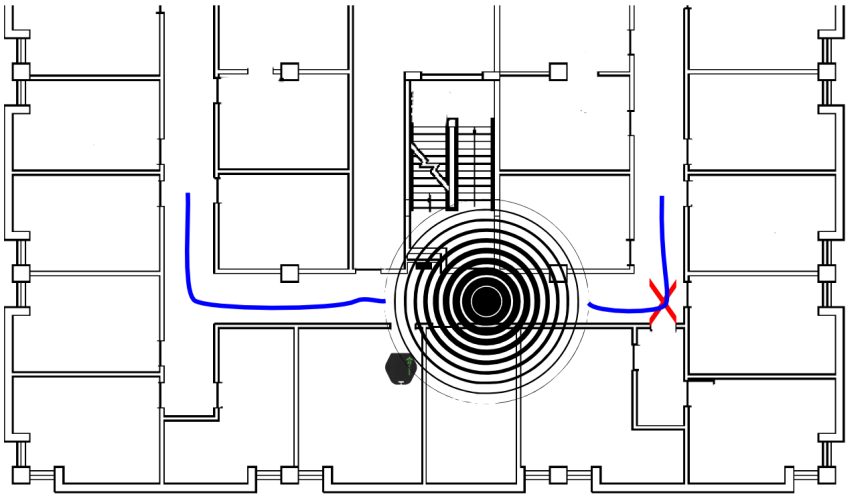


Figure 5.3: The penetrator is hidden in the room and the patroller protecting the goal, indicated by X. The penetrator must learn how the patroller moves in the hallway then reach the goal without being seen by the patroller. The blue line indicates the expert's patrolling path, and the size of the red circle indicates the magnitude of the sound intensity generated from expert's movement.

reach the goal state without being seen by the patroller under 30 minutes. As we expected robust-IRL method when has better performance compare to the most likely trajectory method. Since we only added noise to the sound of patroller's movement the robust-IRL that only uses vision is not affected by the level of noise. As one can see, when the level of noise is very low (signal to noise ratio bigger than 25) the most likely trajectory method produces better results. Even if the level of noise is high robust-IRL that use both vision and sound produces better results. This shows that even if the level of noise is high, we can retrieve some useful information from noisy observations.

Figure 5.5 shows the change in the ILE versus the level of noise added to the sound of patroller's movement sound. The robust-IRL that uses vision has better performance than the most likely trajectory when the level of noise in the sound is considerable. However, the robust-IRL that uses vision and sound outperform the two other methods.

