A Survey of Inverse Reinforcement Learning: Challenges, Methods and Progress

Saurabh Arora

THINC Lab, Dept. of Computer Science, University of Georgia, Athens, GA 30602

Prashant Doshi

THINC Lab and Institute for AI, Dept. of Computer Science, University of Georgia, Athens, GA 30602

Abstract

Inverse reinforcement learning (IRL) is the problem of inferring the reward function of an agent, given its policy or observed behavior. Analogous to RL, IRL is perceived both as a problem and as a class of methods. By categorically surveying the current literature in IRL, this article serves as a reference for researchers and practitioners of machine learning to understand the challenges of IRL and select the approaches best suited for the problem on hand. The survey formally introduces the IRL problem along with its central challenges such as the difficulty in performing accurate inference and its generalizability, its sensitivity to prior knowledge, and the disproportionate growth in solution complexity with problem size. The article elaborates how the current methods mitigate these challenges. We further discuss the extensions to traditional IRL methods for handling: (*i*) inaccurate and incomplete perception, (*ii*) an incomplete model, (*iii*) multiple reward functions, and (*iv*) nonlinear reward functions. This discussion concludes with some broad advances in the research area and currently open research questions.

Keywords: reinforcement learning, reward function, learning from demonstration, generalization, learning accuracy, survey

Email addresses: `sa08751@uga.edu` (Saurabh Arora), `pdoshi@cs.uga.edu` (Prashant Doshi)

2010 MSC: 00-01, 99-00

1. Introduction

Inverse reinforcement learning (IRL) is the problem of modeling the preferences of another agent using its observed behavior, thereby avoiding a manual specification of its reward function [1, 2]. In the past decade or so, IRL has attracted several researchers in the communities of artificial intelligence, psychology, control theory, and machine learning. IRL is appealing because of its poterntial to use data recorded in a task to build autonomous agents capable of modeling others without intervening in the performance of the task.

We study this problem and associated advances in a structured way to address the needs of readers with different levels of familiarity with the field. For clarity, we use a contemporary example to illustrate IRL's use and associated challenges. Consider a self-driving car in role B in Fig. 1. To safely merge into a congested freeway, it may model the behavior of the car in role A; this car forms the immediate traffic. We may use previously collected trajectories of cars in role B, on freeway entry ramps, to learn the safety and speed preferences of a typical driver as she approaches this difficult merge (NGSIM [3] is one such existing data set).

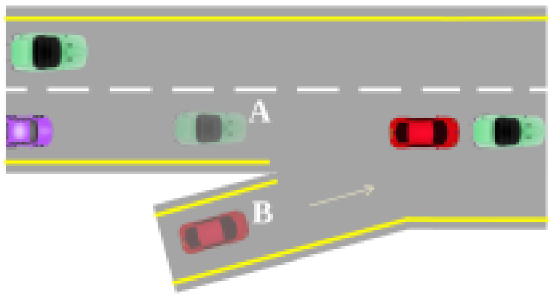

Figure 1: Red car B is trying to merge into the lane, and Green car A is the immediate traffic. The transparent images of cars show their positions before merging, and the opaque images depict one of their possible positions after the merger.

Approaches for IRL predominantly ascribe a Markov decision process (MDP) [4] to the interaction of the observed agent with its environment, and whose solution is a *policy* that maps states to actions. The reward function of this MDP is unknown, and the observed agent is assumed to follow an optimal policy for the MDP. In the traffic merge example, the MDP represents the driving process of car A. The driver of Car A is following action choices (deceleration, braking, low acceleration, and others) based on its optimal policy. Car B needs to reach the end of merging lane before or after Car A for merging safely.

1.1. Significance of IRL

Researchers in the areas of machine learning and artificial intelligence have developed a substantial interest in IRL because it caters to the following needs.

1.1.1. Demonstration substitutes manual specification of reward

Typically, if a designer wants intelligent behavior in an agent, she manually formulates the problem as a forward learning or forward control task solvable using solution techniques in RL, optimal control, or predictive control. A key element of this formulation is a specification of the agent's preferences and goals via a reward function. In the traffic merge example, we may hand design a reward function for Car A. For example, +1 reward if taking an action in a state decreases the relative velocity of Car B w.r.t. Car A within a predefined distance from merging junction, thereby allowing for a safe merge. Analogously, a negative reward of -1 if taking an action in a state increases the relative velocity of Car B w.r.t. Car A. This example specification captures one aspect of a successful merge into a congested freeway: that the merging car must slow down to match the speed of the freeway traffic. However, other aspects such as merging a safe distance behind Car A and not too close in front of the car behind A require further tuning of the reward function. While roughly specified reward functions are sufficient in many domains to obtain expected behavior (indeed affine transformations of the true reward function are sufficient), others require much trial-and-error or a delicate balance of multiple conflicting attributes [5], which becomes cumbersome.

The need to pre-specify the reward function limits the applicability of RL and optimal control to problems where a reward function can be easily specified or simulated. IRL offers a way to broaden the applicability of RL and reduce the manual design of task specification, given a policy or demonstration of desired behavior is available. While acquiring the complete desired policy is usually infeasible, we have easier access to demonstrations of behaviors, often in the form of recorded data. For example, state to action mappings for all contingencies for Car B are not typically available, but datasets such as NGSIM contain trajectories of Cars A and B in real-world driving. Thus, IRL forms a key method for *learning from demonstration* [6].

A topic in control theory related to IRL is inverse optimal control [7]. While the input in both IRL and inverse optimal control (IOC) are trajectories consisting of state-action pairs, the target of learning in the IOC is a function mapping states of observed agent to her actions. The learning agent may use this policy to imitate it or deliberate with it in its own decision-making process.

1.1.2. Improved Generalization

A reward function represents the preferences of an agent in a succinct form, and is amenable to transfer to another agent. The learned reward function may be used as is if the subject agent shares the same environment and goals as the other, otherwise it continues to provide a useful basis when the agent specifications differ mildly – for example, when the subject agent's problem domain exhibits additional states. Indeed, as Russell [1] points out, the reward function is inherently more transferable compared to the observed agent's policy. This is because even slight changes in the environment – for example, changes to the noise levels in the transition function – likely renders the learned policy unusable because it may not be directly revised in straightforward ways. However, this change does not impact the transferability of the reward function. Furthermore, it is likely that the learned reward function simply needs to be extended to any new states in the learner's environment while a learned policy would be discarded if the new states are significant.

1.1.3. Potential Applications

While introducing IRL, Russell [1] 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. [8] and Ullman et al. [9] demonstrate the inference of a human's goal as an inverse planning problem in an MDP. Furthermore, IRL's use toward apprenticeship learning has rapidly expanded the set of visible applications. These can be categorized into:

1. Learning from an expert to create an agent with the expert's preferences. An early and well-known application that brought significant attention to IRL is helicopter flight control [10], illustrated in Fig. 2. In this application, an expert helicopter operator's sophisticated preferences over 24 features were learned from recorded behavior data using IRL. This reward function was then used to teach a physical remotely-controlled helicopter advanced maneuvers using RL. Another application that brought IRL closer to Russell's [1] motivation of modeling animal behavior is that of socially adaptive navigation to avoid colliding into humans by learning from human-walk trajectories [11, 12]. Other important examples include boat sailing [13], learning driving styles [14], and so on;
2. Learning from another agent to predict its behavior. One of the first attempts in this direction was route prediction for taxis [15, 16]. Other such applications are footstep prediction for planning legged locomotion [17], anticipation of pedestrian interactions [18], energy efficient driving [19], and penetration of a perimeter patrol by learning the patrollers' preferences and patrolling route [20].

1.2. Importance of this Survey

This article is a reflection on the research area of IRL with a focus on the following important aspects:

1. Formally introducing IRL and its importance, by means of various examples, to the researchers and practitioners new to the field;

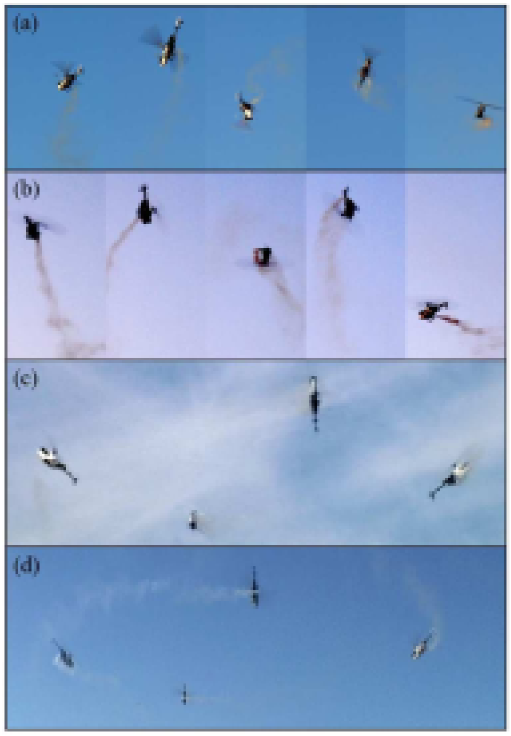

Figure 2: Complex helicopter maneuvers learned using RL on a reward function learned from an expert pilot through IRL. The image is reprinted from [10] with permission from publisher.

2. A study of the challenges that make IRL difficult, and a review of the current (partial) solutions;
3. Qualitative assessment and comparisons among different methods to evaluate them coherently. This will considerably help the readers decide on the approach suitable for the problem at hand;
4. Identification of key milestones achieved by the methods in this field along with their common shortcomings.
5. Identification of the open avenues for future research.

Of course, a single article may not cover all methods in this growing field. Nevertheless, we have sought to make this survey as comprehensive as possible.

1.3. Organization of Contents

As IRL is relatively new and the target reader is likely to be someone who is keen to learn about IRL, the viewpoint of 'IRL as a research problem' is used to guide the organization of this article. Therefore, Section 2 mathematically defines the IRL problem and provides some preliminary technical background that is referenced in later sections. We introduce the core challenges faced by this learning problem in Section 3. These challenges confront all methods and are not specific to any particular technique. Then, we briefly review the foundational methods grouped together by the commonality of their underlying approach in Section 4 that have facilitated progress in IRL, and how these methods mitigate the previously-introduced core challenges in Section 5. This separation of method description across two sections allows a practitioner to quickly identify the methods pertinent to the most egregious challenge she is facing in her IRL problem. This is followed in Section 6 by a review of efforts that generalize or extend the fundamental IRL problem in various directions. Section 7 summarizes the few significant milestones achieved so far and the shortcomings of the existing methods. Finally, the article concludes with a discussion of some open research questions.

2. Formal Definition of IRL

In order to formally define IRL, we must first decide on a framework for modeling the observed agent's behavior. While methods ascribe different frameworks such as an MDP, hidden-parameter MDP, or a POMDP to the expert, 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 $T : S \times A \rightarrow \mathsf{Prob}(S)$ defines a probability distribution over the set of next states conditioned on agent taking action a at state s; $\mathsf{Prob}(S)$ here denotes the set of all probability distributions over S. $T(s'|s, a) \in [0, 1]$ is the probability that the system transitions to state s'. The reward function R can be specified in different ways: $R : S \rightarrow \mathbb{R}$ gives the scalar reinforcement at state s, $R : S \times A \rightarrow \mathbb{R}$ maps a tuple (state s, action a taken in state s) to the reward received on performing the action, and, $R : S \times A \times S \rightarrow \mathbb{R}$ maps a triplet (state s, action a, resultant state s') to the reward obtained on performing the transition. Discount factor γ is the weight for past rewards accumulated in a trajectory, $\langle (s_0, a_0), (s_1, a_1), \ldots (s_j, a_j) \rangle$, where $s_j \in S, a_j \in A, j \in \mathbb{N}$.

A *policy* is a function mapping current state to next action choice(s). It can be deterministic, $\pi : S \rightarrow A$ or stochastic $\pi : S \rightarrow \mathsf{Prob}(A)$. For a policy π, value function $V^\pi : S \rightarrow \mathbb{R}$ 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 for a given start state s_0 is,

$$V^\pi(s_0) = E_{s,\pi(s)} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t)) | s_0 \right] \tag{1}$$

The goal of solving MDP $\mathcal{M}$ is to find an optimal policy π^* such that $V^{\pi^*}(s) = V^*(s) = \sup_{\pi} V^\pi(s)$, for all $s \in S$. The action-value function for π, $Q^\pi : S \times A \rightarrow \mathbb{R}$, maps a state-action pair to the long-term expected cumulative reward incurred after taking action a from s and following policy

π thereafter. We also define the optimal action-value function as $Q^*(s,a) = \sup_\pi Q^\pi(s,a)$. Subsequently, $V^*(s) = \sup_{a\in A} Q^*(s,a)$. Another perspective to the value function involves multiplying the reward with the converged *state-visitation frequency* $\psi^\pi(s)$, which is the number of times the state s is visited on using policy π. The latter is given by:

$$\psi^\pi(s) = \psi^0(s) + \gamma \sum_{s'\in S} T(s,\pi(s),s')\ \psi^\pi(s') \tag{2}$$

where $\psi^0(s)$ is initialized as 0 for all states. Let Ψ be the space of all ψ functions. Iterating the above until the state-visitation frequency stops changing yields the converged frequency function, ψ^π_*. We may write the value function as, $V^*(s) = \sup\limits_\pi \ \sum_{s\in S} \psi^\pi_*(s)\ R(s,\pi(s))$.

We may express the reward function as a linear sum of weighted features:

$$\begin{aligned} R(s,a) &= w_1\phi_1(s,a) + w_2\phi_2(s,a) + \ldots + w_k\phi_k(s,a) \\ &= w^T \boldsymbol{\phi}(s,a). \end{aligned} \tag{3}$$

where $\phi_k : S \to \mathbb{R}$ is a feature function and weight $w_k \in \mathbb{R}$. Then, define the expected feature count for policy π and feature ϕ_k as,

$$\mu^{\phi_k}(\pi) = \sum_{t=0}^{\infty} \psi^\pi(s_t)\ \phi_k(s_t,\pi(s_t)). \tag{4}$$

We will extensively refer to this formulation of the reward function and the expected feature count later in this article. Note that $\mu^{\phi_k}(\pi)$ is also called a successor feature in RL. The expected feature count can be used to define the expected value of a policy:

$$\begin{aligned} V^\pi = \boldsymbol{w}^T \boldsymbol{\mu}^\phi(\pi) &= \sum_{s,a} \psi^\pi(s)\ \boldsymbol{w}^T \boldsymbol{\phi}(s,a) \\ &= \sum_{s,a} \psi^\pi(s)\ R(s,a). \end{aligned} \tag{5}$$

RL offers an online way to solve an MDP. The input for RL is the sequence of sampled experiences in the form (s,a,r) or (s,a,r,s'), which includes the reward or reinforcement due to the agent performing action a in state s. For the model-free setting of RL, the transition function T is unknown. Both the transition function and policy are estimated from the samples and the target of RL is to learn an optimal policy.

We adopt the conventional terminology in IRL, referring to the observed agent as an *expert* and the subject agent as the *learner*. Typically, IRL assumes that the expert is behaving according to an underlying policy π_E, which may not be known. If policy is not known, the learner observes sequences of the expert's state-action pairs called trajectories. The reward function is unknown but the learner usually assumes some structure that helps in the learning. Common functional forms include a linearly-weighted combination of reward features, a probability distribution over reward functions, or a neural network representation. We elaborate on these forms later in this article. The expert's transition function may not be known to the learner. We are now ready to give the formal problem definition of IRL.

