A Review of Inverse Reinforcement Learning Theory and Recent Advances

Shao Zhifei
School of Electrical and
Electronic Engineering
Nanyang Technological University
Singapre
Email: zshao1@e.ntu.edu.sg

Er Meng Joo
School of Electrical and
Electronic Engineering
Nanyang Technological University
Singapore
Email: EMJER@ntu.edu.sg

***Abstract*—A major challenge faced by machine learning community is the decision making problems under uncertainty. *Reinforcement Learning* (RL) techniques provide a powerful solution for it. An agent used by RL interacts with a dynamic environment and finds a policy through a reward function, without using target labels like Supervised Learning (SL).**

However, one fundamental assumption of existing RL algorithms is that reward function, the most succinct representation of the designer's intention, needs to be provided beforehand. In practice, the reward function can be very hard to specify and exhaustive to tune for large and complex problems, and this inspires the development of Inverse Reinforcement Learning (IRL), an extension of RL, which directly tackles this problem by learning the reward function through expert demonstrations. IRL introduces a new way of learning policies by deriving expert's intentions, in contrast to directly learning policies, which can be redundant and have poor generalization ability. In this paper, the original IRL algorithms and its close variants, as well as their recent advances are reviewed and compared.

***Index Terms*—Reinforcement learning, inverse reinforcement learning, reward function, expert demonstration.**

I. Introduction

Reinforcement Learning (RL) techniques solve problems through an agent, which acquires experience through interactions with a dynamic environment. The result is a policy that can resolve complex tasks without specific instructions on how the tasks are to be achieved [1]. Unlike Supervised Learning (SL), RL doesn't require target labels, which can be either unavailable or too expensive to obtain or not representative enough to cover all possible areas in real applications. Because of this, RL can be easily transformed into different scenarios and has better generalization abilities than SL [2]. These promises are beguiling, especially for the complex tasks where the exact executions are hard to specify. However, there is one problem associated with RL, that is the reward function in RL has to be specified in advance, and its design difficulties promoted the introduction of Inverse Reinforcement Learning (IRL), where the reward function can be derived from expert's demonstrations. In this paper, the algorithm developments in IRL are surveyed, and the organization is as follows: The rest of this section introduces the preliminary knowledge of RL, as well as the origin and problem formulation of IRL. Section II is devoted to the introduction of original IRL algorithms. Section III is mainly concerned with new approaches to solve IRL in other perspectives and various improvements. These improvements enable IRL to be implemented in more practical and complex problems. The last section summarizes this paper and provides authors' perspective about the current research situation in IRL.

A. MDP and POMDP

The original RL algorithms assume the problems to be solved satisfy *Markov decision process* (MDP), which is a form of tuple $(S, A, P^a_{ss'}, \gamma, R)$ [3]:

- S: A set of possible states that represent the dynamic environment.
- A: A set of possible actions that the agent can select from at each time step. After executing the selected action, the system is transformed to the next state and gets a reward.
- $P^a_{ss'}$: The state transition probabilities. For an action $a \in A(s)$ taken in a state $s \in S$, the probability of transforming to the next state s' is given by $P^a_{ss'}$.
- γ: A discounting factor in the range of $[0, 1]$, which controls the prediction horizon of the algorithm.
- R: The reward function that specifies the reward gained at a specific state. It contains the information that guides the agent towards the goal.

MDP has *Markov property*, where the effect of taking an action in a state only depends on the current state-action pair and not on the prior history:

$$\begin{aligned} P^a_{ss'} &= P\{s_{t+1} = s', r_{t+1} = r | s_t, a_t\} \\ &= P\{s_{t+1} = s', r_{t+1} = r | s_t, a_t, r_t \dots r_1, s_0, a_0\} \end{aligned} \tag{1}$$

The assumption of MDP in RL provides good theoretical foundation for developing algorithms. However, this assumption natively allows agent to access the true global state of the environment, thus it is too strong in practice because the agent usually has limited sensory abilities [4]. To address this problem, an extension of MDP called *Partially Observable Markov Decision Processes* (POMDPs) [5] was introduced, which is a form of tuple $(S, A, P^a_{ss'}, \gamma, R, Z, O, b_0)$. Besides the elements shared by MDP, three more are introduced in POMDP:

- Z is a finite set of observations.

978-1-4673-1509-8/12/$31.00 ©2012 IEEE

- $O : S \times A \rightarrow \Pi(Z)$ is an observation model, where $\Pi(Z)$ is the space of probability distribution over Z, and $O(s', a, z)$ denotes the probability of observing z when applying action a and arriving at state s'.
- b_0 is a belief function, and $b_0(s)$ gives the probability of starting in state s.

The difference between MDP and POMDP is that in POMDP, true states are unavailable to the agent and can only be estimated through the hints provided by the observations. Therefore the true states s can only be approximated by the belief function $b(s)$, which denotes the probability in s. At each time step, the belief function needs to be updated as follows:

$$b'(s') = \frac{O(s', a, z) \sum_{s \in S} P^a_{ss'} b(s)}{\sum_{s' \in S} O(s', a, z) \sum_{s \in S} P^a_{ss'} b(s)} \tag{2}$$

where $b'(s')$ is the updated belief function of state s'.

Clearly, the update process of the belief function has Markov property: the next belief function only depends on the current belief, current action and observation. This makes the problems in POMDP to be considered in a similar way in MDP and bridges the gap between them. Regarding to the specific RL solutions under POMDP, readers can refer to the survey in [6].