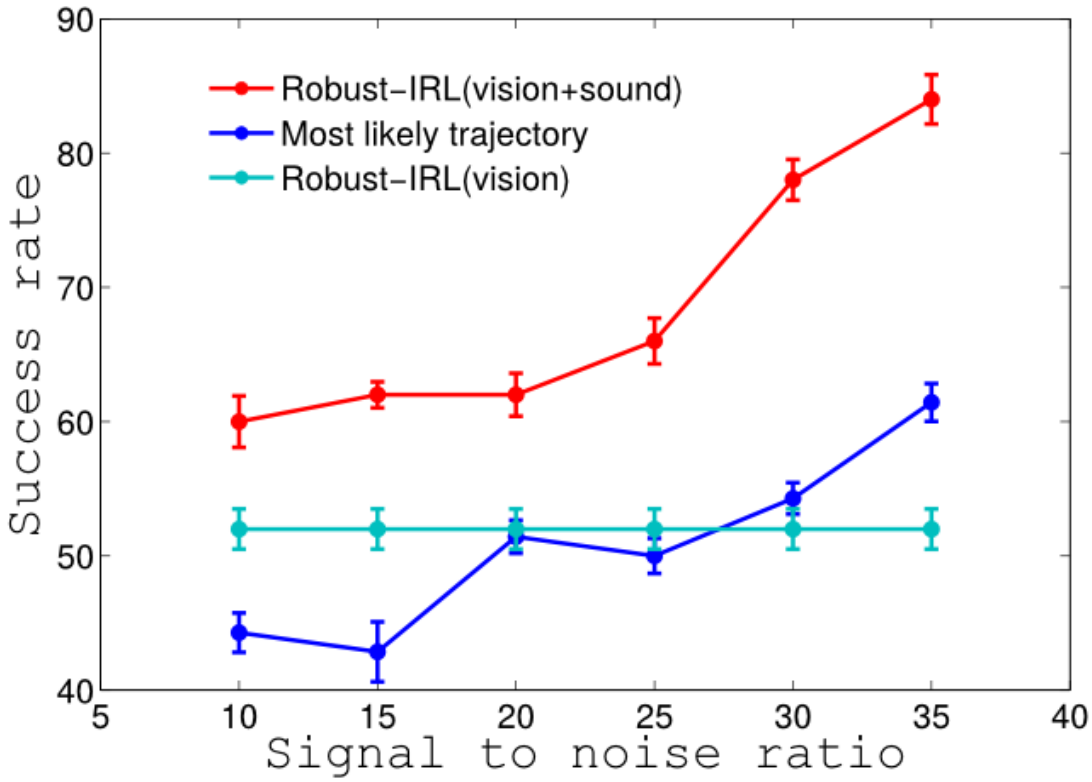


Figure 5.4: Performance evaluation of two methods on the Penetrating a patrol. This comparison is based on the successful runs. The robust-IRL method is evaluated using two different observation models. The horizontal axis shows the amount of noise added to the sound. As expected robust-IRL method has a better performance compare to the other method as the level of noise increases.

Physical runs

In addition to simulation for penetrating a patrol domain, we evaluate the performance of robust IRL with physical robots. Table 5.1 shows the obtained results from 10 physical runs for each method. Due to the limited battery life of Turtlebots, we limited the experiment time to 30 minutes for each run. It means we give the total time of 30 minutes to the learner to observe and learn. If by 30 minutes the learner could reach the goal without being seen by the patroller, we count that as a successful run. All other cases we counted as unsuccessful

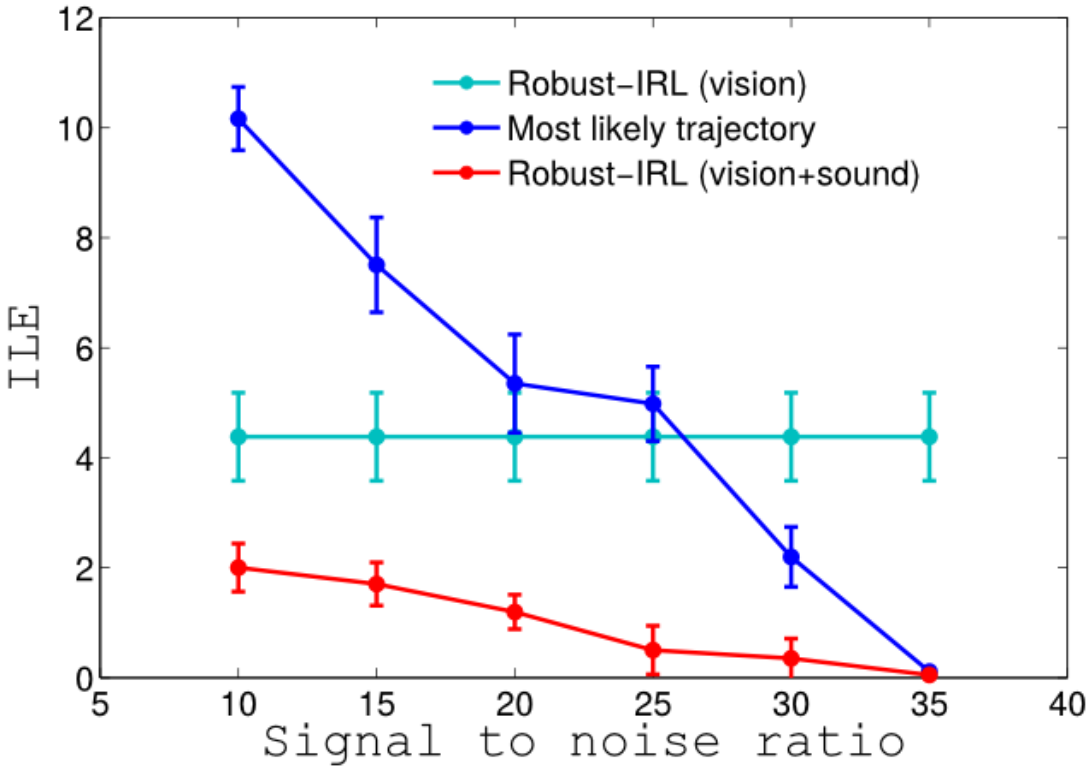


Figure 5.5: Performance evaluation of two methods on the penetrating a patrol simulation. ILE performance for Robust-IRL method and using most likely trajectory method. The robust-IRL method is evaluated using two different observation models. The horizontal axis shows the amount of noise added to the observation. As expected robust-IRL method has a better performance compare to the other method as the level of noise increases.

runs.

