This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCDS.2019.2957831, IEEE Transactions on Cognitive and Developmental Systems

An Efficient Unified Approach using Demonstrations for Inverse Reinforcement Learning

Maxwell Hwang, Wei-Cheng Jiang, Yu-Jen Chen, Kao-Shing Hwang, *Senior Member, IEEE,* and Yi-Chia Tseng

***Abstract*—A reinforcement learning agent is always equipped with a designed reward function to correct policies for optimal decision-making through interactions with an environment. However, it is difficult to design a reward function appropriate for complex reinforcement learning problems. To solve this difficulty, the inverse reinforcement learning (IRL) is introduced to provide an efficient way to design a reward function based on input derived from knowledgeable experts. In the inverse reinforcement learning, experts provide demonstrations so that the agents can imitate the behaviors accordingly. However, even incorrect demonstrations have merits, some of which are similar to correct ones, so as that the agents with these clues can endeavor to avoid the occurrence of that behavior. This paper introduces an IRL method which considers two types of demonstrations, correct and incorrect, in function approximation of a reward function. Given the clues from two opposite demonstrations, agents can iteratively approximate a reward function that can guide them to like expert's correct demonstrations and also, prevent them from making the same mistakes as the expert did. These incorrect demonstrations provide agents with some guidelines to avoid erroneous motions in the initial phase. Two simulated tasks, a labyrinth and robot soccer games are conducted to validate the proposed method. The simulation results show that the proposed method can achieve the objectives of generating an appropriate reward function to accomplish apprentice learning with an efficient learning time in IRL.**

***Index Terms*—Reinforcement Learning; Inverse Reinforcement Learning; Incorrect Demonstrations; Feature Weight**

I. Introduction

Robotic applications have become an imperative research topic [1]-[3], especially in order to make the robots perform intelligent behavior. Learning from demonstration in robotic systems is an approach to retrieve demonstrated motions from tasks and implementing them in new contexts [4]. Machine learning enables robots to perform a range of intelligent behavior after training [1]-[3]. Reinforcement learning (RL) is one of these machine learning methods and has been broadly used to adaptively control robots [5][6]. In the process of a value-based RL, an agent perceives a reinforcement signal to maximize the long-term cumulative reward and learn an optimal policy. Unfortunately, this reinforcement signal, in the form of rewards, is usually handcrafted, instead of a direct response from the physical environment. How to choose a well-defined reward function is always a major challenge for RL [7][8], especially in complex applications. To help resolve this problem, inverse reinforcement learning (IRL) induces a reward function from expert's demonstrated behavior [7]-[10]. The IRL agent handles reward functions based on the sequence of the expert's decision-making by means of linear combination over the state space. A hybrid system based on IRL optimizes the mix of fuel and power battery [11]. A probabilistic method models the routing preferences of drivers using IRL methods and imitation learning [12].

It is worth pointing out that IRL intrinsically represents the demonstrations in terms of the reward functions. Apprenticeship learning (AL) algorithms are also based on an expert's correct demonstrations to learn a policy which can produce the emulating behavior [13]. In AL, the primary interest is a policy which can generate the demonstrations, though it is sometimes not necessarily, obtained through the reward. Unlike the IRL approaches, behavior cloning, which is also closely related, is given some examples of a behavior and simply tries to reproduce it. This could mean generating a behavior that matches the statistics of the observed behavior. It is obvious that it can be resolved by conventional supervised learning. For instance, as in regressions, given some demonstrations, a neural network can be trained to generate similar behaviors in similar situations.

The IRL applications can be formulated as a Markov decision process (MDP) [14], where a reward function should be

Maxwell Hwang is with the Department of Colorectal Surgery, the Second Affiliated Hospital of Zhejiang University School of Medicine, Hangzhou, China, and also with the Cancer Institute (Key Laboratory of Cancer Prevention and Intervention, China National Ministry of Education; Key Laboratory of Molecular Biology in Medical Sciences, Zhejiang Province, China), the Second Affiliated Hospital, Zhejiang University School of Medicine, Hangzhou, China. (e-mail: himax26@zju.edu.cn).

Wei-Cheng Jiang is with the Department of Electrical Engineering, Tunghai University, Taichung 40704, Taiwan. (e-mail: jiangwc@thu.edu.tw).

Kao-Shing Hwang is with the Department of Electrical Engineering, National Sun Yat-sen University, Kaohsiung 80424, Taiwan. (e-mail: hwang@ccu.edu.tw).

Yu-Jen Chen is with the Department of Electrical Engineering, National Chung Cheng University, Chiayi 62102, Taiwan. (e-mail: yujenchn@gmail.com).

Yi-Chia Tseng is with the Department of Electrical Engineering, National Sun Yat-sen University, Kaohsiung 80424, Taiwan. (e-mail: Kero523@hotmail.com).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCDS.2019.2957831, IEEE Transactions on Cognitive and Developmental Systems

explicitly defined by expert demonstrations. Once an approximated reward function is acquired, agents can imitate the expert behaviors driven by this reward function. If the agent's actions and the expert behaviors are not similar, the agent must evaluate the difference, and that is used to correct the current approximated reward function. Then, after obtaining a modified reward function, the agents iteratively initiate a new episode of RL and calculate the difference between the learned behaviors and the demonstrated ones after the optimal policy is found based on the estimated reward function. Through multiple iterations, the agent always can find an appropriate reward function to replay the demonstrations.

In general, only positive demonstrations are shown to the learning agents in order to imitate the similar behaviors as much as possible. However, during the initial stages of the learning process, RL agents usually take an exploratory strategy, leading then to encounter unfavorable situations, such as bumping into walls or moving obstacles, and other mistakes. Nevertheless, these situations are avoidable if some clues about these difficulties can be provided in time. In other words, correct (positive) behavior can drive the agents to imitate the expert, but incorrect (negative) behavior can encourage the agents to avoid that expert behavior. Both types of behavior have the same importance for agents in learning [15][16]. In this paper, an agent is asked to consider the incorrect behaviors, in addition to the correct behaviors, and to thereby establish a reward function for solving IRL applications. This paper is organized as follows. Section II introduces the background, RL and IRL theories. The unified method without and with incorrect demonstrations is illustrated in Section III. In Section IV, the simulations are conducted to show the validity of the proposed methods, confirming the applicability and effectiveness of the algorithms. Finally, conclusions are drawn in the last section.

II. Background

A. Q-Learning

Q-Learning, an RL method, is a model-free algorithm [17] in which the agent collects experiences, (**s**, *a*, **s**', *r*), from the environment. These experiences are used to evaluate the action-value function, called $Q(\boldsymbol{s}, a)$, shown as follows.

$$Q(\boldsymbol{s},a) \leftarrow Q(\boldsymbol{s},a)+\beta(r+\gamma\max_{a'}Q(\boldsymbol{s}',a')-Q(\boldsymbol{s},a)) \quad (1)$$

where $\beta\,(0\le\beta\le1)$ is the learning rate and $\gamma\,(0\le\gamma\le1)$ is the discount factor. The agent collects numerous experiences and updates the *Q*-values, which are converging to an optimal Q^*-value [17].

