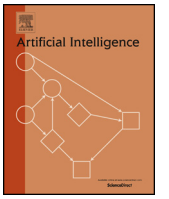
Artificial Intelligence 297 (2021) 103500

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint



A survey of inverse reinforcement learning: Challenges, methods and progress

Saurabh Arora [a], Prashant Doshi [b,*]

[a] THINC Lab, Dept. of Computer Science, University of Georgia, Athens, GA 30602, United States of America
[b] THINC Lab and Institute for AI, Dept. of Computer Science, University of Georgia, Athens, GA 30602, United States of America

ARTICLE INFO

Article history:
Received 30 October 2017
Received in revised form 11 February 2021
Accepted 14 March 2021
Available online 24 March 2021

Keywords:
Reinforcement learning
Reward function
Learning from demonstration
Generalization
Learning accuracy
Survey

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 extant literature in IRL, this article serves as a comprehensive reference for researchers and practitioners of machine learning as well as those new to it 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 surveys a vast collection of foundational methods grouped together by the commonality of their objectives, and elaborates how these methods mitigate the challenges. We further discuss extensions to the traditional IRL methods for handling imperfect perception, an incomplete model, learning multiple reward functions and nonlinear reward functions. The article concludes the survey with a discussion of some broad advances in the research area and currently open research questions.

1. Introduction

Inverse reinforcement learning (IRL) is the problem of inferring the hidden preferences of another agent from its observed behavior, thereby avoiding a manual specification of its reward function [1,2]. Over the past decade, IRL has attracted much interest in the communities of artificial intelligence, control theory, machine learning, and psychology. IRL is appealing because of its potential to use data recorded in performing 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 roles A and B, on freeway entry ramps, to learn the safety and speed preferences of a typical driver in role B as she approaches this difficult merge (NGSIM [3] is one such existing data set).

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

* Corresponding author.
E-mail addresses: sa08751@uga.edu (S. Arora), pdoshi@uga.edu (P. Doshi).

https://doi.org/10.1016/j.artint.2021.103500
0004-3702/

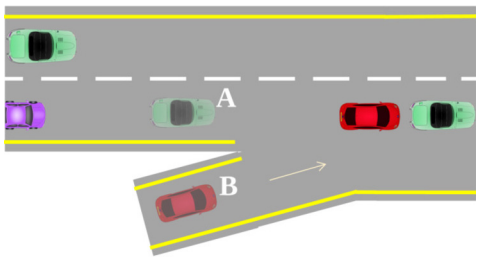


Fig. 1. Red car B is seeking to merge into the lane, and green car A is the immediate traffic. The lighter images of cars show their positions before merging, and the opaque images to the right depict one of their possible positions after the merger. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

unknown, and the observed agent is assumed to follow an optimal policy of the MDP. In the traffic merge example, the MDP represents the driving process of Car B. The driver of Car B is following action choices (deceleration, braking, low acceleration, and others) based on its optimal policy. Car B needs to reach the end of the merging lane 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 B. 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 planning to problems where a reward function can be easily specified. IRL offers a way to broaden the applicability of RL and reduce the manual design of task specification, when 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 are trajectories consisting of state-action pairs, the target of learning in inverse optimal control is a function mapping states of observed agent to her actions, which need not involve recovering the hidden rewards. 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, actions, 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

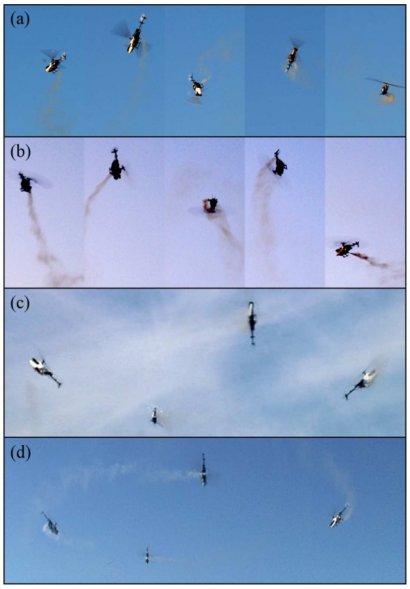


Fig. 2. Complex helicopter maneuvers learned using RL on a reward function learned from an expert pilot through IRL. The image is reprinted from [10] with permission from publisher.

inference of a human's goal as an inverse planning problem in an MDP. Furthermore, IRL's use toward apprenticeship and imitation learning has rapidly expanded the set of visible use cases. 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 expert video game play [15].
2. Learning from another agent to predict its behavior. One of the first attempts in this direction was route prediction for taxis [16,17]. Other such applications are footstep prediction for planning legged locomotion [18], anticipation of pedestrian interactions [19], energy efficient driving [20], and penetration of a perimeter patrol by learning the patrollers' preferences and patrolling route [21].

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 a uncomplicated exposition and various examples, to the researchers and practitioners new to the field;
2. A study of the challenges that make IRL difficult, and a review of the various foundational methods;
3. Qualitative assessments and comparisons among different methods, both those that are foundational and those that extend traditional IRL, to evaluate them coherently. This will considerably help the readers decide on the approach suitable for the problem at hand;
4. Identification of some significant milestones achieved by the methods in this field;
5. Identification of the common shortcomings and 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. To help with this goal, we maintain a distinction between IRL and areas of research pertaining to problems such as behavior cloning, imitation learning, learning from demonstration, and inverse optimal control. Specifically, IRL is simply one way to perform imitation learning or to learn from demonstration (whereas other methods for these problems need not recover the reward function). As the scope of our survey is limited to IRL, we do not focus on reviewing other methods for imitation learning, learning from demonstration, or inverse optimal control. We refer the interested reader to survey articles focused on these problems [22,23] (which include brief references to IRL as one way of approaching the problem).

1.3. Organization of contents

As IRL is an emerging area and the target reader is likely 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 the requisite 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 with some recent extensions grouped together by the commonality of their underlying approaches, in Section 4 that have facilitated much early progress in IRL. We include a tabulated summary and unifying views of these methods. Section 5 then discusses how these methods mitigate the previously introduced core challenges and some achieved milestones. 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. Finally, the article concludes with a discussion of some shortcomings and 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 [4].

Definition 1 (MDP). An MDP $\mathcal{M} := \langle S, A, T, R, \gamma \rangle$ models an agent's sequential decision-making process. S is a finite set of states and A is a set of actions. Mapping $T : S \times A \to \text{Prob}(S)$ defines a probability distribution over the set of next states conditioned on the agent taking action a at state s; $\text{Prob}(S)$ 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: $R : S \to \mathbb{R}$ gives the scalar reinforcement at state s, $R : S \times A \to \mathbb{R}$ maps a tuple (state s, action a taken in state s) to the reward received on performing the action, and, $R : S \times A \times S \to \mathbb{R}$ maps a triplet (state s, action a, resultant state s') to the reward obtained on performing the transition. Discount factor $\gamma \in [0,1]$ is the weight for future rewards 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, $\pi : S \to A$ or stochastic $\pi : S \to \text{Prob}(A)$. For a policy π, value function $V^\pi : S \to \mathbb{R}$ gives the value of a state s as the long-term expected cumulative reward incurred from the state by following π. The value of a policy π 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 the 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, $Q^\pi : S \times A \to \mathbb{R}$, 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 to the initial state distribution. Let Ψ be the space of all ψ functions. Iterating Eq. (2) until the state-visitation frequency stops changing yields the converged frequency function, ψ^π_*. We may write the value function as, $V^*(s) = \sup_\pi \ \sum_{s \in S} \psi^\pi_*(s) \ R(s, \pi(s))$.

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

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

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,

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

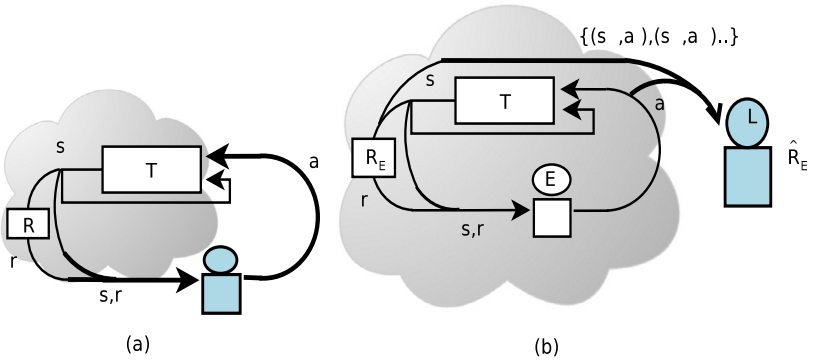


Fig. 3. (*a*) A schematic showing the subject agent (shaded in blue) performing RL [24]. In forward learning or RL, the agent chooses an action at a known state and receives a reward in return generated by a reward function R that may not be known to the agent. The state changes based on the previous 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. L perceives the states and actions $\{(s,a),(s,a),\ldots,(s,a)\}$ 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.

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:

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

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, $\mathcal{M}\backslash_{R_E}$, model the interaction of the expert E with the environment. 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 may 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.

Notice that IRL inverts the RL problem. Whereas RL seeks to learn the optimal behavior based on experiences ((s,r) or (s,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 $\mathcal{D}$ of N trajectories, 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 $\hat{R}_E$.

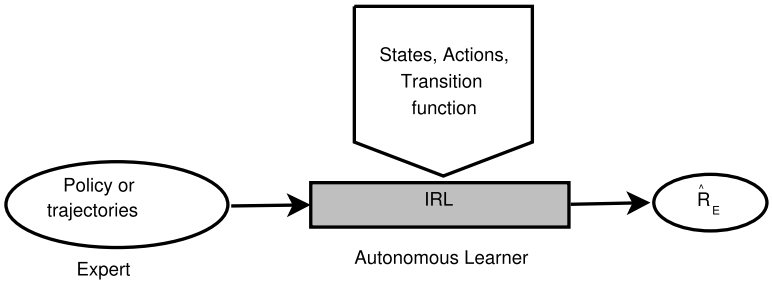


Fig. 4. Pipeline for a classical IRL process. The learner receives an optimal policy or trajectories as input. The prior domain knowledge (shown here as a pentagon) include the completely observable state space, action space, and fully known transition probabilities.