Definition 2 (IRL)**.** Let an MDP without reward, $\mathcal{M}\backslash_{R_E}$, represent the dynamics of the expert E. Let $\mathcal{D} = \{\langle(s_0, a_0), (s_1, a_1), \ldots, (s_j, a_j)\rangle_1, \ldots, \langle(s_0, a_0), (s_1, a_1), \ldots, (s_j, a_j)\rangle_{i=2}^{N}\}$, $s_j \in S$, $a_j \in A$, and $i, j, N \in \mathbb{N}$ be the set of demonstrated trajectories. A trajectory in $\mathcal{D}$ is denoted as τ. We assume that all $\tau \in \mathcal{D}$ are perfectly observed. Then, determine $\hat{R}_E$ that best explains either policy π_E if given or the observed behavior in the form of demonstrated trajectories.

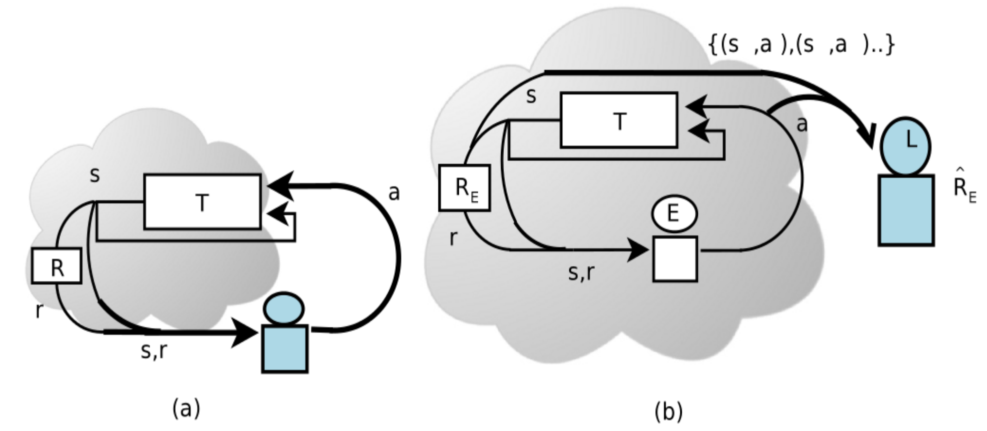

Figure 3: (a) A schematic showing the subject agent (shaded in blue) performing RL [21]. In forward learning or RL, the agent chooses an action at a known state and receives a reward in return generated by a reward function R that may not be known to the agent. The state changes based on the previous state and action, which is modeled using the transition function T that may be unknown as well. (b) In inverse learning or IRL, the input and output for the learner L are reversed. color distinguishes an letter E L perceives the states and actions $\{(s, a)(s, a) \ldots\}$ of expert E (or its policy π_E), and learns a reward function $\hat{R}_E$ that best explains E's behavior, as the output. Note that the learned reward function may not exactly correspond to the true reward function.

Notice that IRL inverts the RL problem. Whereas RL seeks to learn the optimal behavior based on experiences ((s, a, r) or (s, a, r, s')) that include obtained rewards, IRL seeks to best explain the observed behavior by learning the corresponding reward function. We illustrate this relationship between RL and IRL in Fig. 3

We may obtain an estimate of the expected feature count from a given demonstration $\mathcal{D}$, which is the empirical analog of that in Eq. 4,

$$\hat{\mu}^{\phi_k}(\mathcal{D}) = \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{\infty} \gamma^t \phi_k(s_t, a_t). \tag{6}$$

3. Primary Challenges of IRL

IRL is challenging because the optimization associated in finding a reward function that best explains observations is essentially ill-posed. Furthermore, computational costs of solving the problem tend to grow disproportionately with the size of the problem. We discuss these challenges in detail below, but prior to this discussion, we establish some notation. Let $\hat{\pi}_E$ be the policy obtained by optimally solving the MDP with reward function $\hat{R}_E$.

3.1. Difficulty of Accurate Inference

Classical IRL takes an expert demonstration of a task consisting of a finite set of trajectories, knowledge of the environment and expert's dynamics, and finds the expert's potential reward function; this is illustrated in Fig. 4.

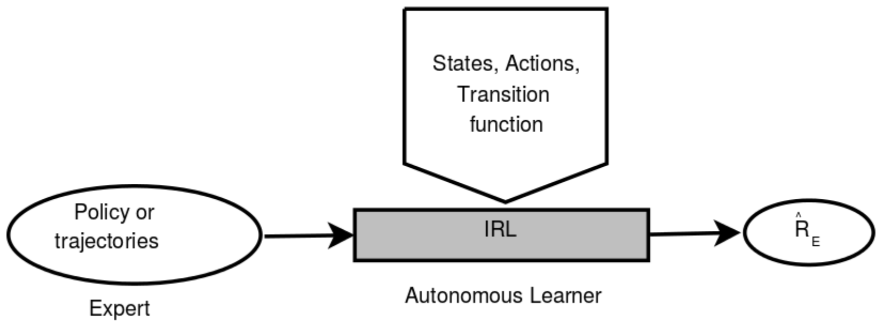

Figure 4: Pipeline for a classical IRL process. The learner receives an optimal policy or trajectories as input. The prior domain knowledge (shown here as a pentagon) include completely observable state space, action space, and fully known transition probabilities.

A critical challenge, first noticed by Ng and Russell [2], is that many reward functions (including highly degenerate ones such as a function with all reward values zero) could explain the observations. This is because the input is usually a finite and small set of trajectories (or a policy) and many reward functions in the set of all reward functions can generate policies that realize the observed demonstration. Thus, IRL suffers from an ambiguity in solution.

Given the difficulty of ensuring accurate inference, its pertinent to contemplate how we may measure accuracy. If the true reward function R_E is available for purposes of evaluation, then one measure of accuracy is the closeness of a learned reward function $\hat{R}_E$ to R_E, $\left|\left|R_E - \hat{R}_E\right|\right|_p$. However, a direct comparison of rewards is not useful because an MDP's optimal policy is invariant under affine transformations of the reward function [22]. On the other hand, two reward functions similar for the most part but differing for some state-action pairs may produce considerably different policies and behaviors. To make the evaluation targeted, a comparison of the behavior generated from the learned reward function with the true behavior of expert is more appropriate. In other words, we may compare the policy $\hat{\pi}_E$ generated from MDP with $\hat{R}_E$ with the true policy π_E. The latter could be given or is generated using the true reward function. A limitation of this measure of accuracy is that a difference between the two policies in just one state could still have a significant impact. This is because performing the correct action at that state may be crucial to realizing the task. Consequently, this measure of closeness is inadequate because it would report just a small difference despite the high significance.

This brings us to the conclusion that we should measure the difference in values of the learned and true policies. Specifically, we may measure the error in inverse learning, called *inverse learning error* (ILE), as $\left|\left|V^{\pi_E} - V^{\hat{\pi}_E}\right|\right|_p$ where V^{π_E} is the value function for actual policy π_E and $V^{\hat{\pi}_E}$ is that for the learned policy $\hat{\pi}_E$ both obtained using the true reward function [23]. Notice that if the true and learned policies are the same, then ILE is zero. However, ILE may also vanish when the two differ if both policies are optimal. On the other hand, ILE requires knowing the true reward function which limits its use to formative evaluations. Another assessment measures the *learned behavior accuracy.* This metric is computed as the number of demonstrated state-action pairs that match between using the true and learned policies expressed as a percentage of the former.

3.2. Generalizability

Generalization refers to the extrapolation of learned information to the states and actions unobserved in the demonstration and to starting the task at different initial states. Observed trajectories typically encompass a subset of the state space and the actions performed from those states. Well-generalized reward functions should reflect expert's overall preferences relevant to the task. The challenge is to generalize correctly to the unobserved space using data that often covers a fraction of the complete space.

Notice that achieving generalizability promotes the temptation of training the learner using fewer examples because the latter now possesses the ability to extrapolate. However, less data may contribute to greater approximation error in $\hat{R}_E$ and inaccurate inference.

ILE continues to be pertinent by offering a way to measure the generalizability of the learned information as well. This is because it compares value functions, which are defined over all states. Another procedure for evaluating generalizability is to simply withhold a few of the demonstration trajectories from the learner. These can be used as labeled test data for comparing with the output of the learned policy on the undemonstrated state-action pairs.

3.3. Sensitivity to Correctness of Prior Knowledge

If we represent the reward function, R_E, as a weighted combination of feature functions, the problem then reduces to finding the values of the weights. Each feature function, $\phi : S \times A \rightarrow \mathbb{R}$, is given and is intended to model a facet of the expert's preferences.

Prior knowledge enters IRL via the specification of feature functions in R_E and the transition function in the MDP ascribed to the expert. Consequently, the accuracy of IRL is sensitive to the selection of feature functions that not only encompass the various facets of the expert's true reward function but also differentiate the facets. Indeed, Neu et al. [13] prove that IRL's accuracy is closely tied to the scaling of correct features. Furthermore, it is also dependent on how accurately are the dynamics of the expert modeled by the ascribed MDP. If the dynamics are not deterministic, due to say some noise in the expert's actuators, the corresponding stochasticity needs to be precisely modeled in the transitions.

Given the significant role of prior knowledge in IRL, the challenge is two-fold: (i) we must ensure its accuracy, but this is often difficult to achieve in practice; (ii) we must reduce the sensitivity of solution methods to the correctness of prior knowledge or replace the knowledge with learned information.

3.4. Disproportionate Growth in Solution Complexity with Problem Size

Methods for IRL are iterative as they involve a constrained search through the space of reward functions. As the number of iterations may vary based on whether the optimization is convex, it is linear, the gradient can be computed quickly, or none of these, we focus on analyzing the complexity of each iteration. Consequently, the computational complexity is expressed as the time complexity of each iteration and its space complexity.

Each iteration's time is dominated by the complexity of solving the ascribed MDP using the reward function currently learned. While the complexity of solving an MDP is polynomial in the size of its parameters, the parameters such as the state space are impacted by the curse of dimensionality – its size is exponential in the number of components of state vector (dimensions). Furthermore, the state space in domains such as robotics is often continuous and an effective discretization also leads to an exponential blowup in the number of discrete states. Therefore, increasing problem size adversely impacts the run time of each iteration of IRL methods.

Another type of complexity affecting IRL is *sample complexity*, which refers to the number of trajectories present in the input demonstration. As the problem size increases, the expert must demonstrate more trajectories in order to maintain the required level of coverage in the training data.

3.5. Direct Learning of Reward or Policy Matching

Two distinct approaches to IRL present themselves, each with its own attendant set of challenges. First seeks to directly approximate the reward function $\hat{R}_E$ by tuning it using input data. The second approach focuses on learning a policy that matches its actions or action values with the demonstrated behavior, thereby learning the reward function as an intermediate step.

Success of the first approach hinges on selecting an adequate and complete reward structure (for example, the set of feature functions) that composes the reward function. Though learning a reward function offers a deeper generalization of the task at hand, it may lead to policies that do not fully reproduce the observed trajectories. For the second approach, Neu et al. [24] point out that the optimization for IRL is convex if the actions are deterministic and the demonstration spans the complete state space. While both approaches are negatively impacted by reduced data, matching the observed policy is particularly sensitive to missing states in the demonstration, which makes the problem non-convex and weakens the objective of matching the given (but now partial) policy.

Next section categorizes the foundational methods in IRL based on the mathematical framework they use for learning, and discusses them in some detail.

4. Foundational Methods for IRL

Many IRL methods fit a template of key steps. We show this template in Algorithm 1, and present the methods in the context of this template. Such presentation allows us to compare and contrast various methods. Algorithm 1 assumes that the expert's MDP sans the reward function is known to the learner as is commonly assumed in most IRL methods although a few methods discussed later allow the transition function to be unknown. Either a demonstration or the expert's policy is provided as input as well as any features for the reward function.

Existing methods seek to learn the expert's preferences, a reward function $\hat{R}_E$, represented in different forms such as a linear combination of weighted feature functions, a probability distribution over multiple real-valued maps from states to reward values, and others. Parameters of $\hat{R}_E$ vary with the type of representation (weights, parameters defining the shape of distribution). IRL involves solving the MDP with the function hypothesized in current iteration and updating the parameters, constituting a search that terminates when the behavior derived from the current solution aligns with the observed behavior.

Rest of this section categorizes IRL methods based on the core idea they use for inverse learning – margin based optimization, entropy based optimization, Bayesian inference, regression, classification, and all others. A second-level grouping within each of these categories clusters methods based on the common technique utilized in realizing the core idea.

4.1. Maximum Margin Optimization

Maximum margin prediction aims to learn a reward function that explains the demonstrated policy better than alternative policies by a margin. The

Algorithm 1: Template for IRL

```
Input: $\mathcal{M}\backslash_{R_E} = \langle S, A, T, \gamma \rangle$,
Set of trajectories demonstrating desired behavior:
$\mathcal{D} = \{\langle (s_0, a_0), (s_1, a_1), \ldots, (s_n, a_n) \rangle, \ldots\}$, $s_n \in S$, $a_n \in A$, $n \in \mathbb{N}$,
or expert's policy: $\pi_E$, and reward function features
Output: $\hat{R}_E$
1 Model the expert's observed behavior as the solution of an MDP whose
    reward function is not known;
2 Initialize the parameterized form of the reward function using any given
    features (linearly weighted sum of feature values, distribution over
    rewards, or other);
3 Solve the MDP with current reward function to generate the learned
    behavior or policy;
4 Update the optimization parameters to minimize the divergence between
    the observed behavior (or policy) and the learned behavior (policy);
5 Repeat the previous two steps till the divergence is reduced to a desired
    level.
```

methods under this category aim to address IRL's solution ambiguity (discussed in Section 3.1) by deciding on a solution that maximizes a margin from others.

4.1.1. Linear and non-linear programming

An early and foundational method for IRL is Ng and Russell's [2], which takes in the expert's policy as input. It formulates a linear program to retrieve the reward function that not only produces the given policy as optimal output from the complete MDP, but also maximizes the sum of differences between the value of the optimal action and that of the next-best action for every state. In addition to maximizing this margin, it also prefers reward functions with smaller values as a form of regularization.

For non-discrete state spaces, the constraints defined for each discrete state must be generalized to the continuous state space, or we may sample a large subset of the state space and restrict the constraints to these sampled states only.

Under the assumption that each trajectory in the demonstration reflects a distinct policy, Ratliff et al.'s [25] maximum margin planning (MMP) associates each trajectory $\tau \in \mathcal{D}$ with an MDP. While these MDPs could differ in their state and action sets, and the transition functions, they share the same reward function. MMP and most recent methods express the reward function as a linear and weighted sum of basis functions as shown in Eq. 3. MMP's loss function quantifies the closeness between the learned and demonstrated behaviors by utilizing the state visitation frequency ψ as defined in Eq. 2 and a loss vector l_i that defines the learner's loss due to mismatch in state visitation frequencies between the demonstrated and currently learned behaviors:

$$\mathcal{L}(\psi, \psi_i) = l_i^T \psi. \tag{7}$$

The reward weight vector is obtained by solving a quadratic program that is constrained to have a positive margin between the expected value of the policy from the observed behavior and the expected values of all other policies combined with the loss function of Eq. 7. The former is obtained by multiplying the empirical state visitation frequency from the observed trajectory with the weighted feature function values $\boldsymbol{\phi}(\cdot)$ obtained using the trajectory. The solution is the weight vector that makes the expected value of the optimal policy to be closest to the expected value of the observed policy, for each of the MDPs.

