Neural Networks 144 (2021) 138–153

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

2021 Special Issue

Forward and inverse reinforcement learning sharing network weights and hyperparameters

Eiji Uchibe [a,*], Kenji Doya [b]

[a] *Department of Brain Robot Interface, ATR Computational Neuroscience Laboratories, 2-2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-0288, Japan*
[b] *Neural Computation Unit, Okinawa Institute of Science and Technology Graduate University, 1919-1 Tancha, Onna-son, Okinawa 904-0495, Japan*

ARTICLE INFO

Article history:
Available online 20 August 2021

Keywords:
Reinforcement learning
Inverse reinforcement learning
Imitation learning
Entropy regularization

ABSTRACT

This paper proposes model-free imitation learning named Entropy-Regularized Imitation Learning (ERIL) that minimizes the reverse Kullback–Leibler (KL) divergence. ERIL combines forward and inverse reinforcement learning (RL) under the framework of an entropy-regularized Markov decision process. An inverse RL step computes the log-ratio between two distributions by evaluating two binary discriminators. The first discriminator distinguishes the state generated by the forward RL step from the expert's state. The second discriminator, which is structured by the theory of entropy regularization, distinguishes the state–action–next-state tuples generated by the learner from the expert ones. One notable feature is that the second discriminator shares hyperparameters with the forward RL, which can be used to control the discriminator's ability. A forward RL step minimizes the reverse KL estimated by the inverse RL step. We show that minimizing the reverse KL divergence is equivalent to finding an optimal policy. Our experimental results on MuJoCo-simulated environments and vision-based reaching tasks with a robotic arm show that ERIL is more sample-efficient than the baseline methods. We apply the method to human behaviors that perform a pole-balancing task and describe how the estimated reward functions show how every subject achieves her goal.

1. Introduction

Reinforcement Learning (RL) is a computational framework for investigating the decision-making processes of both biological and artificial systems that can learn an optimal policy by interacting with an environment (Doya, 2007; Kober, Bagnell, & Peters, 2013; Sutton & Barto, 1998). Modern RL algorithms have achieved remarkable performance in playing Atari games (Mnih et al., 2015), Go (Silver et al., 2017), Dota 2 (OpenAI, Berner et al., 2019), and Starcraft II (Vinyals et al., 2019). They have also been successfully applied to dexterous manipulation tasks (OpenAI, Akkaya et al., 2019), folding a T-shirt (Tsurumine, Cui, Uchibe, & Matsubara, 2019), quadruped locomotion (Haarnoja, Zhou, Hartikainen et al., 2018), the optimal state feedback control of non-affine nonlinear systems (Wang & Qiao, 2019), and non-zero-sum game output regulation problems (Odekunle, Gao, & Jiang, 2020). However, one critical open question in RL is designing and preparing an appropriate reward function for a given task. Although it is easy to design a sparse reward function that gives a positive reward when a task is accomplished and zero otherwise, such an approach complicates finding an optimal policy due to prohibitive learning times. On the other hand, we can accelerate the learning speed with a complicated function that generally gives a non-zero reward signal. However, optimized behaviors often deviate from an experimenter's intention if the reward function is too complicated (Doya & Uchibe, 2005).

In some situations, it is easier to prepare examples of a desired behavior provided by an expert than handcrafting an appropriate reward function. Behavior Cloning (BC) is a straightforward approach in imitation learning formulated as supervised learning. BC minimizes the forward Kullback–Leibler (KL) divergence, which is known as moment projection (M-projection). Forward KL is the expectation of the log-likelihood ratio under expert distribution. Although BC requires no interaction with the environment, it suffers from a state covariate shift problem: small errors in actions introduce the learner to unseen states that are not included in the training dataset (Ross, Gordon, & Bagnell, 2011). Also, BC shows poor performance if the model is misspecified because minimizing the forward KL has a mode-covering property. To overcome the covariate shift problem, several inverse RL (Ng & Russell, 2000) and apprenticeship learning (Abbeel & Ng, 2004) methods have been proposed to retrieve a reward function from expert behaviors and implement imitation learning. Currently available applications include a probabilistic driver route prediction system (Liu et al., 2013;

[*] Corresponding author.
E-mail addresses: uchibe@atr.jp (E. Uchibe), doya@oist.jp (K. Doya).