For physical runs, we used a random attack approach as an extra baseline. In random attack the learner wait for a random amount of time then it attacks.

Table 5.1: Results from physical runs

Method	Successful runs	Unsuccessful runs
Robust IRL	7	3
Most likely trajectory	4	6
Random Attack	1	9

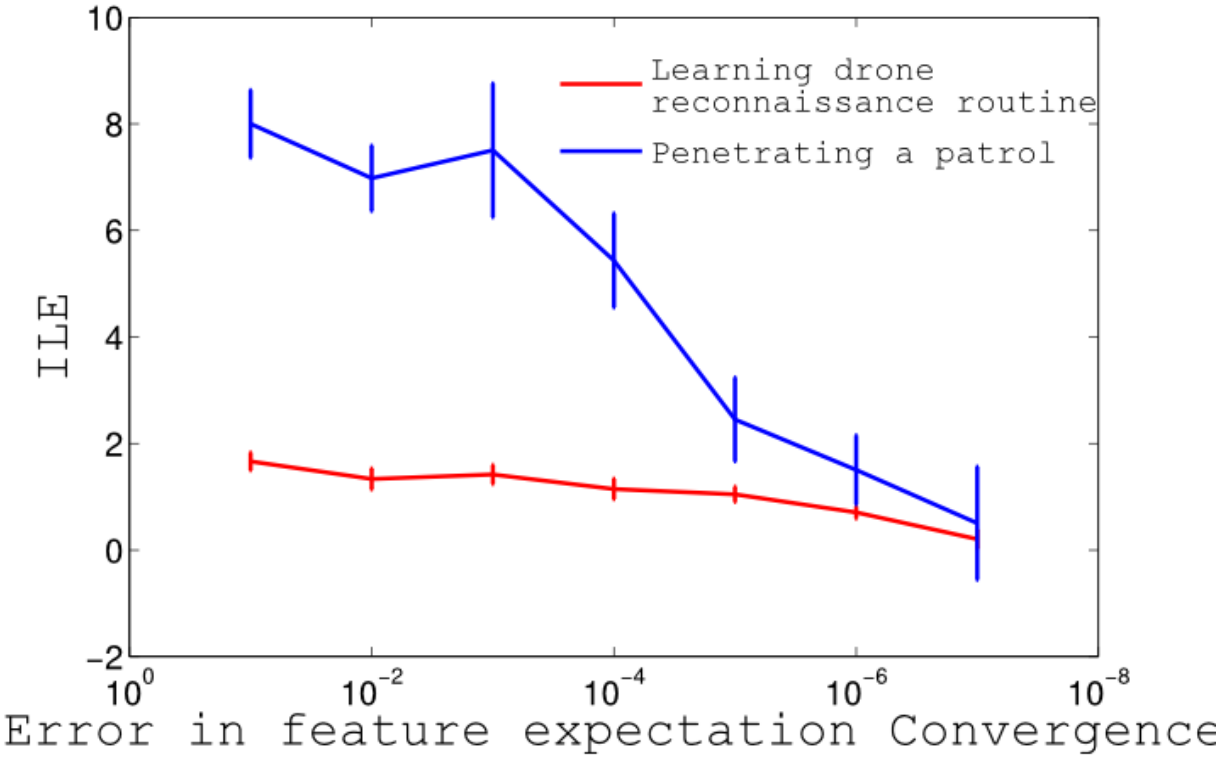


Figure 5.6: Performance evaluation of robust-IRL in two domains. The horizontal axis shows the convergence threshold in calculating the feature expectation in the E-step. The robust-IRL method is evaluated using two different observation models. Lower numbers means a tighter convergence condition. Lower ILE means higher accuracy.

Figure 5.7: This is the Turtlebot that has been used on our experiment. Each Turtlebot is equipped with a Microsoft XBox 360 Kinect, which provides a camera, a ranging sensor, and a microphone.

Chapter 6

Conclusion and future work

In the context of real world robotics problem presence of noise in observation is usually unavoidable. Our method proposes a mathematical framework to deal with noise. Our solution is to incorporate an observation model and try to recover a distribution over trajectories given observation. The experiments show promising results. They indicate that useful policies can be learned by the learner robot even if the amount of noise is considerable. Also, we show that the learner robot can integrate information from various sensors with different level of accuracy through robust IRL.