B. Inverse Reinforcement Learning

As mentioned above, the concept of IRL is similar to apprenticeship learning acquiring skilled behaviors from experts' demonstrations [13][18]-[20]. The demonstrations are formed by a set of state-action pairs, (**s**, *a*), and the experts indicate how to take actions in each state, referred to as demonstrations. After obtaining the demonstrations, the reward function is approximated by using a linear combination of weight vector and function of features as follows,

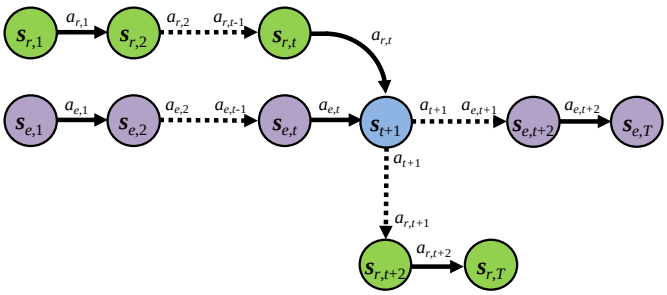


Fig. 1. Example of two trajectories.

$$R(\boldsymbol{s}) = \boldsymbol{\omega}^T \cdot \phi(\boldsymbol{s}, a) \quad (2)$$

where $\phi: \boldsymbol{S}\times\boldsymbol{A}\rightarrow[0,1]^n$ is a function of features over states and actions that map from states and actions to features with *n* dimensions and $\boldsymbol{\omega}\in R^n$ is the weight vector tuned during the process of learning.

The action-value function $Q^{\pi}(\boldsymbol{s}_0, a_0)$ under policy π is formulated as follows.

$$\begin{aligned} E_{\boldsymbol{s}_0\sim D}[Q^{\pi}(\boldsymbol{s}_0, a_0)] &= E\left[\sum_{t=0}^{\infty}\gamma^t R(\boldsymbol{s}_t, a_t)|\pi\right] \\ &= E\left[\sum_{t=0}^{\infty}\gamma^t \omega^T\cdot\phi(\boldsymbol{s}_t, a_t)|\pi\right] \\ &= \omega^T\cdot E\left[\sum_{t=0}^{\infty}\gamma^t \phi(\boldsymbol{s}_t, a_t)|\pi\right] \end{aligned} \quad (3)$$

where $\boldsymbol{s}_0$ id the initial-state with a distribution *D*; $\mu(\pi)=E\left[\sum_{t=0}^{\infty}\gamma^t\phi(\boldsymbol{s}_t,a_t)|\pi\right]$ is defined as the expected discounted accumulated feature value or the feature expectation.

An expert's demonstrations can be formulated as feature expectations, $\mu_e=\mu(\pi_e)$ and the agent takes actions following its own policy, π. The difference of feature expectations between the expert and the agent is calculated as follows [13][18]-[20],

$$\begin{aligned} &\left| E\left[\sum_{t=0}^{\infty}\gamma^t\phi(\boldsymbol{s}_t,a_t)|\pi_e\right] - E\left[\sum_{t=0}^{\infty}\gamma^t\phi(\boldsymbol{s}_t,a_t)|\pi\right]\right| \\ &= \left|\omega^T\mu_{\pi}-\omega^T\mu_e\right| \\ &\le \|\omega\|_2\,\|\mu_{\pi}-\mu_e\|_2 \\ &\le 1\times\theta \end{aligned} \quad (4)$$

where θ is a predefined threshold. The Cauchy Inequality finds the policy that is most similar to expert's feature expectations if the norm of $\boldsymbol{\omega}$ is 1 in equation (4). If experts demonstrate many trajectories, the expert's feature expectations are computed using equation (5):

$$\mu_e = \left(\sum_{j=1}^{m}\sum_{t=0}^{\infty}\gamma^t\phi(\boldsymbol{s}_t^{(j)}, a_t^{(j)})\right)\Big/m \quad (5)$$

where *m* is the number of trajectories.

III. A Unified Approach with Two Types of Demonstrations

The proposed method adopts the concept of boosting in binary classification problems and eligibility trace in [21], which is a temporary record of the occurrence of an event, such

as the visiting of a state or the taking of an action. This problem is required as a sequence classification task, where each of the feature expectations represents a different class. The trace marks the memory parameters associated with the event as eligible for undergoing learning changes. Since there is no desired output or correct label provided in IRL problems, it is difficult to evaluate the gradient vector that would encourage the approximator, such as a linear combination function or neural network, to be slightly more likely to take actions causing fewer errors in the future. But the problem is that it is always premature to determine during the process of learning if a potential action is really better than the other candidates. But the critical point is that the problem can be attenuated, because the learning process could be triggered until the end of the trials, and then take the reward obtained and use the scalar of that as the gradient for the action taken. Therefore, if a penalty scales filled in to discourage the action for that reward input in the future. That is; a stochastic policy samples actions that happen to eventually lead to good outcomes will be encouraged in the future, and actions taken that lead to bad outcomes will be discouraged. Furthermore, the reward does not even need to be a real scalar value if the goal is achieved eventually. Instead, it can be an arbitrary measure of some kinds of eventual quality. The training protocol proposed is depicted as follows.

A. Unified Method for Inverse Reinforcement Learning

In IRL, the agent must imitate the expert's demonstrations which is represented as feature expectation μ_e. An agent in an environment selects a sequence of actions following its own policy π, and the trajectory is represented as feature expectation μ_π. In equation (4), the difference between the agent's and the expert's feature expectations is calculated. If the action-value functions of the expert and the agent are similar to each other, the agent has behaviors similar to the expert. Otherwise, the agent must update a weight vector, $\boldsymbol{\omega}_i$, and obtain a new policy, $\pi^{(i+1)}$. When the difference, d_i, is smaller than the threshold, θ, the learning iteration is terminated. The difference is defined as follows,

$$d_i \leftarrow \left\| \mu_e - \mu_\pi^{(i)} \right\| \tag{6}$$

where i means the number of iteration and $d_i \le \theta$ represents the policy of the agent whose behavior is close to the expert's demonstration.