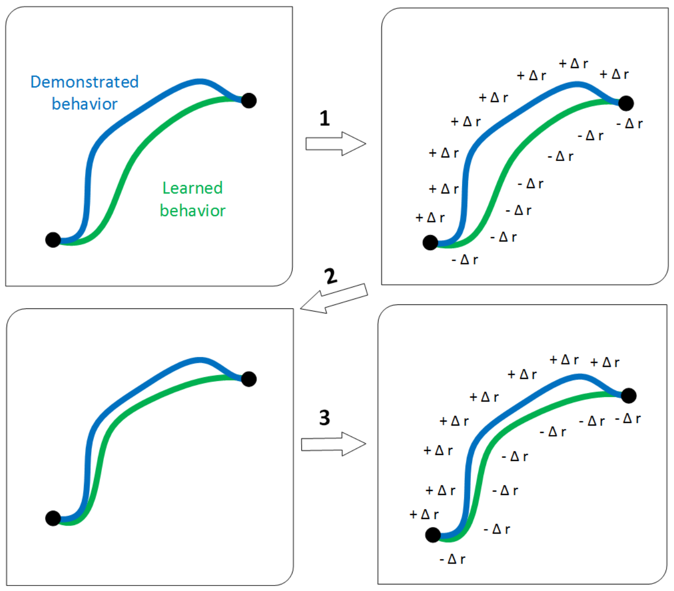

Figure 5: An iteration of LEARCH in the feature space $\Phi = \{\phi(s,a)|\forall(s,a) \in S \times A\}$ (Fig 3 in Ratliff et al. [17] excerpted with permission). The method considers a reward function as negative of a cost function. Blue path depicts the demonstrated trajectory, and green path shows the maximum return (or minimum cost) trajectory according to the current intermediate reward hypothesis. Step 1 is the determination of the points where reward should be modified, shown as $-\Delta r$ for a decrease and $+\Delta r$ for an increase. Step 2 generalizes the modifications to entire space Φ computing the next hypothesis $\hat{R}_E$ and corresponding maximum return trajectory. Next iteration repeats these two steps. Color should be used for this figure in print.

Ratliff et al. [17] improves on MMP in a subsequent method called *learn to search (*LEARCH*)*. Figure 5 explains how an iteration of LEARCH increases the cost (decreases the reward) for the actions that cause deviation between the learned and demonstrated behaviors. For optimization, LEARCH uses an exponentiated functional gradient descent in the space of reward functions (represented as cost maps). One of the techniques in LEARCH comprises the mechanism introduced by Silver et al. [26] to handle suboptimal or infeasible demonstrations.

4.1.2. Apprenticeship learning

Two methods [27] under this general label also perform maximum margin optimization but in a different way. Noting that the learner does not typically have access to the expert's policy, *max-margin* and *projection* take a demonstration (defined in Def. 2) as input. Both methods continue to represent the reward function as a linear, weighted sum of feature functions.

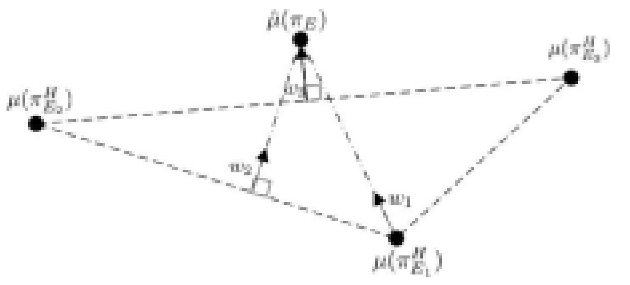

Figure 6: Iterations of the max-margin method computing the weight vector, **w**, and the feature expectations, μ^ϕ. $\hat{\mu}^\phi(\pi_E)$ is the estimation of the feature counts $\mu^\phi(\pi_E)$ of the expert. w_j is the learned weight vector in the j^{th} iteration and $\pi^H_{E_j}$ is the corresponding optimal policy for intermediate hypothesis. This figure is redrawn with slight changes from the one in Abbeel and Ng [27].

These methods seek a reward function that minimizes the margin between the feature expectations of a policy computed by the learner (Eq. 4) and the empirically computed feature expectations from the expert's trajectory (Eq. 6); we refer to this margin as the *value loss*. Both methods iteratively tune weight vector $\boldsymbol{w}$ by computing a policy as an intermediate hypothesis at each step and using it to obtain intermediate feature counts. These counts are compared with the empirical feature counts $\hat{\mu}^\phi(\pi_E)$ of expert, as shown in Fig. 6 and the weights are updated. Abbeel and Ng [27] point out that the performance of these methods is contingent on matching the feature expectations, which may not yield an accurate $\hat{R}_E$ because feature expectations are based on the policy. An advantage of these methods is that their sample complexity depends on the number of features and not on the complexity of expert's policy or the size of the state space.

A variant of the projection method described above is Syed et al's. multiplicative weights for apprenticeship learning (MWAL) [28]. The initial model and input are the same in both methods. However, MWAL presents a learner as a max player choosing a policy and its environment as an adversary selecting a reward hypothesis. This formulation transforms the value-loss objective in the *projection method* to a *minimax* objective for a zero-sum game between the learner and its environment, with weights $\boldsymbol{w}$ as output.

An alternative to minimizing the value-loss in the projection method is to minimize the probability difference between stochastic policies $\hat{\pi}_E(a|s)$ - $\pi_E(a|s)$ for each state. Viewing a policy as a Boltzmann function of the action-value, Neu and Szepesvari [13] present HYBRID-IRL as a gradient descent method that searches the space of reward hypotheses. As the behavior of expert is available instead of its policy, the difference above is computed using the empirically computed frequencies of the visitations of each state (Eq. 2) and the frequencies of taking specific actions in that state as given by the hypothesized policy.

4.2. Entropy Optimization

IRL is essentially an ill-posed problem because multiple reward functions can explain the expert's behavior. The maximum margin approaches of Section 4.1 introduce a bias into the learned reward function. Thus, multiple methods take recourse to the maximum entropy principle [29] to obtain a distribution over potential reward functions while avoiding any bias. The distribution that maximizes entropy makes minimal commitments beyond the constraints and is least wrong.

4.2.1. Maximum entropy IRL

Ziebart et al.'s maximum entropy IRL (MAXENTIRL) [15] recovers a distribution over all trajectories $Pr(\tau)$, which has the maximum entropy among all such distributions under the constraint that feature expectations of learned policy match that of demonstrated behavior. Mathematically, this problem can be formulated as a convex, nonlinear optimization:

$$\begin{aligned}
&\max_{\Delta} \left(-\sum_{\tau \in \mathcal{D}} Pr(\tau) \log Pr(\tau) \right) \\
&\textbf{subject to} \quad \sum_{\tau \in \mathcal{D}} Pr(\tau) = 1 \\
&\sum_{\tau \in \mathcal{D}} Pr(\tau) \sum_{t=1}^{\infty} \gamma^t \phi_k(s_t, a_t) = \hat{\mu}^{\phi_k}(\tau) \qquad \forall k
\end{aligned} \tag{8}$$

where Δ is the space of all possible distributions $Pr(\tau)$. Lagrangian relaxation allows us to bring both constraints into the objective function and the dual is solved to obtain the reward weight vector $\boldsymbol{w}$ by utilizing exponentiated gradient descent.

Extending this maximum entropy IRL to a nonlinear reward function, Wulfmeier et al. [30] point out that the difference in the feature expectations can be used as an error signal or loss function for backpropagation in a neural network approximation of the reward function.

A disadvantage of the trajectory-based formulation of (8) is that it becomes intractable for long trajectories because the space of trajectories grows exponentially with the length. A related method [31] formulates the problem as one of finding a distribution over the space of deterministic policies Π, which has the maximum entropy. In other words, this method maximizes $-\sum\limits_{\pi \in \Pi} Pr(\pi)\, log\, Pr(\pi)$ over the space of distributions of policies while matching the convex combination of feature expectations from all policies with those from the observed expert's trajectory.

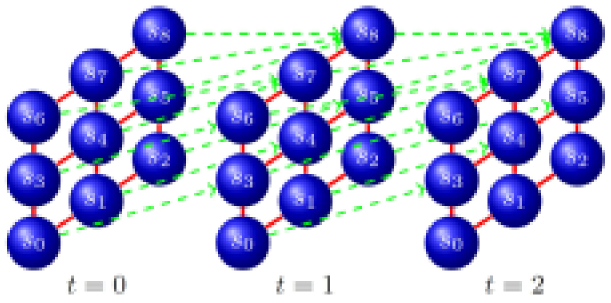

Figure 7: A Markov random field which favors the policies choosing similar actions in neighboring states. This structure generalizes the feature information beyond individual states. The figure is reprinted from [31] with permission from publisher.

Due to the constraint of matching feature expectations, the process is equivalent to maximizing the likelihood of expert's behavior under the maximum entropy distribution. The set of deterministic policies has size $\mathcal{O}(|A|^{|S|})$, which is independent of the length and number of demonstrated trajectories, but it could get considerably large with increase in the size of the state space.

Applications may exhibit *structure* that most neighboring states have similar optimal actions. To exploit this information, Boularias et al. [31] introduced special features in the constraints to make the distribution favor policies for which neighboring states have similar optimal actions. This method called structured apprenticeship learning results in a Markov random field like distribution (see Fig. 7).

4.2.2. Relative entropy IRL

Entropy optimization is also useful when the transition dynamics are not known. In REIRL [32], the relative entropy (known as Kullbach-Leibler divergence [33]) between two distributions on trajectories is minimized and the corresponding reward function is obtained.

From the space of all possible distributions, learner chooses an arbitrary distribution on trajectories with feature expectations within pre-determined closeness to true empirical feature expectations of the demonstration. Using a given baseline policy π_b, learner empirically computes another distribution by sampling trajectories. REIRL then learns the reward function by minimizing the relative entropy between these two probability distributions. Notably, the input behavior need not be optimal for the method. An analytical solution of this optimization problem requires a known transition function. To estimate the latter, the learner uses importance sampling.

Recently, Finn et al. [34] extend the above sample based estimation approach (Guided cost learning, GCL) to model-free maximum entropy optimization by allowing a neural network to represent the reward function [30]. Replacing an uninformed baseline policy that effectively leads to a uniform distribution over trajectories, sample generation is guided by policy optimization that uses the current reward hypothesis to update the baseline policy which in turn guides the sample generation. In this way, RL is interleaved with IRL to generate more trajectories that are supported by a larger reward in the reward function.

4.3. Bayesian Update

An important class of methods treat the state-action pairs in a trajectory as observations that facilitate a Bayesian update of a prior distribution over candidate reward functions. This approach yields a different but principled way for IRL that has spawned various improvements.

4.3.1. Bayesian IRL

A posterior distribution over candidate reward functions is obtained using the following Bayesian update:

$$P(\hat{R}_E|\tau) \propto P(\tau|\hat{R}_E)\ P(\hat{R}_E)\ \ \forall\tau. \tag{9}$$

where $P(\tau|\hat{R}_E) = \prod_{\langle s,a\rangle\in\tau} Pr(\langle s,a\rangle|\hat{R}_E)$.

Ramachandran and Amir [35] define the likelihood $P(\langle s,a\rangle|\hat{R}_E)$ as a logit distribution of the Q-value of the state-action pair:

$$P(\langle s,a\rangle|\hat{R}_E) \propto e^{\left(\frac{Q^*(s,a;\hat{R}_E)}{\beta}\right)} \tag{10}$$

where β controls the randomness in action selection (lower the β, more exploratory is the action). Given a candidate reward hypothesis, some state-action pairs are more likely than others as given by the likelihood function. As the space of reward functions is continuous, Ramachandran and Amir present a random walk algorithm (BIRL) for implementing the update and obtain a sampled approximation of the posterior in Eq. 9. Notice that Eq. 10 requires the Q-value function of the MDP completed using each $\hat{R}_E$. In this regard, Francisco et al. [36] utilize bisimulation [37] to measure the closeness of two MDPs in order to avoid solving an MDP that is similar to a previously solved one.

An extension of this method measures the state-specific entropy of the posterior over the reward functions [38]. The method defines a set $\{\hat{R}_E(p)\}$ of reward functions such that for a given state-action pair under a reward function $\hat{R}_E$, $\pi_E(s,a;\hat{R}_E) = p$. The posterior distribution over rewards induces a distribution over intervals of probability p: $I_k \in [0,1]$ where $I_k = (\frac{k}{K}, \frac{k+1}{K}], k \in \{0,1,\ldots(K-1)\}$. State-based entropy is then computed as the entropy of this distribution for some state and averaged over all actions.

In an instance of introducing active learning to IRL, the learner in the above method can query the expert for sample demonstrations in states exhibiting high entropy with the aim of learning a posterior that is well informed at all states. Prior knowledge about the structural similarities between states improves the effectiveness of this approach.

4.3.2. Bayesian nonparametrics

Michini et al. [39, 40] introduce CRP-BNIRL (Chinese restaurant process Bayesian nonparametric IRL), which is a straightforward application of nonparametric clustering by letting a Chinese Restaurant process partition a trajectory into as many subtrajectories as needed and by using Bayesian inference to assign a reward function to each subtrajectory to best fit the observed data. Subtrajectories are presumed to represent subtasks with their own subgoals. However, large state spaces and long trajectories make obtaining the demonstration-likelihood computationally intensive. Interestingly, the authors use real-time dynamic programming and action comparison with existing closed-loop controllers to avoid discretizing the state and action spaces.

4.4. Classification and Regression

Classical machine learning techniques such as classification and regression have also played a significant role in IRL. However, these methods are challenged by the fact that IRL is not a straightforward supervised learning problem. Nevertheless, the methods below show that IRL can be cast into this framework.

4.4.1. Multi-label classification

Klein et al. [41] views IRL as a multi-label classification problem (SCIRL) in which the demonstration is utilized for training, and the state-action pairs are data-label pairs. The scoring function of the classifier is the action-value function $Q^{\pi_E}(s,a)$, which is obtained using feature expectations, $Q^{\pi}(s,a) = \sum_{k=1}^{K} w_k \, \mu^{\phi_k}(\pi)(s,a)$ (see Eq. 4). Thus, the weight vector $\boldsymbol{w}$ becomes the shared parameter between the Q- and reward functions. A multi-label classification algorithm computes the solution by inferring a weight vector that minimizes the classification error.

An extension of the above method, called CSI [42], estimates the transition probabilities in the computation of the Q-function if they are unknown. CSI utilizes standard regression on a simulated demonstration dataset to estimate the transition model and thereafter learn the reward function.

