The current issue and full text archive of this journal is available at
www.emeraldinsight.com/1756-378X.htm

A survey of inverse reinforcement learning techniques

Shao Zhifei and Er Meng Joo
School of Electrical and Electronics Engineering, Nanyang Technological University, Singapore

Received 6 August 2011
Revised 22 October 2011, 11 December 2011
Accepted 27 February 2012

Abstract

Purpose – This purpose of this paper is to provide an overview of the theoretical background and applications of inverse reinforcement learning (IRL).

Design/methodology/approach – Reinforcement learning (RL) techniques provide a powerful solution for sequential decision making problems under uncertainty. RL uses an agent equipped with a reward function to find a policy through interactions with a dynamic environment. However, one major 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 IRL, an extension of RL, which directly tackles this problem by learning the reward function through expert demonstrations. In this paper, the original IRL algorithms and its close variants, as well as their recent advances are reviewed and compared.

Findings – This paper can serve as an introduction guide of fundamental theory and developments, as well as the applications of IRL.

Originality/value – This paper surveys the theories and applications of IRL, which is the latest development of RL and has not been done so far.

Keywords Inverse reinforcement learning, Reward function, Reinforcement learning, Artificial intelligence, Learning methods

Paper type General review

1. Introduction

Reinforcement learning (RL) techniques solve problems through an agent, which acquires experiences 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 (Kaelbling *et al.*, 1996).

Unlike supervised learning (SL), RL does not 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 (Sutton and Barto, 1998). 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.

IRL can be considered as a branch of Learning from Demonstration (LfD) (Argall *et al.*, 2009) or imitation learning (Schaal, 1999), where a policy is learned through examples, and the objective of the agent is to reproduce the demonstrated behavior.

International Journal of Intelligent Computing and Cybernetics
Vol. 5 No. 3, 2012
pp. 293-311

1756-378X
DOI 10.1108/17563781211255862

Generally, the methods of deriving a policy can be divided into two categories: one is through function mapping as in SL, which normally is to minimize the deviation between demonstrated and derived policies. The other one is to find an optimal policy according to a reward function as in RL. As stated before, SL has the disadvantage of requiring target labels and deriving the reward function in RL can be non-trivial in complex tasks. To overcome these problems, two major methods have been proposed: one is to build movements from a small set of motor primitives (MPs), which can generate either discrete or rhythmic movement. This approach first learns MPs with a relatively small set of parameters, and this can be done using various LfD methods (including RL), therefore the problem complexity can be reduced. The major difficulty is to combine these MPs into full movements and some practical methods have been developed (Kober and Peters, 2010). The other method is IRL, which directly targets at deriving the reward function through 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 2 is devoted to the introduction of original IRL algorithms. Section 3 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.

1.1 MDP and POMDP

The original RL algorithms assume the problems to be solved satisfy Markov decision process (MDP), which is a form of tuple $(S, A, P^a_{ss'}, \gamma, R)$ (Puterman, 1994):

- *S:* a set of possible states that represent the dynamic environment. The agent finds itself in a particular state at each time step.
- *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 is usually provided by the user or derived from experiences, and 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:

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

The assumption of MDP in RL provides a good theoretical foundation for developing algorithms. However, this assumption natively allows the 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 (Choi and Kim, 2009). To address this problem,

an extension of MDP called "Partially Observable Markov Decision Processes" (POMDPs) (Sondik, 1971) 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:

(1) Z is a finite set of observations.

(2) 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'.

(3) b_0 is a belief function, and $b_0(s)$ gives the probability of starting in state s.

In fact, the underlying mechanism of POMDP is MDP and the effect of the actions over the environment is as same as MDP. The difference is that the true states are unavailable to the agent and can only be estimated through the hints provided by the observations. So 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. Therefore, a belief state B in POMDP can be considered as the state S in MDP. For simplicity, the preliminary theories of RL will be addressed under the MDP formulation. Regarding to the specific RL solutions under POMDP, readers can refer to the survey by Murphy (2000).

1.2 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 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 was introduced to estimate the accumulated rewards starting from a state:

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

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 equation (3) becomes a Q-value function. Q^{π}:$S \times A \rightarrow \mathbb{R}$ 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:

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

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

Equations (6) and (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.