The unified method in this paper seeks to decrease the difference in fewer iterations. In Fig. 1, there are two trajectories, $\boldsymbol{S}_r=\{\boldsymbol{s}_{r,1}, \boldsymbol{s}_{r,2},\ldots, \boldsymbol{s}_{r,t+2},\ldots, \boldsymbol{s}_{r,T}\}$, $\boldsymbol{A}_r=\{a_{r,1}, a_{r,2},\ldots, a_{r,t+2},\ldots, a_{r,T}\}$ and $\boldsymbol{S}_e=\{\boldsymbol{s}_{e,1}, \boldsymbol{s}_{e,2},\ldots, \boldsymbol{s}_{e,t+2},\ldots, \boldsymbol{s}_{e,T}\}$, $\boldsymbol{A}_e=\{a_{e,1}, a_{e,2},\ldots, a_{e,t+2},\ldots, a_{e,T}\}$, of which the trajectory $\boldsymbol{S}_r$ is demonstrated by the agent and trajectory $\boldsymbol{S}_e$, is demonstrated by the expert, $\boldsymbol{S}_a$ and $\boldsymbol{S}_e \subseteq \boldsymbol{S}$, $\boldsymbol{A}_r$ and $\boldsymbol{A}_e \subseteq \boldsymbol{A}$, $\boldsymbol{S}$ is state space and $\boldsymbol{A}$ is action space. The state, $\boldsymbol{s}_{t+1}$, means that it is visited by both the agent and the expert. The two trajectories are formulated as feature expectations listed as follows.

$$\begin{aligned} \mu_e &= E\left[\sum_{t=0}^{\infty} \gamma^t \phi(\boldsymbol{s}_t, a_t) | \pi_e\right] \\ \mu_\pi^{(i)} &= E\left[\sum_{t=0}^{\infty} \gamma^t \phi(\boldsymbol{s}_t, a_t) | \pi^{(i)}\right] \end{aligned} \tag{7}$$

where μ_e and $\mu_\pi^{(i)}$ represent feature expectations of the expert and of the agent, respectively.

According to equation (6), in each learning iteration, there are three types of cases between feature expectations of the agent and the expert after obtaining the expert's demonstration. In the first type, the all elements of expert's feature expectation are larger than all elements of agent's feature expectation. Because the discount rate of each state leads to different values of feature expectation, this indicates that the expert visits some states with actions earlier than the agent or that these states with actions are visited only by the expert. In the second type, all elements of expert's feature expectation are smaller than all elements of agent's feature expectation. This indicates that the expert visits some state-action pairs later than the agent or that these states are visited only by the agent. In the third type, all elements of expert's feature expectation are equal to all elements of agent's feature expectation. This indicates that the expert and the agent visit the same states with actions such that the order pairs, ($\boldsymbol{s}_{r,t} = \boldsymbol{s}_{e,t}$, $a_{r,t} = a_{e,t}$), $t = 1, 2,\ldots, T$, are the same. This is, the two trajectories in Fig. 1 are the same.

Based on the expert's trajectory, in the initial phase if the behaviors between the expert and the agent are different, the agent needs to adjust the weight vector of state-action pairs, $\omega_i(\boldsymbol{s}, a)$, and to decrease the difference. In the later phase, if the behaviors are still not similar to each other, the adjustment must continue iteratively. Based on the three cases as mentioned above, if the discounted accumulated feature expectations of all states between the agent and the expert are different, the error in each state must be evaluated. The error in each state is defined as follows,

$$\Delta\mu_i(\boldsymbol{s}, a) \leftarrow \mu_e(\boldsymbol{s}, a) - \mu_\pi^{(i)}(\boldsymbol{s}, a) \tag{8}$$

where i means the number of iteration and $\boldsymbol{s}$ represents states, $\boldsymbol{s} \in \boldsymbol{S}$, and a represents actions, $a \in \boldsymbol{A}$, $\mu_e(\boldsymbol{s}, a)$ represents discounted accumulated feature expectation of state-action pairs of the expert, and $\mu_\pi^{(i)}(\boldsymbol{s}, a)$ represents discounted accumulated feature expectation of state-action pairs of the agent.

Because the agent must adjust the weight vector of state-action pairs, $\omega_i(\boldsymbol{s}, a)$, the unified method introduces an updating rule as follows.

$$\omega_{i+1}(\boldsymbol{s}, a) \leftarrow \omega_i(\boldsymbol{s}, a) + F_i(\boldsymbol{s}, a) \times \Delta\mu_i(\boldsymbol{s}, a) \tag{9}$$

where $F_i(\boldsymbol{s}, a)$ can be regarded as a composite function of boosting and eligibility trace and $F_i(\boldsymbol{s}, a) \in (0, 1]$. It weakens the weights of feature expectations where the expert's demonstration matches the one generated by the current policy and vice versa. In terms of eligibility, once mismatches are classified at an expected feature, its eligibility should be activated to enhance the effect on weight updating for next iterations.

In the first type, because an expert visits some state-action

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCDS.2019.2957831, IEEE Transactions on Cognitive and Developmental Systems

Weight	$F_i(s_1, a_1)$	$F_i(s_2, a_2)$	$F_i(s_3, a_3)$	…	$F_i(s_{\|S\|-1}, a_{\|S\|-1})$	$F_i(s_{\|S\|}, a_{\|S\|})$
$\mu_e(s, a)$	0	γ	γ	…	γ	0
$\mu_i(s, a)$	γ^2	0	γ^2	…	γ	0
$\Delta\mu_i(s, a)$	x	x	x	…	0	0
adjust weight	increase	increase	increase	…	decrease	decrease

Fig. 2. Example for the adjustment of adaptive weight.

pairs that are not visited by the agent or the expert visits these state-action pairs earlier than the agent, the errors, $\Delta\mu_i(s, a)$, are positive, while the weight vector of state-action pairs, $\omega_{i+1}(s, a)$, is small and must be increased. In the second type, the agent visits some state-action pairs that are not visited by the expert or it visits these state-action pairs later. The error, $\Delta\mu_i(s, a)$, is negative, which means that the weight vector of state-action pairs, $\omega_{i+1}(s, a)$, is large and must be decreased. In the third case, the error, $\Delta\mu_i(s, a)$, is equal to zero, which means that the weight vector of state-action pairs $\omega_{i+1}(s, a)$ is correct in these state-action pairs. In the first and second cases, the weight vector of state-action pairs, $\omega_{i+1}(s, a)$, must be adjusted and the eligibility trace, $F_i(s, a)$, is adjusted flexibly in each state-action pair. The adjustment of eligibility trace, $F_i(s, a)$, is defined as follows.