3.1. Obstacles to 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.

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 [25]. 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 another metric, which is to 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 [26]. Notice that if the true and learned policies are the same, then ILE is zero. However, ILE may also vanish when the two differ but if both policies are optimal. On the other hand, ILE requires knowing the true transition and reward functions 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. However, it is limited to the demonstration. Clearly, no single metric gives a complete evaluation.

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 then 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.

The metric 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 and Szepesvári [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 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, is linear, the gradient can be computed quickly, or none of these, researchers 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. The first one 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, and explicitly 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 and better transferability of the task at hand, it may lead to policies that do not fully reproduce the observed trajectories. For the second approach, Neu and Szespesvári. [27] 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.

The following 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. As such, it involves repeatedly solving the embedded forward learning problem.

Algorithm 1: Template for IRL.

Input: $\mathcal{M}\backslash_{R_E} = \langle S, A, T, \gamma \rangle$,
Set of trajectories demonstrating desired behavior:
$\mathcal{D} = \{\langle (s_0, a_0), (s_1, a_1), \ldots, (s_t, a_t) \rangle, \ldots\}$, $s_t \in S$, $a_t \in A$, $t \in \mathbb{N}$,
or expert's policy: π_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.

The remainder of this section categorizes IRL methods based on the core approach they use for inverse learning – margin based optimization, entropy based optimization, Bayesian inference, classification, and regression. A second-level grouping within each of these categories clusters methods based on the specific objective function utilized in realizing the core approach. Our presentation emphasizes the commonalities between the various methods. Recall the notation introduced in Section 2 before continuing.

4.1. Margin optimization

Maximum margin prediction aims to learn a reward function that explains the demonstrated policy better than alternative policies by a margin. The methods under this category aim to address IRL's solution ambiguity (discussed in Section 3.1) by converging on a solution that maximizes some margin. We broadly organize the methods that engage in margin optimization based on the type of the margin that is used.

4.1.1. Margin of optimal from other actions or policies

One of the earliest and simplest margins chosen for optimization is the sum of differences between the expected value of the optimal action and that of the next-best action over all states,

$$\sum_{s \in S} Q^{\pi}(s, a^*) - \max_{a \in A \backslash \{a^*\}} Q^{\pi}(s, a) \tag{7}$$

where a^* is the optimal action for s.

If the reward function is feature-based, whose form is given in Eq. (3), a similar margin that takes the difference between the expected value of the behavior from each observed trajectory and the largest of the expected values of behaviors from all other trajectories can be used to learn the feature weights. The expected value of a policy is obtained by multiplying the empirical state visitation frequency from the observed trajectory with the weighted feature function values $\boldsymbol{\phi}(\cdot)$ obtained for the trajectory. For each trajectory τ_i in the demonstration, this margin is expressed as,

$$\sum_{\langle s,a \rangle \in \tau_i} \psi(s)\ \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{s},\boldsymbol{a}) - \max_{\tau \in (S \times A)^l \backslash \{\tau_i\}} \sum_{\langle s,a \rangle \in \tau} \psi(s)\ \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{s},\boldsymbol{a}) \tag{8}$$

where $(S \times A)^l$ is the set of all trajectories of length l.

An early and foundational method that optimized the margin given in Eq. (7) is Ng and Russell's [2], which takes in the expert's policy as input thereby instantiating π as π_E in the margin. 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 margin shown above. In addition to maximizing this margin, it also prefers reward functions with smaller values as a form of regularization.

Under the assumption that each trajectory in a demonstration reflects a distinct policy and the reward function is expressed as a linear and weighted sum of feature functions, Ratliff et al.'s [28] maximum margin planning (MMP) associates each trajectory $\tau_i \in \mathcal{D}$ with an MDP. While these MDPs could differ in their state and action sets, and the transition functions, they share the same reward function. The desired reward weight vector $\boldsymbol{w}$ is obtained by solving a quadratic program that is constrained to have a positive value on the margin given in Eq. (8) with the right-hand side of the margin augmented by a regularizing loss term $l_i^T \psi$ that quantifies the closeness between the demonstrated and other behaviors.

Using the same margin as in Eq. (8) and the regularizer, Ratliff et al. [18] improves on MMP in a subsequent method called *learn to search* (LEARCH). Fig. 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). Later, Silver et al. [29] introduced a gradient normalization technique for LEARCH to allow for suboptimal demonstrations as input.

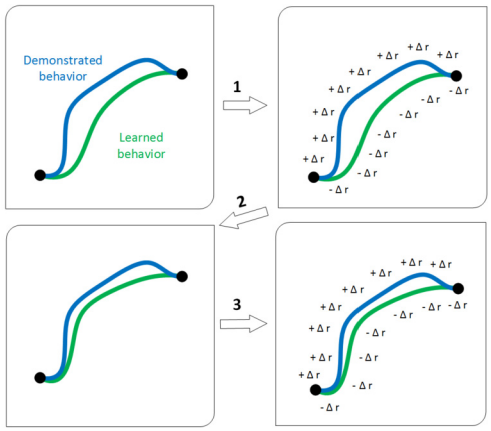


Fig. 5. An iteration of LEARCH in the feature space $\Phi = \{\phi(s,a) | \forall (s,a) \in S \times A\}$ (Fig. 3 in Ratliff et al. [18] excerpted with permission). The method considers a reward function as negative of a cost function. Blue path depicts the demonstrated trajectory, and green path shows 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. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

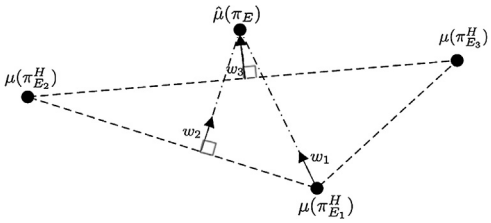


Fig. 6. Iterations of the max-margin method computing the weight vector, $\mathbf{w}$, and the feature expectations, $\boldsymbol{\mu}^{\boldsymbol{\phi}}$. $\hat{\boldsymbol{\mu}}^{\boldsymbol{\phi}}(\mathcal{D})$ is the estimation of the feature counts $\boldsymbol{\mu}^{\boldsymbol{\phi}}(\pi_E)$ 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 [30].

4.1.2. Margin of observed from learned feature expectations

Adoption of the feature-based reward function led to several methods optimizing margins that utilized feature expectations. Some of 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)):

$$|\boldsymbol{\mu}^{\boldsymbol{\phi}}(\pi) - \hat{\boldsymbol{\mu}}^{\boldsymbol{\phi}}(\mathcal{D})|. \tag{9}$$

We refer to this margin as the *feature expectation loss*.

Two foundational methods [30] that minimize the feature expectation loss of Eq. (9) are MAX-MARGIN and PROJECTION. Noting that the learner does not typically have access to the expert's policy, both these methods take a demonstration (defined in Definition 2) as input. The methods represent the reward function as a linear, weighted sum of feature functions.

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 $\hat{\boldsymbol{\mu}}^{\boldsymbol{\phi}}(\mathcal{D})$ of expert and the weights are updated, as shown in Fig. 6. Abbeel and Ng [30] points 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) [31]. 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 margin to a minimax objective for a zero-sum game between the learner and its environment, $\max_{\hat{\pi}_E} \min_{\boldsymbol{w}} \ \boldsymbol{w}^T(\boldsymbol{\mu}^{\boldsymbol{\phi}}(\hat{\pi}_E) - \hat{\boldsymbol{\mu}}^{\boldsymbol{\phi}}(\mathcal{D}))$, and the optimization uses the exponentiated gradient ascent to obtain the weights $\boldsymbol{w}$.

4.1.3. Observed and learned policy distributions over actions

An alternative to minimizing the feature expectation loss is to minimize the probability difference between stochastic policies

$$\hat{\pi}_E(a|s) - \pi_E(a|s) \tag{10}$$

for each state. As the behavior of expert is available instead of its policy, the difference above is computed using the empirically estimated state visitation frequencies (Eq. (2)) and the frequencies of taking specific actions in the states. HYBRID-IRL [13] uses Eq. (10) in the margin optimization problem, solving the optimization using gradient descent in the space of reward hypotheses.

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. To avoid this bias, multiple methods take recourse to the maximum entropy principle [32] to obtain a distribution over behaviors, parameterized by the reward function weights. According to this principle, the distribution that maximizes the entropy makes minimal commitments beyond the constraints and is least wrong. We broadly categorize the methods that optimize entropy based on the distribution chosen by the method, whose entropy is considered.

4.2.1. Entropy of the distribution over trajectories or policies

We may learn a reward function that yields the distribution over all trajectories with the maximum entropy

$$\max_{\Delta} \; - \sum_{\tau \in (S \times A)^l} Pr(\tau) \log Pr(\tau) \tag{11}$$

while being constrained by the observed demonstration. However, the search space of trajectories in this optimization $(S \times A)^l$ grows exponentially with the length of the trajectory l. To avoid this disproportionate growth, we may learn a reward function that alternately yields the distribution over all policies with the maximum entropy

$$\max_{\Delta} \; - \sum_{\pi \in (S \times A)} Pr(\pi) \log Pr(\pi) \tag{12}$$

where Δ is the space of all distributions. Notice that the space of policies grows with the sizes of the state and action sets as $\mathcal{O}(|A|^{|S|})$ but not with the length of the trajectory.

A foundational and popular IRL technique by Ziebart et al. [16] MAXENTIRL optimizes the entropy formulation of Eq. (11) while adding two constraints. First, the distribution over all trajectories should be a probability distribution. Second, the expected feature count of the demonstrated trajectories $\sum_{\tau \in \mathcal{D}} Pr(\tau) \sum_{t=1}^{l} \gamma^t \phi_k(s_t, a_t)$ must match the empirical feature count obtained using Eq. (6).