1.3 The 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 the expert's demonstrations. For example, regression techniques have been successfully applied to autonomous navigation (Pomerleau, 1991) and human-to-robot skill transfer (Grudic and Lawrence, 1996). However, these appraoches 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 senarios, but the robot may be required to do a slightly different task (Atkeson and Schaal, 1997). Generally speaking, the strategy of SL is to simply 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 given reward function can be wrong and the parameters of a multi-attribute reward function are hard to be determined a priori (Abbeel and Ng, 2004). Therefore, the problem of IRL is first formulated as follows (Russell, 1998).

(1) Given:
   - measurements of an agent's behavior over time under various circumstances;
   - sensory inputs to that agent if needed; and
   - a model of the environment if available.

(2) 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 (Keeney and Raiffa, 1993). In the machine learning community, IRL problems are first formally studied by Ng and Russell (2000), and described as follows:

(1) Given:
- a finite state space S;
- a set of actions $A = \{a_1, a_2, \ldots a_k\}$;
- transition probability $P^a_{ss'}$;
- a discount factor γ, and
- a policy π.

(2) Determine a set of possible reward functions R such that π is the optimal policy for the given MDP $\left(S, A, \left\{P^a_{ss'}\right\}, \gamma, R\right)$.

In summary, IRL can be considered as a reverse procedure of RL problems. The assumption is that the expert's demonstration is an optimal policy π^*, which is 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 (Abbeel and Ng, 2004), 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, and the derived reward function from IRL can be used to generate approximate optimal policies.

2. Original IRL algorithms

The original IRL algorithms were introduced by Ng and Russell (2000) and Abbeel and Ng (2004). 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:

(1) Finite-state MDP with known optimal policy.

(2) Infinite-state MDP with known optimal policy.

(3) 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 Ng and Russell (2000).

2.1 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 $\pi(s) \equiv a_1$, and the bold letters represent vectors. Then the given policy is optimal if and only if the reward function $\mathbf{R}$ satisfies:

$$(\mathbf{P}_{a1} - \mathbf{P}_a)(\mathbf{I} - \gamma \mathbf{P}_{a1})^{-1}\mathbf{R} \geq 0 \tag{8}$$

where $\mathbf{P}_{a1}$ is the transition probability matrix of taking the optimal action a_1, and $\mathbf{P}_a$ is the transition probability of other policies with $a \in A/a_1$. The detailed proof can be found in Ng and Russell (2000).

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 (Lopes *et al.*, 2009). For instance, $\mathbf{R} = 0$ 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 $\mathbf{R}$ from the other solutions:

(1) Maximize the difference between the best and second best solutions.

(2) Simple solutions are preferred.

The first criterion specifies a reward function that makes the expert policy clearly stands out among other policies:

$$\max\left\{\sum_{i=1}^{N}\min_{a\in A/a_1}\{(\mathbf{P}_{a_1}(i)-\mathbf{P}_a(i))(I-\gamma\mathbf{P}_{a_1}(i))^{-1}\mathbf{R}\}\right\} \tag{9}$$

where $\min_{a\in A/a_1}\{(\mathbf{P}_{a_1}(i)-\mathbf{P}_a(i))(I-\gamma\mathbf{P}_{a_1}(i))^{-1}\mathbf{R}$ 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 $\lambda\|\mathbf{R}\|_1$ 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 (Ng and Russell, 2000):

$$\begin{aligned}&\max\left\{\sum_{i=1}^{N}\min_{a\in A/a_1}\{(\mathbf{P}_{a_1}(i)-\mathbf{P}_a(i))(I-\gamma\mathbf{P}_{a_1})^{-1}\mathbf{P}\}-\lambda\|\mathbf{R}\|_1\right\}\\ &s.t.\quad(\mathbf{P}_{a_1}-\mathbf{P}_a)(I-\gamma\mathbf{P}_{a_1})^{-1}\mathbf{R}\geq 0\quad\text{and}\quad|\mathbf{R}_i|\leq R_{\max}\end{aligned} \tag{10}$$

2.2 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 all useful features:

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

where $\phi(s)=[\,\phi_1(s)\quad\phi_2(s)\quad\ldots\quad\phi_d(s)\,]^T$ are predefined basis functions, i.e. features. d is the number of useful features in the reward function, and $\alpha=[\,\alpha_1\quad\alpha_2\quad\ldots\quad\alpha_d\,]$ are the parameters to be tuned during the learning process.