Algorithm 1 Algorithm of the unified method

$\boldsymbol{S}$:State space
$\boldsymbol{A}$:Action space
$\boldsymbol{D_G}$: A set of correct demonstrations

1. **Initialize:**
   $\boldsymbol{D_G}$
2. Calculate the expert's feature expectation, μ_e.
3. **Initialize** feature weight of all state-action pairs:
   $F_0(s, a)$, $s \in S$, $a \in A$.
4. $i \leftarrow 0$.
5. The agent demonstrates a trajectory following its own policy, $\pi^{(i)}$, and derives the agent's feature expectation, $\mu_\pi^{(i)}$.
6. **Repeat** (for each iteration):
7. **if** $d_i \geq \theta$ **then**
8. **for** k = 1 to | $\boldsymbol{S}$ |
9. **for** z = 1 to | $\boldsymbol{A}$ |
10. Calculate the eligibility trace for all states-action pairs as follows.
$$F_{i+1}(s, a) \leftarrow \begin{cases} F_i(s, a) + \alpha(1 - F_i(s, a)), & \text{if } \Delta\mu_i(s, a) \neq 0 \\ F_i(s, a) + \alpha(0 - F_i(s, a)), & \text{if } \Delta\mu_i(s, a) = 0 \end{cases}$$
11. Update the reward weight for all state-action pairs as follows.
$$\omega_{i+1}(s, a) \leftarrow \omega_i(s, a) + F_i(s, a) \times \Delta\mu_i(s, a)$$
12. $z \leftarrow z + 1$
13. **end for**
14. $k \leftarrow k+1$
15. **end for**
16. A reward function is established as follows.
$$R(s) = \omega^T \cdot \phi(s, a)$$
17. Q-Learning uses the reward function to search for a new policy, $\pi^{(i+1)}$.
18. Compute the agent's feature expectation as follows.
$$\mu_\pi^{(i+1)} = E\left[\sum_{t=0}^{\infty} \gamma^t \phi(s_t, a) \middle| \pi^{(i+1)}\right]$$
19. $i \leftarrow i + 1$.
20. **end if**
21. **Until** $d_i \leq \varepsilon$.

$$F_{i+1}(s, a) \leftarrow \begin{cases} F_i(s, a) + \alpha(1 - F_i(s, a)), & \text{if } \Delta\mu_i(s, a) \neq 0 \\ F_i(s, a) + \alpha(0 - F_i(s, a)), & \text{if } \Delta\mu_i(s, a) = 0 \end{cases} \quad (10)$$

where the α ($0 \leq \alpha \leq 1$) is an update rate.

An example of the adjustment is shown as Fig. 2. When the error, $\Delta\mu_i(s, a)$, is not equal to zero, the weight vector of state-action pairs, $\omega_{i+1}(s, a)$, must be adjusted. The errors in some state-action pairs, (s_1, a_1), (s_2, a_2), (s_3, a_3),…, need to be changed. The error of state-action pair, (s_1, a_1), has a negative value and the difference is large, so the weight, $F_i(s_1, a_1)$, is large. In other cases, such as states 2 and 3, the weight is small. The error in state-action pairs $(s_{|S|-1}, a_{|S|-1})$ and $(s_{|S|}, a_{|S|})$ is zero, so the weight must approach zero.

After updating the weight vector, the reward function is established and Q-Learning uses it to find a new trajectory. The new trajectory is used to repeat the same steps for obtaining a new feature expectation. The steps calculate the difference, d_i, smaller than the threshold, θ. The algorithm of the unified method is summarized in **Algorithm** 1.

B. Unified Method Integrated with Incorrect Demonstrations

In the previous section, the unified method only considers the correct demonstration which is demonstrated by the expert. The underlying assumption is that the demonstrations are successful and are suitable for reproduction [22][23]. However, the agent in the initial phase has no knowledge about the environment, so it may visit some state-action pairs that may be dangerous. An expert with knowledge can avoid these state-action pairs that have uncertainty, but the agent does not have this ability. So, it should explore state-action pairs that have not been visited by the expert. Therefore, to enhance the abilities of the agent, the unified method is integrated with incorrect demonstrations so that the method can teach the agent not only what to do but also what not to do. Therefore, having both correct and incorrect demonstrations is important for the agent during the process of learning. The expert not only performs the correct trajectory, but also performs the incorrect trajectory, $\boldsymbol{S}_b = \{s_{b,1}, a_{b,1}, s_{b,2}, a_{b,2}, \ldots, s_{b,t+2}, a_{b,t+2}, \ldots, s_{b,T}, a_{b,T}\}$, $\boldsymbol{S}_b \subseteq \boldsymbol{S}$. Then, the trajectory is represented as a feature expectation. The feature expectation of the incorrect demonstration is defined as follows,

$$\mu_B = E\left[\sum_{t=0}^{\infty} \phi(s_t, a_t) \middle| \pi_B\right] \quad (11)$$

Since the state-action pairs visited by incorrect demonstration in any moment are viewed as the same. Regardless of when the state-action pairs are visited in the incorrect demonstration, it is always considered to incorrect feature expectation by the agent. If a state-action pair is never being visited by both the agent and the expert, the trace, $F_i(s, a)$ would be decaying.

As mentioned above, the unified method considers both correct and incorrect demonstrations and thereby increases the learning speed with fewer iterations. In the beginning, the feature expectations of incorrect demonstrations are evaluated based on the policy, π_B. Only the state-action pairs that are visited by the agent and are not visited by the expert need to be considered in the incorrect demonstrations. A case of incorrect demonstrations is specified and shown in Fig. 3. In this case,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCDS.2019.2957831, IEEE Transactions on Cognitive and Developmental Systems

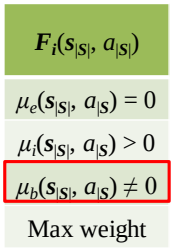


Fig. 3. Situation is solved by incorrect demonstrations.

Algorithm 2 Algorithm of the unified method with incorrect demonstrations

$\boldsymbol{S}$:State space
$\boldsymbol{A}$:Action space
$\boldsymbol{D_G}$: A set of correct demonstrations
$\boldsymbol{D_B}$: A set of incorrect demonstrations