https://doi.org/10.1016/j.neunet.2021.08.017

Vogel, Ramachandran, Gupta, & Raux, 2012), modeling risk anticipation and defensive driving (Shimosaka, Kaneko, & Nishi, 2014), investigating human behaviors in table tennis (Muelling, Boularias, Mohler, Schölkopf, & Peters, 2014), robot navigation tasks (Kretzschmar, Spies, Sprunk, & Burgard, 2016; Xia & El Kamel, 2016), analyzing animal behaviors (Ashida, Kato, Hotta, & Oka, 2019; Hirakawa et al., 2018; Yamaguchi et al., 2018), and parser training (Neu & Szepesvári, 2009). A recent functional magnetic resonance imaging (fMRI) study suggests that the anterior part of the dorsomedial prefrontal cortex (dmPFC) is likely to encode the inverse reinforcement learning algorithm (Collette, Pauli, Bossaerts, & O'Doherty, 2017). Combining inverse RL with a standard RL is a promising approach to find an optimal policy from expert demonstrations. Hereafter, we use the term "forward" reinforcement learning to clarify the difference.

Recently, some works (Fu, Luo, & Levine, 2018; Ho & Ermon, 2016) have connected forward and inverse RL and Generative Adversarial Networks (GANs) (Goodfellow et al., 2014), which exhibited remarkable success in image generation, video prediction, and machine translation domains. In this view, inverse RL is interpreted as a GAN discriminator whose goal is to determine whether experiences are drawn from an expert or generated by a forward RL step. The GAN generator is implemented by a forward RL step and creates experiences that are indistinguishable by an inverse RL. Generative Adversarial Imitation Learning (GAIL) (Ho & Ermon, 2016) showed that the iterative process of forward and inverse RL produces policies that outperformed BC. However, GAIL minimizes the Jensen–Shannon divergence, which has a similar forward KL property. In addition, GAIL is sample-inefficient because an on-policy RL algorithm used in the forward RL step simply trains a policy from a reward calculated by the inverse RL step. To improve the sample efficiency in the forward RL step, Jena, Liu, and Sycara (2020) added BC loss to the loss of the GAIL generator. Kinose and Taniguchi (2020) integrated the GAIL discriminator with reinforcement learning, in which the policy is trained with both the original reward and additional rewards calculated by the discriminator. However, their approaches remain sample-inefficient. Utilizing the result of the inverse RL step to the forward RL step and vice versa is difficult because the discriminator's structure is designed independently from the generator.

To further improve the sample efficiency, this paper proposes a model-free imitation learning algorithm named Entropy-Regularized Imitation Learning (ERIL), which minimizes the information projection or the I-projection, which is also known as the reverse KL divergence between two probability distributions induced by a learner and an expert. A reverse KL, which is the expectation of the log-likelihood ratio under the learner's distribution, has a mode-seeking property that focuses on the distribution mode that the policy can represent. A reverse KL is more appropriate than a forward KL when the policy is misspecified. Unfortunately, reverse KL divergence cannot be computed because the expert distribution is unknown. Our idea applies the density ratio trick (Sugiyama, Suzuki, & Kanamori, 2012) to evaluate the log-ratio between two distributions from samples drawn from them. In addition, we exploit the framework of the entropy-regularized Markov Decision Process (MDP), where the reward function is augmented by the differential entropy of a learner's policy and the KL divergence between the learner and expert policies. Consequently, the log-ratio can be computed by training two binary discriminators. One is a state-only discriminator, which distinguishes the state generated by the learner from the expert's state. The second discriminator, which is a function of a tuple of a state, an action, and the next state, also distinguishes between the experiences of learners and experts. The second discriminator is represented by reward, a state value function, and the log-ratio of the first discriminator. We show that Adversarial Inverse Reinforcement Learning (AIRL) (Fu et al., 2018) and Logistic Regression-based Inverse RL (LogReg-IRL) (Uchibe, 2018; Uchibe & Doya, 2014) discriminators are a special case of an ERIL. The loss function is essentially identical as that of GAN for training discriminators, which are efficiently trained by logistic regression.