If fact, equation (11) provides another way to represent the value function using feature expectations (Abbeel and Ng, 2004):

$$\begin{aligned}V(\pi)&=E_{s_0\sim D}[V^{\pi}(s_0)]=E\left[\sum_{t=0}^{\infty}\gamma^t R(s_t)|\pi\right]\\&=E\left[\sum_{t=0}^{\infty}\gamma^t\sum_{i=1}^{d}\alpha_i\phi_i(s_t)|\pi\right]\\&=\sum_{i=1}^{d}\alpha_i E\left[\sum_{t=0}^{\infty}\gamma^t\phi_i(s_t)|\pi\right]\end{aligned} \tag{12}$$

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:

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

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

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

where $\boldsymbol{\alpha}$ is a weight vector: $\boldsymbol{\alpha} = \begin{bmatrix} \alpha_1 & \alpha_2 & \cdots & \alpha_d \end{bmatrix}$, and $\boldsymbol{\mu}(\pi)$ is a feature vector: $\boldsymbol{\mu}(\pi) = [\, \mu_1(\pi) \quad \mu_2(\pi) \quad \cdots \quad \mu_d(\pi) \,]^T$.

Obviously, according to equation (6) and $\pi(s) \equiv a_1$ is the optimal policy, the following holds:

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

This means that given the appropriate reward function, the expert always performs better or equal to any other policy π^i. Since it is difficult to evaluate the infinite state space, a subsample of $S - S_0$, 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. So the optimization problem is relaxed with a penalty when the constraint in equation (15) is violated. In summary, the IRL problem under infinite-space is formulated as follows:

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

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

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

2.3 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 some 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. Furthermore, a good helicopter model is required (Abbeel *et al.*, 2010).

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 (Atkeson and Schaal, 1997). Unfortunately, some applications cannot be solved by blindly following the expert, where the intention of the expert is the objective. Thus, a new version of IRL learning based on sample trajectories was proposed by Abbeel and Ng (2004).

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

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

where $\boldsymbol{\mu}(\pi^i)$ 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:

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

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 $t_j \le \varepsilon$, 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).

Equation (19) is a little different from equation (16) since $\|\boldsymbol{\alpha}\|_2$ is used instead of $\|\boldsymbol{\alpha}\|_1$, and this is valid because $\|\boldsymbol{\alpha}\|_2 \le \|\boldsymbol{\alpha}\|_1$. This simple tweak enable the optimization problem in equation (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 $\boldsymbol{\mu}(\pi^E)$ and $\{\boldsymbol{\mu}(\pi^j)|\, j \in (0, 1, \ldots, i-1)\}$. The geometric description of the policy iteration is shown in Figure 1.

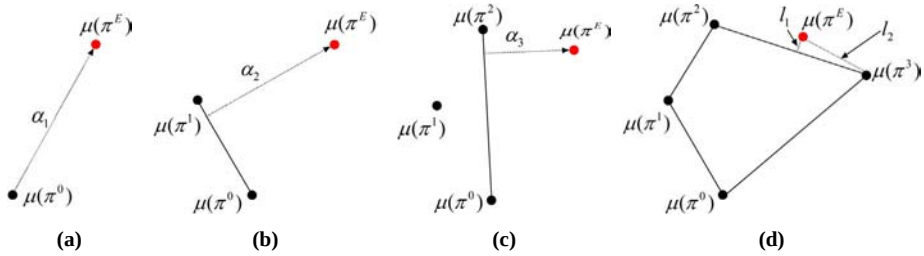


Figure 1.
Geometric description of policy iteration: the algorithm terminates when $l_1 < \varepsilon$. l_2 is the distance between the best and second best policy

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 (Abbeel and Ng, 2004):

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

As shown in Figure 1, this algorithm does not try to recover the true reward function ($l_2 \geq l_1$) and returned 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. Abbeel *et al.* (2008) has successfully demonstrated an application of parking lot navigation using the above algorithm.

2.4 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. Abbeel *et al.* (2007) demonstrated a set of helicopter aerobatic maneuvers 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 (Coates *et al.*, 2008). Autorotation, a challenging emergency helicopter landing procedure performed during an engine failure, were also successfully demonstrated (Abbeel *et al.*, 2009). Furthermore, Abbeel *et al.* (2010) designed a controller 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.