Robust-IRL demonstrated promising results for robotic domains. One can study the use of the robust-IRL technique for other similar domains such as self-driving cars. Robust-IRL can be useful under heavy weather conditions when the data from the sensors is not as reliable as the normal weather condition. Recently there has been lots of advancements in the field of deep learning [15]. Deep recurrent neural networks and long-term-short-term-memory networks (LSTM) [30] showed promising results for processing and extracting features from sound information. It is possible to use these techniques instead for regression to construct a complicated observation model for robust-IRL.

An open question, which remains from this study is how we can extend robust-IRL to suit

multi-agent settings. For example in [3] Bogert and Doshi extended Max-Ent IRL for situations with multiple experts. Also solving a robust-IRL program requires lots of computation power and time, which makes it impossible to solve the problem exactly. Therefore, we performed an approximation step through Gibbs sampling. However, it might be possible to solve the robust-IRL program exactly by taking advantage of GPU programming [22].

Bibliography

[1] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pages 1–, New York, NY, USA, 2004. ACM.

[2] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[3] K. Bogert and P. 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, pages 173–180, Richland, SC, 2014. International Foundation for Autonomous Agents and Multiagent Systems.

[4] K. Bogert, J. F.-S. Lin, P. Doshi, and D. Kulic. Expectation-maximization for inverse reinforcement learning with hidden data. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, AAMAS '16, pages 1034–1042, Richland, SC, 2016. International Foundation for Autonomous Agents and Multiagent Systems.

[5] A. Boularias, J. Kober, and J. Peters. Relative entropy inverse reinforcement learning. In *AISTATS*, pages 182–189, 2011.

[6] J. Choi and K.-E. Kim. Inverse reinforcement learning in partially observable environments. *Journal of Machine Learning Research*, 12(Mar):691–730, 2011.

[7] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

[8] Y. Gao, J. Peters, A. Tsourdos, S. Zhifei, and E. Meng Joo. A survey of inverse reinforcement learning techniques. *International Journal of Intelligent Computing and Cybernetics*, 5(3):293–311, 2012.

[9] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence*, (6):721–741, 1984.

[10] W. R. Gilks. *Markov chain monte carlo*. Wiley Online Library, 2005.

[11] W. R. Gilks, N. Best, and K. Tan. Adaptive rejection metropolis sampling within gibbs sampling. *Applied Statistics*, pages 455–472, 1995.

[12] W. R. Gilks and P. Wild. Adaptive rejection sampling for gibbs sampling. *Applied Statistics*, pages 337–348, 1992.

[13] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[14] K. M. Kitani, B. D. Ziebart, J. A. Bagnell, and M. Hebert. Activity forecasting. In *European Conference on Computer Vision*, pages 201–214. Springer, 2012.

[15] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[16] J. W. Lee. Stock price prediction using reinforcement learning. In *Industrial Electronics, 2001. Proceedings. ISIE 2001. IEEE International Symposium on*, volume 1, pages 690–695. IEEE, 2001.

[17] S. Levine, Z. Popovic, and V. Koltun. Nonlinear inverse reinforcement learning with gaussian processes. In *Advances in Neural Information Processing Systems*, pages 19–27, 2011.

[18] L. Martino, J. Read, and D. Luengo. Independent doubly adaptive rejection metropolis sampling within gibbs sampling. *IEEE Transactions on Signal Processing*, 63(12):3123–3138, 2015.

[19] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[20] G. Neu and C. Szepesvári. Apprenticeship learning using inverse reinforcement learning and gradient methods. *arXiv preprint arXiv:1206.5264*, 2012.

[21] A. Y. Ng, S. J. Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, pages 663–670, 2000.

[22] M. Pharr and R. Fernando. *Gpu gems 2: programming techniques for high-performance graphics and general-purpose computation*. Addison-Wesley Professional, 2005.

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

[24] D. Ramachandran and E. Amir. Bayesian inverse reinforcement learning. *Urbana*, 51(61801):1–4, 2007.

[25] N. D. Ratliff, J. A. Bagnell, and M. A. Zinkevich. Maximum margin planning. In *Proceedings of the 23rd international conference on Machine learning*, pages 729–736. ACM, 2006.

[26] S. J. Russell, P. Norvig, J. F. Canny, J. M. Malik, and D. D. Edwards. *Artificial intelligence: a modern approach*, volume 2. Prentice hall Upper Saddle River, 2003.

[27] S. Wang, D. Schuurmans, and Y. Zhao. The latent maximum entropy principle. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 6(2):8, 2012.

[28] C. J. Watkins and P. Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[29] C. J. Wu. On the convergence properties of the em algorithm. *The Annals of statistics*, pages 95–103, 1983.

[30] W. Zaremba, I. Sutskever, and O. Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.

[31] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, pages 1433–1438, 2008.