Mathematically, this problem is a convex but nonlinear optimization whose Lagrangian dual reveals that the distributions of maximum entropy belong to the exponential family. Therefore, the problem reduces to finding the reward weights $\boldsymbol{w}$ that parameterize the exponential distribution over trajectories and exhibit the highest likelihood of the demonstration,

$$\arg\max_{\boldsymbol{w}} \sum_{\tau \in \mathcal{D}} log \; Pr(\tau; \boldsymbol{w}) \text{ where } \; Pr(\tau; \boldsymbol{w}) = \frac{e^{\sum_{\langle s,a \rangle \in \tau} \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{s},\boldsymbol{a})}}{Z(\boldsymbol{w})} \tag{13}$$

where the normalization constant $Z(\boldsymbol{w})$ is the well-known partition function. We show this distribution here as it is the subject of other methods as well. Ziebart et al. solves the Lagrangian dual thereby maximizing the likelihood, using gradient descent.

A subsequent method by Ziebart et al. [33] replaces the maximization objective of Eq. (11) for sequential environments with that of maximizing the causal entropy. More formally, the objective function becomes, $\max_{\Delta} \; - \sum_{\tau \in (S \times A)^l} Pr(\tau) \times \sum_{t=1}^{l} \log Pr(a_t | s_{1:t}, a_{1:t-1})$, where Δ is the space of all causally-conditioned probabilities $Pr(a_t | s_{1:t}, a_{1:t-1})$ for $t = 1, 2, \ldots, l$. All other constraints remain the same. Notice that the causal entropy allows the actions a_t to be conditioned on the prior sequences of states and actions, and not on any future information (which may not have a causal relationship); this serves to upper bound the entropy that would be obtained by conditioning on the entire sequence of states and actions. A recent iteration on this method suggests the use of causal Tsallis entropy in the objective function [34]. We may then utilize sparse Tsallis MDPs to model the expert, which scale better to larger problems and yield policies which may assign zero probabilities to unseen actions in contrast to softmax policies.

Wulfmeier et al. [35] shows that the linearly-weighted reward function in MAXENTIRL can be easily generalized to a nonlinear reward function represented by a neural network. The corresponding DEEP MAXENTIRL technique continues to maximize the likelihood of Eq. (13) by using its known gradient in the backpropagation to update the neural network's weights. Though a neural network representation does not require the use of explicit feature functions, it typically uses far more reward parameters, whose learning requires more data.

The optimization of the likelihood of Eq. (13) is also the subject of the PI-IRL method [36], which generalizes MAXENTIRL to continuous state spaces. To enable this, it replaces the traditional feature functions in a reward with its path integral

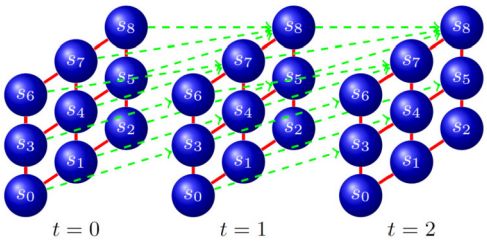


Fig. 7. A Markov random field, representing the distribution resulting from structured apprenticeship learning, favors the policies which prescribe similar actions in neighboring states. This structure generalizes the feature information beyond individual states and helps manage reward ambiguity. (Figure reprinted from [38] with permission from publisher.)

formulations [37] that involves not only the known features, but also the rate of change in the continuous state, and a matrix giving the state and goal costs. PI-IRL uses an iterated sampling approach to continually improve both the region of trajectory sampling and the path integral functions used in determining the demonstrated trajectory rewards.

Subsequent to MAXENTIRL, Boularias et al.'s structured apprenticeship learning [38] maximizes the entropy shown in Eq. (12) under constraints that are analogous to those of MAXENTIRL but pertaining to policy. In particular, the second constraint matches the expected feature count from the distribution over all the *policies* with the empirical feature count obtained using Eq. (6). Additionally, the method introduces a third constraint that brings a preference for those policies that assign the same action to adjacent states, while still conforming to the state-action pairs in the given trajectory. The resulting convex, nonlinear optimization problem is made somewhat tractable by observing that domains often exhibit *structure* that spatially neighboring states have similar optimal actions, which can be used to guide the search. This results in a Markov random field like distribution over policies as illustrated in Fig. 7.

4.2.2. Relative entropy of the distribution over trajectories

A different approach to entropy optimization for IRL involves minimizing the relative entropy (also known as Kullbach-Leibler divergence [39]) between two distributions P and Q over the trajectories. More formally,

$$\min_{P \in \Delta} \sum_{\tau \in (S \times A)^l} P(\tau) \log \frac{P(\tau)}{Q(\tau)}. \tag{14}$$

REIRL [40] is a prominent technique that utilizes the optimization objective given in Eq. (14). Distribution Q in REIRL is obtained empirically by sampling trajectories under a *baseline policy*. Distribution P is obtained such that the expected feature count of the trajectories matches the empirical feature count obtained using Eq. (6). This constraint is similar to previous approaches in this section and constrains REIRL to the demonstration data. The baseline policy serves as a way to provide domain-specific guidance to the method. While an analytical solution would need the transition dynamics to be pre-specified, Boularias et al. shows that the presence of the baseline policy allows importance sampling and REIRL can be solved model-free using stochastic gradient descent.

4.3. Bayesian update

An important class of IRL methods treats 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 methods. No structure is typically imposed on the reward function. Let $Pr(\hat{R}_E)$ be a prior distribution over the reward functions and $Pr(\tau|\hat{R}_E)$ be the observation likelihood of the reward hypothesis. Then, a posterior distribution over candidate reward functions is obtained using the following Bayesian update:

$$Pr(\hat{R}_E|\tau) \propto Pr(\tau|\hat{R}_E)\ Pr(\hat{R}_E) \tag{15}$$

Here, the likelihood is typically factored as $Pr(\tau|\hat{R}_E) = \prod_{\langle s,a\rangle \in \tau} Pr(\langle s,a\rangle|\hat{R}_E)$. The update above is performed as many times as the number of trajectories in the observed demonstration.

We categorize the Bayesian IRL methods based on how they model the observation likelihood in Eq. (15).

4.3.1. Boltzmann distribution

A popular choice for the likelihood function is the Boltzmann distribution (also known as the Gibbs distribution) with the Q-value of the state-action pair as the energy function:

$$Pr(\langle s,a\rangle|\hat{R}_E) \propto e^{\left(\frac{Q^*(s,a;\hat{R}_E)}{\beta}\right)} \tag{16}$$

where parameter β controls the randomness of the state-action probability (lower the β, more random is the state-action pair). Given a candidate reward hypothesis, some state-action pairs are more likely than others as given by the likelihood function.

The earliest Bayesian IRL technique is BIRL [41], which models the likelihood as shown in Eq. (16). BIRL suggests several different priors over the continuous space of reward functions. While the uniform density function is an agnostic choice, several real-world MDPs have a sparse reward structure for which a Gaussian or Laplacian prior is better suited. If the MDP reflects a planning type problem with a large dichotomy in rewards (goal states exhibiting a large positive reward), then a Beta density could be appropriate. The continuous space of reward functions brings the challenge that analytically obtaining the posterior is difficult. To address this issue, BIRL presents a random walk based MCMC algorithm in the space of policies for obtaining a sample-based empirical approximation of the posterior. On the other hand, we may also converge to the maximum-a-posteriori reward function directly using gradient descent [42].

Lopes et al. [43] shows that the posterior computed in BIRL can be used to incorporate active learning in IRL. The learner queries the expert for additional samples of actions in those states where a distribution induced from the posterior exhibits a high entropy. The induced distribution is a measure of how discriminating is the learned expert policy at each state. The new samples help learn an improved posterior.

4.3.2. Gaussian process

An influential Bayesian approach introduced after BIRL lets the reward function be a nonlinear function of features, $\hat{R}_E = f(\boldsymbol{r}, \boldsymbol{\phi})$, and models it as a Gaussian process whose underlying structure is given by a kernel function parameterized by $\boldsymbol{\theta}$. Thus, the posterior $Pr(\hat{R}_E|\tau)$ in Eq. (15) now becomes $Pr(\boldsymbol{r}, \boldsymbol{\theta}|\tau, \boldsymbol{\phi})$ where $\boldsymbol{r}$ are the rewards associated with the feature functions $\boldsymbol{\phi}$ at select states and actions.

The GPIRL technique [44] computes this posterior as a Bayesian update with the likelihood being $Pr(\tau|\hat{R}_E)Pr(\hat{R}_E|\boldsymbol{r}, \boldsymbol{\theta}, \boldsymbol{\phi})$. The first factor is computed as shown previously in Eq. (16). However, the distinctive second factor (also called the Gaussian process posterior) is a Gaussian distribution with analytically derived mean and covariance matrices. GPIRL generalizes from just a small subset of reward values – those contained in the observed trajectories and a few additional rewards at random states. It utilized L-BFGS with restarts to optimize the likelihood, finding most likely $\boldsymbol{r}$, thereby most likely reward functions.

4.3.3. Maximum likelihood estimation

Apart from the posterior estimation in BIRL methods, we may just directly maximize the likelihood of the input data ($Pr(\tau|\hat{R}_E)$ in Eq. (15)). The standard expression for the data likelihood involves the policy and the Bellman operator, which is not differentiable. However, a softmax Boltzmann exploration can be used to change the policy expression from maximization to $\pi_{\boldsymbol{w}}(s, a) = \frac{e^{\beta Q(s,a;\hat{R}_E)}}{\sum_{a' \in A} e^{\beta Q(s,a';\hat{R}_E)}}$, which in turn makes the likelihood $P(\tau|\hat{R}_E)$ differentiable. Vroman et al. [45] in the method MLIRL chooses this Boltzmann exploration policy to infer the reward feature weights $\boldsymbol{w}$ that leads to the maximum likelihood estimate. As the likelihood is now differentiable, the algorithm uses a standard gradient ascent for converging to the (locally-) optimal weights.