After evaluating the log-ratio, the forward RL in ERIL minimizes the estimated I-projection. We show that its minimization is equivalent to maximizing the entropy-regularized reward. Consequently, the forward RL algorithm is implemented by off-policy reinforcement learning that resembles Dynamic Policy Programming (Azar, Gómez, & Kappen, 2012), Soft Actor–Critic (SAC) (Haarnoja, Zhou, Abbeel, & Levine, 2018), and conservative value iteration (Kozuno, Uchibe, & Doya, 2019). In the forward RL step, the state value, the state–action value, and the stochastic policy are trained by the actor–critic algorithm, and the reward function estimated by the inverse RL step is fixed. This step allows the learner to generalize the expert policy to unseen states that are not included in the demonstrations.

We experimented with the MuJoCo benchmark tasks (Todorov, Erez, & Tassa, 2012) in the OpenAI gym (Brockman et al., 2016). Our experimental results demonstrate that ERIL resembles some modern imitation learning algorithms in terms of the number of trajectories from expert data and outperformed sample efficiency in terms of the number of trajectories in the forward RL step. Ablation studies show that entropy regularization plays a critical role in improving sample efficiency. Next we conducted a vision-based target-reaching task with a manipulator in three-dimensional space and demonstrated that using two discriminators is vital when the learner's initial state distribution differs from the expert one. Then we applied ERIL to human behaviors for performing a pole-balancing task. Since the actions of human subjects are not observable, the task is an example of realistic situations. ERIL recovers the subjects' policies better than the baselines. We also showed that the estimated reward functions show how every subject achieved her goal.

The following are the main contributions of our paper: (1) We proposed a structured discriminator with hyperparameters derived from entropy-regularized reinforcement learning. (2) The hyperparameters, which are shared between forward and inverse RL, can be used to control the discriminator's ability. (3) The state value function is trained by both forward and inverse RL, which improves the sample efficiency in terms of the number of environmental interactions.

2. Related work

2.1. Regularization in RL

The role of regularization is to prevent overfitting and to encourage generalization. Many regularization methods have been proposed from various aspects, such as dropout (Liu, Li, Kang, & Darrell, 2021), temporal difference error (Parisi, Tangkaratt, Peters, & Khan, 2019), value function difference (Ohnishi et al., 2019), and manifold regularization for feature representation learning (Li, Liu, & Wang, 2018). Amit, Meir, and Ciosek (2020) investigated the property of a discount factor as a regularizer.

The most widely used regularization method is entropy regularization (Belousov & Peters, 2019; Ziebart, Maas, Bagnell, & Dey, 2008). It supports exploration by favoring more stochastic policies (Haarnoja, Zhou, Abbeel et al., 2018) and smoothens the optimization landscape (Ahmed, Le Roux, & Schuurmans, 2019). The advantage of entropy regularization in inverse RL is its robustness against noisy and stochastic demonstrations because an optimal policy becomes stochastic.

2.2. Behavior cloning

BC directly maximizes the log-likelihood of an expert action. Pomerleau (1989) achieved an Autonomous Land Vehicle In a Neural Network (ALVINN), which is a potential precursor of autonomous driving cars. ALVINN's policy is implemented by a 3-layer neural network and learns mappings from video and range finder inputs to steering directions by supervised learning. However, ALVINN suffers from the covariate shift problem. To reduce covariate shift, Ross et al. (2011) proposed an iterative method called Dataset Aggregation (DAGGER), where the learner runs its policy while the expert provides the correct action for the states visited by the learner. Laskey, Lee, Fox, Dragan, and Goldberg (2017) proposed Disturbances for Augmented Robot Trajectories (DART), which collects expert datasets with injected noise.

It is often difficult to provide such expert data as state–action pairs in such realistic situations as analyzing animal behaviors and learning from videos. Torabi, Warnell, and Stone (2018) studied a situation in which expert action is unavailable and proposed Behavioral Cloning from Observation (BCO) that estimates actions from an inverse dynamics model. Soft Q Imitation Learning (SQIL) (Reddy, Dragan, & Levine, 2020) is a BC with a regularization term that penalizes large squared soft Bellman error. SQIL is implemented by SAC, which assigns the reward of the expert data to 1 and the generated data to 0. SQIL can learn from both the expert and learner's samples because SAC is an off-policy algorithm.