B. Value Function and Policy

A policy $\pi(s, a) : S \rightarrow A$ is the probability of taking action $a \in A(s)$ in state $s \in S$. In order to estimate the performance of a policy, we need to specify its performance metric, i.e., the reward function $r(s, a) : S \times A \rightarrow \mathbb{R}$, which denotes the immediate reward obtained after executing action a in state s. It is the most succinct representation of the user's intention since it specifies the agent the intrinsic desirability of an event for the system. However, to achieve as many rewards as possible, the ones from the future have to be taken into account. Hence the value function is introduced to estimate the accumulated rewards starting from a state:

$$\begin{aligned} V^\pi(s) =& E_\pi\{R_t | s_t = s\} \\ =& \sum_{a \in A(s)} \pi(s, a) \sum_{s' \in S} P^a_{ss'} \{r(s, a) + \gamma V^\pi(s')\} \end{aligned} \tag{3}$$

where π denotes the current policy being followed, and R_t is the reward estimation formula which usually takes a discounted form:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{4}$$

Sometimes it is easier to directly estimate the desirability of a state-action pair (s, a), i.e., $\sum_{a \in A(s)} \pi(s, a) = 1$ and Eq. (3) becomes a *Q-value function*. $Q^\pi : S \times A \rightarrow \mathbb{R}$ is the expected value return when taking action a in state s, and following policy π thereafter:

$$Q^\pi(s, a) = E_\pi\{R_t | s_t = s, a_t = a\} \tag{5}$$

The objective of the agent is to find an optimal policy π^* which gives the optimal state value function or Q-value function:

$$V^*(s) = \max_a \sum_{s' \in S} P^a_{ss'} (R^a_{ss'} + \gamma V^*(s')) \tag{6}$$

$$Q^*(s, a) = \sum_{s' \in S} P^a_{ss'} (R^a_{ss'} + \gamma \max_{a'} Q^*(s', a')) \tag{7}$$

Eq. (6) and Eq. (7) are also known as the *Bellman optimality equations*, and they essentially explained the objective of the optimal policy π^*, which is to find the policy that gives the highest reward return. This fact is also used as the fundamental strategy for many IRL algorithms.

C. Problem Origin and Formulation of IRL

The most intuitive representation of an expert's intention is its policy, and it is possible to directly learn a policy from expert's demonstrations. For example, regression techniques have been successfully applied to autonomous navigation [7] and human-to-robot skill transfer [8]. However, these approaches are based on SL, which usually suffers from insufficient number of training samples and poor generalization ability. Furthermore, policy representation of expert's intention is rather redundant and usually restricted to certain scenarios, but the robot may be required to do a different task [9]. Generally speaking, the strategy of SL is to penalize the behaviors that deviate from the target trajectories, without considerations of the underlying system dynamics. On the other hand, the reward function can succinctly represent the expert's knowledge, and this knowledge is transferable to other scenarios.

The conventional RL theory usually assumes the reward function to be predetermined and fixed. However, the appropriate reward function is hard to hand specify for complex problems [10]. Therefore, the problem of IRL is first formulated as follows [11]:

- **Given**
  1) measurements of an agent's behavior over time,
  2) sensory inputs to that agent if needed,
  3) a model of the environment if available.
- **Determine** the reward function that can mostly justify the agent's behavior.

The idea of IRL has been well studied in the economics area, which is the decision making process of a person when facing multiple choices with various attributes [12]. In the machine learning community, IRL problem is first formally studied by [13], and described as follows:

- **Given**
  1) a finite state space S,
  2) a set of actions $A = \{a_1, a_2, \ldots a_k\}$,
  3) transition probability $P^a_{ss'}$,
  4) a discount factor γ,
  5) a policy π.
- **Determine** a set of possible reward functions R such that π is the optimal policy for the given MDP $(S, A, \{P^a_{ss'}\}, \gamma, R)$.

In summary, IRL can be considered as a reverse procedure of RL problems, which assumes that the expert's demonstration is an optimal policy π^*, derived according to some reward function R^*. The objective of IRL is to learn this unknown R^*. IRL is also closely related to apprenticeship learning [10], which aims at deriving a policy that is almost as good as the expert's demonstrations under this reward function R^*, for a known initial state. Apparently, apprenticeship learning has a weaker objective since it does not necessarily need to derive the true reward function.

II. Original IRL Algorithms

The original IRL algorithms were introduced in [13] and [10]. These algorithms basically formulate the IRL problem as a linear programming procedure with constraints corresponding to the optimal condition. There are mainly three cases existed in the IRL problem formulation:

- Finite-state MDP with known optimal policy.
- Infinite-state MDP with known optimal policy.
- Infinite-state MDP with unknown optimal policy, but demonstrations are given.

Among these cases, the last one is the closest to practical problems because usually only the expert's demonstrations are available rather than the explicit policy. However, for the completeness of the review, all three cases will be briefly discussed. The formulas and theories of the first two cases are mainly adopted from [13].

A. IRL Problems under Finite-state with Known Optimal Policy

The easiest scenario is a finite-state MDP with a known and completely observed policy. For the ease of representation, the optimal policy is given by $\pi(s) \equiv a_1$. Then the given policy is optimal if and only if the reward function R satisfies

$$(P_{a1} - P_a)(I - \gamma P_{a1})^{-1} R \geq 0 \tag{8}$$

where P_{a1} is the transition probability matrix of taking the optimal action a_1, and P_a is the transition probability of other policies with $a \in A/a_1$. The proof can be found in [13].

However, this IRL problem is *ill-posed*. In the same sense that there usually exist multiple optimal policies under the same reward function, and multiple reward functions can provide the same optimal policy [14]. For instance, $R = 0$ will always be a solution, which means there are no distinctions among actions and makes $\pi(s) \equiv a_1$ one of the optimal solutions. Other than this solution, there may still be many reward functions that can generate the given optimal policy. Therefore, two additional criteria are introduced to differentiate a good choice of R from the other solutions:

- Maximize the difference between the best and second best solutions.
- Simple solutions are preferred.