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. Classification based on action-value scores

IRL may be formulated as a multi-class classification problem by viewing the state-action pairs in a trajectory as data-label pairs. For each state in a pair, the label is the corresponding action performed at that state as prescribed by the expert's policy. As there are usually more than two actions in most domains, the classification is into one of multiple classes. An obvious way to score the classification is to use the action-value function, which is derived from Eq. (5) as:

$$Q^{\pi}(s, a) = \boldsymbol{w}^T \, \boldsymbol{\mu}^{\phi}(\pi)(s, a). \tag{17}$$

Notice that this is a linear scoring function which uses the same feature weight vector $\boldsymbol{w}$ as used in the reward function. Subsequently, a classifier aims to learn the weights that minimize the classification error between the labels in the state-action pairs of a trajectory and the action label predicted as $\arg\max_{a \in A} Q^{\pi}(s, a)$.

Klein et al. [46] introduced this multi-class classification formulation of IRL in a method called SCIRL. The demonstration is utilized for training a classifier with Eq. (17) as the scoring function. Any linear score based multi-class classification algorithm may be used to solve for the weight vector $\boldsymbol{w}$. SCIRL chose the large margin approach of structured prediction [47] as the algorithm for classification.

While SCIRL assumes the presence of a transition model to compute the scoring function, an extension called CSI [48] takes a step further and estimates the transition probabilities if they are not known. CSI utilizes standard regression on a simulated demonstration data set to estimate the transition model and thereafter learn the reward function using SCIRL.

Table 1
A compilation of the notation introduced in the text and abbreviations, which is referenced in Tables 2 and 3.

$\boldsymbol{\phi}(\cdot)$	reward features
$\boldsymbol{w}$	reward feature weights
$\psi^{\pi}(s)$	state visitation frequency
$\boldsymbol{\mu}^{\phi}(\pi)$	feature expectations
τ	trajectory
$\mathcal{D}$	demonstration (set of trajectories)
β	temperature in Boltzmann distribution
M	number of target reward functions
IBP	Indian buffet Gaussian process

Notice that SCIRL is predicated on the action in the state-action pair being the desired optimal action and uses it as the label. However, if we allow for the demonstration to be suboptimal, this classification approach may not work. In this context, Brown et al. [49] shows that if the input to IRL also includes a ranking of the trajectories based on the degree of suboptimality of the trajectory, we may utilize this additional preference information to fit a neural network representation of the reward function. A cross-entropy based loss function trains the neural network to obtain a higher cumulative reward for the trajectory that is preferred over another per the given ranking.

4.4.2. Regression tree for state space partitions

The linearly weighted sum of feature functions represents a global reward function model over the entire state and action spaces. It may be inadequate or require many features when these spaces are large. An alternate model could be a regression tree whose intermediate nodes are the individual feature functions and whose leaves represent a conjunction of indicator feature functions. Each path of this tree then captures a region of the state and action space, and the whole tree induces a partition of this space.

Subscribing to this representation of the reward function, FIRL [50] iteratively constructs both the features and the regression tree. To arrive at the reward values for the current regression tree, it solves a quadratic program with the objective function

$$\min_{\hat{R}_E} ||\hat{R}_E - Proj_{\boldsymbol{\phi}^{(i-1)}}(\hat{R}_E)||_2$$

under the constraint that the policy obtained by solving the MDP with current $\hat{R}_E$ gives actions that match those in the demonstrations for the associated states. Here, $Proj_{\boldsymbol{\phi}^{(i-1)}}(\cdot)$ is the projection of the reward function on the linear combination of the set of features $\boldsymbol{\phi}^{(i-1)}$ from the previous iteration. FIRL interleaves this optimization step with a fitting step during which a new set of features $\boldsymbol{\phi}^{(i)}$ is learned by splitting those leaves of the tree that are too coarse or merging leaves that yield the same average reward values. The iterations stop when the learner detects that further node splitting is unnecessary to maintain consistency with the demonstration.

Recently, regression has been used to extend IRL to linearly-solvable MDPs [51] – a class of MDPs for discrete states but continuous actions. These MDPs define the reward function as the magnitude of the impact of actions on an uncontrolled transition function. Uchibe [52] uses a combination of three neural networks – for the reward function and value-function approximations and third for the ratio of controlled and uncontrolled state transitions – all of which are trained using logistic regression on the trajectory data. Logistic regression is also used in Fu et al. [53] to train a deep neural network (utilized in representing the reward function) by minimizing the cross-entropy between the given trajectories and those generated by a policy from optimizing the currently evolved reward function. A key focus in this technique, labeled AIRL, is on inversely learning rewards that are not tied to any particular transition function, yet yield the same policy as the true rewards.

4.5. Summary and unified views of methods

In the previous subsections, we briefly described the early and foundational IRL methods and briefly remarked on their notable extensions. These have influenced various subsequent methods and spawned improvements as discussed in the later sections. Table 2 abstracts and summarizes the key insights of these methods. It identifies the parameters that are learned, the metric used in the optimization objective, and a distinguishing contribution of the method. This facilitates a convenient comparison across the techniques and helps in aligning the methods with the template given in Algorithm 1.

Our groupings of the methods based on their optimization objectives emphasizes the common grounding of multiple IRL methods. More formal unifications also exist and we review the prominent ones here.

Neu and Szepesvari [27] adopts the function $J(\hat{R}_E, \mathcal{D})$, which quantifies the dissimilarity between the optimal behavior with respect to the learned expert's reward function $\hat{R}_E$ and the demonstrated behavior. This real-valued function that assigns higher values when the two behaviors are less similar serves to unify several methods discussed in Section 4. For instance, the margins defined in Section 4.1 (Eqs. (7), (9), and others) naturally serve as J thereby reiterating our common

Table 2
A categorized summarization of the foundational IRL methods presented in Section 4. We focus on the key aspects of each method and abstract out the shared representations. For example, notice how popular is the linearly-weighted representation of the reward function. Refer to Table 1 for a quick explanation of abbreviations and notations used here.

Method	$\hat{R}_E$ params	Optimization objective	Notable aspect
Max margin methods - maximize the margin between value of observed behavior and the hypothesis			
MMP	$\boldsymbol{w}$	value of obs. τ - max of values from all other τ (Eq. (8))	provable convergence
MAX-MARGIN		feature exp. of policy - empirical feature exp. (Eq. (9))	sample bounds
MWAL		min diff. in value of policy and observed τ across features	first bound on iteration complexity
HYBRID-IRL		empirical stochastic policy - computed policy of expert (Eq. (10))	natural gradients and efficient optimization
LEARCH	$R(\boldsymbol{\phi})$	value of obs. τ - max of values from all other τ (Eq. (8))	nonlinear reward with suboptimal input
Silver et al. [29]			normalization of outlier inputs
Max entropy methods - maximize the entropy of the distribution over behaviors			
MAXENTIRL	$\boldsymbol{w}$	entropy of distribution over trajectories (Eq. (11))	low learning bias
STRUCTURED		entropy of distribution over policies (Eq. (12))	efficient optimization
APPRENTICESHIP			
DEEP MAXENTIRL		gradient of likelihood equivalent of MaxEnt (Eq. (13))	nonlinear reward
PI-IRL			continuous state-action spaces
REIRL		relative entropy of distribution from baseline policy (Eq. (14))	suboptimal input and unknown dynamics
Bayesian learning methods - learn posterior over hypothesis space using Bayes rule			
BIRL	$R(s)$	posterior with Boltzmann data likelihood (Eq. (16))	first Bayesian IRL formulation
Lopes et al. [43]		entropy of multinomial $(p_1(s), p_2(s), \ldots, p_{\lvert A\rvert-1}(s))$ derived from posterior	active learning
GP-IRL	$f(\boldsymbol{r},\boldsymbol{\theta})$	Gaussian process posterior	nonlinear reward
MLIRL	$\boldsymbol{w}$	differentiable likelihood with Boltzmann policy (Eq. (16))	first ML approach
Classification and regression - learn a prediction model that imitates observed behavior			
SCIRL	$\boldsymbol{w}$	Q-function as classifier scoring function	actions as state labels provable convergence
CSI			unknown dynamics
FIRL	regression tree	norm of ($\hat{R}_E$ - projection of $\hat{R}_E$)	avoids manual feature engineering
AIRL	R(s)	regression error between expert demonstration $\mathcal{D}$ and $\mathcal{D}_\pi$	suitable for transfer learning

perspective to MMP, MAX-MARGIN/PROJECTION, MWAL, and HYBRID-IRL techniques. Interestingly, the MAXENTIRL method of Section 4.2 can also be brought under the umbrella of the dissimilarity function! Neu and Szepesvari show that its entropy optimization is equivalent to minimizing the Kullbach-Leibler divergence in $\boldsymbol{w}$ between the unknown $Pr(\tau)$ induced by the expert's behavior and the given empirical $Pr(\tau)$ as given in Eq. (13). Therefore, we may write the dissimilarity function as $J(\hat{R}_E, \mathcal{D}) = -\boldsymbol{w}^T \hat{\mu}^{\boldsymbol{\phi}}(\mathcal{D}) + log\ Z(\boldsymbol{w})$. Along a similar vein, Ghasemipour et al. [54] recently suggests that the dissimilarity function, f-divergence, which generalizes several divergences including the Kullbach-Leibler divergence, could also be used to unify some recent IRL methods. For e.g., AIRL's objective function could be viewed as the Kullbach-Leibler divergence between the expert's trajectory distribution and the learner neural network's distribution of the trajectories.