Besides the applications of helicopter aerobatic maneuvers, original IRL has also been used in other areas. Abbeel and Ng (2004) presented a car driving simulator that can learn different driving styles of the demonstrator. Using similar ideas, a navigation controller was developed that enables a real size robotic car to learn different parking styles (Abbeel *et al.*, 2008). Ng's algorithm (Ng and Russell, 2000) has also been used to predict pedestrian intentions in a robot application, where the robot can imitate pedestrian behaviors (Chung and Huang, 2010). Most recently, Chandramohan *et al.* (2011) built a user simulation database for dialogue systems using the original IRL algorithm.

3. 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 many assumptions and constraints that make them incapable to be applied to a broader range of applications. Even Abbeel *et al.* (2007), 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 by the authors, but generally there are several assumptions and problems that make the original IRL not suitable for practical applications:

- The reward function is usually assumed 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. The algorithms should be able to handle imperfect and noisy demonstrations.
- 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 number of demonstrations can be small and the demonstrations maybe incomplete. The algorithms 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 problems 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. Some key IRL variants are first discussed and it follows with a short introduction to other improvements.

3.1 Maximum margin planning

Maximum margin planning (MMP) (Ratliff *et al.*, 2006) uses similar ideas as the original IRL algorithm (Ng and Russell, 2000), where the solver attempts to make the provided demonstrations look better than any other solutions by a margin, through aquadratic 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 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, which is another advantage compared to the apprenticeship IRL (Abbeel and Ng, 2004).

Since MMP still assumes the reward function to have a linear form, it was then extended to learn non-linear reward functions and a LEArning and seaRCH (LEARCH) algorithm was introduced (Ratliff *et al.*, 2009). Then the LEARCH algorithm was implemented in an autonomous navigation problem where the vehicle was operated in a complex unstructured terrain (Silver *et al.*, 2010). Using a similar approach, Silver *et al.* (2011) designed a visual navigation system that automatically assigns costs to different detected objects and derives a path that is most suitable under the current situation.

3.2 Bayesian IRL

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 (Ramachandran and Amir, 2007).

Consider one of the expert's demonstrations $O_\chi = \{(s_1, a_1), (s_2, a_2) \ldots (s_k, a_k)\}$, which means 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, according to reward function R, thus the resulting policy is stationary. The probability of observing this demonstration sequence is given by:

$$P_\chi(O_\chi|R) = P_\chi((s_1, a_1)|R)P_\chi((s_2, a_2)|R)\ldots P_\chi((s_k, a_k)|R) \tag{21}$$

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

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

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 $P_R(R)$ (Ramachandran and Amir, 2007).

Using the similar Bayesian framework, Rothkopf and Dimitrakakis (2011) generalized it to a preference elicitation formulation (Friedman and Savage, 1952), 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 sub-optimal demonstrations provided by the expert according to his own preferences. Then a Gaussian prior is assigned to the reward function and the assumption of its linear form is removed (Qiao and Beling, 2011). It has been shown that IRL via Gaussian process can deal with noisy observations, incomplete policies, and small number of observations. Furthermore, Dimitrakakis and Rothkopf (2011) extended the Bayesian IRL to a multitask setting.

Specifically, the algorithm derives preferences from a series of tasks demonstrations using a hierarchical (Heskes, 1998) and empirical Bayesian (Robbins, 1992) approaches that assign multitask priors on reward functions and policies.

3.3 Maximum entropy IRL

Similar to the idea of Bayesian IRL (Ramachandran and Amir, 2007), maximum entropy IRL use a probability approach to resolve the ill-posed problem of the original IRL. Specifically, the principle of *maximum entropy*, which gives the least biasedestimate based on the given information (Jaynes, 1957), 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:

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

where θ is the parameter vector of the reward function R, and R is considered as a linear combination features as in equation (11).

Then the optimal value of θ is given by maximizing the likelihood of the observed trajectory through maximum entropy (Ziebart *et al.*, 2008):

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

Equation (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 (Lafferty *et al.*, 2011). As a consequence, policy with the highest reward may not be the one with the highest probability (Ziebart *et al.*, 2008). Maximum entropy IRL avoids this problem by focusing on the distribution over trajectories rather than actions (equation (26)).

Ziebart *et al.* (2008) applied the above algorithm 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. Then the algorithm was extended to a POMDP setting (Melo and Lopes, 2010). Specifically, a Gaussian process was used to integrate the 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, the planned trajectories has to be revised according to the change of surrounding people, thus the environment is updated every time step and a new route is planned every $H(H > 1)$ steps. Furthermore, Boularias *et al.* (2011) introduced the maximum entropy framework to a model-free setting. 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.