The first criterion corresponds to the following formula with $a \in A/a_1$:

$$\max\{\sum_{i=1}^{N} \min_{a}\{(P_{a_1}(i) - P_a(i))(I - \gamma P_{a_1}(i))^{-1} R\}\} \tag{9}$$

where $\min_{a \in A/a_1}\{(P_{a_1}(i) - P_a(i))(I - \gamma P_{a_1}(i))^{-1} R$ is the return difference between the best and second best policy at time step i and N is the total number of states.

Furthermore, solutions that mainly assign small rewards to most states and large rewards to a few states are considered *simple* and preferred. Therefore a penalty term $\lambda \|R\|_1$ is added to the objective function to balance between the two additional criteria. In summary, the reward function optimization problem can be formulated as follows and solved via linear programming [13]:

$$\max\{\sum_{i=1}^{N} \min_{a}\{(P_{a_1}(i) - P_a(i))(I - \gamma P_{a_1})^{-1} P\} - \lambda \|R\|_1\}$$
$$s.t.\ (P_{a_1} - P_a)(I - \gamma P_{a_1})^{-1} R \geq 0\ \&\ |R_i| \leq R_{\max} \tag{10}$$

B. IRL Problems under Infinite-state and Known Policy

For infinite state space problems, searching through all reward functions is impractical. Instead, the reward function is approximated using a linear combination of useful features:

$$R(s) = \alpha_1 \phi_1(s) + \alpha_2 \phi_2(s) + \cdots + \alpha_d \phi_d(s) = \alpha \phi(s) \tag{11}$$

where $\phi(s) = \begin{bmatrix} \phi_1(s) & \dots & \phi_d(s) \end{bmatrix}^T$ are predefined basis functions, i.e., features, and d is the number of useful features in the reward function, and $\alpha = \begin{bmatrix} \alpha_1 & \dots & \alpha_d \end{bmatrix}$ are the parameters to be tuned during the learning process.

If fact, Eq. (11) provides another way to represent the value function using *feature expectations* [10]:

$$V(\pi) = E[\sum_{t=0}^{\infty} \gamma^t R(s_t)|\pi] = E[\sum_{t=0}^{\infty} \gamma^t \sum_{i=1}^{d} \alpha_i \phi_i(s_t)|\pi]$$
$$= \sum_{i=1}^{d} \alpha_i E[\sum_{t=0}^{\infty} \gamma^t \phi_i(s_t)|\pi] \tag{12}$$

where D is the initial state distribution, and $s_0 \sim D$ means the expectation is taken from random state sequence starting from s_0 and continuing according to D. Then the individual feature expectation can be represented as:

$$\mu_i(\pi) = E[\sum_{t=0}^{\infty} \gamma^t \phi_i(s_t)|\pi] \tag{13}$$

Clearly, $V(\pi)$ can be represented as a linear combination of individual feature expectations:

$$V(\pi) = V(\pi) = \sum_{i=1}^{d} \alpha_i \mu_i(\pi) = \alpha \mu(\pi) \tag{14}$$

where α is a weight vector, and $\mu(\pi)$ is a feature vector: $\mu(\pi) = \begin{bmatrix} \mu_1(\pi) & \mu_2(\pi) & \cdots & \mu_d(\pi) \end{bmatrix}^T$. Since $\pi(s) \equiv a_1$ is the optimal policy, the following holds:

$$\alpha \mu(\pi^E) \geq \alpha \mu(\pi^{(i)}) \tag{15}$$

This means that given the appropriate reward function, the expert always performs better or equal to any other policy $\pi^{(i)}$. Since it is difficult to evaluate the infinite state space, a subsample of S_0 is used. Because approximation has been

applied to R, there may not exists R other than $R = 0$ that can give the provided optimal policy, the optimization problem is relaxed with a penalty when the constraint in Eq. (15) is violated:

$$\begin{aligned} &\max\{\sum_{s\in S_0}\min_{a\in A/a_1}\{q(\alpha\mu(\pi^E)\geq\alpha\mu(\pi^{(i)}))\}\} \\ &s.t. \quad |\alpha_i|\leq 1,\ i=1,2,\ldots,d \end{aligned} \tag{16}$$

where $q(x)$ is a penalty function and c is a positive constant:

$$q(x)=\begin{cases} x & \text{if } x\geq 0 \\ cx & \text{if } x<0 \end{cases} \tag{17}$$

C. Apprenticeship Learning with Sample Trajectories

Many robotic tasks need to specify a trajectory for a robot to follow, such as moving a robotic arm. However, for complex applications, the desired trajectory is hard to describe. Such as the helicopter aerobatic maneuver trajectory, not only it should correspond to the job, but also be consistent with the helicopter dynamics [15].

The demonstrations provided by the expert can be used to extract the desired trajectory through apprenticeship learning. The traditional apprenticeship learning algorithms use SL to directly mimic the expert's behavior. For example, a robot arm is asked to follow a desired trajectory and penalized for deviations from it [9]. Unfortunately, some applications cannot be solved by blindly following the expert, where the intention of the expert is the objective. Thus an IRL learning based on sample trajectories was proposed in [10].

Similar to IRL, the estimation of expert's feature expectations $\mu(\pi^E)$ is required. Specifically, it is estimated from multiple demonstrations by the expert, which can be done by simply averaging the estimations from m demonstrations:

$$\hat{\mu}(\pi^E)=\frac{1}{m}\sum_{i=1}^{m}\mu(\pi^{(i)}) \tag{18}$$

where $\mu(\pi^{(i)})$ is the observed return from demonstration i.