Another approach toward unification [55] assumes that the expert's policy is given and adopts the maximum causal entropy technique (reviewed in Section 4.2) as a representative and unifying IRL method. First, the approach shows that if the reward function is not limited to a weighted sum of feature functions (it is allowed to be $\mathbb{R}^{S\times A}$) and a closed and convex reward regularizer is additionally added to the optimization objective of the IRL method (to minimize overfitting), then maximizing the regularized causal entropy implicitly yields a policy whose state-visitation frequency is very close to that of the expert's policy if available. Indeed, IRL then can be seen as a dual of a state-visitation frequency matching problem. Second, Ho and Ermon [55] shows that the regularized variant of the maximum causal entropy method also yields the MAX-MARGIN and MWAL methods of Section 4.1 as special cases, thereby offering a unifying approach.

The relatedness between the MAX-MARGIN (and PROJECTION) and MWAL margins can also be analyzed using restrictions of the reward function space. In particular, Ho and Ermon [55] notes that MAX-MARGIN restricts the space of reward functions to $\{\boldsymbol{w}^T\boldsymbol{\phi} : ||\boldsymbol{w}||_2 \leq 1\}$ whereas MWAL imposes a convexity constraint on the weights, $\{\boldsymbol{w}^T\boldsymbol{\phi} : ||\boldsymbol{w}||_1 = 1,\ w_i \geq 0\ \forall i\}$. Thus, the former restriction on the reward space leads to a feature expectation matching that minimizes the L_2 distance between expected feature functions while the latter restriction maximizes the worst-case excess reward among the basis features.

5. Mitigating the challenges

Next, we elaborate how the foundational methods reviewed in Section 4 mitigate the various challenges for IRL introduced in Section 3. Technical challenges often drive the development of methods. Hence, in addition to situating the overall progress of the field, this section will help the reader make an informed choice about the method that may address the

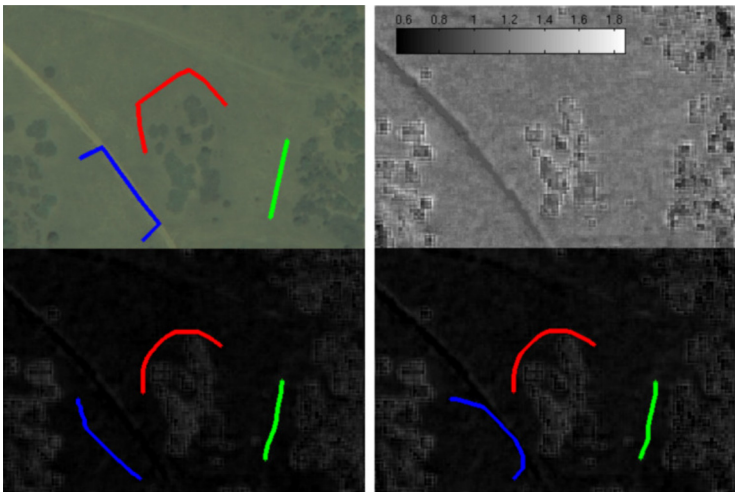


Fig. 8. Learning with a perturbed demonstration in an unstructured terrain. Figure reprinted [29] with permission from MIT Press. An expert provides three demonstration trajectories - red, blue, and green [top left]. The portion of terrain traveled by a presumably achievable red trajectory should have 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 [29] mitigates the lowering of costs [bottom right]. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

challenges in her particular domain. We have also included techniques that purposefully extend some of the foundational methods to address a specific challenge.

5.1. Improving the accuracy of inference

As we mentioned in Section 3.1, IRL's inference accuracy depends on several components of the learning process. Most existing methods aim at ensuring that the input is accurate, reducing the ambiguity among multiple solutions, improving feature selection, and offering algorithmic performance guarantees.

5.1.1. Learning from noisy input

Perturbed demonstrations may be due to noisy sensing or suboptimal actions by the expert. Methods such as REIRL stay robust to perturbations whereas other IRL methods may learn inaccurate feature weights [10] or predict the action poorly [28]. 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. (16) to allow more randomness into the learned policy $\hat{\pi}_E$ 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. [56] shows that the robustness of MAXENTIRL to an imperfect demonstration is better than that of MMP. The recent method by Brown et al. (called T-REX) [49] was shown to learn a reward function from sub-optimal input, which then surpasses the performances of the demonstrator in simulated domains.

To specifically address noisy input, Coates et al. [57] introduces 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 then applied to the resultant noise-free but unobserved trajectories [5]. Melo et al. [58] formally analyzed and characterized the space of solutions for the case when some actions in a demonstration are not optimal and when the demonstration does not include all states. Such demonstrations were obtained by perturbing the distribution that modeled the expert's policy. Taking a step further, Shiarlis et al. [59] performs IRL with some demonstrations that fail to even complete the task.

Suboptimal demonstrations may also include trajectories whose lengths are much longer than expected. As we mentioned in Section 4.1, MMP minimizes the cost of simulated trajectories diverging from the demonstrated ones by noting the difference in state-visitation frequencies of two trajectories. MMP attempts this minimization for a suboptimal demonstration as well, but avoids it if the learning method distinguishes an unusually long demonstration from the optimal ones. Silver et al. [29,18] 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

Several 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 a loss function, which the degenerate $\hat{R}_E = 0$ can not minimize because the function is proportional to state-visitation frequencies [27]. HYBRID-IRL avoids degeneracy in the same way as MMP, and makes the solution less ambiguous 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 makes the demonstration optimal, ambiguous output cannot be entirely avoided using these methods [33].

Making this more stringent, Bayesian and entropy optimization 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 [43] and query for further information in those states. The selection mechanism builds on BIRL and reduces the solution ambiguity compared to BIRL. In general, these methods add optimization constraints and exploit domain knowledge to distinguish between the multiple hypotheses.

We believe that the progress made collectively by the methods in significantly mitigating this challenge of IRL – that it is an underconstrained learning problem – represents a key *milestone* in the progression of this relatively new field.

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 [33]. 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.

5.1.3. Theoretical bounds on accuracy

From a theoretical viewpoint, some methods have better performance guarantees than others. The maximum entropy probability distribution over the space of trajectories (or policies) minimizes the worst-case expected loss [60]. Consequently, MAXENTIRL learns a behavior which is neither much better nor much worse than the expert's [61]. 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 bound 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 [31].

Some methods also analyze and bound the ILE metric for a given threshold of success and a given minimum number of demonstrations. The analysis relies on determining the value of a policy using its generated feature expectations $\boldsymbol{\mu}^{\boldsymbol{\phi}}(\pi)$ [30, 16,26] or state-visitation frequencies $\psi^{\pi}(s)$ [13,16,28,45] 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 [30,62,63]. These bounds apply to methods such as MMP, HYBRID-IRL, and MAXENTIRL that use state-visitation frequencies. Subsequently, the derived bound on bias can be used to analytically compute the maximum error in learning for a given minimum sample complexity Lee et al. [64] 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. A recent extension of BIRL introduced the bounding of approximated ILE as an alternative to the bounding of the difference in feature expectations as a learning objective [65]. The method demonstrated confidence error bounds tighter than the methods that used the latter objective.

In the context of Ng and Russell's early IRL method [2] that takes the policy as input, a recent sample complexity analysis [66] bounds the number of samples needed so that the estimated transition probabilities yield the reward function that would have generated the input policy with the true transition function. The analysis uses a notion of separability (as in support vector machines) and rests on formulating a variant of Ng and Russell's method whose solution not only admits infinitely-many nonzero reward functions but also a reward function for which the input policy is strictly optimal.

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 learn the reward for unseen states lying within the domains of the features of 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. And,

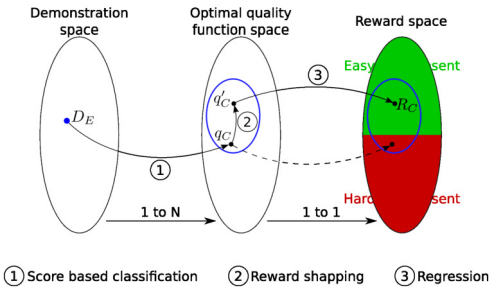


Fig. 9. The 3-step process of IRL in a relational domain - classification outputs a score function, reward shaping optimizes the output, and regression generates an approximate reward function $\hat{R}_E$ corresponding to the optimal score function. Included with permission from authors.

AIRL specifically targets transfer learning by learning a reward function that is disentangled from the demonstrated environment's transition dynamics. Similarly, Finn et al.'s guided cost learning (GCL) [67], which extends REIRL to include a neural network representation for the reward function and a baseline distribution learned using RL, admits an enhanced generalization by learning new instances of a previously-learned task without repeated reward learning. Melo and Lopes [68] shows that the use of bisimulation metrics allows BIRL to achieve improved generalization by partitioning the state space based on a relaxed equivalence between the states.

Munzer et al. [69] extends the classification-regression steps in CSI to include relational learning in order to benefit from the strong generalization and transfer properties that are associated with relational-learning representations. The process shapes the reward function using the scoring 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. 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 trajectories or policies, which reduces the impact that feature selection has on the performance of IRL [27].

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, and MAXENTIRL. If a reward function is linear taking the form of Eq. (3) and the value of each of its k features is bounded from the above, then the probable bound on the error scales linearly with k [16]. However, maximum entropy based methods show an improvement in this aspect with $O(\log k)$ dependence.

Orthogonal to reducing feature dependence, AIRL explicitly focuses on learning robust reward functions that continue to yield the same optimal policy as the true reward function regardless of the underlying transition function. This reduces AIRL's sensitivity to the given transition model and supports transfer learning.

5.4. Analysis and reduction of complexity

The intractability of this machine learning problem due to its large hypothesis space has been significantly mitigated through the widespread 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 MAXENTIRL. Consequently, we view this adoption as another *milestone* for this field.

Next, we discuss ways by which IRL methods have sought to reduce the time and space complexity of an iteration, and mitigate the input sample complexity. Early IRL methods such as Ng and Russell [2] and PROJECTION were mostly demonstrated on grid problems exhibiting less than a hundred states and four actions. Later methods based on entropy optimization and GP-IRL scaled up with MAXENTIRL demonstrating results on a deterministic MDP with thousands of states and actions while taking recourse to approximations.