1. **Initialize:**
   $\boldsymbol{D_G}$ and $\boldsymbol{D_B}$
2. Calculating the expert's feature expectations, μ_e and μ_B, from two sets, $\boldsymbol{D_G}$ and $\boldsymbol{D_B}$
3. **Initialize** feature weight of all state-action pairs:
   $F_0(\boldsymbol{s}, a)$, $\boldsymbol{s} \in \boldsymbol{S}$, $a \in \boldsymbol{A}$.
4. $i \leftarrow 0$.
5. The agent demonstrates a trajectory following its own policy, $\pi^{(i)}$, and derives the agent's feature expectation, $\mu_\pi^{(i)}$.
6. **Repeat** (for each iteration):
7. **if** $d_i \geq \theta$ **then**
8. **for** k = 1 to | $\boldsymbol{S}$ |
9. **for** z = 1 to | $\boldsymbol{A}$ |
10. Calculating the weight for all state-action pairs as follows.
11. **if** $\mu_e(\boldsymbol{s}, a)$==0 && $\mu_i(\boldsymbol{s}, a)$>0 && $\mu_b(\boldsymbol{s}, a)$ == 1
12. $F_{i+1}(\boldsymbol{s}, a) \leftarrow 1$(Maximum weight)
13. **else**
$$F_{i+1}(\boldsymbol{s}, a) \leftarrow \begin{cases} F_i(\boldsymbol{s}, a) + \alpha(1 - F_i(\boldsymbol{s}, a)), & \text{if } \Delta\mu_i(\boldsymbol{s}, a) \neq 0 \\ F_i(\boldsymbol{s}, a) + \alpha(0 - F_i(\boldsymbol{s}, a)), & \text{if } \Delta\mu_i(\boldsymbol{s}, a) = 0 \end{cases}$$
14. **end if**
15. Updating the reward weight for all state-action pairs as follows.
$$\omega_{i+1}(\boldsymbol{s}, a) \leftarrow \omega_i(\boldsymbol{s}, a) + F_i(\boldsymbol{s}, a) \times \Delta\mu_i(\boldsymbol{s}, a)$$
16. $z \leftarrow z + 1$
17. **end for**
18. $k \leftarrow k + 1$
19. **end for**
20. A reward function is established as follows.
$$R(\boldsymbol{s}) = \boldsymbol{\omega}^T \cdot \boldsymbol{\phi}(\boldsymbol{s}, a)$$
21. Q-Learning uses the reward function to search for a new policy, $\pi^{(i+1)}$.
22. Computing the agent's feature expectation as follows.
$$\mu_\pi^{(i+1)} = E\left[\sum_{t=0}^{\infty} \gamma^t \boldsymbol{\phi}(\boldsymbol{s}_t, a) \middle| \pi^{(i+1)}\right]$$
23. $i \leftarrow i + 1$.
24. **end if**
25. **Until** $d_i \leq \varepsilon$.

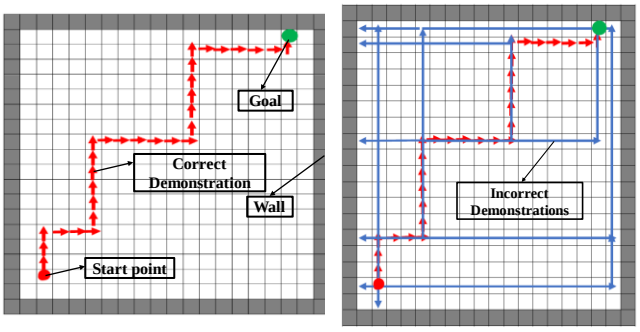
the feature expectation of state-action pairs, $\mu_b(\boldsymbol{s}, a)$, is not equal to zero, indicating that the state-action pairs are appearing as incorrect demonstrations. If a state-action pair appears in the incorrect demonstrations, it is apparent that the visiting time must be decreased for the agent. Thus, to adjust the weight vector of state-action pair, $\omega_i(\boldsymbol{s}, a)$ as soon as possible, the weight, $F_i(\boldsymbol{s}, a)$, is set as a maximum value. Based on the settings, the incorrect demonstrations can unload the learning burden for the agent and speed up learning for giving instructive commands and placing constraints on the state-action pairs which may or may not be explored in the next episode. In this paper, the incorrect demonstrations are similar to "hit the objects", "make a detour", "over speed", "go in the opposite direction intentionally", and so on. And these incorrect demonstrations are used in the simulation environment. The algorithm of the unified method with incorrect demonstrations is summarized in **Algorithm** 2.

IV. Simulation

In this section, two simulations, a maze environment (without and with obstacles) and robot soccer, are introduced to verify the unified method and the unified method integrated with incorrect demonstrations. In each case, an expert performs correct demonstrations and incorrect demonstrations that are used to form a reward function. A learning agent must imitate the two demonstrations based on the reward function. In the initial phase of learning, the agent's behaviors are different from the expert's, so the reward function must be corrected. After multiple iterative processes, the correct reward function appropriate for the task can be established. Therefore, for both cases, the proposed methods are compared with Adaboost-IRL [24] to show that the proposed method can imitate the expert's behaviors as quickly as possible.

A. Maze without obstacles

The maze environment shown in Fig. 4, has 20 x 20 grid surrounded by walls. In this maze, the green goal is set in the right top of the maze and the starting point, which is red, is set in the left bottom of the maze as shown in Fig 4(a). In this simulation, an expert produces a trajectory with 31 steps from the starting point to the goal, each having a red arrow. Thus, the trajectory is a correct demonstration to guide the agent to arrive at the goal. Therefore, the learning agent must imitate the expert's behaviors. In Fig. 4(b), the blue trajectories are incorrect demonstrations that involve hitting the obstacles and making a detour. Because the incorrect demonstrations are considered, the agent can be taught to avoid incorrect actions. The action space includes four actions, {"up", "down", "right"



(a) (b)

Fig. 4. Maze environment for the agent. (a) Map of the maze with the expert's demonstration. (b) Map of the maze with the expert's correct demonstration and incorrect demonstrations.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCDS.2019.2957831, IEEE Transactions on Cognitive and Developmental Systems

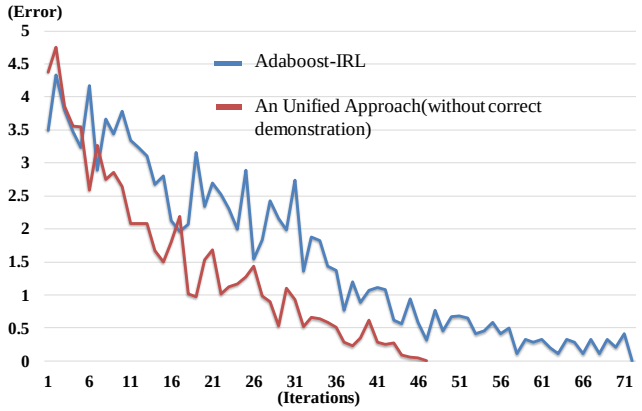


Fig. 5. Compare the error of two methods.

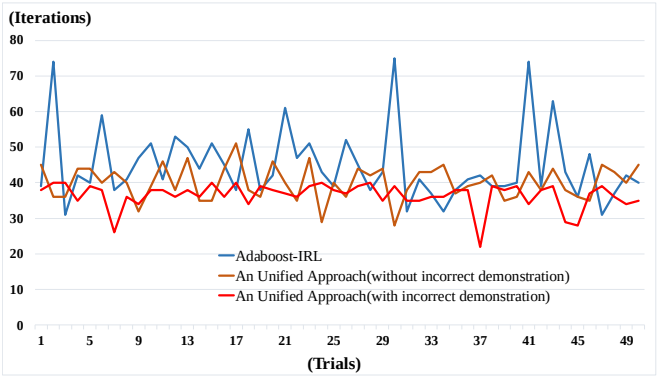


Fig. 6. Comparison of the iterations by two methods.

and "left"}, and an agent moves one grid in each action. In this example of learning, the learning rate (β) is set as 0.7 and the discount factor (γ) is set as 0.9. The learning policy is ε-Greedy with a 10% exploration probability (ε). There are 2000 training episodes in each iteration, and each episode has 500 steps. In each learning, if an agent touches a goal or cannot reach the goal within 500 steps, the episode is terminated, and the agent is reset to the next episode.