3.4 Other developments in IRL

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

3.4.1 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^*. Neu and Szepesvári (2007) used the idea of SL, 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* (Amari, 1998). Using a gradient ascent approach, IRL is also extended to a multiple intention setting (Babes *et al.*, 2010).

3.4.2 Active learning. For many cases, the demonstrated trajectories by the expert cannot cover the whole area that the agent wants to learn, or the provided examples are incomplete. Instead of blindly 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-SL 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 (Lopes *et al.*, 2009). Similarly, Cohn *et al.* (2010) modeled the incomplete knowledge about the MDP using Bayesian and made the agent query the expert whenever it chooses in an online manner.

3.4.3 Non-linear 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 (Boularias and Chaib-Draa, 2011). Ratliff *et al.* (2009) used the LEARCH algorithm to learn a non-linear reward function, and Qiao and Beling (2011) treated the reward function as a Gaussian process and make no assumptions about its format.

3.4.4 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. Melo and Lopes (2010) used MDP induced metrics and Kalakrishnan *et al.* (2010) used the PI^2 (Policy Improvement with Path Integrals) algorithm (Theodorou *et al.*, 2010) and avoided the computational burden since they do not require iteratively solving different MDPs.

3.4.5 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 Silva *et al.* (2006) used an evaluator, which can distinguish between two policies, but has difficulties in giving direct instructions. 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. Syed and Schapire (2008) approached this problem 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 (Kolter *et al.*, 2008). Similarly, Melo and Lopes (2010) focused on problems when the demonstrations of the expert is imperfect or incomplete using MDP induced metrics. 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 the expert's mistakes (Grollman and Billard, 2011).

3.4.6 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 (Choi and Kim, 2009; Henry *et al.*, 2010). For certain scenarios, it is preferred to influence the expert instead of taking instructions. Such as an online shopping web site, it is better to provide an amiable interface so that the customers can spend more. To tackle these situations, Zhang and Parkes (2008) used the idea of active indirect preference elicitation to learn the reward function from the expert's reactions in response to incentives. 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 $\min_{j\in(0,1,\ldots,i-1)}\|\boldsymbol{\alpha}(\boldsymbol{\mu}(\boldsymbol{\pi}^E) - \boldsymbol{\mu}(\boldsymbol{\pi}^j))\|$, i.e. to minimize l_2 instead of l_1 (Lee and Popovi, 2010) (Figure 1). In this way, a deterministic policy can be derived instead of a suboptimal mixed policy which can be unsuitable for deterministic problems. Normally, 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 Principle Component Analysis (PCA) to extract useful features from demonstrations (Shen-yi *et al.*, 2010). 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 (Tang *et al.*, 2010). 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. Mason and Lopes (2011) introduced a simplified reward structure, where the final state of the demonstration is the target but the agent is left free to choose the approaches to achieve the target.

4. Conclusions

This paper is a generalization of IRL algorithms, which is a new branch of RL and 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 (Abbeel and Ng, 2004). We first present the origin and problem formulation of IRL to give the readers a clear understanding of its importance. Then it follows with 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 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 3, 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. Second, there should be no assumption about the form of the reward function, since this assumption 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.

Notes

1. Q^* is retrieved from the estimated reward function R using RL algorithms.
2. This is because of the definition of value function, see equation (3).

References

Abbeel, P. and Ng, A. (2004), "Apprenticeship learning via inverse reinforcement learning", *Proceedings of the 21st International Conference on Machine Learning*, p. 1.

Abbeel, P., Coates, A. and Ng, A. (2010), "Autonomous helicopter aerobatics through apprenticeship learning", *International Journal of Robotics Research*, Vol. 29 No. 13, pp. 1608-39.

Abbeel, P., Coates, A., Hunter, T. and Ng, A. (2009), "Autonomous autorotation of an RC helicopter", *Proceedings of the International Symposium on Experimental Robotics*, pp. 385-94.

Abbeel, P., Coates, A., Quigley, M. and Ng, A. (2007), "An application of reinforcement learning to aerobatic helicopter flight", *Advances in Neural Information Processing Systems 19: Proceedings of the 2006 Conference*, p. 1.

Abbeel, P., Dolgov, D., Ng, A. and Thrun, S. (2008), "Apprenticeship learning for motion planning with application to parking lot navigation", *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pp. 1083-90.

Amari, S. (1998), "Natural gradient works efficiently in learning", *Neural Computation*, Vol. 10 No. 2, pp. 251-76.