While an emphasis on reducing the 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 [41]. Specifically, the Markov chain with a uniform prior that approximates the Bayesian posterior converges in polynomial time. Enhancing BIRL via bisimulation also exhibits low computational cost because it need not solve equivalent intermediate MDPs; the computation of the bisimulation metric over space S occurs once regardless of how many times the metric is used as shown in Fig. 10. MWAL requires $\mathcal{O}(\ln k)$ (k is number of features) iterations for convergence, which

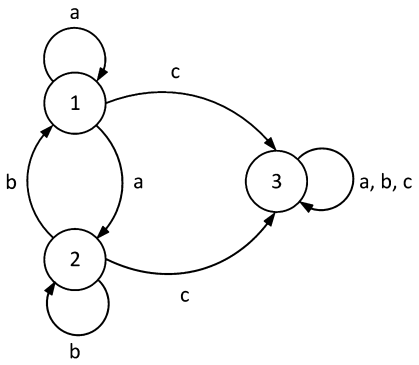


Fig. 10. State equivalence in an MDP with states $S = \{1, 2, 3\}$ and actions $A = \{a, b, c\}$. The similarity in actions and transitions for states 1 and 2 makes them equivalent. Therefore, the selection of optimal actions through expert's policy π_E will be similar in both the states. Demonstration of c in one implies the optimality of c in other. The illustration redraws Fig. 1 in Melo et al. [68].

is lower than the $\mathcal{O}(k \ln k)$ for the PROJECTION method. Though an iteration of FIRL is slower than both MMP and PROJECTION due to the computationally expensive step of regression, FIRL converges in fewer iterations than the latter two methods.

Some optimization methods employ more affordable techniques of gradient computations. In contrast with the fixed-point method in HYBRID-IRL, the approximation method in BIRL with active learning (reviewed in Section 4.3) has a complexity that is polynomial in the number of states. For maximum entropy based parameter estimation, gradient-based methods (e.g., BFGS [70]) outperform iterative scaling approaches [71].

BIRL with active learning 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 [40].

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 a 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. Kretzschmar et al. [11] applies 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 complexity of computing the partition function Z, which appears in the normalization constant of the likelihood (Eq. (13)), increases exponentially with the dimensionality of the state space because it requires finding the complete policy under the current solution $\hat{R}_E$. Approaches for making the likelihood computation in a high-dimensional state space tractable include the use of importance sampling as utilized by REIRL and GCL, down-scaling the state space using low-dimensional features [72], and the assumption by PI-IRL that demonstrations are locally optimal.

For the 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 [71] thereby limiting the memory consumption.

Instead of demonstrating complete trajectories for large tasks, the designer may decompose the task hierarchically. An expert may then give demonstrations at different levels of implementation. The modularity of this process significantly reduces the complexity of learning. For example, Kolter et al. [73] applies such task decomposition toward learning quadruped locomotion by scaling IRL from low- to high-dimensional spaces. Likewise, Rothkopf et al. [74] utilizes the independence between components of a task – each modeled using a stochastic reward function of its own – to introduce decomposition in BIRL.

We may speed up forward learning by quickly computing the values of the intermediate policies learned in IRL. Both LPAL [75] and LPIRL [45] are incremental extensions of MWAL that solve the underlying MDP in MWAL using the dual and primal linear programs, respectively. These linear program formulations make 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 the MDP by their classification-based models.

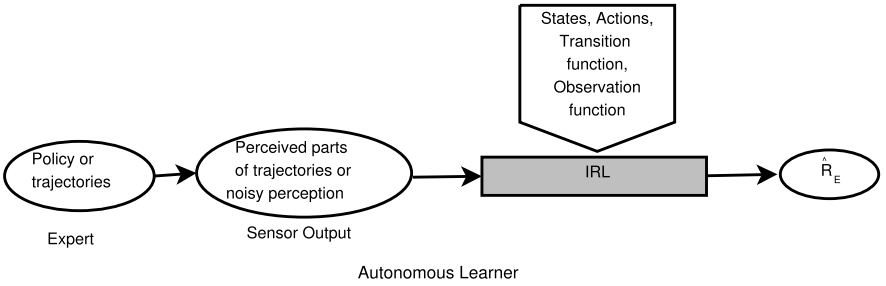


Fig. 11. IRL with imperfect perception of the input trajectory. The learner is limited to using just the perceived portions.

6. Extensions of basic IRL

Having surveyed the foundational methods for IRL in Section 4 and discussed how they and their extensions 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.

Fig. 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.

6.1.2. Methods

Bogert et al. [21] introduces 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 the structured apprenticeship learning method by Boularias et al. 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.

The principle of latent maximum entropy [76,77] allows us to extend the maximum entropy principle to problems with hidden variables. By using this extension, Bogert et al. [78] 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.

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) [79] 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.

Choi and Kim [26] take a different perspective involving an expert that senses its state imperfectly. The expert is modeled as a partially observable MDP (POMDP) [80]. 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

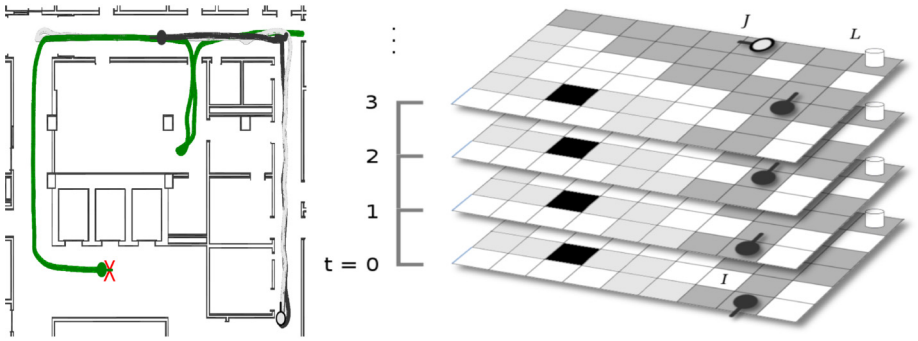


Fig. 12. Prediction of experts' behaviors using multi-robot IRL* [21] in a multi-robot patrolling problem (left). Learner L (green) needs to cross hallways patrolled by experts I (black, reward R_{E_1}) and J (gray, reward R_{E_2}). It has to reach goal 'X' without being detected. Due to occlusion, just 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.

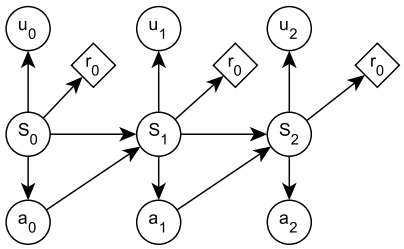


Fig. 13. Hidden-variable MDP. u_i is a noisy observation, by the learner, of the state s_i reached after taking action a_{i-1}. The source of illustration is [79]. The figure is shown here with permission from publisher.

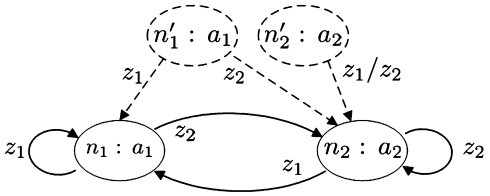


Fig. 14. In this illustration, similar to the one in Choi et al. [26], consider a POMDP with two actions and two observations. π_E (solid lines) is a FSM with nodes $\{n_1, n_2\}$ associated to actions and edges as observations $\{z_1, z_2\}$. The one-step deviating policies (dashed lines) $\{\pi_i\}_{i=1}^2$ 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.

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. Fig. 14 illustrates this approach. However, a 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.

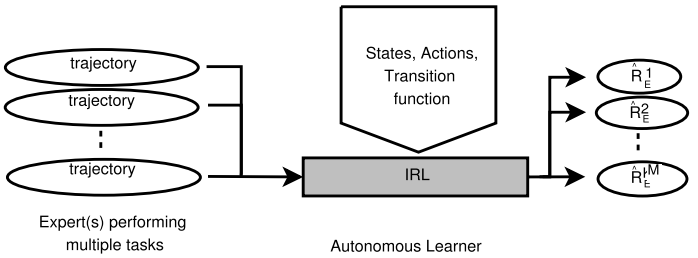


Fig. 15. Multi-task IRL involves learning multiple reward functions. The input is mixed trajectories executed by a single or multiple expert(s) realizing behavior driven by different reward functions but in the same environment. The output is a set of reward functions, each one associated with a subset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

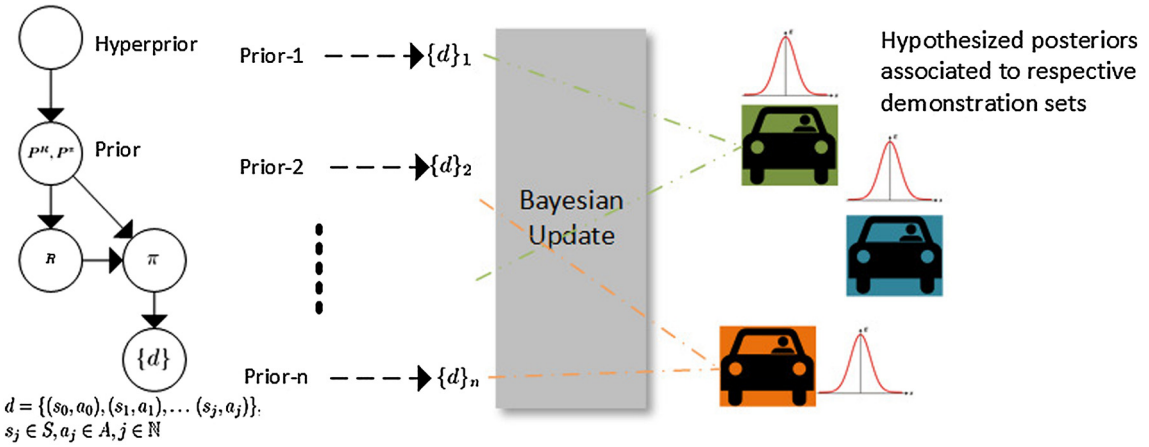


Fig. 16. Updating the hyperprior in parametric multi-task BIRL modifies the dependent priors P^R on reward functions and priors P^π on policies – the figure shows it as a sequence of priors. The variation in priors is assumed to influence the observed trajectories $\{\tau\}_i, i \in \mathbb{N}$. Bayesian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

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 M MDPs each without the reward function where M 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 $\hat{R}_E^1$, $\hat{R}_E^2$, …, $\hat{R}_E^M$ that best explain the observed behavior.