The apprenticeship learning algorithm of finding a reward function with sample trajectories goes as follows:

1) Randomly generalize a policy π_0, and i starts from 1.
2) Perform the following optimization:

$$\begin{aligned} &\max\{t_i=\min_{j\in\{0,1,\ldots i-1\}}\alpha(\mu(\pi^E)-\mu(\pi^j))\} \\ &s.t. \quad \|\alpha\|_2=1 \end{aligned} \tag{19}$$

   where π^j is one of the policy from the generated policy reservoir. This optimization is consistent with previous IRL algorithms that the reward function should greatly distinguish the best and second best policy.
3) If $t_j\leq\varepsilon$, then terminate, where ε is a predefined threshold.
4) Find a new policy π_j that maximize V^{π_j} under the new reward function $R=\alpha\phi(s)$ using RL algorithms, and add policy π_j into the policy reservoir.
5) Set $i=i+1$ and go to step 2.

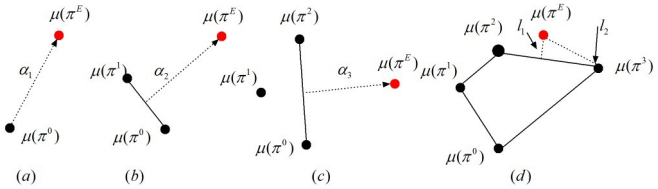

Fig. 1. Geometric description of policy iteration: The algorithm terminates when $l_1<\varepsilon$. l_2 is the distance between the best and second best policy.

Eq. (19) is a little different from Eq. (16) since $\|\alpha\|_2$ is used instead of $\|\alpha\|_1$, and this is valid because $\|\alpha\|_2\leq\|\alpha\|_1$. This simple tweak enable the optimization problem in Eq. (19) to be solved by support vector machines (SVM), and α can be viewed as the unit vector orthogonal to the maximum margin that separates $\mu(\pi^E)$ and $\{\mu(\pi^j)|j\in(0,1,\ldots,i-1)\}$. The geometric description of the policy iteration is shown in Figure 1.

Upon termination, the algorithm returns a policy reservoir: $\{\pi_i : i=0,1,\ldots n\}$, where $(n+1)$ is the total number of generated policies. Then the near optimal policy can be either selected manually from the policy reservoir or solved using a linear combination of generated policies [10]:

$$\begin{aligned} &\min\left\|\mu(\pi^E)-\mu\right\| \\ &s.t. \quad \mu=\sum_{i=0}^{n}\lambda_i\mu(\pi^i),\ \lambda_i\geq 0,\ \sum_{i=0}^{n}\lambda_i=1 \end{aligned} \tag{20}$$

As shown in Figure 1, this algorithm does not try to recover the true reward function ($l_2\geq l_1$) and returns a mixed policies that performs approximately as good as the expert's demonstration. Apparently, if the expert's behavior is deterministic, the mixed policy can be inaccurate because of its stochastic nature. An application of parking lot navigation has been demonstrated using the above algorithm [16].

D. Applications Using Original IRL Algorithms

The most notable series of applications in IRL field is the autonomous helicopter aerobatic demonstrations carried out by Stanford University. They all used apprenticeship IRL theory discussed before to find the trade-off among features of the reward function through expert's demonstrations. In [17], a set of helicopter aerobatic maneuvers were demonstrated, including flip, roll, tail-in funnel and nose-in funnel. By learning a desired trajectory from a number of sub-optimal demonstrations, the helicopter performance has been greatly increased, and some new aerobatic maneuvers were performed [18]. Autorotation, a challenging emergency helicopter landing procedure performed during an engine failure, were also successfully demonstrated [19]. Furthermore, a controller was designed that enables the helicopter to perform chaos, considered as the most challenging helicopter aerobatic maneuver, by observing the expert's demonstrations of in-place flips rather than observing chaos [15].

Besides the applications of helicopter aerobatic maneuvers, original IRL has also been used in other areas. In [10], a car

driving simulator that can learn different driving styles of the demonstrator was presented . Using similar ideas, a navigation controller was developed that enables a real size robotic car to learn different parking styles [16]. Original IRL algorithm [13] has also been used to predict pedestrian intentions in a robot application, where the robot can imitate pedestrian behaviors [20]. Most recently, a user simulation database for dialogue systems was built using the same algorithm [21].

III. Refinements on the Original IRL Algorithms

Although original IRL algorithms can somewhat achieve their objectives and their effectiveness has been proved by some applications, there are still assumptions and constraints that make them incapable to be applied to a broader range of applications. Even the authors in [17], who introduced the original apprenticeship IRL, pointed out that the reward functions derived from IRL are often unsafe to fly the helicopter, and a fine tuning process by hand is needed. The reasons behind this fact are not explained, but generally there are several assumptions and problems that make the original IRL not suitable for practical applications:

- The reward function is considered to be a linear combination of features, which can be wrong when the expert acts according to a reward function with other forms.
- Original IRL algorithms assume the expert's demonstrations are optimal, which is usually not true in practice when the demonstrations are noisy and imperfect.
- The IRL problem is ill-posed
- The policy derived from the apprenticeship IRL is stochastic, which may not be a good choice if the expert's policy is deterministic.
- The demonstrations can be not representative enough, and the algorithm should be able to generalize demonstrations to uncovered areas.
- IRL is solved in an MDP setting, which can be impractical since usually the agent cannot access the true global state of the environment. A generalization to a POMDP scenario is prefered since it is closer to reality.
- The computational expense is heavy since IRL usually requires iteratively solving RL problems with each new reward function provided.