The first comparison does not consider incorrect demonstration and it compares the unified method (without incorrect demonstrations) with the Adaboost-IRL method [24].

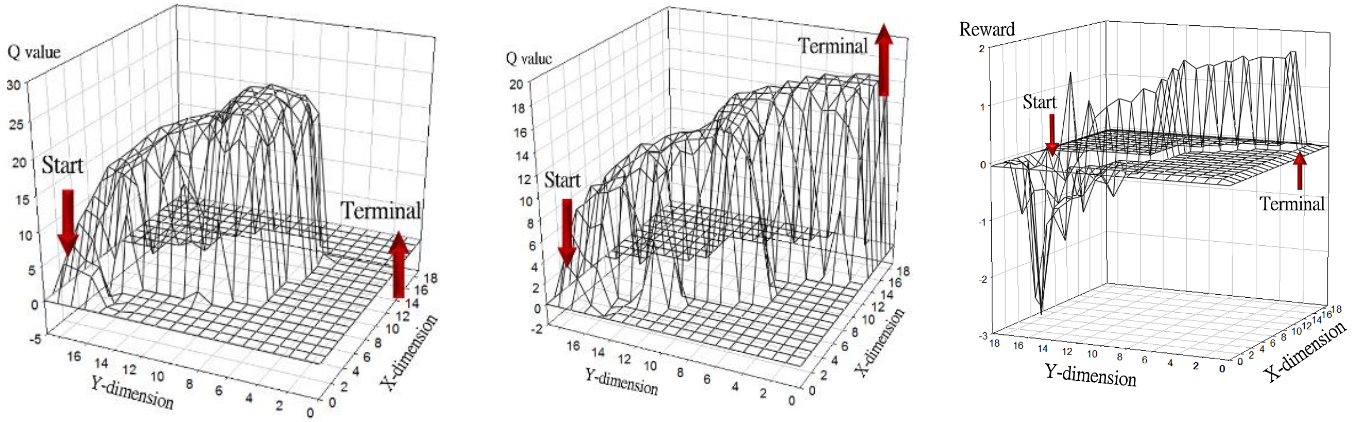
TABLE I
COMPARISON OF NUMBER OF LEARNING ITERATIONS IN MAZE ENVIRONMENT

Methods	*Mean*	*Standard deviation*
Adaboost-IRL	44.94	10.27
Unified method (without incorrect demonstrations)	40.10	4.73
Unified method (with incorrect demonstrations)	36.48	3.58

In each learning iteration, the difference of feature expectation, d_i, between the agent and the expert is calculated and this difference must converge to zero gradually, as shown in Fig. 5. When the difference converges to zero, this means that the agent can follow the same behaviors as the expert. In the figure, the x-axis is the number of the iterations and the y-axis is the error. The unified method can decrease the error to zero in the 47rd iterations and the Adaboost-IRL method takes until the 73th iterations. Therefore, the proposed unified method can efficiently decrease the error in only a few iterations.

Fig. 6 depicts the average number of iterations; the x-axis is the 100 number of the trials and the y-axis is the number of iterations. The figure shows that the learning curve of the Adaboost-IRL method drastically changed. But the proposed unified method can stably finish the learning tasks with the fewest iterations. The means and the standard deviations of the iterations are listed in Table I.

The unified method uses demonstrations performed by the expert to update the *Q*-values and to correct the reward function iteratively. Fig. 7 depicts the learning situations of *Q*-values and the reward value of all states that are stored in a *Q*-table and a reward table. The x-axis and y-axis represent the state space with two dimensions and the z-axis represents *Q*-values and reward values in each state. In the initial learning, all the *Q*-table is zero. In Fig. 7 (a), the *Q*-values of the agent become large corresponding to the trajectory which is demonstrated by the expert in the mid-term of learning. But, the reward function



(a) (b) (c)

Fig. 7. Q-values during the process of learning. (a) Q-values at the mid-term of learning. (b) Q-values at the end of learning. (c) Accumulated reward at the end of learning

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCDS.2019.2957831, IEEE Transactions on Cognitive and Developmental Systems

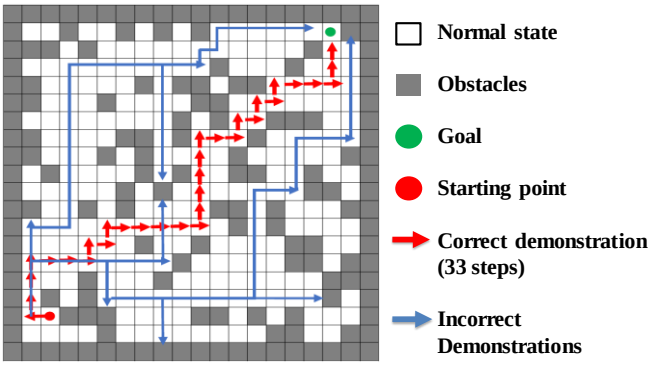


Fig. 8. Maze environment with obstacles for the agent.

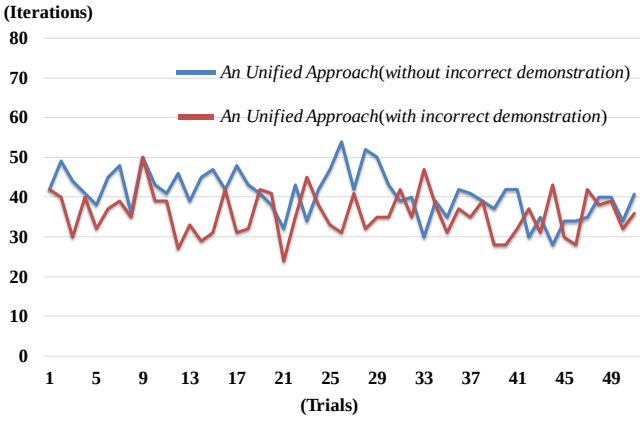


Fig. 9. Compare the performance without and with incorrect demonstrations.

TABLE II
COMPARISON OF NUMBER OF LEARNING ITERATIONS IN MAZE ENVIRONMENT WITH OBSTACLES.

Methods	*Mean*	*Standard deviation*
Unified method (without incorrect demonstrations)	40.82	5.74
Unified method (with incorrect demonstrations)	35.84	5.54

is not yet well established, so *Q*-values near the terminal states are not well-trained. In the final phase, it is obvious that the *Q*-values and reward values in each state became larger to guide the agent toward the goal, as shown in Fig. 7 (b) and Fig. 7 (c), respectively.