Fig. 15 gives the schematic for this important IRL extension. Having to associate a subset of input trajectories from the demonstration 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. Diverse methods have sought to address the problem defined in Definition 4, and we briefly review them below.

6.2.2. Methods

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 functions with common features $\{\hat{R}_E^i = \boldsymbol{w}_i^T \phi\}_{i=1}^{M}, M \in \mathbb{N}$. Given prior knowledge of M, the solution is a pair of weight vector $\boldsymbol{w}_i \in \mathbb{R}^k$ and a correspondence probability for each reward function $\hat{R}_E^i$. 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.

Continuing with the assumption of knowing M, 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 [61] 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.

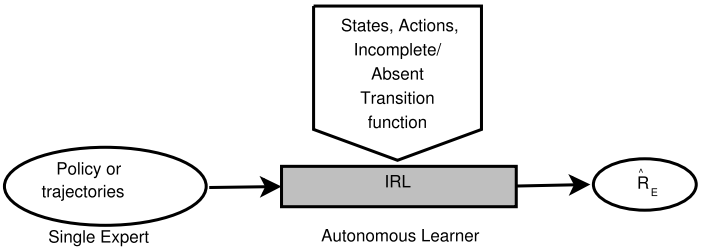


Fig. 17. IRL with incomplete model of transition probabilities.

In contrast to parametric clustering, DPM-BIRL is a clustering method that learns multiple reward specifications from unlabeled fixed-length trajectories [81]. It differs from the previous methods in this section in that the number of experts are not known. Therefore, it addresses the problem in Definition 4 with M 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. [21] extend IRL* and MAXENTIRL to multiple experts who may interact, albeit sparsely. These experts are mobile robots patrolling a narrow hallway. 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. [82] adopt this approach and model multiple interacting experts as a decentralized general-sum stochastic game. Similarly, Lin et al. [83] presents a Bayesian method that learns 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 $\mathcal{D}$.

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

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 the remaining state-action pairs to an absorbing state. To formally guarantee the accuracy of learned dynamics and thereby the reward function, the algorithm

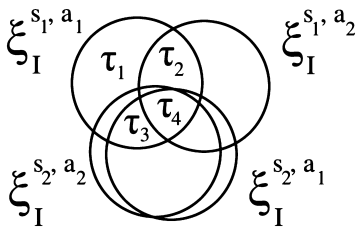


Fig. 18. In mIRL*\T, $\xi_i^{(s,a)} = \{\tau_1, \tau_2, \ldots \tau_b\}$ denotes the transition-features for transition (s, a, s') of expert i, s' is intended next state. Computation of unknown probabilities by using probabilities of transition-features, $\prod_{\tau \in \xi_i^{(s,a)}} P(\tau) = T_{sa}(s, a, s')$, is feasible because different transitions share transition features among them. Source of illustration is [84] and figure is reprinted with author's permission.

leverages a theoretical upper bound on the accuracy of the learned transition model if the learner receives a given minimum amount of demonstration [62].

While MWAL assumes that a learner fully observes the states, mIRL*$_{\backslash T}$ [21] 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 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 local frequency based estimation.

Furthermore, for estimating the unknown dynamics, GCL [85] 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. On the other hand, Boularias et al. [40] shows that the transition models approximated from a small set of demonstrations may result in highly inaccurate solutions.

Of course, model-free IRL completely bypasses learning the transition dynamics. Prominent model-free methods include AIRL and PI-IRL, which were reviewed in Sections 4.4 and Section 5.4, respectively. A recent method [86] builds on MLIRL by replacing the traditional Bellman update in MLIRL to perform model-free Q-learning, which is modified to be differentiable. One modification replaces the max operator in the update with an averaging operator. An alternative modification replaces the max operator with a Boltzmann-weighted mean. These modifications demonstrated good results on the real-world NGSIM data set toward learning driver preferences during a freeway merge.

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 [87]. 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 L_1 regularization objectives can learn robustly when input features are not completely relevant [88]. 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.

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. Recall that FIRL constructs features by capturing these relationships in a regression tree. 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 [89]. BNP-FIRL is demonstrated to construct features more succinct than those by FIRL. Of course, all these methods are applicable only when the feature space is sufficient to 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 simplistic in complex real tasks especially when raw sensory input is used to compute the reward values [67]. 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, and promote overfitting.

As our definition of IRL given in Definition 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.

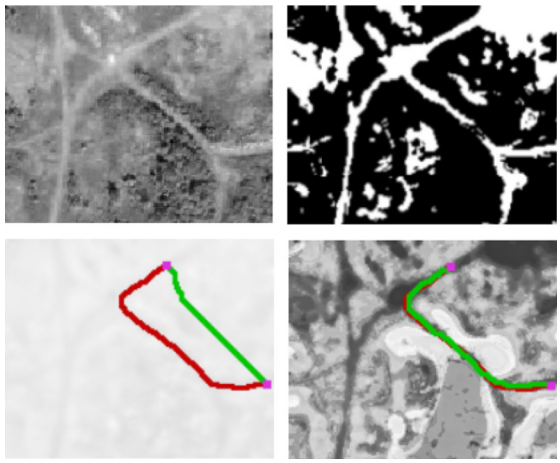


Fig. 19. Learning a nonlinear reward function with boosted features improves performance over linear reward. Learner needs to imitate example paths drawn by humans in overhead imagery. Upper left panel - base features for a region. Upper right panel - image of the region used for testing. Red path is 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 [87] with permission from MIT press. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

Table 3
A comparative analysis of the challenges addressed by the extensions introduced in Section 6. Please refer to Table 1 for the explanation of abbreviations and notations used here.

Method	$\hat{R}_E$ params	Optimization Obj.	Notable aspects
Methods for incomplete and noisy observations			
POMDP-IRL	$\boldsymbol{w}$	feature expectation of policy - empirical feature exp.	MAX-MARGIN with noisy observations of expert
HIOC		entropy of distribution	modeling noise in input
IRL*		over trajectories	IRL with hidden variables
Methods for multiple tasks			
Dimitrakakis et al. [61]	$\{R_E^i\}_{i=1}^M$	joint dist. over rewards and policies	hierarchical BIRL for multiple hypotheses
DPM-BIRL	$\{R_E^i\}^*$	generative DP governing $\mathcal{D}$	first nonparametric multi-task technique
Reddy et al. [82]	$\{R_E^i\}_{i=1}^M$	joint policy value	modeling expert interactions
Lin et al. [83]			using game theory
Methods for incomplete model parameters			
MMPBOOST		value of observed τ- max of values from all other τ	max. likelihood derived classifier to fit ϕ
MWAL	$\boldsymbol{w}$	min diff. in value of policy and observed τ across features	first formal bound on learning dynamics
Model-free MLIRL [86]		differentiable Q-learning update rule	good performance on real-world driving data
BNP-FIRL	$\boldsymbol{w}$, primitive features	generative IBP governing {primitive features, $\boldsymbol{w}$, $\mathcal{D}$}	integrating feature learning in BIRL

To overcome the constraint 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 $\hat{R}_E = f(\phi)$. Fig. 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. [35], Finn et al. [67], and AIRL represent a complex nonlinear cost function using a neural network approximation, thereby avoiding the assumption of a linear form.

Table 3 abstracts and summarizes the key properties of the methods reviewed in this section. Some of these methods build on the foundational methods reviewed in Section 4 while others are new introduced with the aim of generalizing IRL in pragmatic ways.

7. 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 but accessible look at IRL, and focuses on the specific ways by

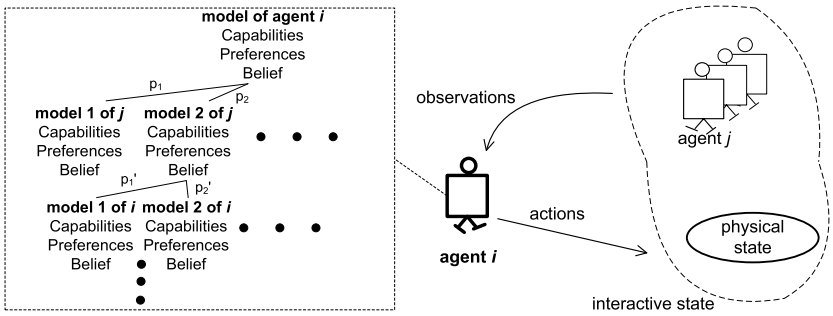


Fig. 20. The state of I-POMDP evolves as an *interactive state space* that encompasses the computable models (beliefs, preferences, and capabilities) for other agents and the physical states of the environment. Agent i maintains and updates his models of other agents.

which various methods mitigated challenges and contribute to the ongoing advance of IRL. 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. 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, until last year, PROJECTION and MWAL were the only methods among the foundational ones that provided a sample complexity analysis. These methods use Hoeffding's inequality to relate the error in estimated feature expectations with the minimum sample complexity. Only recently was the sample complexity of Ng and Russell's early IRL method analyzed [66]. 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. 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.

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.

In addition to these immediate avenues of future work, we also discuss lines of inquiry below that could lead to a better understanding of IRL and lead to progress over the longer term.

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. [26] observes that when the values of learned policy $\hat{\pi}_E$ and expert's policy π_E are evaluated on the true reward 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 learned reward $\hat{R}_E$ 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.