Inspired by the idea of IRL and its successful applications, as well as the bottlenecks existed in the original algorithms, many researchers participate into the further refinements of IRL. They either tackle the IRL from a new perspective, or focus on solving one or more problems mentioned above. In this section, some key IRL variants are first discussed and it follows with a short introduction to other improvements.

A. Maximum Margin Planning

Maximum margin planning (MMP) [22] uses similar ideas as the original IRL algorithm [13], where the solver attempts to make the provided demonstrations look better than any other solutions by a margin, through a quadratic programming formulation. MMP solves the ill-posed problem by the introduction of the loss-functions, which can have different forms and usually are used to penalize choosing actions that are different from expert's at a certain state or arriving in states that the expert chooses not to enter. The difference between MMP and the original IRL is that the margin scales with these loss-functions in MMP. Furthermore, MMP is agnostic about the underling MDP and thus the demonstrations with different start and goal states can be used, since only the expected rewards are compared. MMP seeks to reproduce the expert's behaviors instead of returning a mixed of policies, another advantage compared to the apprenticeship IRL [10].

Since MMP still assumes the reward function to have a linear form, it was then extended to learn non-linear reward functions and a LEARCH (LEArning and seaRCH) algorithm was introduced [23]. The LEARCH algorithm was implemented in a autonomous navigation problem where a vehicle was operated in a complex unstructured terrain [24]. Using a similar approach, a visual navigation system was designed that automatically assigns costs to different detected objects and derives a most suitable path under the current situation [25].

B. Bayesian Inverse Reinforcement Learning

As stated before, the IRL problem is ill-posed since there is uncertainty existed in the obtained reward function. Therefore, it is natural to use probability distribution to model this uncertainty. Bayesian IRL approaches the IRL problem from this perspective, and treats the demonstration sequences as the evidence and calculates a prior on the reward function. The basic idea of Bayesian IRL is given below [26].

Consider one of expert's demonstrations $O_\chi = \{(s_1, a_1), (s_2, a_2) \dots (s_k, a_k)\}$, meaning this episode lasts k time steps and the expert chooses action a_i at state s_i. Bayesian IRL assumes the expert is acting greedily, i.e., always choosing actions with the highest Q-value, thus the resulting policy is stationary. The probability of observing this demonstration sequence is given by:

$$P_\chi(O_\chi|R) = P_\chi((s_1, a_1)|R) \dots P_\chi((s_k, a_k)|R) \tag{21}$$

Since the expert executes a greedy policy, the higher the $Q^*(s, a)$[1] is, the more likely that this state action pair is to be selected. Thus the likelihood of selecting (s_i, a_i) can be given as a potential function with $Q^*(s_i, a_i)$:

$$P_\chi((s_i, a_i)|R) = \frac{1}{Z_i} e^{\alpha_\chi Q^*(s_i, a_i, R)} \tag{22}$$

where α_χ represents the confidence we have in the expert's ability to choose actions with high value, and Z_i is a normalizing constant. Consequently, the likelihood of the entire episode is given by:

$$P_\chi(O_\chi|R) = \frac{1}{Z} e^{\alpha_\chi E(O_\chi, R)} \tag{23}$$

where $E(O_\chi, R) = \sum_i Q^*(s_i, a_i, R)$ represents the summation of Q-values of the entire episode. Then according to Bayes

[1] $Q^*(s, a)$ is retrieved from the estimated reward function R using RL algorithms

theorem, the posterior probability of the reward function R is given by:

$$P_\chi(R|O_\chi) = \frac{P_\chi(O_\chi|R)P_R(R)}{P(O_\chi)} = \frac{1}{Z \cdot P(O_\chi)} e^{\alpha_\chi E(O_\chi,R)} P_R(R) \quad (24)$$

The normalizing factor $Z \cdot P(O_\chi)$ can be solved through sampling algorithms. $P_R(R)$ can be considered as independently identically distributed if we are agnostic about the reward distribution. Otherwise, prior information can be implemented through $P_R(R)$ [26].

Using similar Bayesian framework, it was generalized to a preference elicitation formulation [27], in which the goal is to determine the posterior distribution on the expert's preferences. By correctly identifying the expert's preferences, the derived policies can even surpass the suboptimal demonstrations provided by the expert. Then a Gaussian prior is assigned to the reward function and the assumption of its linear form is removed [28]. It has been shown that IRL via Gaussian process can deal with noisy observations, incomplete policies, and small number of observations. Furthermore, the Bayesian IRL was extended to a multitask setting [29]. Specifically, the algorithm derives preferences from a series of tasks demonstrations using a hierarchical [30] and empirical Bayesian [31] approaches that assign multitask priors on reward functions and policies.

C. Maximum Entropy Inverse Reinforcement Learning

Similar to Bayesian IRL [26], maximum entropy IRL uses a probability approach to resolve the ill-posed problem of the original IRL. Specifically, the principle of *maximum entropy*, which gives the least biased estimate based on the given information [32], is implemented. The probability of the observed expert trajectory O_χ is also weighted by the estimated rewards like Bayesian IRL, and policies with higher rewards are exponentially more preferred:

$$P_\chi(O_\chi|\theta) = \frac{1}{Z} e^{\alpha_\chi E(O_\chi,\theta)} \quad (25)$$

where θ is the parameter vector of the reward function R, and R is considered as a linear combination features as in Eq. (11). Then the optimal value of θ is given by maximizing the likelihood of the observed trajectory through maximum entropy [33]:

$$\theta^* = \arg\max_\theta L(\theta) = \arg\max_\theta \sum_{examples} \log P_\chi(O_\chi|\theta) \quad (26)$$

Eq. (26) is convex for deterministic MDP and can be solved through gradient-based methods. Previous IRL methods focus on actions, where the trajectories after the actions are compared, instead of the trajectories before[2], and this causes the problem of label bias [34]. As a consequence, policy with the highest reward may not be the one with the highest probability [33]. Maximum entropy IRL avoids this problem by focusing on the distribution over trajectories rather than actions (see Eq. (26)).