B. Maze environment (with obstacles)

The maze environment with obstacles shown in Fig. 8 has a 20 x 20 grid. In this maze, the environment and the settings of parameters are almost all the same as the previous one. The only difference is that this maze has some grey obstacles. In this simulation, an expert produces a trajectory composed of the 33 steps with red arrows from the starting point to the goal. The blue trajectories are incorrect demonstrations driving the agent to impact the obstacles and make a detour. The curves of learning iterations are shown in Fig. 9. The proposed method can stably finish the learning tasks within only a few iterations. The means and the standard deviations of all iterations are calculated and listed in Table II.

C. Robot Soccer

Fig. 10 presents a soccer field 1.4 meters long and 1.2 meters wide, surrounded by walls. There are four differential wheeled robots and a goal gate in the field. Three robots with the yellow and red stripes are moving obstacles that cannot be controlled, but are just moving back and forth following the blue trajectories on the ground. A ball is placed in front of the gate. The controlled robot with blue and red is used to implement the proposed methods. It is equipped with 15 infrared sensors for detecting the obstacles. Correct demonstrations are shown in pink and incorrect demonstrations in yellow are both produced by an expert conducting the robots, as shown in Fig. 11. When the expert performs the correct demonstration, a moving robot appears in the trajectory highlighted by a red circle. The expert decreases the speed to avoid the obstacles, the controlled robot tries to learn the reward function to imitate the same behaviors. In Fig. 11, the incorrect demonstrations are induced by four behaviors, “hit the obstacles”, “make a detour”, “overspeed” and “go in the opposite direction intentionally”.

The state space of the soccer field includes three dimensions, x-coordinate, y-coordinate, and a Boolean variable. The x-coordinate and y-coordinate are continuous and then discretized into 16 and 12 intervals separately. The Boolean variable has

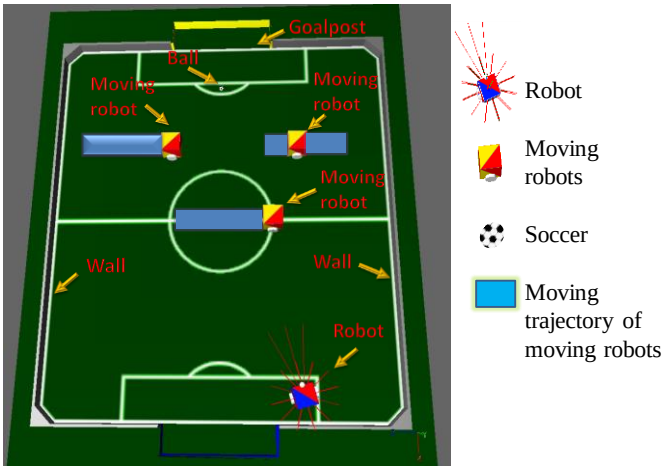


Fig. 10. Simulation scene of soccer field.

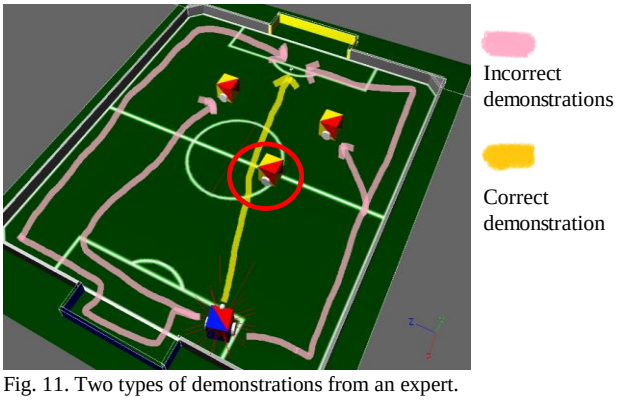


Fig. 11. Two types of demonstrations from an expert.

the value of a state. When the robot detects obstacles, the state is set to "true" and vice versa. The action space consists of five actions, such as "go forward towards the ball", "go backward from the ball", "turn right and go forward", "turn left and go forward", and "go forward toward the ball more slowly." The goal of the controlled robot is to avoid moving robots and kick the ball into the goal gate with the shortest path, as demonstrated. The learning rate (β) is set to 0.6 and the discount factor (γ) is 0.9. The exploration probability is decreasing over the time from 30% to 5%. The total number of episodes is 100 in this task. If the robot is unable to kick the ball into the goal within five minutes, an episode will be terminated immediately. The simulation results are shown in Fig. 12. Two curves show that the proposed method can decrease the error to zero. Nevertheless, the method without incorrect demonstrations reduces the error to zero in the 15th iterations and the method with incorrect demonstrations reduces the error to zero in the 7th iterations.

V. Conclusion

In general, the basic approaches in learning from demonstrations can be categorized into two different settings, learning from a given expert's policy and learning from expert trajectories. Learning from a given policy, also called apprentice learning [10], is a simpler problem than learning from trajectories but always with computational complexity. In addition, reward functions are often sparse, thereby providing a natural means of generalizing from a small amount of training data, even in very large state spaces. In addition, the human's behavior may encode a great deal of background information about the task that is easy to encode in the reward function but more complex to encode in a policy function, and which can be reused in later contexts. With active inverse reinforcement learning, an implicit reward function can be learned by a computational method with either parametric and non-parametric models [9].

The non-parametric inverse reinforcement learning algorithms are capable of learning multiple reward functions from unstructured demonstrations. The method has been extended with a number of optimizations making it suitable for large state spaces but is restricted to reward functions representing the goal of reaching particular sub-goal states. These reward functions much return a positive reward in a single state, and zero elsewhere. It is limited in practices. On the other hand, the parametric IRL paradigm has several advantages. First, if the reward function is a function of the objects or features in the world and not the agent's kinematics, then it can be naturally ported from human to robot (or between different robots) without encountering the correspondence problem. The proposed method an IRL method accommodates positive and negative demonstrations in the same time by a simple Procetron-like update rule to mitigate computational complexity carried by the gradient methods or the orthogonal projection methods [7, 8, 9]. This unified approach takes negative examples for exploration and speeds up exploitation with positive ones so as to accelerate learning iterations. In other words, this approach discourages an agent from encountering bad situations by using the clues provided. The outcomes of two simulations are conducted to demonstrate the performance of the proposed methods. The simulation results also indicate that even mistakes can be useful to enhance the knowledge basis of a learning agent.

References

[1] C. Fu, and K. Chen, "Gait Synthesis and Sensory Control of Stair Climbing for a Humanoid Robot," *IEEE Transactions on Industrial Electronics*, vol. 55, no. 5, pp. 2111-2120, 2008.