Argall, B., Chernova, S., Veloso, M. and Browning, B. (2009), "A survey of robot learning from demonstration", *Robotics and Autonomous Systems*, Vol. 57 No. 5, pp. 469-83.

Atkeson, C. and Schaal, S. (1997), "Robot learning from demonstration", *Proceedings of the International Conference on Machine Learning* (*ICML'97*), Morgan Kaufmann, Burlington, MA, pp. 12-20.

Babes, M., Marivate, V., Littman, M. and Subramanian, K. (2010), "Apprenticeship learning about multiple intentions", *Proceedings of International Conference on Machine Learning* (*ICML 2011*).

Boularias, A. and Chaib-Draa, B. (2011), "Bootstrapping apprenticeship learning", *Proceedings of Neural Information Processing Systems, 2010*.

Boularias, A., Kober, J. and Peters, J. (2011), "Relative entropy inverse reinforcement learning", *Proceedings of Fourteenth International Conference on Artificial Intelligence and Statistics* (*AISTATS 2011*), *JMLR WC&P*, Vol. 15, pp. 182-9.

Chandramohan, S., Geist, M., Lefevre, F. and Pietquin, O. (2011), "User simulation in dialogue systems using inverse reinforcement learning", *Proceedings of the 12th Annual Conference of the International Speech Communication Association* (*Interspeech 2011*), *Florence* (*Italy*), *August*.

Choi, J. and Kim, K. (2009), "Inverse reinforcement learning in partially observable environments", *Proceedings of the 21st International Joint Conference on Artifical Intelligence* (*IJCAI*), pp. 1028-33.

Chung, S. and Huang, H. (2010), "A mobile robot that understands pedestrian spatial behaviors", *Intelligent Robots and Systems* (*IROS*), *2010 IEEE/RSJ International Conference on*, pp. 5861-6.

Coates, A., Abbeel, P. and Ng, A. (2008), "Learning for control from multiple demonstrations", *Proceedings of the 25th International Conference on Machine Learning*, pp. 144-51.

Cohn, R., Maxim, M., Durfee, E. and Singh, S. (2010), "Selecting operator queries using expected myopic gain", *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, pp. 40-7.

Dimitrakakis, C. and Rothkopf, C. (2011), "Bayesian multitask inverse reinforcement learning", *paper presented at the 9th European Workshop on Reinforcement Learning* (*EWRL 2011*), *Athens, Greece, 9-11 September*.

Friedman, M. and Savage, L. (1952), "The expected-utility hypothesis and the measurability of utility", *The Journal of Political Economy*, No. 6, pp. 463-74.

Grollman, D. and Billard, A. (2011), "Donut as I do: learning from failed demonstrations", *IEEE International Conference on Robotics and Automation, Shanghai, 9-13 May*, pp. 9-13.

Grudic, G. and Lawrence, P. (1996), "Human-to-robot skill transfer using the spore approximation", *Robotics and Automation, Proceedings, 1996 IEEE International Conference on*, Vol. 4, pp. 2962-7.

Henry, P., Vollmer, C., Ferris, B. and Fox, D. (2010), "Learning to navigate through crowded environments", *Robotics and Automation* (*ICRA*), *2010 IEEE International Conference on*, pp. 981-6.

Heskes, T. (1998), "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* (*ICML'98*), pp. 233-41.

Jaynes, E. (1957), "Information theory and statistical mechanics", *Physical Review*, Vol. 108 No. 2, p. 171.

Kaelbling, L., Littman, M. and Moore, A. (1996), "Reinforcement learning: a survey", *Journal of Artificial Intelligence Research*, Vol. 4, pp. 237-85.

Kalakrishnan, M., Theodorou, E. and Schaal, S. (2010), "Inverse reinforcement learning with PI 2", The Snowbird Workshop (submitted to).

Keeney, R. and Raiffa, H. (1993), *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*, Cambridge University Press, Cambridge.

Kober, J. and Peters, J. (2010), "Imitation and reinforcement learning, practical algorithms for motor primitives in robotics", *Robotics and Automation Magazine, IEEE*, Vol. 17 No. 2, pp. 55-62.

Kolter, J., Abbeel, P. and Ng, A. (2008), "Hierarchical apprenticeship learning with application to quadruped locomotion", *Advances in Neural Information Processing Systems*, MIT Press, Cambridge, MA.