The above algorithm was applied to a set of GPS data collected from taxi drivers, and recover a reward function that can be used for route recommendation and predicting driver's behavior [33]. Then the algorithm was extended to a POMDP setting [35]. Specifically, a Gaussian process was used to integrate robot's local sensing ability with priors of the environment and produced a joint distribution of the expected environment. Since the objective is to enable the robot to navigate through crowded environments, planned trajectories has to be revised according to the change of surrounding people, and thus the environment is updated every time step and a new route is planned every $H(H > 1)$ steps. Furthermore, the maximum entropy framework was also introduced to a model-free setting [36]. Since the algorithm derives θ^* by minimizing the relative entropy (KL divergence) between the demonstrated policy and derived policy, and this divergence can be empirically estimated without an MDP model.

D. Other Developments in Inverse Reinforcement Learning

Apart from the major IRL variants discussed above, there are still many other new developments focussing on solving the problems mentioned at the beginning of this section and improving various aspects of the original IRL.

1) Gradient Methods: The policy of the agent is derived according to the reward function through RL algorithms, thus a slight change in the parameter space of the reward function can also cause a change in the policy space. Then it is natural to consider gradient methods to solve θ^*. In [37] the idea of SL was used, where the deviations from the expert's trajectory were penalized. However, instead of directly tuning the policy, it was done by tuning the reward function through *natural gradient* [38]. Using a gradient ascent approach, IRL was also extended to a multiple intention setting [39].

2) Active Learning: For many cases, the demonstrated trajectories by the expert cannot cover the whole learning area, or the provided examples are incomplete. Instead of supplying more examples, it is better that the agent can actively select the most appropriate scenarios to ask for guidance from the expert, and this is the idea of *active learning*. It can be considered as a semi-supervised learning approach where the agent can query the expert when the decision uncertainty rises to a certain level. Through this, the number of required demonstrations can be reduced. A full Bayesian approach is also used to select the most potentially useful state to ask for the expert's support [14]. Similarly, the incomplete knowledge about the MDP was modeled using Bayesian and made the agent query the expert whenever it chooses in an online manner [40].

3) Nonlinear Reward Function: The original IRL algorithms and its many variants assume the reward function to have a linear combination of features. This assumption usually is not reasonable for practical problems, since it has been shown that the quality of the learned policies can be greatly jeopardized by the error of value estimation [41]. The LEARCH algorithm was used to learn a nonlinear reward func-

[2]This is because of the definition of value function, see Eq. (3.)

tion [23], and the reward function was treated as a Gaussian process with no assumptions about its format [28].

4) Computational Burden: The computational expense on the original IRL can be immense for large domain problems. To evaluate the performance of the revised reward function, especially when the comparison is among policies, it is done by solving a RL problem with respect to this reward function. Therefore, IRL typically needs to solve one RL problem per iteration, and the computational burden with this can be considered as one of the major bottlenecks of existing IRL algorithms, which prevents them from being implemented efficiently. MDP induced metrics [35] and PI^2 (Policy Improvement with Path Integrals) algorithm [42] are used to avoid this computational burden since they do not require iteratively solving different MDPs.

5) Expert with Imperfect/Incomplete Demonstrations: IRL is an approach that infers preferences from demonstrations, which means its performance heavily relies on the quality of these expert's demonstrations. In practical applications, the demonstrations can be suboptimal, incomplete or even total failures. Therefore it is essential to take into account the expert performance factors when building the IRL algorithms. There are times when the expert has difficulties of demonstrating the entire trajectories, thus an evaluator, which can distinguish between two policies, but has difficulties in giving direct instructions, was used [43]. For normal cases, the performance of the agent is bounded by the performance of the expert, and this is not a favorable case if the expert is not good enough. In [44], this problem was approached from a game-theoretic view, which may produce a policy that exceeds the performance of the expert. Alternatively, a hierarchical method can be implemented if the direct demonstration is difficult. Using this idea, the control of a quadruped robot was divided into two levels and the expert can give instructions more easily [45]. Similarly, MDP induced metrics was used to focus on problems when the demonstrations of the expert is imperfect or incomplete [35]. For most IRL problems, the trajectories provided by the expert are considered as the correct behaviors. However, the failed attempts can also have meaningful information that might help the agent to learn, and the agent's performance can be improved by deliberately avoiding them [46].

6) Other IRL Improvements: In practical applications, the agent usually has limited access to the true state of the environment. Therefore the IRL problems has to be extended to a POMDP setting to deal with realistic problems [4] [47]. For certain scenarios, it is preferred to influence the expert instead of taking instructions. Such as an online shopping website, it is better to provide an amiable interface so that the customers can spend more. To tackle these situations, the idea of active indirect preference elicitation was used to learn the reward function from the expert's reactions in response to incentives [48]. The original apprenticeship IRL focuses on deriving the approximate optimal policy rather than the true reward function that generates it, therefore a convex apprenticeship learning algorithm was proposed, whose target is the true approximate reward function $\min_{j\in(0,1,\ldots,i-1)} \left\|\alpha(\mu(\pi^E) - \mu(\pi^j))\right\|$, i.e., to minimize l_2 instead of l_1 [49] (see Figure 1). In this way, a deterministic policy can be derived instead of a suboptimal mixed policy which can be unsuitable for deterministic problems. Usually, the focus of the reward function is to derive the parameters of the corresponding features. However, the feature selection process itself can be of great difficulty when building a concise, meaningful reward function. This can be solved by using dimension reduction methods such as PCA (Principle Component Analysis) to extract useful features from demonstrations [50]. In the face of insufficient number of expert's demonstrations, a method of learning parameterized version of demonstrations was introduced, where new trajectories can be generated by tuning the parameters [51]. In this way, a large number of trajectories can be generated without the need for more expert demonstrations. In some practical problems, the goal state is what matters instead of the whole trajectories. Therefore, it is a waste of effort or meaningless to blindly follow the expert's whole trajectories. In [52], a simplified reward structure was introduced , where the final state of the demonstration is the target but the agent is left free to choose the approaches to achieve the target.