4.4.2. Feature construction using regression

A unique effort to automatically learn the feature functions [43] begins with *primitive* binary feature functions that form components of the final features used in the reward function. We may use a logical conjunction of the primitive binary features. Starting with an empty set of features, the method labeled FIRL iteratively updates reward hypothesis $\hat{R}_E$ and adds hypothesized feature functions ϕ^H to it. The search involves building a minimal regression tree on the state space that encodes the conjunctions, resulting in new binary feature functions which make $\hat{R}_E$ consistent with demonstrated behavior.

4.5. Other Miscellaneous Techniques

We briefly review a few other IRL methods that do not fall in the broad categories discussed previously.

Gaussian Process IRL (GPIRL) uses a Gaussian process to approximate R_E as a nonlinear function of base features ϕ, $R_E = f(\phi)$ [44]. Rather than learning a single reward function, a reward hypothesis here is the mean of a parametric Gaussian posterior over possible rewards. The method maximizes the likelihood of the actions in the demonstration in order to compute the mean.

Rather than obtain the maximum-a-posteriori estimate of the expert's reward function obtained from Bayesian learning (as by methods in Section 4.3), Vroman et al. [45] choose a reward function that leads to the maximum likelihood estimate. In MLIRL, the formulation of the maximum likelihood estimate is analogous to the posterior in Bayesian learning. However, this method remains challenged by degeneracy and ill-posedness of the IRL problem.

Syed et al. [46] presents a linear program for imitating the expert's policy, which scales to MDPs with larger state spaces and many basis feature functions. Babes-Vroman et al. [45] takes the dual of this linear program to allow reward learning in the method (calling it LPIRL) while keeping its computational benefits intact.

Kalakrishnan et al. [47] and Aghasadeghi et al. [48] each introduced path integral [49] based approaches to learn continuous-state continuous-action reward functions by sampling a local region around each demonstrated trajectory. The two methods differ only in how the sample set from the trajectory spaces is obtained for approximating the reward function. Neither approach considers the trajectories that deviate from the demonstration and thereby incur significant control input. The techniques apply to high-dimensional problems due to the assumed local optimality of demonstration.

In this section, we briefly described the foundational IRL methods. Table A.1 in the Appendix succinctly lists all the methods along with the parameters that need to be learned and the divergence metric that is minimized by each method. This facilitates a convenient comparison across the techniques and helps in aligning the methods with the template given in Algorithm 1.

5. Mitigating Challenges

This section elaborates how the methods reviewed in previous section mitigate the various challenges introduced in Section 3. This will help the reader make an informed choice about the method that may address the challenges in her domain. In addition, we also note techniques that purposefully extend some of the foundational methods to address a specific challenge.

5.1. Improving the Accuracy of IRL

IRL's accuracy depends on several aspects of the learning process. Most existing methods aim at ensuring that the input is accurate, reducing the ambiguity among solutions, improving feature selection, and offering algorithmic performance guarantees.

5.1.1. Learning from Faulty Input

Suboptimal actions characterize a perturbed demonstration. Some methods such as REIRL stay robust to perturbations whereas other IRL methods may learn inaccurate feature weights [10] or predict the action poorly [25]. Methods such as MAXENTIRL, BIRL, MLIRL, and GPIRL use probabilistic frameworks to account for the perturbation. For example, MLIRL allows tuning of its model parameter β in Eq. 10 to allow more randomness into the learned policy π_E when the demonstrated behavior is expected to be noisy and suboptimal [45]. On the other hand, methods such as MMP and LEARCH introduce slack variables in their optimization objective for this purpose. Using the application of helicopter flight control, Ziebart et al. [50] show that the robustness of MAXENTIRL against an imperfect demonstration is better than that of MMP.

To specifically address noisy input, Coates et al. [51] introduce a model-based technique of trajectory learning that de-noises the noisy demonstration by learning a generative trajectory model and then utilizing it to produce noise-free trajectories. Apprenticeship learning is subsequently applied to the resultant noise-free but unobserved trajectories [5]. As an instance of learning from suboptimal input, Shiarlis et al. [52] performs IRL with demonstrations that fail to complete a task. Melo et al. [53] focused on a formal analysis and characterization of the space of solutions for the case when some actions in a demonstration are not optimal (a perturbation in the distribution modeling the expert's policy) and when demonstration does not include samples in all states.

A suboptimal demonstration may also be a trajectory with length much longer than desired. As we mentioned in Section 4.1, MMP converges to a solution by minimizing the cost of simulated trajectories diverging from the demonstrated ones. This divergence is computed by noting the difference in state-visitation frequencies of two trajectories. However, MMP attempts this minimization for a suboptimal demonstration as well, which can be avoided if the learning method distinguishes an unusually long demonstration from the optimal ones. Silver et al. [26, 17] specifically target this issue by implementing an MMP-based imitation learning approach that applies a functional gradient normalized by the state-visitation frequencies of a whole trajectory (see Fig. 8 for an illustration).

5.1.2. Ambiguity and Degeneracy of Reward Hypotheses

Various methods mitigate this challenge of ambiguity and degeneracy by better characterizing the space of solutions. This includes using heuristics and prior domain knowledge, and adding optimization constraints.

MMP and MWAL avoid degenerate solutions by using *heuristics* that favor the learned value $V^{\hat{\pi}_E}$ to be close to expert's V^{π_E}. Specifically, MMP avoids degeneracy by using the loss function of Eq. 7 in the objective, which the degenerate $\hat{R}_E = 0$ can not minimize because the function is proportional to state-visitation frequencies [24]. HYBRID-IRL avoids degeneracy in the same way as MMP, and makes the solution unique by preferring a reward function that corresponds to a stochastic policy $\hat{\pi}_E$ with action selection same as the expert's ($\hat{\pi}_E(a|s) \approx \pi_E(a|s)$). Naturally, if no single non-degenerate solution

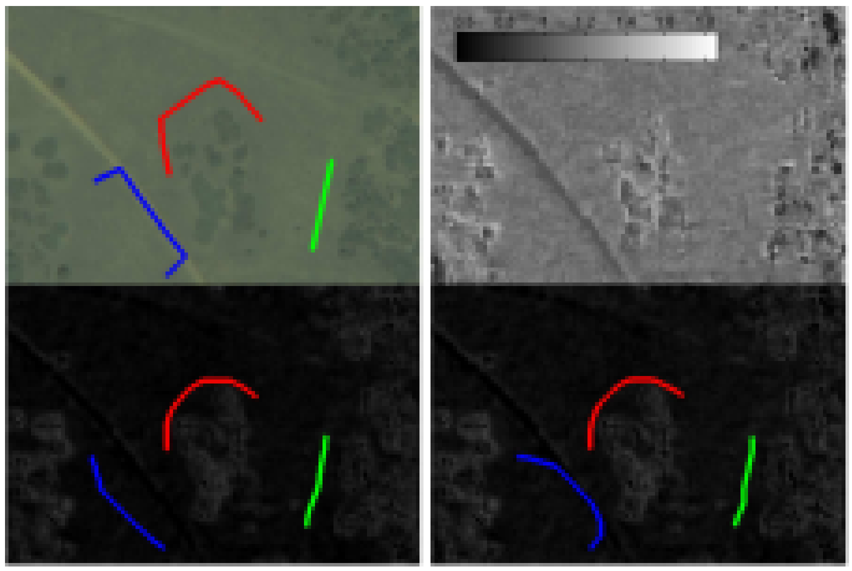

Figure 8: Learning with a perturbed demonstration in an unstructured terrain. Figure reprinted [26] with permission from MIT press. An expert provides three demonstration trajectories - red, blue, and green [top left]. The portion of terrain traveled by a presumably achievable red trajectory should have low cost (high reward) as the expert is presumably optimal. But the path is not optimal. It is not even achievable by any planning system with predefined features because passing through the grass is always cheaper than taking a wide berth around it. The assumed optimality of expert forces the optimization procedure in IRL methods to lower the cost (increase the reward) for features encountered along the path, i.e., features for grass. This influences the learning behavior in other paths such as the blue path [bottom left]. Using a normalized functional gradient [26] makes this lowering effect vanish [bottom right].

makes the demonstration optimal, ambiguous output cannot be entirely avoided using these methods [54].

Many methods embrace the ambiguity by modeling the uncertainty of the hypothesized rewards as a probability distribution over reward functions or that over the trajectories corresponding to the rewards. In this regard, MAXENTIRL infers a single reward function by using a probabilistic framework that avoids any constraint other than making the value-loss zero, $V^{\hat{\pi}_E} = V^{\pi_E}$. On the other hand, the maximum-a-posteriori objective of Bayesian inference techniques and GPIRL limit the probability mass of the posterior distribution to the specific subset of reward functions that supports the demonstrated behavior. This change in probability mass shapes the mean of the posterior, which is output by these methods. Active learning of the reward function uses the state-conditional entropy of the posterior to select the least informative states [38] and query for further information in those states. The selection mechanism builds on BIRL and reduces the ambiguity of the solution as compared to BIRL. In general, these methods add optimization constraints and exploit domain knowledge to distinguish between hypotheses.

5.1.3. Theoretically Guaranteed Accuracy

From a theoretical viewpoint, some methods have better performance guarantees than others. The maximum entropy probability distribution over space of policies (or trajectories) minimizes the worst-case expected loss [55]. Consequently, MAXENTIRL learns a behavior which is neither much better nor much worse than the expert's [56]. However, the worst-case analysis may not represent the performance in practice because the performance of optimization-based learning methods can be improved by exploiting favorable properties of the application domain. Classification based approaches such as CSI and SCIRL admit a theoretical guarantee for the quality of $\hat{R}_E$ in terms of optimality of the learned behavior $\hat{\pi}_E$, given that both classification and regression errors are small. Nevertheless, these methods may not reduce the loss as much as MWAL as the latter is the only method, in our knowledge, which has no lower bound on the incurred value-loss [28].

Some methods also analyze and bound the ILE metric for a given confidence of success and a given minimum number of demonstrations. The analysis relies on determining the value of a policy using feature expectations $\mu^{\phi}(\pi)$ [27, 15, 23] or state visitation frequencies $\psi^{\pi}(s)$ [13, 15, 25, 45] as shown in Eq. 5.

For any method based on feature expectations or state-visitation frequencies, there exists a probabilistic upper bound on the bias in $\hat{V}^{\pi_E}$ and thereby on ILE for a given minimum sample complexity [27, 57, 58]. These bounds also apply to methods such as MMP, HYBRID-IRL, and MAXENTIRL that use state-visitation frequency. Subsequently, each derived bound on bias can be used to analytically compute the maximum error in learning for a given minimum sample complexity Lee et al. [59] change the criterion (and thereby the direction) for updating the current solution in MAX-MARGIN and PROJECTION methods to formally prove an improvement in the accuracy of the solution as compared to that of the original method.

5.2. Generalizability

While early approaches such as apprenticeship learning required a demonstration that spanned all states, later approaches sought to explicitly learn a reward function that correctly represented expert's preferences for unseen state-action pairs, or one that is valid in an environment that mildly differs from the input. An added benefit is that such methods may need less demonstrations. GPIRL can predict the reward for unseen states lying within the domains of the features for a Gaussian process. Furthermore, FIRL can use the learned reward function in an environment that is slightly different from the original environment used for demonstration but with base features similar to those in the original environment. Similarly, GCL admits an enhanced generalization by learning new instances of a previously-learned task without repeated reward learning. BIRL with bisimulation achieves improved generalization by partitioning the state space based on a relaxed equivalence between states.

Munzer et al. [60] extend the classification-regression steps in CSI to include relational learning in order to benefit from the strong generalization and transfer

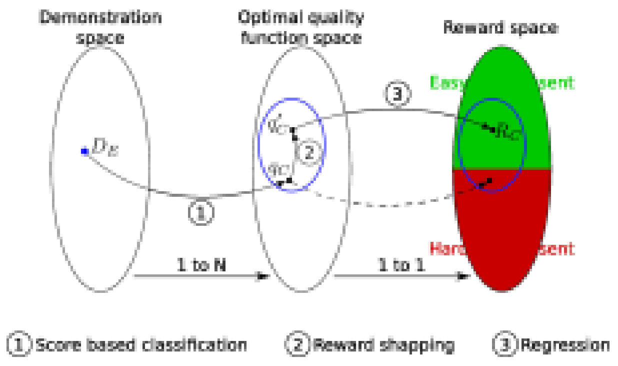

Figure 9: The 3 step process of IRL in relational domain - classification outputs a score function, reward shaping optimizes the output, and regression approximates reward function $\hat{R}_E$ corresponding to the optimal score function. Included with permission from authors.

properties that are associated with relational-learning representations. The process shapes the reward function using the score function as computed by the classification (see Fig. 9 for more details).

5.3. Lowering Sensitivity to Prior Knowledge

In this section, we discuss techniques in the context of the challenge introduced in Section 3.3. Performance of the foundational methods such as PROJECTION, MAX-MARGIN, MMP, MWAL, LEARCH, and MLIRL are all highly sensitive to the selection of features – a challenge pointed out in Section 3.3. While we are unaware of methods that explicitly seek to reduce their dependence on feature selection, some methods are less impacted by virtue of their approach. These include HYBRID-IRL that uses policy matching and all maximum entropy based methods tune distributions over the policies or trajectories, which reduces the impact that feature selection has on the performance of IRL [24].

Apart from selecting appropriate features, the size of the feature space influences the error in learned feature expectations for the methods that rely on $\hat{V}^{\pi_E}$, e.g., PROJECTION, MMP, MWAL, MAXENT, and IRL. If a reward function is linear in the form of Eq. 3 and the magnitude of its k features is bounded from the above, then the probable bound on the error scales linearly with k [15]. However, maximum entropy based methods show an improvement in this aspect with $O(\log k)$ dependence.

5.4. Analysis and Reduction of Complexity

We discuss ways by which the various IRL methods reduce the time and space complexity of an iteration, and mitigate the input sample complexity.

While emphasis on reducing time and space complexity is generally lacking among IRL techniques, a small subset does seek to reduce the time complexity. An analysis of BIRL shows that computing the policy π_E using the mean of the posterior distribution is computationally more efficient than the direct minimization of expected value-loss over the posterior [35]. Specifically, the Markov chain with a uniform prior that approximates the Bayesian posterior converges in polynomial time. Performing BIRL via bisimulation, a method noted in Section 4.3, also exhibits low computational cost because it need not solve equivalent intermediate MDPs and the computation of the bisimulation metric over space S occurs once regardless of the number of applications. Next, MWAL requires $\mathcal{O}(\ln k)$ (k is number of features) iterations for convergence, which is lower than $\mathcal{O}(k \ln k)$ for the PROJECTION method. Although an iteration of FIRL is slower than both MMP and PROJECTION due to the computationally expensive step of regression, FIRL converges in fewer iterations as compared to latter.

Some optimization methods employ more affordable techniques of gradient computations. In contrast with the fixed-point method in HYBRID-IRL, the approximation method in Active BIRL has a cost that is polynomial in the number of states. For maximum entropy based parameter estimation, gradient-based methods (e.g., BFGS [61]) outperform iterative scaling approaches [62].