Lafferty, J., McCallum, A. and Pereira, F. (2011), "Conditional random fields: probabilistic models for segmenting and labeling sequence data", *Proceedings of the International Conference on Machine Learning* (*ICML 2011*), Morgan Kaufmann, Burlington, MA, pp. 282-9.

Lee, S. and Popovi, Z. (2010), "Learning behavior styles with inverse reinforcement learning", ACM SIGGRAPH 2010 papers, ACM, New York, NY, pp. 1-7.

Lopes, M., Melo, F. and Montesano, L. (2009), "Active learning for reward estimation in inverse reinforcement learning", *Machine Learning and Knowledge Discovery in Databases*, Vol. 5782 No. 1, pp. 31-46.

Mason, M. and Lopes, M. (2011), "Robot self-initiative and personalization by learning through repeated interactions", *Proceedings of the 6th International Conference on Human-robot Interaction*, pp. 433-40.

Melo, F. and Lopes, M. (2010), "Learning from demonstration using MDP induced metrics", *Machine Learning and Knowledge Discovery in Databases*, Springer, Berlin, pp. 385-401.

Murphy, K. (2000), "A survey of POMDP solution techniques", *Environment*, Vol. 2, p. X3.

Neu, G. and Szepesvári, C. (2007), "Apprenticeship learning using inverse reinforcement learning and gradient methods", *Proceedings of the Twenty-third Conference Annual Conference on Uncertainty in Artificial Intelligence* (*UAI-07*), pp. 295-302.

Ng, A. and Russell, S. (2000), "Algorithms for inverse reinforcement learning", *Proceedings of the Seventeenth International Conference on Machine Learning*, pp. 663-70.

Pomerleau, D. (1991), "Efficient training of artificial neural networks for autonomous navigation", *Neural Computation*, Vol. 3 No. 1, pp. 88-97.

Puterman, M. (1994), *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, Wiley, New York, NY.

Qiao, Q. and Beling, P. (2011), "Inverse reinforcement learning with Gaussian process", *American Control Conference* (*ACC*), pp. 113-18.

Ramachandran, D. and Amir, E. (2007), "Bayesian inverse reinforcement learning", *Proceedings of the 20th International Joint Conference on Artificial Intelligence*.

Ratliff, N., Bagnell, J. and Zinkevich, M. (2006), "Maximum margin planning", *Proceedings of the 23rd International Conference on Machine Learning*, pp. 729-36.

Ratliff, N., Silver, D. and Bagnell, J. (2009), "Learning to search: functional gradient techniques for imitation learning", *Autonomous Robots*, No. 1, pp. 25-53.

Robbins, H. (1992), "An empirical Bayes approach to statistics", *Breakthroughs in Statistics: Foundations and Basic Theory*, Vol. 1, p. 388.

Rothkopf, C. and Dimitrakakis, C. (2011), "Preference elicitation and inverse reinforcement learning", *Proceedings of 22nd European Conference on Machine Learning ECML, Part III, LNAI 6913*, pp. 34-48.

Russell, S. (1998), "Learning agents for uncertain environments (extended abstract)", *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, pp. 101-3.

Schaal, S. (1999), "Is imitation learning the route to humanoid robots?", *Trends in Cognitive Sciences*, Vol. 3 No. 6, pp. 233-42.

Shen-yi, C., Hui, Q., Jia, F., Zhuo-jun, J., Miao-liang, Z. and Springer (2010), "Modified reward function on abstract features in inverse reinforcement learning", *Journal of Zhejiang University – Science C*, Vol. 11 No. 9, pp. 718-23.

Silva, V., Costa, A. and Lima, P. (2006), "Inverse reinforcement learning with evaluation", *IEEE International Conference on Robotics and Automation* (*ICRA06*), *Orlando, FL, USA*, pp. 4246-51.

Silver, D., Bagnell, J. and Stentz, A. (2010), "Learning from demonstration for autonomous navigation in complex unstructured terrain", *The International Journal of Robotics Research*, Vol. 29 No. 12, p. 1565.

Silver, D., Bagnell, J. and Stentz, A. (2011), "Perceptual interpretation for autonomous navigation through dynamic imitation learning", *International Symposium on Robotics Research*, pp. 433-49.

Sondik, E. (1971), "The optimal control of partially observable Markov processes", PhD thesis, Stanford University, Stanford, CA.

Sutton, R. and Barto, A. (1998), *Introduction to Reinforcement Learning*, MIT Press, Cambridge, MA.