IV. Conclusions

This paper introduces IRL algorithms, which is a new branch of RL that originated from last decade. The objective of IRL is to derive a reward function, the most succinct representation of the expert's intention, from a group of expert's demonstrations. This reward function is usually engineered by hand in RL algorithms, and proved to be of great difficulty in complex problems [10]. We first presents the origin and problem formulation of IRL to give readers a clear understanding of its importance. Then it follows the introduction of the original IRL algorithms and their applications. We believe they are of great importance because the series of aerobatic helicopter applications using original IRL are well known and their success is arguably the reason why IRL becomes popular. Since IRL can be approached from different perspectives, some major variants of IRL algorithms are also discussed, including MMP, Bayesian theory, maximum entropy, and gradient methods. Finally, algorithms focusing on other improvements of the original IRL are also briefly introduced.

The perspectives about IRL have changed a lot from its first introduction, and many improvements enable IRL to be implemented in more practical and complex applications. Among the aspects discussed in Section III, we believe that there are several factors that future IRL algorithms should pay special attention to. First of all, the computational requirement is a severe bottleneck of existing IRL algorithms. So new ones should avoid iteratively solving RL problems. Secondly, there should be no assumption about the form of the reward function, since this assumed form maybe quite different from the one expert is using. Last but not least, IRL algorithms should be able to be applied to realistic scenarios, such as POMDP and continuous settings. The idea of IRL in control fields is still rather new, and we believe the promise of *understanding*

people's intention broadly expands the learning capabilities of machines and may open a new era for artificial intelligence.

REFERENCES

[1] K. Kaelbling, M. Littman, and A. Moore, "Reinforcement learning: A survey," *Journal of Artificial Intelligence Research*, pp. 237–285, 1996.

[2] R. Sutton and A. Barto, *Introduction to reinforcement learning*, 1998.

[3] M. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley Sons, Inc., 1994.

[4] J. Choi and K. Kim, "Inverse reinforcement learning in partially observable environments," *Proceedings of the 21st International Joint Conference on Artifical Intelligence (IJCAI)*, pp. 1028–1033, 2009.

[5] E. Sondik, "The optimal control of partially observable markov processes." 1971.

[6] K. Murphy, "A survey of pomdp solution techniques," *Environment*, vol. 2, p. X3, 2000.

[7] D. Pomerleau, "Efficient training of artificial neural networks for autonomous navigation," *Neural Computation*, vol. 3, pp. 88–97, 1991.

[8] G. Grudic and P. Lawrence, "Human-to-robot skill transfer using the spore approximation," *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 4, pp. 2962–2967, 1996.

[9] C. Atkeson and S. Schaal, "Robot learning from demonstration," *Machine Learning International Conference*, pp. 12–20, 1997.

[10] P. Abbeel and A. Ng, "Apprenticeship learning via inverse reinforcement learning," *Proceedings of the 21st international conference on Machine learning*, p. 1, 2004.

[11] S. Russell, "Learning agents for uncertain environments (extended abstract)," *Proceedings of the 11st annual conference on Computational learning theory*, pp. 101–103, 1998.

[12] R. Keeney and H. Raiffa, *Decisions with multiple objectives: Preferences and value tradeoffs*. Cambridge University Press, 1993.

[13] A. Ng and S. Russell, "Algorithms for inverse reinforcement learning," *Proceedings of the 17th International Conference on Machine Learning*, pp. 663–670, 2000.

[14] M. Lopes, F. Melo, and L. Montesano, "Active learning for reward estimation in inverse reinforcement learning," *Machine Learning and Knowledge Discovery in Databases*, pp. 31–46, 2009.

[15] P. Abbeel, A. Coates, and A. Ng, "Autonomous helicopter aerobatics through apprenticeship learning," *International Journal of Robotics Research*, vol. 29, no. 13, pp. 1608–1639, 2010.

[16] P. Abbeel, D. Dolgov, A. Ng, and S. Thrun, "Apprenticeship learning for motion planning with application to parking lot navigation," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1083–1090, 2008.

[17] P. Abbeel, A. Coates, Quigle, and A. Ng, "An application of reinforcement learning to aerobatic helicopter flight," *Proceedings of Advances in Neural Information Processing Systems*, p. 1, 2007.

[18] A. Coates, P. Abbeel, and A. Ng, "Learning for control from multiple demonstrations," *Proceedings of the 25th international conference on Machine learning*, pp. 144–151, 2008.

[19] P. Abbeel, A. Coates, T. Hunter, and A. Ng, "Autonomous autorotation of an rc helicopter," *Experimental Robotics*, pp. 385–394, 2009.

[20] S. Chung and H. Huang, "A mobile robot that understands pedestrian spatial behaviors," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5861–5866, 2010.

[21] S. Chandramohan, M. Geist, F. Lefevre, and O. Pietquin, "User simulation in dialogue systems using inverse reinforcement learning," *Proceedings of the 12th Annual Conference of the International Speech Communication Association*, 2011.