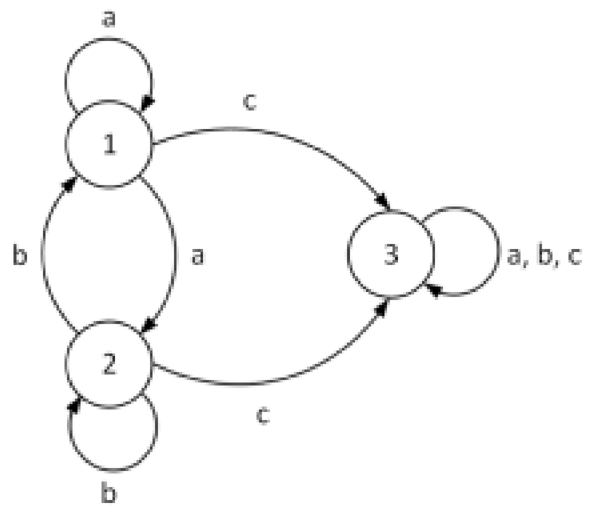

Figure 10: State equivalence in an MDP with states $S = \{1, 2, 3\}$ and actions $A = \{a, b, c\}$. The similarity in actions and transitions for states 1 and 2 makes them equivalent. Therefore, the selection of optimal actions through expert's policy π_E will be similar in both the states. Demonstration of c in one implies the optimality of c in other. The illustration redraws Fig. 1 in Francisco et al. [36].

Active BIRL offers a benefit over traditional BIRL by exhibiting reduced sample complexity. This is because it seeks to ascertain the most informative states where a demonstration is needed, and queries for it. Consequently, less demonstrations are needed and the method becomes more targeted. Of course, this efficiency exists at the computational expense of interacting with the expert. Model-free REIRL uses fewer samples (input trajectories) as compared to alternative methods including a model-free variant of MMP [32]. In a different direction, BIRL with bisimulation exploits any equivalence among the states to reduce the sample complexity, as shown in Fig. 10.

5.4.1. Continuous State Spaces

While most approaches for IRL target discrete state spaces, a group of prominent methods that operate on continuous state spaces are path integral based approaches, PI-IRL. These aim for local optimality of demonstrations to avoid the complexity of full forward learning in continuous space. This approximation makes it scale well to high-dimensional continuous spaces and large demonstration data. Although the performance of path integral algorithms is sensitive to the choice of samples in the demonstration, they show promising progress in scalability. Further, to avoid an expensive discretization in large continuous spaces, CRP-BNIRL approximates the demonstration likelihood by comparing actions to an existing closed-loop controller. Kretzschmar et al. [11] apply MAXENTIRL to learn the probability distribution over navigation trajectories of interacting pedestrians using a subset of their continuous space trajectories. A mixture distribution models both, the discrete as well as the continuous navigation decisions.

5.4.2. High Dimensional and Large Spaces

In IRL methods such as MAXENTIRL and BIRL, the probability of choosing actions in demonstration $\tau \in \mathcal{D}$ is

$$P(\tau|\hat{R}_E) = \frac{1}{Z} e^{\left(\sum_{t=0}^{\infty} Q(s^t, a^t; \hat{R}_E)\right)}. \tag{11}$$

The complexity of computing the partition function Z increases exponentially with the dimensionality of the state space, because it requires finding the complete policy under the current solution ($\hat{R}_E$) [54].

Approaches for likelihood approximation in a high-dimensional state space include the use of importance sampling by REIRL, down-scaling the state space using low-dimensional features [63], and the assumption by PI-IRL that demonstrations are locally optimal. For optimizations involved in maximum entropy methods, limited memory variable metric optimization methods such as L-BFGS are shown to perform better than other alternatives because they implicitly approximate the likelihood in the vicinity of the current solution [62].

Instead of demonstrating complete trajectories for large tasks, a problem designer may decompose the task hierarchically. An expert may then easily give demonstrations at different levels of implementation. The modularity of this process significantly reduces the complexity of learning. For example, Kolter et al. [64] apply such task decomposition toward learning quadruped locomotion by scaling IRL from low- to high-dimensional spaces. Likewise, Rothkopf et al. [65] utilize the independence between the components of a task – , each modeled using a stochastic reward function of its own – to introduce decomposition in BIRL. By decomposing a task into subtasks wherever possible, CRP-BNIRL allows a parallelized pre-computation of an approximate action-value function.

We may speed up forward learning by quickly computing the values of the intermediate policies learned in IRL. The linear program formulation of LPIRL makes solving the MDP less expensive in large state spaces with many basis functions (ϕ for $R_E = w^T\phi$). Similarly, CSI and SCIRL do not need to solve MDPs repeatedly because they update the previous solution by exploiting the structure imposed on MDP by their classification-based models. CRP-BNIRL avoids repeated forward learning by limiting the computation to those states which improve the action values around the observations. The process depends on the size of the input demonstration only rather than that of the complete state space.

6. Extensions of Basic IRL

Having surveyed the foundational methods for IRL in Section 4 and discussed how they and other methods mitigate the various challenges in Section 5, we now discuss important ways in which the assumptions of the basic IRL problem have been relaxed to enable advances toward real-world applications.

6.1. Incomplete and Imperfect Observations

Learners in the real world must deal with noisy sensors and may not perceive the full demonstration trajectory. For example, the merging car B in our illustrative example of Fig. 1 described in Section 1 may not see car A in the merging lane until it comes into its sensor view. This is often complicated by the fact that car B's sensor may be partially blocked by other cars in front of it, which further occludes car A. Additionally, the expert itself may possess noisy sensors and may not observe its own state perfectly.

6.1.1. Extended Definition

The property of incomplete and noisy observations by the learner modifies the traditional IRL problem and we provide a new definition below for completeness.

Definition 3 (IRL with imperfect perception)**.** Let $\mathcal{M}\backslash_{R_E}$ represent the dynamics of the expert E. Let the set of demonstrated trajectories be, $\mathcal{D} = \{\langle(s_0, a_0), (s_1, a_1), \ldots (s_j, a_j)\rangle_{i=1}^{N}\}$, $s_j \in Obs(S)$, $a_j \in Obs(A)$, $i, j, N \in \mathbb{N}$. Either some state-action pairs of a trajectory, $\tau \in \mathcal{D}$, are not observed or some of the observed state-action pairs could be different from the actual demonstrated ones. Thus, let $Obs(S)$ and $Obs(A)$ be the subsets of states and actions respectively that are observed. Then, determine $\hat{R}_E$ that best explains either given policy π_E or the demonstrated trajectories.

Figure 11 revises the schematic for the traditional IRL shown in Fig. 3 to allow for incomplete and imperfect observations. Observing the trajectories imperfectly may require the learner to draw inferences about the unobserved state-action pairs or the true ones from available information, which is challenging.

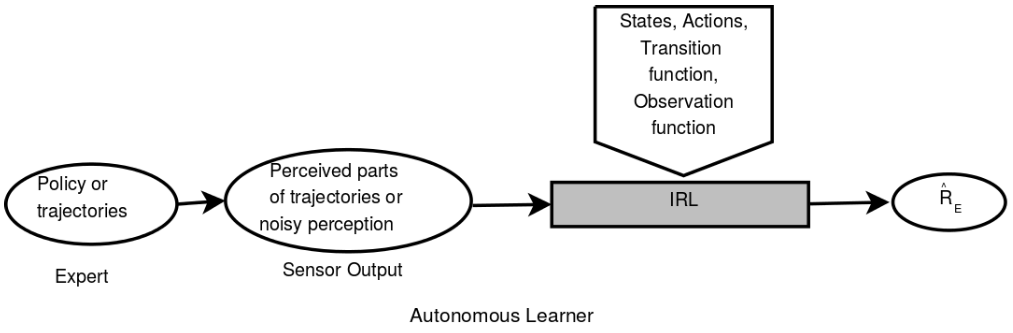
States, Actions, Transition function, Observation function
Policy or trajectories
Perceived parts of trajectories or noisy perception
IRL
$\hat{R}_E$
Expert
Sensor Output
Autonomous Learner

Figure 11: IRL with imperfect perception of the input trajectory. The learner is limited to using just the perceived portions.

6.1.2. Methods

Bogert et al. [20] introduce IRL* for settings where the learner is unable to see some state-action pairs of the demonstrated trajectories due to occlusion. The maximum entropy formulation of Boularias et al. [31] is generalized to allow feature expectations that span the observable state space only. This method is applied to a new domain of multi-robot patrolling as illustrated in Fig. 12.

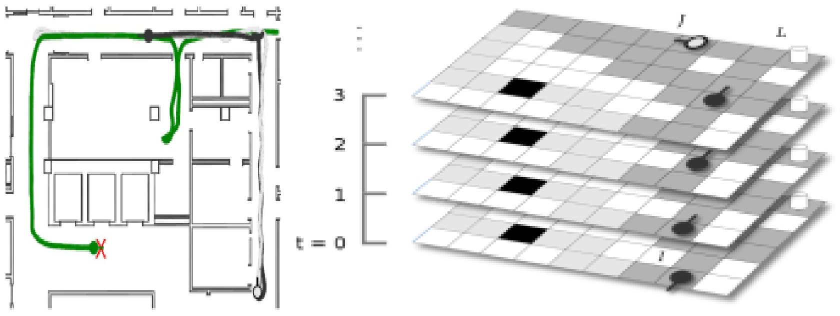

Figure 12: Prediction of experts' behaviors using multi-robot IRL* [20] in a multi-robot patrolling problem (left). Learner L (green) needs to cross hallways patrolled by experts I (black, reward R_{E_1}) and J (gray, reward R_{E_2}). It has to reach goal 'X' without being detected. Due to occlusion, just portions of the trajectory are visible to L. After learning R_{E_1} and R_{E_2}, L computes their policies and projects their trajectories forward in time and space to know the possible locations of patrollers at each future time step. These projections over future time steps help create L's own policy as shown in the figure on the right.

The principle of latent maximum entropy [66, 67] allows us to extend the maximum entropy principle to problems with hidden variables. By using this extension, Bogert et al. [68] continued along the vein of incomplete observations and generalized MAXENTIRL to the context where a dimension of the expert's actions are hidden from the learner. For example, the amount of force applied by a human while picking ripe and unripe fruit usually differs but this would be hidden from an observing co-worker robot. An expectation-maximization scheme is introduced with the E-step involving an expectation of the hidden variables while the M-step performs the MAXENT optimization.

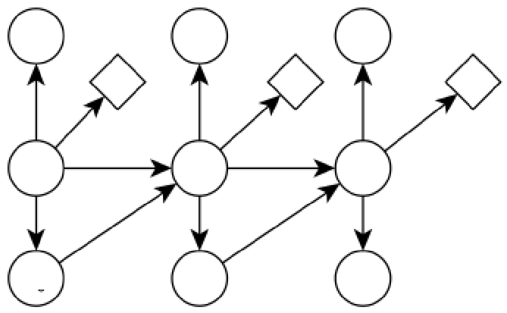

Figure 13: Hidden variable MDP. u_i is a noisy observation, by the learner, of the state s_i reached after taking action a_{i-1}. The source of illustration is [69]. The figure is shown here with permission from publisher.

Taking the context of noisy observations, a hidden variable MDP incorporates the probability of learner's noisy observation conditioned on the current state (u in Fig. 13), as an additional feature ϕ_u in the feature vector ϕ. Hidden variable inverse optimal control (HIOC) [69] then modifies MAXENTIRL to a problem where the dynamics are modeled by the hidden variable MDP with a linearly-weighted reward function. Consequently, the expression for the likelihood of expert's behavior incorporates the additional feature and its weight (ϕ_u, w_u). The tuning of weights during optimization also adjusts w_u to determine the reliability of the imperfect observations.

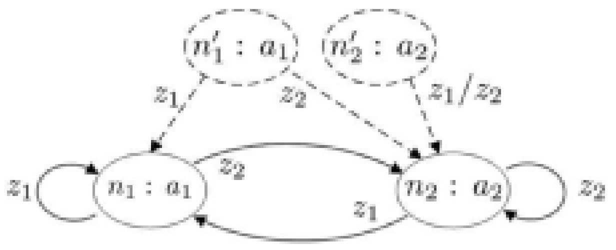

Figure 14: In this illustration, similar to the one in Choi et al. [23], consider a POMDP with two actions and two observations. π_E (solid lines) is a FSM with nodes $\{n_1, n_2\}$ associated to actions and edges as observations $\{z_1, z_2\}$. The one-step deviating policies (dashed lines) $\{\pi_i\}_{i=1}^2$ are policies which are slightly modified from π_E. Each π_i visits n'_i instead of n_i and then becomes same as π_E. The comparison of $\hat{V}^{\pi_E}$ with $\{V(\pi_i)\}_{i=1}^2$ characterizes the set of potential solutions. Since such policies are suboptimal yet similar to expert's, to reduces computations, they are preferable for comparison instead of comparing $\hat{V}^{\pi_E}$ with all possible policies.

Choi and Kim [23] take a different perspective involving an expert that senses its state imperfectly. The expert is modeled as a partially observable MDP

(POMDP) [70]. The expert's uncertainty about its current physical state is modeled as a belief (distribution) over its state space. The expert's policy is then a mapping from its beliefs to optimal actions. The method POMDP-IRL makes either this policy available to the learner or the prior belief along with the sequence of expert's observations and actions (that can be used to reconstruct the expert's sequence of beliefs). The POMDP policy is represented as a finite-state machine whose nodes are the actions to perform on receiving observations that form the edge labels. The learner conducts a search through the space of reward functions as it gradually improves on the previous policy until the policy explains the observed behavior. Figure 14 illustrates this approach. However, a well-known limitation of utilizing POMDPs is that the exact solution of a POMDP is PSPACE-hard, which makes them difficult to scale to pragmatic settings.

In POMDP-IRL, the expert may not observe its state perfectly. However, IRL* and HIOC differ from this setup. They model the learner observing the expert's state and action imperfectly whereas the expert is perfectly aware of its state.

6.2. Multiple Tasks

Human drivers often exhibit differing driving styles based on traffic conditions as they drive toward their destination. For example, the style of driving on the rightmost lane of a freeway is distinctly different prior to the joining of a merging lane, at joining of the lane, and post joining the lane. We may model such distinct behaviors of expert(s) as guided by differing reward functions. Consequently, there is a need to investigate methods that learn multiple reward functions simultaneously.

6.2.1. Extended Definition

To accommodate demonstrations involving multiple tasks, we revise the traditional IRL problem definition as given below.

Definition 4 (Multi-task IRL)**.** Let the dynamics of the expert(s) be represented by K MDPs each without the reward function where K may not be known. Let the set of demonstrated trajectories be, $\mathcal{D} = \{\langle(s_0, a_0), (s_1, a_1), \ldots, (s_j, a_j)\rangle_{i=1}^{N}\}$, $s_j \in S$, $a_j \in A$, $i, j, N \in \mathbb{N}$. Determine $\hat{R}_E^1$, $\hat{R}_E^2$, ..., $\hat{R}_E^K$ that best explain the observed behavior.

Figure 15 gives the schematic for this important IRL extension. Having to associate a subset of input trajectories to a reward function that likely generates it (also called the data association problem) makes this extension challenging. This becomes further complex when the number of involved tasks is not known.

6.2.2. Methods

Diverse methods have sought to address the problem defined in Def. 4, and we briefly review them below.