Syed, U. and Schapire, R. (2008), "A game-theoretic approach to apprenticeship learning", *Advances in Neural Information Processing Systems*, MIT Press, Cambridge, MA, pp. 1449-56.

Tang, J., Singh, A., Goehausen, N. and Abbeel, P. (2010), "Parameterized maneuver learning for autonomous helicopter flight", *Robotics and Automation* (*ICRA*), *2010 IEEE International Conference on*, pp. 1142-8.

Theodorou, E., Buchli, J. and Schaal, S. (2010), "Reinforcement learning of motor skills in high dimensions: a path integral approach", *Robotics and Automation* (*ICRA*), *2010 IEEE International Conference on*, pp. 2397-403.

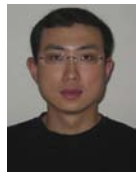
Zhang, H. and Parkes, D. (2008), "Enabling environment design via active indirect elicitation", *Proceedings Workshop on Preference Handling, Chicago, IL*.

Ziebart, B., Maas, A., Bagnell, J. and Dey, A. (2008), "Maximum entropy inverse reinforcement learning", *Proceedings 23rd AAAI Conference Artificial Intelligence*, pp. 1433-8.

Further reading

Leishman, J. (2006), *Principles of Helicopter Aerodynamics*, Cambridge University Press, New York, NY.

About the authors



Shao Zhifei received his BEng and MSc degrees in Technology of Measure and Control and Instrumentation from Harbin Institute of Technology, China in 2009, and Computer Control and Automation from Nanyang Technological University, Singapore in 2010, respectively. He has been a full-time PhD student in Nanyang Technological University, Singapore since 2010. His research interests include reinforcement learning, fuzzy neural networks and robotics.

Professor Er Meng Joo is currently a Full Professor in Electrical and Electronic Engineering and Director of Renaissance Engineering Programme, College of Engineering (CoE). He is also an elected member of the NTU Advisory Board, the NTU Senate Steering Committee and the Institution of Engineers, Singapore (IES) Council. He has authored three books entitled *Dynamic Fuzzy Neural Networks: Architectures, Algorithms and Applications and Engineering Mathematics with Real-World Applications* published by McGraw Hill in 2003 and 2005, respectively, and *Theory and Novel Applications of Machine Learning* published by In-Tech in 2009, 14 book chapters and more than 400 refereed journal and conference papers in his research areas of interest. He was the winner of the IES Prestigious Publication

(Application) Award in 1996 and IES Prestigious Publication (Theory) Award in 2001. He was awarded a Commonwealth Fellowship tenable at University of Strathclyde in 2000. He received the Teacher of the Year Award for the School of EEE in 1999, School of EEE Year 2 Teaching Excellence Award in 2008 and the Most Zealous Professor of the Year Award 2009. He also received the Best Session Presentation Award at the World Congress on Computational Intelligence in 2006. Furthermore, together with his students, he has won more than 30 awards at international and local competitions. Currently, he serves as the Editor-in-Chief for the *IES Journal B on Intelligent Devices and Systems*, an Area Editor of *International Journal of Intelligent Systems Science* and an Associate Editor of 12 refereed international journals, namely *IEEE Transactions on Fuzzy Systems*, *International Journal of Fuzzy Systems*, *Neurocomputing*, *International Journal of Humanoid Robots*, *Journal of Robotics*, *International Journal of Mathematical Control Science and Applications*, *International Journal of Applied Computational Intelligence and Soft Computing*, *International Journal of Fuzzy and Uncertain Systems*, *International Journal of Automation and Smart Technology*, *International Journal of Modelling, Simulation and Scientific Computing*, *International Journal of Intelligent Information Processing* and the *Evolving Systems*. Furthermore, he serves a Guest Editor of *International Journal of Neural Systems* and an editorial member of *Open Electrical and Electronic Engineering Journal*, *Birkhauser Autonomous Editorial Board* and *EE Times-Asia*. He has been invited to deliver more than 60 workshops, seminars, technical talks overseas. He has also been active in professional bodies. Currently, he is the Vice-Chairman of IEEE Computational Intelligence Society Standards Committee, Chairman of IEEE Computational Intelligence Society Singapore Chapter and Chairman of IES Electrical and Electronic Engineering Technical Committee. Er Meng Joo is the corresponding author and can be contacted at: emjer@ntu.edu.sg