[22] N. Ratliff, J. Bagnell, and M. Zinkevich, "Maximum margin planning," *Proceedings of the 23rd international conference on Machine learning*, pp. 729–736, 2006.

[23] N. Ratliff, D. Silver, and J. Bagnell, "Learning to search: Functional gradient techniques for imitation learning," *Autonomous Robots*, vol. 27, no. 1, pp. 25–53, 2009.

[24] D. Silver, J. Bagnell, and A. Stentz, "Learning from demonstration for autonomous navigation in complex unstructured terrain," *The International Journal of Robotics Research*, vol. 29, no. 12, p. 1565, 2010.

[25] ——, "Perceptual interpretation for autonomous navigation through dynamic imitation learning," *Robotics Research*, pp. 433–449, 2011.

[26] D. Ramachandran and E. Amir, "Bayesian inverse reinforcement learning," *Proceedings of 20th International Joint Conference of Artificial Intelligence*, 2007.

[27] C. Rothkopf and C. Dimitrakakis, "Preference elicitation and inverse reinforcement learning," *Machine Learning and Knowledge Discovery in Databases, Lecture Notes in Computer Science*, 2011.

[28] Q. Qiao and P. Beling, "Inverse reinforcement learning with gaussian process," *Proceedings of American Control Conference (ACC)*, pp. 113–118, 2011.

[29] C. Dimitrakakis and C. Rothkopf, "Bayesian multitask inverse reinforcement learning," *Arxiv preprint arXiv:1106.3655*, 2011.

[30] T. Heskes, "Solving a huge number of similar tasks: a combination of multi-task learning and a hierarchical bayesian approach," *Proceedings of the 15th International Conference on Machine Learning*, pp. 233–241, 1998.

[31] H. Robbins, "An empirical bayes approach to statistics," *Breakthroughs in Statistics: Foundations and basic theory*, vol. 1, p. 388, 1992.

[32] E. Jaynes, "Information theory and statistical mechanics. ii," *Physical review*, vol. 108, no. 2, p. 171, 1957.

[33] B. Ziebart, A. Maas, J. Bagnell, and A. Dey, "Maximum entropy inverse reinforcement learning," *Proceedings of 23rd AAAI Conference on Artificial Intelligence*, vol. 1433, p. 1438, 2008.

[34] J. Lafferty, A. McCallum, and F. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*, pp. 282–289, 2001.

[35] F. Melo and M. Lopes, "Learning from demonstration using mdp induced metrics," *Machine Learning and Knowledge Discovery in Databases*, pp. 385–401, 2010.

[36] A. Boularias, J. Kober, and J. Peters, "Relative entropy inverse reinforcement learning," *Proceedings of 21st International Conference on Automated Planning and Scheduling*, vol. 15, pp. 20–27, 2011.

[37] G. Neu and C. Szepesvri, "Apprenticeship learning using inverse reinforcement learning and gradient methods," *Proc. UAI*, pp. 295–302, 2007.

[38] S. Amari, "Natural gradient works efficiently in learning," *Neural computation*, vol. 10, no. 2, pp. 251–276, 1998.

[39] M. Babes, V. Marivate, M. Littman, and K. Subramanian, "Apprenticeship learning about multiple intentions," 2010.

[40] R. Cohn, M. Maxim, E. Durfee, and S. Singh, "Selecting operator queries using expected myopic gain," *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, pp. 40–47, 2010.

[41] A. Boularias and B. Chaib-Draa, "Bootstrapping apprenticeship learning," *Proceedings of ICAPS Multiagent Planning Workshop*, vol. 15, pp. 20–27, 2011.

[42] M. Kalakrishnan, E. Theodorou, and S. Schaal, "Inverse reinforcement learning with pi 2," *The Snowbird Workshop*, 2010.

[43] V. Silva, A. Costa, and P. Lima, "Inverse reinforcement learning with evaluation," *IEEE International Conference on Robotics and Automation*, pp. 4246–4251, 2006.

[44] U. Syed and R. Schapire, "A game-theoretic approach to apprenticeship learning," *Advances in neural information processing systems*, vol. 20, pp. 1449–1456, 2008.

[45] J. Kolter, P. Abbeel, and A. Ng, "Hierarchical apprenticeship learning with application to quadruped locomotion," *Advances in Neural Information Processing Systems*, vol. 20, 2008.

[46] D. Grollman and A. Billard, "Donut as i do: Learning from failed demonstrations," *International Conference on Robotics and Automation, Shanghai*, 2011.

[47] P. Henry, C. Vollmer, B. Ferris, and D. Fox, "Learning to navigate through crowded environments," *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 981–986, 2010.

[48] H. Zhang and D. Parkes, "Enabling environment design via active indirect elicitation," *Proceedings of Workshop on Preference Handling*, 2008.

[49] S. Lee and Z. Popovi, "Learning behavior styles with inverse reinforcement learning," *ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH*, pp. 1–7, 2010.

[50] S.-Y. Chen, H. Qian, J. Fan, Z.-J. Jin, and M.-L. Zhu, "Modified reward function on abstract features in inverse reinforcement learning," *JOURNAL OF ZHEJIANG UNIVERSITY - SCIENCE C*, vol. 11, no. 9, 2010.

[51] J. Tang, A. Singh, N. Goehausen, and P. Abbeel, "Parameterized maneuver learning for autonomous helicopter flight," *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1142–1148, 2010.

[52] M. Mason and M. Lopes, "Robot self-initiative and personalization by learning through repeated interactions," *Proceedings of the 6th international conference on Human-robot interaction*, pp. 433–440, 2011.