Babes-Vroman et al. [45] assume that a linear reward function of an expert can change over time in a chain of tasks. The method aims to learn multiple reward

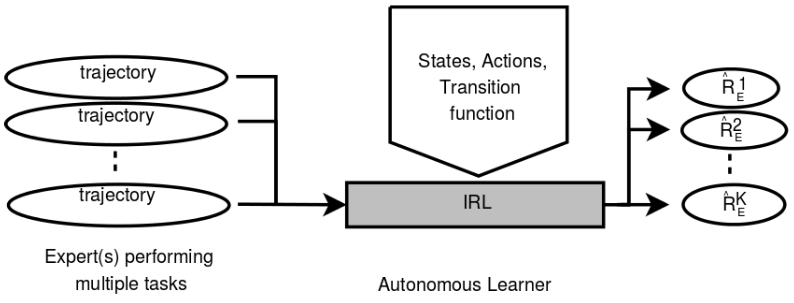

Figure 15: Multi-task IRL involves learning multiple reward functions. The input is mixed trajectories executed by a single or multiple expert(s) realizing behavior driven by different reward functions. The output is a set of reward functions, each one associated with a subset of input trajectories potentially generated by it.

functions with common features $\{R_i = w_i^T \phi\}_{i=1}^n, n \in \mathbb{N}$. Given prior knowledge of K, the solution is a pair of weight vector $\boldsymbol{w}_i \in \mathbb{R}^k$ and a correspondence probability for each reward function $\hat{R}_E^K$. This probabilistically ties a cluster of trajectories to a reward function. The process iteratively clusters trajectories based on current hypothesis, followed by implementation of MLIRL for updating the weights. This approach is reminiscent of using expectation-maximization for Gaussian data clustering.

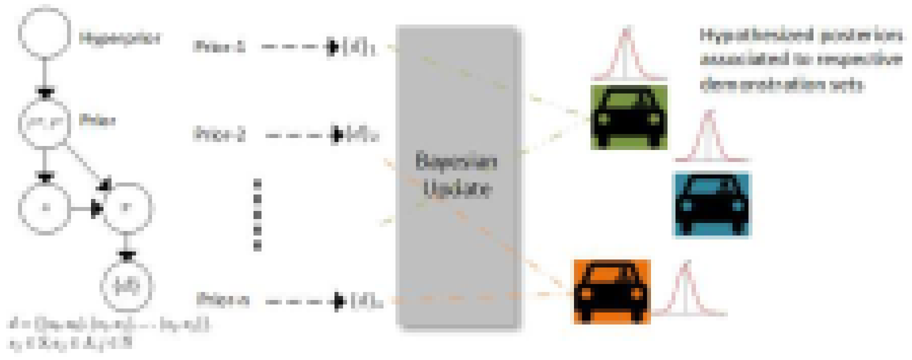
Hyperprior
Prior
Prior-1
Prior-2
Prior-n
Bayesian Update
Hypothesized posteriors associated to respective demonstration sets

Figure 16: Parametric multi-task BIRL.Tuning of independent hyperprior modifies the dependent priors P^R on reward functions and priors P^π on policies, the diagram shows it as a sequence of priors. The variation in priors is assumed to influence the observed trajectories $\{\tau\}_i, i \in \mathbb{N}$. Bayesian update outputs an approximated posterior over rewards and policies.

Continuing with the assumption of knowing K, BIRL can be generalized to a hierarchical Bayes network by introducing a hyperprior that imposes a probability measure on the space of priors over the joint reward-policy space. Dimitrakakis and Rothkopf [56] show how the prior is sampled from an updated posterior given an input demonstration. This posterior (and thus the sampled prior) may differ for an expert performing different tasks or multiple experts involved in different tasks. Within the context of our running example, Fig. 16

illustrates how this approach may be used to learn posteriors for multiple drivers on a merging lane of a freeway.

In contrast to parametric clustering, DPM-BIRL is a clustering method that learns multiple reward specifications from unlabeled fixed-length trajectories [71]. It differs from the previous methods in this section in that the number of experts are not known. Therefore, it addresses the problem in Def. 4 with K unknown. The method initializes a nonparametric Dirichlet process of priors over the reward functions and aims to assign each input trajectory to a reward function that potentially generates it, thereby forming clusters of trajectories. Learning occurs by implementing a Bayesian update to compute the joint posterior over the reward functions and the probabilistic assignments to clusters. The procedure iterates until the reward functions and clusters stabilize.

In settings populated by multiple experts, Bogert et al. [20] extend IRL* and MAXENTIRL to multiple experts who may interact, albeit sparsely. While the motion dynamics of each expert is modeled separately, the interaction is modeled as a strategic game between the two experts; this approach promotes scalability to many experts. Experts are assumed to play one of the Nash equilibria profiles during the interaction, although the precise one is unknown to the learner. Alternatively, all experts may be modeled jointly as a multiagent system. Reddy et al. [72] adopt this approach and model multiple interacting experts as a decentralized general-sum stochastic game. Lin et al. [73] propose a Bayesian method learning the distribution over rewards in a sequential zero-sum stochastic multi-agent game.

6.3. Incomplete Model

Definition 2 for IRL assumes full knowledge of the transition model T and the reward feature functions. However, knowing the transition probabilities that represent the dynamics or specifying the complete feature set is challenging and often unrealistic. Hand-designed features introduce structure to the reward, but they increase the engineering burden. Inverse learning is difficult when the learner is partially unaware of the expert's dynamics or when the known features do not sufficiently model the expert's preferences. Subsequently, the learner must estimate the missing components for inferring $\hat{R}_E$. Readers familiar with RL may notice that these extensions share similarity with model-free RL where the transition model and reward function features are also unknown.

6.3.1. Extended Definition

Definition 5 (Incomplete dynamics)**.** Let an MDP without reward, $\mathcal{M}\backslash_{R_E} = (S, A, \hat{T}, \gamma)$, represent the dynamics between an expert and its environment, where $\hat{T}$ specifies the probabilities for a subset of all possible transitions. The input is demonstration $\mathcal{D} = \{\langle (s_0, a_0), (s_1, a_1), \ldots (s_j, a_j)\rangle_{i=1}^{N}\}$, $s_j \in S$, $a_j \in A$, $i, j, N \in \mathbb{N}$ or expert's policy π_E. Then, determine reward $\hat{R}_E$ that best explains either the input policy π_E or the observed demonstration $\mathcal{D}$.

Figure 17 illustrates the corresponding generalized IRL pipeline. Next, we define the IRL problem when the set of basis feature functions is incomplete.

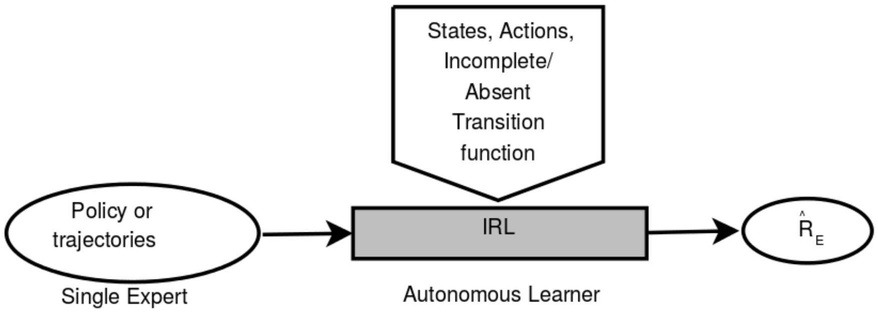

Figure 17: IRL with incomplete model of transition probabilities.

Definition 6. (Incomplete features) Let an MDP without reward, $\mathcal{M}\backslash_{R_E} = (S, A, T, \gamma)$, represent the dynamics of an expert and its environment. Let the reward function $\hat{R}_E = f(\boldsymbol{\phi})$ depend on the feature set $\boldsymbol{\phi}$. The input is demonstration $\mathcal{D} = \{\langle(s_0^i, a_0^i), (s_1^i, a_1^i), \ldots (s_j^i, a_j^i)\rangle_{i=1}^N\}$, $s_j \in S$, $a_j \in A$, $i, j, N \in \mathbb{N}$ or expert's policy π_E. If the given feature set $\boldsymbol{\phi}$ is incomplete, find the features and function $\hat{R}_E$ that best explains the input.

6.3.2. Methods

While the majority of IRL methods assume completely specified dynamics, we briefly review two that learn the dynamics in addition to the reward function. MWAL obtains the maximum likelihood estimate of unknown transition probabilities by computing the frequencies of state-action pairs which are observed more than a preset threshold number of times. The process determines the complete transition function by routing the transitions for remaining state-action pairs to an absorbing state. To give formal guarantees for the accuracy of learned dynamics and thereby the reward function, the algorithm leverages a theoretical upper bound on the accuracy of the learned transition model if the learner receives a given minimum amount of demonstration. Then, the upper bound on learning error in the MDP with learned transition model naturally extends to the actual MDP of expert [57].

While MWAL assumes that a learner fully observes the states, mIRL$^*_{\backslash T}$ [20] focuses on limited observations with unknown transition probabilities and multiple experts. Bogert and Doshi model each transition as an event composed of underlying components. For example, movement by a robot may be decomposed into its left and right wheels moving at some angular velocity. Therefore, the probability of reaching intended location by moving forward is the joint probability of left and right wheels rotating with the same velocities. The learner is assumed to know the intended next state for a state-action pair, and probability not assigned to the intended state is distributed equally among the unintended next states. Importantly, the components, also called transition features, are likely to be shared between observable and unobserved transitions as shown in Fig. 18. Therefore, a fixed distribution over the transition features determines T. The frequencies of a state-action pair in the demonstration provide a set of

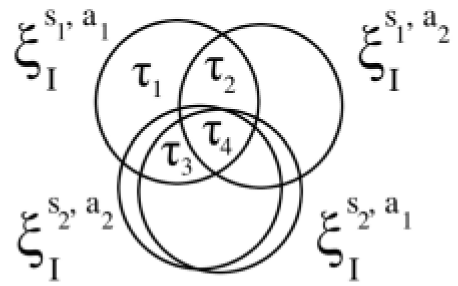

Figure 18: In mIRL*\T, $\xi_i^{(s,a)} = \{\tau_1, \tau_2, \ldots \tau_b\}$ denotes the transition-features for transition (s, a, s') of expert i, s' is intended next state. Computation of unknown probabilities by using probabilities of transition-features, $\prod_{\tau \in \xi_i^{(s,a)}} P(\tau) = T_{sa}(s, a, s')$, is feasible because different transitions share transition features among them. Source of illustration is [74] and figure is reprinted with author's permission.

empirical joint probabilities, as potential solutions. The preferred solution is the distribution of component probabilities with the maximum entropy. mIRL*\T generalizes better than MWAL because the structure introduced by shared features is more generalizable in the space of transition probabilities than a local frequency based estimation.

On the other hand, Boularias et al. [32] shows that the transition models approximated from a small set of demonstrations may result in highly inaccurate solutions. To this end, PI-IRL learns a reward function without any input transition probabilities. Furthermore, for estimating the unknown dynamics, GCL [75] iteratively runs a linear-Gaussian controller (current policy) to generate trajectory samples, fits local linear-Gaussian dynamics to them by using linear regression, and updates the controller under the fitted dynamics.

A generalization of MMP that focuses on IRL when the feature vector is known to be insufficient to explain the expert's behavior is MMPBOOST [76]. In this case, the method assumes that a predefined set of primitive features, which are easier to specify, create the reward feature functions. In the space of nonlinear functions of base features, MMPBOOST searches new features that make the demonstrated trajectories more likely and any alternative (simulated) trajectories less likely. Consequently, the hypothesized reward function performs better than the one with original feature functions. Further, it is well known that methods employing L1 regularization objectives can learn robustly when input features are not completely relevant [77]. In addition to MMPBOOST, GPIRL also uses this concept of base features to learn a new set of features which correspond better to the observed behavior. Wulfmeier et al. [30] and Finn et al. [34] utilize neural networks as function approximators that avoid the cumbersome hand-engineering of appropriate reward features.

In some applications, it is important to capture the *logical relationships* between base features to learn an optimum function representing the expert's reward. Most methods do not determine these relationships automatically. FIRL constructs features by capturing these relationships. In contrast, BNP-FIRL uses an Indian buffet process to derive a Markov Chain Monte Carlo procedure for Bayesian inference of the features and weights of a linear reward function [78]. BNP-FIRL is demonstrated to construct features more succinct than those by FIRL. Of course, all these methods are applicable only in domains where the feature space is sufficient to satisfactorily express the reward function.

6.4. Nonlinear Reward Function

A majority of the IRL methods such as PROJECTION, MMP, and MAXENTIRL assume that the solution is a weighted linear combination of a set of reward features (Eq. 3). While this is sufficient for many domains, a linear representation may be over simplistic in complex real tasks especially when raw sensory input is used to compute the reward values [34]. Also, analyzing the learner's performance w.r.t. the best solution seems compromised when a linear form restricts the class of possible solutions. But a significant challenge for relaxing this assumption is that nonlinear reward functions may take any shape, which could lead to a very large number of parameters and search space.

As our definition of IRL given in Def. 2 does not involve the structure of the learned reward function, it continues to represent the problem in the context of nonlinear reward functions as well.

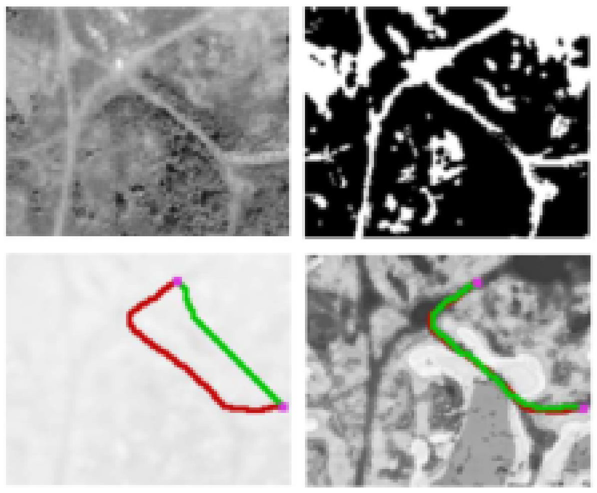

Figure 19: Learning a nonlinear reward function with boosted features improves performance over linear reward. Learner needs to imitate example paths drawn by humans in overhead imagery. Upper left panel - base features for a region. Upper right panel - image of the region used for testing. Red path is a demonstrated path and Green path is a learned path. Lower left panel - a map (un-boosted linear reward function) inferred using MMP with uniformly high cost everywhere. Lower right panel shows results of MMPBOOST. Since MMPBOOST creates new features by a search through a space of nonlinear reward functions, it performs significantly better. We reprinted this figure from [76] with permission from MIT press.

To overcome the shortcomings of using a linear reward function, methods MMPBOOST, LEARCH, and GPIRL infer a nonlinear reward function. MMPBOOST and LEARCH use a matrix of features in an image (cost map) and GPIRL uses a Gaussian process for representing the nonlinear function $R_E = f(\phi)$. Figure 19 shows the benefit of a nonlinear form with boosted features as compared to a restrictive linear form. In addition to these, Wulfmeier et al. [30] and Finn et al. [34] represent a complex nonlinear cost function using a neural network approximation, thereby avoiding the assumption of a linear form.