[2] K. S. Hwang, J. L. Lin, and K. H. Yeh, "Learning to Adjust and Refine Gait Patterns for a Biped Robot," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 12, pp. 1481-1490, 2015.

[3] L. Wang, Z. Liu, C. L. P. Chen, Y. Zhang, S. Lee, and X. Chen, "Energy-efficient SVM learning control system for biped walking robots," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 5, pp. 831-837, 2013.

[4] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," Robot. Autonomous Syst., vol. 57, no. 5, pp. 469-483, 2009.

[5] T. S. Li, Y.-T. Su, S.-W. Lai and J.-J. Hu, "Walking Motion Generation, Synthesis, and Control for Biped Robot by Using PGRL, LPI, and Fuzzy Logic," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 41, no. 3, pp. 736-748, 2011.

[6] H. b. Shi, X. Li, K. S. Hwang, W. P. and G. Xu, "Decoupled Visual Servoing with Fuzzy Q-Learning," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 1, pp. 241-252, 2018.

[7] J. Choi and K. E. Kim, "Hierarchical Bayesian Inverse Reinforcement Learning," *IEEE Transactions on Cybernetics*, vol. 45, no. 4, pp. 793-805, 2015.

[8] B. Michini, T. J. Walsh, A.-A. Agha-Mohammadi and J. P. How, "Bayesian nonparametric reward learning from demonstration," *IEEE Transactions on Robotic*s, vol. 31, no. 2, pp. 369-386, 2015.

[9] Y. Ng and S. Russell, "Algorithms for inverse reinforcement learning," in *proc. of 17th International Conference on Machine Learning*, pp. 663–670, 2000.

[10] P. Abbeel, A. Coates and A. Y. Ng, "Autonomous helicopter aerobatics through apprenticeship learning," *International Journal of Robotics Research*, vol. 29, pp. 1608-1639, 2010.

[11] A. Vogel, D. Ramachandran, R. Gupta and A. Raux, "Improving hybrid vehicle fuel efficiency using inverse reinforcement learning," *in Proc. of the 26th AAAI Conf. Artif. Intell.*, 2012.

[12] B. D. Ziebart, A. Maas, J. A. Bagnell and A. K. Dey, "Maximum entropy inverse reinforcement learning," in *Proc. of the 23th National Conference of Articiial Intelli- gence (AAAI)*, 2008.

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

[14] S. Zhifei and E. M. Joo, "A survey of inverse reinforcement learning techniques," *International Journal of Intelligent Computing and Cybernetics*, vol. 5, no. 3, pp. 293-311, 2012.

[15] R. E. Schapire, "A brief introduction to boosting," in p*roceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pp. 1401-1406, 1999.

[16] W. Dai, Q. Yang, G. Xue, and Y. Yu, "Boosting for transfer learning," in *Proceedings of the 24th International Conference on Machine Learning*, pp. 193–200 , 2007.

[17] C. J. C. H. Watkins, and P. Dayan, "Technical note: Q-Learning, " *Machine Learning*, 8(3-4): pp. 279-292, 1992.

[18] 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.

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

[20] U. Syed, M. Bowling and R. E. Schapire, "Apprenticeship learning using linear programming," in *Proc. of the Twenty-Fifth International Conference on Machine Learning (ICML)*, pp. 1032–1039, 2008.

[21] R. S. Sutton, and A. G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, 1998.

[22] B. Argall, B. Browning, and M. Veloso, "Learning by demonstration with critique from a human teacher," in *Proc. of International Conference on Human-Robot Interaction*, pp. 57–64, 2007.

[23] B. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robot. Auton. Syst.*, vol. 57, no. 5, pp. 469–483, 2009.

[24] C. C. Lee, "Imitation learning based on inverse reinforcement learning," *National Chung Cheng University*, *Master Thesis*, 2014.

Maxwell Hwang received the B. Med. And M Med. degree at t The School of Medicine, Zhejiang University Hangzhou, China, in 2013, 2015, respectively. He is currently working toward the Ph.D. degree at the same university.

He is an interdisciplinary scientist, and his current research interests include the applications of artificial intelligence in medical image analysis, machine learning for medical applications, in addition to the areas related to Colorectal Surgery, Colorectal Cancers, Laparoscopic Surgery.

Wei-Cheng Jiang received the B.S. degree in Computer Science and information Engineering and M.S. degree in Electro-Optical and Materials Science from National Formosa University, Yunlin, Taiwan, in 2007 and 2009, respectively. He received the Ph.D. degree in Electric Engineering from National Chung Cheng University, Chiayi, Taiwan, in 2013. He was a posdoc with the Electrical Engineering Department at National Sun Yat-sen University, Kaohsiung, Taiwan. He was a visiting scholar with Department of Electrical and Systems Engineering at Washington University in St. Louis, MO, USA. Since 2019, he is currently an Assistant Professor with the Department of Electrical Engineering, Tunghai University. His research interests include machine learning, multi-agent system, and intelligent control.

Yu-Jen Chen received the B.S. degree in electrical engineering from the Tatung Institute of Technology, Taipei, Taiwan, in 1994, and the M.S. and Ph.D. degrees in electrical engineering from National Chung Cheng University, Chia-Yi, Taiwan, in 1997 and 2009, respectively. From 2004 to 2009, he was an Adjunct Lecturer with the Center for General Education, National Chung Cheng University. Since 2010, he has been an Assistant Professor with the Electrical Engineering Department, National Chung Cheng University. His current research interests include machine learning, robotics, neural networks, and embedded systems.

Kao-Shing Hwang (M'93_SM'09) is a professor of the Department of Electrical Engineering at National Sun Yat-sen University, an adjunct professor of the Department of Healthcare Administration and Medical Informatic, Kaohsiung Medical University, Taiwan, and a visiting chair professor of the Department of Computer Engineering, Northwestern Polytech. University, Xian, China. He received the M.M.E. and Ph.D. degrees in Electrical and Computer Engineering from Northwestern University, Evanston, IL, U.S.A., in 1989 and 1993, respectively. He had been with National Chung Cheng University in Taiwan from 1993-2011. He was the deputy director of Computer Center (1998-1999), the chairman of the Electrical Engineering Department (2003~2006), and the director of the Opti-mechatronics Institute of the university (2010~2011) in National Chung Cheng University. He has been a member of IEEE since 1993 and a Fellow of the Institution of Engineering and Technology (FIET). His research interest includes methodologies and analysis for various intelligent robot systems, visual servoing, and reinforcement learning for robotic applications.

Yi-Chia Tseng received the M.S. degree in electrical engineering from the National Sun Yat-sen University, Kaohsiung, Taiwan, in 2015. She is currently an Electrical Engineer with a renowned high-tech electronics company at Hsinchu. Her current research interests include methodologies and analysis for various intelligent robot systems, machine learning, and embedded system design for robotic applications.