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 [83,82]. These classes include models for fully-observable state spaces (Markov games [94], multi-agent Markov decision processes [95], interaction-driven Markov games [96]) and for partially-observable states (partially observable identical-payoff stochastic games [97], multi-agent team decision problems [98], decentralized Markov decision processes [99], and interactive POMDPs [100] illustrated in Fig. 20). researchers must extend the existing approaches to this level of inference. A special case worth initiative is a single learner with multiple interactive experts in partial observation settings. Outside the domain of IRL, we note behavior prediction approaches related to inverse optimal control in multi-agent game-theoretic settings [101]. 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, the preferences of agent(s) may change with time, and the reward function can be time-variant i.e., $R : S \times A \times \eta \rightarrow \mathbb{R}$. 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 Kalakrishnan et al. [102].

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This research was partly funded by a grant from the Toyota Research Institute of North America (TRI-NA) and a grant from NSF's NRI program IIS-1830421. We acknowledge feedback from Scott Niekum and several anonymous reviewers that helped improve the manuscript.

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, in: Proceedings of the Seventeenth International Conference on Machine Learning, 2000, pp. 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, Commun. 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 Rev. 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: 22nd International Conference on Neural Information Processing Systems, 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, Int. J. Robot. Res. 35 (11) (2016) 1289–1307, https://doi.org/10.1177/0278364915619772.
[12] B. Kim, J. Pineau, Socially adaptive path planning in human environments using inverse reinforcement learning, Int. J. Soc. Robot. 8 (1) (2016) 51–66.
[13] G. Neu, C. Szepesvári, Apprenticeship learning using inverse reinforcement learning and gradient methods, in: Twenty-Third Conference on Uncertainty in Artificial Intelligence, 2007, pp. 295–302, arXiv: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] A. Tucker, A. Gleave, S. Russell, Inverse reinforcement learning for video games, arXiv:1810.10593, 2018.
[16] 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.
[17] 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.
[18] N.D. Ratliff, D. Silver, J.A. Bagnell, Learning to search: functional gradient techniques for imitation learning, Auton. Robots 27 (1) (2009) 25–53.
[19] 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.
[20] A. Vogel, D. Ramachandran, R. Gupta, A. Raux, Improving hybrid vehicle fuel efficiency using inverse reinforcement learning, in: AAAI Conference on Artificial Intelligence, 2012.
[21] 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.
[22] A. Hussein, M.M. Gaber, E. Elyan, C. Jayne, Imitation learning: a survey of learning methods, ACM Comput. Surv. 50 (2) (2017) 21–35.
[23] N. Ab Azar, A. Shahmansoorian, M. Davoudi, From inverse optimal control to inverse reinforcement learning: a historical review, Annu. Rev. Control 50 (2020) 119–138.
[24] L.P. Kaelbling, M.L. Littman, A.W. Moore, Reinforcement learning: a survey, J. Artif. Intell. Res. 4 (1) (1996) 237–285.
[25] S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, second edition, Prentice Hall, 2003.
[26] J. Choi, K.-E. Kim, Inverse reinforcement learning in partially observable environments, J. Mach. Learn. Res. 12 (2011) 691–730.
[27] G. Neu, C. Szepesvári, Training parsers by inverse reinforcement learning, Mach. Learn. 77 (2–3) (2009) 303–337.
[28] 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.
[29] 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.
[30] 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.

[31] 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.
[32] E.T. Jaynes, Information theory and statistical mechanics, Phys. Rev. 106 (1957) 620–630.
[33] B. Ziebart, J. Bagnell, A. Dey, Modeling interaction via the principle of maximum causal entropy, in: International Conference on Machine Learning, ICML, 2010, pp. 1255–1262.
[34] K. Lee, S. Choi, S. Oh, Maximum causal Tsallis entropy imitation learning, in: Advances in Neural Information Processing Systems, Curran Associates, Inc., 2018, pp. 4403–4413.
[35] M. Wulfmeier, I. Posner, Maximum entropy deep inverse reinforcement learning, arXiv preprint.
[36] 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.
[37] E. Theodorou, J. Buchli, S. Schaal, A generalized path integral control approach to reinforcement learning, J. Mach. Learn. Res. 11 (2010) 3137–3181.
[38] 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, 2012, pp. 227–242.
[39] S. Kullback, Information theory and statistics, 1968.
[40] 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.
[41] 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.
[42] J. Choi, K. eung Kim, Map inference for bayesian inverse reinforcement learning, in: Advances in Neural Information Processing Systems, vol. 24, 2011, pp. 1989–1997.
[43] 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.
[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] 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.
[47] B. Taskar, V. Chatalbashev, D. Koller, C. Guestrin, Learning structured prediction models: a large margin approach, in: 22nd International Conference on Machine Learning, 2005, pp. 896–903.
[48] 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, ECML PKDD 2013, in: LNAI, vol. 8188, Springer-Verlag New York, Inc., New York, NY, USA, 2013, pp. 1–16.
[49] D. Brown, W. Goo, P. Nagarajan, S. Niekum, Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations, in: Proceedings of the 36th International Conference on Machine Learning, in: Proceedings of Machine Learning Research, vol. 97, 2019, pp. 783–792.
[50] 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.
[51] E. Todorov, Linearly-solvable Markov decision problems, in: Advances in Neural Information Processing Systems, 2007, pp. 1369–1376.
[52] E. Uchibe, Model-free inverse reinforcement learning by logistic regression, Neural Process. Lett. 47 (2018) 891–905.
[53] J. Fu, K. Luo, S. Levine, Learning robust rewards with adverserial inverse reinforcement learning, in: International Conference on Learning Representations, 2018, https://openreview.net/forum?id=rkHywl-A.
[54] S.K.S. Ghasemipour, R. Zemel, S. Gu, A divergence minimization perspective on imitation learning methods, in: Conference on Robot Learning, 2020, pp. 1259–1277.
[55] J. Ho, S. Ermon, Generative adversarial imitation learning, in: Advances in Neural Information Processing Systems, NIPS, vol. 29, 2016, pp. 4565–4573.
[56] B.D. Ziebart, J.A. Bagnell, A.K. Dey, Modeling interaction via the principle of maximum causal entropy, in: J. Fürnkranz, T. Joachims (Eds.), Proceedings of the 27th International Conference on Machine Learning, ICML-10, Omnipress, 2010, pp. 1255–1262.
[57] 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.
[58] 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, 2010, pp. 349–354.
[59] 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.
[60] P.D. Grünwald, A.P. Dawid, Game theory, maximum entropy, minimum discrepancy and robust Bayesian decision theory, Ann. Stat. 32 (1) (2004) 1367–1433, http://arxiv.org/abs/0410076v1.
[61] 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.
[62] U. Syed, R.E. Schapire, A game-theoretic approach to apprenticeship learning–supplement, 2007.
[63] M.C. Vroman, Maximum likelihood inverse reinforcement learning, Ph.D. thesis, Rutgers, The State University of New Jersey, 2014.
[64] S.J. Lee, Z. Popović, Learning behavior styles with inverse reinforcement learning, ACM Trans. Graph. 29 (4) (2010) 122:1–122:7.
[65] D.S. Brown, S. Niekum, Efficient probabilistic performance bounds for inverse reinforcement learning, in: Thirty-Second AAAI Conference on Artificial Intelligence, 2018.
[66] A. Komanduru, J. Honorio, On the correctness and sample complexity of inverse reinforcement learning, in: Advances in Neural Information Processing Systems, vol. 32, 2019, pp. 7112–7121.
[67] C. Finn, S. Levine, P. Abbeel, Guided cost learning: deep inverse optimal control via policy optimization, preprint, arXiv:1603.00448.
[68] 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.
[69] 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.
[70] R. Fletcher, Practical Methods of Optimization, Wiley-Interscience Publication, Wiley, 1987.
[71] 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.
[72] 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.

[73] 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.
[74] C.A. Rothkopf, D.H. Ballard, Modular inverse reinforcement learning for visuomotor behavior, Biol. Cybern. 107 (4) (2013) 477–490.
[75] 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.
[76] S. Wang, R. Rosenfeld, Y. Zhao, D. Schuurmans, The latent maximum entropy principle, in: IEEE International Symposium on Information Theory, 2002, p. 131.
[77] S. Wang, D. Schuurmans, Yunxin Zhao, The latent maximum entropy principle, ACM Trans. Knowl. Discov. Data 6 (8) (2012).
[78] 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.
[79] 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.
[80] 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.
[81] 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.
[82] 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.
[83] X. Lin, P.A. Beling, R. Cogill, Multi-agent inverse reinforcement learning for zero-sum games, CoRR, arXiv:1403.6508 [abs].
[84] 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.
[85] 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.
[86] V. Jain, P. Doshi, B. Banerjee, Model-free irl using maximum likelihood estimation, in: AAAI Conference on Artificial Intelligence, vol. 19, 2019, pp. 3951–3958.
[87] 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.
[88] 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, p. 78.
[89] 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.
[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, preprint, arXiv:1005.2908.
[92] R. Eberhart, J. Kennedy, Particle swarm optimization, in: Proceedings of the IEEE International Conference on Neural Networks, vol. 4, 1995, pp. 1942–1948.
[93] X.-S. Yang, Firefly algorithm, stochastic test functions and design optimisation, preprint, arXiv:1003.1409.
[94] 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.
[95] 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.
[96] 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.
[97] 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.
[98] D.V. Pynadath, M. Tambe, The communicative multiagent team decision problem: analyzing teamwork theories and models, J. Artif. Intell. Res. 16 (1) (2002) 389–423.
[99] 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.
[100] P.J. Gmytrasiewicz, P. Doshi, A framework for sequential planning in multi-agent settings, J. Artif. Intell. Res. 24 (1) (2005) 49–79.
[101] K. Waugh, B.D. Ziebart, J.A. Bagnell, Computational rationalization: the inverse equilibrium problem, CoRR, arXiv:1308.3506 [abs].
[102] 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.