7. Discussion

This section distills the methods discussed previously (Section 4), and how they mitigated some of the challenges and the progress made so far (Sections 5 and 6) into a concise discussion of the significant milestones that have been achieved by the methods in this relatively new area so far, and the significant shortcomings of the presented methods. As such, this section is useful to a reader who is interested in IRL and wishes to know what has been accomplished and what remains largely unaddressed.

7.1. Milestones

Through the use of principled techniques such as maximum entropy optimization, maximum posterior and likelihood, and Gaussian process regression, solution methods have made significant progress in addressing IRL's primary challenge of being an underconstrained learning problem and admitting several satisficing hypotheses without introducing unnecessary bias.

The intractability of this machine learning problem due to its large hypothesis space has been significantly mitigated through the adoption of a reward function composed of linearly-weighted features. Though this imposed structure limits the hypothesis class, it often adequately represents the reward function in many problem domains. Importantly, it allowed the use of feature expectations as a sufficient statistic for representing the value of trajectories or the value of an expert's policy. This has contributed significantly to the success of early methods such as PROJECTION, MMP and MAXENT.

The presence of degenerate and multiple solutions led early methods such as MAX-MARGIN and MWAL to introduce bias in their optimizations. However, a side effect of this bias is that these methods may compute a policy $\hat{\pi}_E$ with zero probability assigned to some of the demonstrated actions [54]. Indeed, this is also observed in maximum likelihood based approaches such as MLIRL. Subsequent methods have largely solved this issue. For example, MMP makes the solution policy have state-action visitations that align with those in the expert's demonstration. MAXENT distributes probability mass based on entropy but under the constraint of feature expectation matching. Further, GPIRL addresses it by assigning a higher probability mass to the reward function corresponding to the demonstrated behavior, seeking posterior distributions with low variance.

Finally, among the various evaluation metrics discussed in Section 3.1, ILE has gained prominence as a robust measure of solution quality during formative evaluations. And, there is significant recognition that practical applications of IRL must deal with issues such as partial occlusion of the trajectory and that real-world demonstrations may interleave trajectories generated by multiple distinct reward functions.

7.2. Shortcomings

IRL aims to reduce hand tuning the reward function to obtain the observed behavior in an agent. Though a variety of remarkable endeavors exist for solving this problem, there are some parameters in all the methods which could get hard to tune in state spaces larger than few dozens of states. For example, base features of a reward function, an appropriate prior distribution in Bayesian methods, or the control parameter β in Eq. 10 continue to require manual input.

In our survey of several IRL methods, we observed that very few methods provably analyzed the sample or time complexity of their techniques, and compared it with those of other methods. Indeed, PROJECTION and MWAL are the only methods among the foundational ones that provide a sample complexity analysis. These methods use Hoeffding's inequality to relate the error in estimated feature expectations with the minimum sample complexity. As such, there is a general lack of theoretical guidance on the complexity of IRL as a problem, and on the complexity and accuracy of IRL methods, with most focusing on empirical comparisons to establish improvement.

A particularly egregious shortcoming is that the existing set of methods do not scale reasonably to beyond a few dozens of states or more than ten possible actions. This is a critical limitation that limits IRL demonstrations mainly to toy problems and prevents IRL from being applied in more pragmatic applications.

Finally, there is a distinct lack of a standard testbed of problem domains for evaluating IRL methods, despite the prevalence of empirical evaluations in this area. Well designed testbeds allow methods to be evaluated along various relevant dimensions, point out shared deficiencies, and typically speed up the advance of a particular field. For example, UCI's machine learning repository and OpenAI's Gym library are playing significant roles in advancing the progress of supervised and reinforcement learning techniques, respectively.

8. Concluding Remarks and Future Work

Since the introduction of IRL in 1998 by Russell, researchers have demonstrated a significantly improved understanding of the inherent challenges, developed various methods for their mitigation, and investigated the extension of these challenges toward real-world applications. This survey takes a rigorous look at IRL, and focuses on the specific ways by which various methods mitigated challenges and contributed to the ongoing progress of IRL research. The reason for this focus is that we believe that successful approaches in IRL will eventually combine the synergy of different methods to solve complex learning problems that typically exhibit many challenges. Our improved understanding has also revealed more novel questions. More development and clarifications are needed to answer these new questions.

Efficient MAXENTIRL. The optimizations in [79, 80] are likely to give an improvement in maximum entropy IRL algorithms. When input data (demonstration) is structured accordingly, these algorithms make the evaluation of partition function efficient [62].

Direct and indirect learning. When the state space is large and precise identification of $\hat{\pi}_E$ is cumbersome, directly learning a reward function results in a better generalization as compared to policy matching [13] (see Section 3.5). However, the issue of choosing between these two ways of learning from demonstrations or exploiting their synergies warrants a more thorough analysis.

Heuristics. Choi et al. [23] observes that when the authors evaluated the values of learned policy $\hat{\pi}_E$ and expert's policy π_E on the learned reward $\hat{R}_E$, both are optimal and about equal. However, $\hat{\pi}_E$ obtained using $\hat{R}_E$ does not achieve the same value as π_E when they use the true reward R_E for the evaluation. This is, of course, a quantification of the reward ambiguity challenge, which we pointed out earlier in this survey. It significantly limits learning accuracy. We believe that the choice of heuristics in the optimization may mitigate this issue.

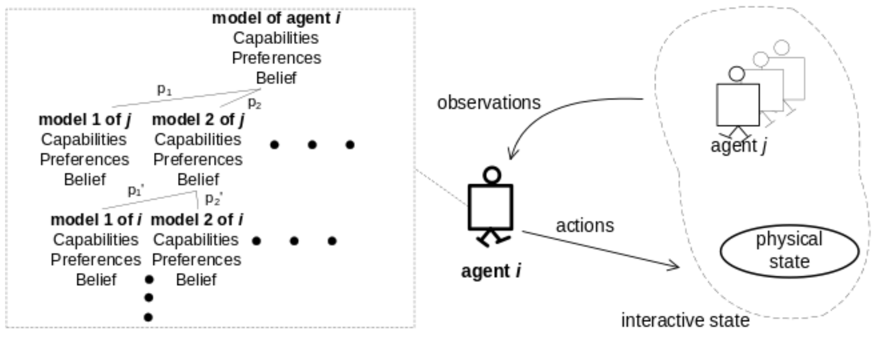

Figure 20: The state of I-POMDP evolves as an *interactive state space* that encompasses the computable models (beliefs, preferences, and capabilities) for other agents and the physical states of the environment. Agent i maintains and updates his models of other agents.

Multi-expert interaction. Recent work on IRL for multi-agent interactive systems can be extended to include more general classes of interaction-based models to increase the potential for applications [73, 72]. These classes include models for fully-observable state spaces (Markov games [81], multi-agent Markov decision processes [82], interaction-driven Markov games [83]) and partially-observable states (partially observable identical-payoff stochastic games [84], multi-agent team decision problems [85], decentralized Markov decision processes [86], and I-POMDPs [87] illusrated in Fig. 20). Outside the domain of IRL, we note behavior prediction approaches related to inverse optimal control in multi-agent game-theoretic settings [88]. The regret-based criterion in this work can be used for Markov games too: for any linear reward function, the learned behavior of agents should have regret less than or equal to that in observed behavior.

Non-stationary rewards. Most methods assume a fixed reward function that does not change. However, some authors believe that preferences of agent(s) may change with time, and the reward function can be time-variant i.e., $R : S \times A \times \eta \to \mathbb{R}$. Babes-Vroman et al. [45] capture such dynamic reward functions as multiple reward functions, but this approximation is crude. A more reasonable start in this research direction is the reward model in [47].

Function approximations. As mentioned in Section 3.4, we need computationally inexpensive approximations of complex nonlinear reward functions in large state spaces. Unlike shallow local architectures, a deep neural network architecture is expressive given a large number of demonstration trajectories [89], Recent preliminary work has investigated the use of neural network based function approximation and computation of the gradient of MAXENT objective function using back-propagation making IRL scalable to large spaces with nonlinear rewards [30, 34]. Future work in this direction may help IRL get closer to real-world applications.

Some more observations merit further enquiry and provide good avenues for future research. Many methods in IRL rely on parameter estimation techniques, and current trends show that meta-heuristic algorithms can estimate the optimal parameters efficiently. Some prominent meta-heuristic methods are cuckoo search algorithm [90, 91], particle swarm optimization [92], and the firefly algorithm [93]. As their noticeable benefits, meta-heuristic algorithms do not rely on the optimization being convex, rather they can search general spaces relatively fast, and they can find a global minimum. Thus, studying the performance of these techniques in IRL should reveal new insights. When IRL methods include a model for perturbations in trajectories, they typically assume that the noise is Gaussian. However, real-world applications often admit non-Gaussian disturbances. Investigations into how non-linear, non-Gaussian filtering methods [94, 95, 96] can address the influence of non-Gaussian outliers on IRL is another direction for future research.

9. Acknowledgments

This research was partly funded by a grant from the Toyota Research Institute of North America and a grant from ONR N-00-0-141310870.

10. References

[1] S. Russell, Learning agents for uncertain environments (extended abstract), in: Proceedings of the Eleventh Annual Conference on Computational Learning Theory, COLT' 98, ACM, New York, NY, USA, 1998, pp. 101–103.

[2] A. Ng, S. Russell, Algorithms for inverse reinforcement learning, Proceedings of the Seventeenth International Conference on Machine Learning 0 (2000) 663–670.

[3] Next generation simulation (NGSIM), http://ops.fhwa.dot.gov/trafficanalysistools/ngsim.htm.

[4] M. L. Puterman, Markov Decision Processes: Discrete Stochastic Dynamic Programming, 1st Edition, John Wiley & Sons, Inc., New York, NY, USA, 1994.

[5] A. Coates, P. Abbeel, A. Y. Ng, Apprenticeship learning for helicopter control, Communications of the ACM 52 (7) (2009) 97–105.

[6] B. D. Argall, S. Chernova, M. Veloso, B. Browning, A survey of robot learning from demonstration, Robot. Auton. Syst. 57 (5) (2009) 469–483.

[7] S. P. Boyd, L. El Ghaoui, E. Feron, V. Balakrishnan, Linear matrix inequalities in system and control theory, SIAM 37 (3) (1995) 479–481.

[8] C. L. Baker, R. Saxe, J. B. Tenenbaum, Action understanding as inverse planning, Cognition 113 (3) (2009) 329 – 349, reinforcement learning and higher cognition.

[9] T. D. Ullman, C. L. Baker, O. Macindoe, O. Evans, N. D. Goodman, J. B. Tenenbaum, Help or hinder: Bayesian models of social goal inference, in: Proceedings of the 22Nd International Conference on Neural Information Processing Systems, NIPS'09, Curran Associates Inc., USA, 2009, pp. 1874–1882.

[10] P. Abbeel, A. Coates, M. Quigley, A. Y. Ng, An application of reinforcement learning to aerobatic helicopter flight, in: Proceedings of the 19th International Conference on Neural Information Processing Systems, NIPS'06, MIT Press, Cambridge, MA, USA, 2006, pp. 1–8.

[11] H. Kretzschmar, M. Spies, C. Sprunk, W. Burgard, Socially compliant mobile robot navigation via inverse reinforcement learning, The International Journal of Robotics Research 35 (11) (2016) 1289–1307. doi:10.1177/0278364915619772.

[12] B. Kim, J. Pineau, Socially adaptive path planning in human environments using inverse reinforcement learning, International Journal of Social Robotics 8 (1) (2016) 51–66.

[13] G. Neu, C. Szepesvári, Apprenticeship Learning using Inverse Reinforcement Learning and Gradient Methods, Twenty-Third Conference on Uncertainty in Artificial Intelligence (2007) 295–302arXiv:1206.5264.

[14] M. Kuderer, S. Gulati, W. Burgard, Learning driving styles for autonomous vehicles from demonstration, in: IEEE International Conference on Robotics and Automation (ICRA), 2015, pp. 2641–2646.

[15] B. D. Ziebart, A. Maas, J. A. Bagnell, A. K. Dey, Maximum entropy inverse reinforcement learning, in: Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3, AAAI'08, AAAI Press, 2008, pp. 1433–1438.

[16] B. D. Ziebart, A. L. Maas, A. K. Dey, J. A. Bagnell, Navigate like a cabbie: Probabilistic reasoning from observed context-aware behavior, in: Proceedings of the 10th International Conference on Ubiquitous Computing, UbiComp '08, ACM, New York, NY, USA, 2008, pp. 322–331.

[17] N. D. Ratliff, D. Silver, J. A. Bagnell, Learning to search: Functional gradient techniques for imitation learning, Auton. Robots 27 (1) (2009) 25–53.

[18] B. D. Ziebart, N. Ratliff, G. Gallagher, C. Mertz, K. Peterson, J. A. Bagnell, M. Hebert, A. K. Dey, S. Srinivasa, Planning-based prediction for pedestrians, in: Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS'09, IEEE Press, Piscataway, NJ, USA, 2009, pp. 3931–3936.

[19] A. Vogel, D. Ramachandran, R. Gupta, A. Raux, Improving hybrid vehicle fuel efficiency using inverse reinforcement learning, in: AAAI Conference on Artificial Intelligence, 2012.

[20] K. Bogert, P. 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, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2015, pp. 1837–1838.

[21] L. P. Kaelbling, M. L. Littman, A. W. Moore, Reinforcement learning: A survey, J. Artif. Int. Res. 4 (1) (1996) 237–285.

[22] S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach (Second Edition), Prentice Hall, 2003.

[23] J. Choi, K.-E. Kim, Inverse reinforcement learning in partially observable environments, J. Mach. Learn. Res. 12 (2011) 691–730.

[24] G. Neu, C. Szepesvári, Training parsers by inverse reinforcement learning, Mach. Learn. 77 (2-3) (2009) 303–337.

[25] N. D. Ratliff, J. A. Bagnell, M. A. Zinkevich, Maximum margin planning, in: Proceedings of the 23rd International Conference on Machine Learning, ICML '06, ACM, New York, NY, USA, 2006, pp. 729–736.

[26] A. S. David Silver, James Bagnell, High performance outdoor navigation from overhead data using imitation learning, in: Robotics: Science and Systems IV, Zurich, Switzerland, 2008.

[27] P. Abbeel, A. Y. Ng, Apprenticeship learning via inverse reinforcement learning, in: Proceedings of the Twenty-first International Conference on Machine Learning, ICML '04, ACM, New York, NY, USA, 2004, pp. 1–8.

[28] U. Syed, R. E. Schapire, A game-theoretic approach to apprenticeship learning, in: Proceedings of the 20th International Conference on Neural Information Processing Systems, NIPS'07, Curran Associates Inc., USA, 2007, pp. 1449–1456.

[29] E. T. Jaynes, Information theory and statistical mechanics, Phys. Rev. 106 (1957) 620–630.

[30] M. Wulfmeier, I. Posner, Maximum Entropy Deep Inverse Reinforcement Learning, arXiv preprint.

[31] A. Boularias, O. Krömer, J. Peters, Structured apprenticeship learning, in: P. A. Flach, T. De Bie, N. Cristianini (Eds.), Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2012, Bristol, UK, September 24-28, 2012. Proceedings, Part II, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 227–242.

[32] A. Boularias, J. Kober, J. 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.

[33] S. Kullback, Information theory and statistics (1968).

[34] C. Finn, S. Levine, P. Abbeel, Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization, arXiv preprint arXiv:1603.00448.

[35] D. Ramachandran, E. Amir, Bayesian inverse reinforcement learning, in: Proceedings of the 20th International Joint Conference on Artifical Intelligence, IJCAI'07, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007, pp. 2586–2591.

[36] F. S. Melo, M. Lopes, Learning from demonstration using mdp induced metrics, in: Proceedings of the 2010 European Conference on Machine Learning and Knowledge Discovery in Databases: Part II, ECML PKDD'10, Springer-Verlag, Berlin, Heidelberg, 2010, pp. 385–401.

[37] P. Castro, P. Panangaden, D. Precup, Equivalence relations in fully and partially observable Markov decision processes, in: International Joint Conference on Artificial Intelligence (IJCAI), 2009, pp. 1653–1658.

[38] M. Lopes, F. Melo, L. Montesano, Active learning for reward estimation in inverse reinforcement learning, in: Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases: Part II, ECML PKDD '09, Springer-Verlag, Berlin, Heidelberg, 2009, pp. 31–46.

[39] B. Michini, J. P. How, Bayesian nonparametric inverse reinforcement learning, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2012, pp. 148–163.

[40] B. Michini, M. Cutler, J. P. How, Scalable reward learning from demonstration, in: IEEE International Conference on Robotics and Automation (ICRA), 2013, 2013, pp. 303–308.

[41] E. Klein, M. Geist, B. Piot, O. Pietquin, Inverse reinforcement learning through structured classification, in: Proceedings of the 25th International Conference on Neural Information Processing Systems, NIPS'12, Curran Associates Inc., USA, 2012, pp. 1007–1015.

[42] E. Klein, B. Piot, M. Geist, O. Pietquin, A cascaded supervised learning approach to inverse reinforcement learning, in: Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases - Volume 8188, ECML PKDD 2013, Springer-Verlag New York, Inc., New York, NY, USA, 2013, pp. 1–16.

[43] S. Levine, Z. Popović, V. Koltun, Feature construction for inverse reinforcement learning, in: Proceedings of the 23rd International Conference on Neural Information Processing Systems, NIPS'10, Curran Associates Inc., USA, 2010, pp. 1342–1350.

[44] S. Levine, Z. Popović, V. Koltun, Nonlinear inverse reinforcement learning with gaussian processes, in: Proceedings of the 24th International Conference on Neural Information Processing Systems, NIPS'11, Curran Associates Inc., USA, 2011, pp. 19–27.

[45] M. Babes-Vroman, V. Marivate, K. Subramanian, M. Littman, Apprenticeship learning about multiple intentions, in: 28th International Conference on Machine Learning, ICML 2011, 2011, pp. 897–904.

[46] U. Syed, M. Bowling, R. E. Schapire, Apprenticeship learning using linear programming, in: Proceedings of the 25th International Conference on Machine Learning, ICML '08, ACM, New York, NY, USA, 2008, pp. 1032–1039.

[47] M. Kalakrishnan, P. Pastor, L. Righetti, S. Schaal, Learning objective functions for manipulation, in: IEEE International Conference on Robotics and Automation (ICRA), 2013, 2013, pp. 1331–1336.

[48] N. Aghasadeghi, T. Bretl, Maximum entropy inverse reinforcement learning in continuous state spaces with path integrals, in: 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2011, pp. 1561–1566.

[49] E. Theodorou, J. Buchli, S. Schaal, A generalized path integral control approach to reinforcement learning, J. Mach. Learn. Res. 11 (2010) 3137–3181.

[50] B. D. Ziebart, J. A. Bagnell, A. K. Dey, Modeling interaction via the principle of maximum causal entropy, in: J. Frnkranz, T. Joachims (Eds.), Proceedings of the 27th International Conference on Machine Learning (ICML-10), Omnipress, 2010, pp. 1255–1262.

[51] A. Coates, P. Abbeel, A. Y. Ng, Learning for control from multiple demonstrations, in: Proceedings of the 25th International Conference on Machine Learning, ICML '08, ACM, New York, NY, USA, 2008, pp. 144–151.

[52] K. Shiarlis, J. Messias, S. Whiteson, Inverse reinforcement learning from failure, in: Proceedings of the 2016 International Conference on Autonomous Agents and Multiagent Systems, AAMAS '16, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2016, pp. 1060–1068.

[53] F. S. Melo, M. Lopes, R. Ferreira, Analysis of inverse reinforcement learning with perturbed demonstrations, in: Proceedings of the 2010 Conference on ECAI 2010: 19th European Conference on Artificial Intelligence, IOS Press, Amsterdam, The Netherlands, The Netherlands, 2010, pp. 349–354.

[54] B. Ziebart, Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy, Ph.D. thesis, Carnegie Mellon University (December 2010).

[55] P. D. Grünwald, A. P. Dawid, Game theory, maximum entropy, minimum discrepancy and robust bayesian decision theory, The Annals of Statistics 32 (1) (2004) 1367–1433. arXiv:0410076v1.

[56] C. Dimitrakakis, C. A. Rothkopf, Bayesian multitask inverse reinforcement learning, in: Proceedings of the 9th European Conference on Recent Advances in Reinforcement Learning, EWRL'11, Springer-Verlag, Berlin, Heidelberg, 2012, pp. 273–284.

[57] U. Syed, R. E. Schapire, A Game-Theoretic Approach to Apprenticeship LearningSupplement (2007).

[58] M. C. Vroman, MAXIMUM LIKELIHOOD INVERSE REINFORCEMENT LEARNING, Ph.D. thesis, Rutgers, The State University of New Jersey (2014).

[59] S. J. Lee, Z. Popović, Learning behavior styles with inverse reinforcement learning, ACM Trans. Graph. 29 (4) (2010) 122:1–122:7.

[60] T. Munzer, B. Piot, M. Geist, O. Pietquin, M. Lopes, Inverse reinforcement learning in relational domains, in: Proceedings of the 24th International Conference on Artificial Intelligence, IJCAI'15, AAAI Press, 2015, pp. 3735–3741.

[61] R. Fletcher, Practical methods of optimization, Wiley-Interscience publication, Wiley, 1987.

[62] R. Malouf, A comparison of algorithms for maximum entropy parameter estimation, in: Proceedings of the 6th Conference on Natural Language Learning - Volume 20, COLING-02, Association for Computational Linguistics, Stroudsburg, PA, USA, 2002, pp. 1–7.

[63] P. Vernaza, J. A. Bagnell, Efficient high-dimensional maximum entropy modeling via symmetric partition functions, in: Proceedings of the 25th International Conference on Neural Information Processing Systems, NIPS'12, Curran Associates Inc., USA, 2012, pp. 575–583.

[64] J. Z. Kolter, P. Abbeel, A. Y. Ng, Hierarchical apprenticeship learning, with application to quadruped locomotion, in: Proceedings of the 20th International Conference on Neural Information Processing Systems, NIPS'07, Curran Associates Inc., USA, 2007, pp. 769–776.

[65] C. A. Rothkopf, D. H. Ballard, Modular inverse reinforcement learning for visuomotor behavior, Biol. Cybern. 107 (4) (2013) 477–490.

[66] S. Wang, R. Rosenfeld, Y. Zhao, D. Schuurmans, The Latent Maximum Entropy Principle, in: IEEE International Symposium on Information Theory, 2002, pp. 131–131.

[67] S. Wang, D. Schuurmans Yunxin Zhao, The Latent Maximum Entropy Principle, ACM Transactions on Knowledge Discovery from Data 6 (8).

[68] K. Bogert, J. F.-S. Lin, P. Doshi, D. Kulic, 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.

[69] K. M. Kitani, B. D. Ziebart, J. A. Bagnell, M. Hebert, Activity forecasting, in: Proceedings of the 12th European Conference on Computer Vision - Volume Part IV, ECCV'12, Springer-Verlag, Berlin, Heidelberg, 2012, pp. 201–214.

[70] L. P. Kaelbling, M. L. Littman, A. R. Cassandra, Planning and acting in partially observable stochastic domains, Artif. Intell. 101 (1-2) (1998) 99–134.

[71] J. Choi, K.-E. Kim, Nonparametric bayesian inverse reinforcement learning for multiple reward functions, in: Proceedings of the 25th International Conference on Neural Information Processing Systems, NIPS'12, Curran Associates Inc., USA, 2012, pp. 305–313.

[72] T. S. Reddy, V. Gopikrishna, G. Zaruba, M. Huber, Inverse reinforcement learning for decentralized non-cooperative multiagent systems, in: 2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2012, pp. 1930–1935.

[73] X. Lin, P. A. Beling, R. Cogill, Multi-agent inverse reinforcement learning for zero-sum games, CoRR abs/1403.6508.

[74] K. Bogert, P. Doshi, Toward estimating others' transition models under occlusion for multi-robot irl, in: Proceedings of the 24th International Conference on Artificial Intelligence, IJCAI'15, AAAI Press, 2015, pp. 1867–1873.

[75] S. Levine, P. Abbeel, Learning neural network policies with guided policy search under unknown dynamics, in: Proceedings of the 27th International Conference on Neural Information Processing Systems, NIPS'14, MIT Press, Cambridge, MA, USA, 2014, pp. 1071–1079.

[76] N. Ratliff, D. Bradley, J. A. Bagnell, J. Chestnutt, Boosting structured prediction for imitation learning, in: Proceedings of the 19th International Conference on Neural Information Processing Systems, NIPS'06, MIT Press, Cambridge, MA, USA, 2006, pp. 1153–1160.

[77] A. Y. Ng, Feature selection, l1 vs. l2 regularization, and rotational invariance, in: Proceedings of the Twenty-first International Conference on Machine Learning, ICML '04, ACM, New York, NY, USA, 2004, pp. 78–.

[78] J. Choi, K.-E. Kim, Bayesian nonparametric feature construction for inverse reinforcement learning, in: Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence, IJCAI '13, AAAI Press, 2013, pp. 1287–1293.

[79] J. D. Lafferty, B. Suhm, Cluster expansions and iterative scaling for maximum entropy language models, in: K. M. Hanson, R. N. Silver (Eds.), Maximum Entropy and Bayesian Methods: Santa Fe, New Mexico, U.S.A., 1995 Proceedings of the Fifteenth International Workshop on Maximum Entropy and Bayesian Methods, Springer Netherlands, Dordrecht, 1996, pp. 195–202.

[80] J. Wu, S. Khudanpur, Efficient training methods for maximum entropy language modeling, in: Sixth International Conference on Spoken Language Processing, ICSLP 2000 / INTERSPEECH 2000, Beijing, China, October 16-20, 2000, 2000, pp. 114–118.

[81] M. L. Littman, Markov games as a framework for multi-agent reinforcement learning, in: Proceedings of the eleventh international conference on machine learning, Vol. 157, 1994, pp. 157–163.

[82] C. Boutilier, Sequential optimality and coordination in multiagent systems, in: Proceedings of the 16th International Joint Conference on Artifical Intelligence - Volume 1, IJCAI'99, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999, pp. 478–485.

[83] M. T. J. Spaan, F. S. Melo, Interaction-driven markov games for decentralized multiagent planning under uncertainty, in: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 1, AAMAS '08, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2008, pp. 525–532.

[84] L. Peshkin, K.-E. Kim, N. Meuleau, L. P. Kaelbling, Learning to cooperate via policy search, in: Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence, UAI '00, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2000, pp. 489–496.

[85] D. V. Pynadath, M. Tambe, The communicative multiagent team decision problem: Analyzing teamwork theories and models, J. Artif. Int. Res. 16 (1) (2002) 389–423.

[86] D. S. Bernstein, R. Givan, N. Immerman, S. Zilberstein, The complexity of decentralized control of markov decision processes, Math. Oper. Res. 27 (4) (2002) 819–840.

[87] P. J. Gmytrasiewicz, P. Doshi, A framework for sequential planning in multi-agent settings, J. Artif. Int. Res. 24 (1) (2005) 49–79.

[88] K. Waugh, B. D. Ziebart, J. A. Bagnell, Computational rationalization: The inverse equilibrium problem, CoRR abs/1308.3506.

[89] Y. Bengio, Y. LeCun, Scaling Learning Algorithms towards AI, in: L. Bottou, O. Chapelle, D. DeCoste, J. Weston (Eds.), Large Scale Kernel Machines, MIT Press, 2007.

[90] X.-S. Yang, S. Deb, Cuckoo search via lévy flights, in: 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC), IEEE, 2009, pp. 210–214.

[91] X.-S. Yang, S. Deb, Engineering optimisation by cuckoo search, arXiv preprint arXiv:1005.2908.

[92] R. Eberhart, J. Kennedy, Particle swarm optimization, in: Proceedings of the IEEE international conference on neural networks, Vol. 4, Citeseer, 1995, pp. 1942–1948.

[93] X.-S. Yang, Firefly algorithm, stochastic test functions and design optimisation, arXiv preprint arXiv:1003.1409.

[94] C. Masreliez, Approximate non-gaussian filtering with linear state and observation relations, IEEE Transactions on Automatic Control 20 (1) (1975) 107–110. `doi:10.1109/TAC.1975.1100882`.

[95] N. J. Gordon, D. J. Salmond, A. F. Smith, Novel approach to nonlinear/non-gaussian bayesian state estimation, in: IEEE Proceedings F (Radar and Signal Processing), IET, 1993, pp. 107–113.

[96] M. S. Arulampalam, S. Maskell, N. Gordon, T. Clapp, A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking, IEEE Transactions on signal processing 50 (2) (2002) 174–188.

Appendix A. Appendix

Table A.1 below lists and compares the foundational methods surveyed in Section 4. We identify the parameters that are learned and the learning objective in the each method. The latter often involves minimizing a divergence metric.

Method	**Parameters of Hypothesis $\hat{R}_E$**	**Divergence metric between π_E and $\hat{\pi}_E$**
LPIRL	weights for linear approximation	difference in values of policies
MAX-MARGIN and PROJECTION	weights for linear approximation	difference in (expected) features
MMP	weights for linear approximation	difference in state-visitation frequencies
LEARCH	weights for linear approximation	difference in state-visitation frequencies
MWAL	weights for linear approximation	difference in (expected) features
MAXENTIRL	weights for linear approximation	divergence in distributions over trajectories
REIRL	weights for linear approximation	divergence in distributions over trajectories
SAL	weights for linear approximation	divergence in distributions over deterministic policies
GCL	synaptic weights for neural network	divergence in distributions over trajectories
BIRL	action values of learned policy	divergence in distributions over reward functions
GPIRL	hyperparameters of Gaussian distribution	divergence in distributions over reward functions
FIRL	state features	difference in action selection
CSI	action values (score function) of learned policy	difference in action selection at demonstrated states
SCIRL	weights for linear approximation	difference in action selection at demonstrated states
HYBRID-IRL or Policy Matching	action values of learned policy	difference in action selection at demonstrated states
MLIRL	action values of learned policy	difference in action selection at demonstrated states
PI-IRL	weights for linear approximation	divergence in distributions over trajectories

Table A.1: Key elements used in the foundational methods for IRL.