2.3. Generative adversarial imitation learning

GAIL (Ho & Ermon, 2016), which is a very popular imitation learning algorithm, formulates the objectives of imitation learning as GAN training objectives. Its discriminator differentiates between generated state–action pairs and those of experts, and the generator acts as a forward reinforcement learning algorithm to maximize the sum of rewards computed by the discriminator. There are several extensions of GAN. AIRL (Fu et al., 2018) used an optimal discriminator whose expert distribution was approximated by a disentangled reward function. A similar, independently proposed discriminator (Uchibe, 2018; Uchibe & Doya, 2014) was derived from the framework of entropy-regularized reinforcement learning and density ratio estimation. AIRL experimentally showed that the learned reward function can be transferred to new, unseen environments. Situated GAIL (Kobayashi, Horii, Iwaki, Nagai, & Asada, 2019) extended GAIL to learn multiple reward functions and multiple policies by introducing a task variable to both the discriminator and the generator.

As discussed in the previous section, expert action is not always available. IRLGAN (Henderson, Chang et al., 2018) is a special GAN case where the discriminator is given as a state function. Torabi, Warnell, and Stone (2019) proposed Generative Adversarial Imitation from Observation (GAIfO) whose functions are characterized by state transitions. Sun and Ma (2014) proposed Action Guided Adversarial Imitation Learning (AGAIL) that can deal with expert demonstrations with incomplete action sequences. AGAIL uses mutual information between expert and generated actions as an additional regularizer for training objectives.

To reduce the number of interactions in the forward RL step, Blondé and Kalousis (2019) proposed Sample-efficient Adversarial Mimic (SAM), which adopts an off-policy method called the Deep Deterministic Policy Gradients (DDPG) algorithm (Lillicrap et al., 2016). SAM maintains three different neural networks, which approximate a reward function, a state–action value function, and a policy. The reward function is estimated in the same way as in GAN, and the state–action value function and the policy are trained by DDPG. Kostrikov, Agrawal, Dwibedi, Levine, and Tompson (2019) showed that GAIL's reward function is biased and that the absorbing states are not treated appropriately. They proposed a preprocessing technique for expert data before learning and developed the Discriminator Actor–Critic (DAC) algorithm. DAC utilizes the Twin Delayed Deep Deterministic policy gradient (TD3) algorithm (Fujimoto, van Hoof, & Meger, 2018), which is an extension of DDPG. Sasaki, Yohira, and Kawaguchi (2019) exploited the Bellman equation to represent a reward function, and a reward's exponential transformation is trained as a kind of discriminator. They improved the policy using an off-policy actor–critic (Degris, White, & Sutton, 2012). Zuo, Chen, Lu, and Huang (2020) proposed a Deterministic GAIL that adopts the modified DDPG algorithm that incorporates the behavior cloning loss in the forward RL step. Discriminator Soft Actor–Critic (Nishio, Kuyoshi, Tsuneda, & Yamane, 2020) extended SQIL by estimating the reward function by an AIRL-like discriminator. Ghasemipour, Zemel, and Gu (2019) and Ke et al. (2020) described the relationship between several imitation learning algorithms from the viewpoint of objective functions. Since our study focuses on sample efficiency with respect to the number of interactions with the environment, the most related works are SAM, DAC, and Sasaki's method. We compared our method with these three methods in the experiments.

3. Preliminaries

3.1. Markov decision process

Here we briefly introduce MDP for a discrete-time domain. Let $\mathcal{X}$ and $\mathcal{U}$ be continuous or discrete state and action spaces. At time step t, a learning agent observes environmental current state $\boldsymbol{x}_t \in \mathcal{X}$ and executes action $\boldsymbol{u}_t \in \mathcal{U}$ sampled from stochastic policy $\pi(\boldsymbol{u}_t \mid \boldsymbol{x}_t)$. Consequently, the learning agent receives from the environment immediate reward $\tilde{r}(\boldsymbol{x}_t, \boldsymbol{u}_t)$, which is an arbitrary bounded function that evaluates the goodness of action $\boldsymbol{u}_t$ at state $\boldsymbol{x}_t$. The environment shifts to next state $\boldsymbol{x}'_t = \boldsymbol{x}_{t+1} \in \mathcal{X}$ according to state transition probability $p_T(\boldsymbol{x}'_t \mid \boldsymbol{x}_t, \boldsymbol{u}_t)$.

Forward reinforcement learning's goal is to construct optimal policy $\pi(\boldsymbol{u} \mid \boldsymbol{x})$ that maximizes the given objective function. Among several available objective functions, the most widely used is a discounted sum of rewards:

$$V(\boldsymbol{x}) \triangleq \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \tilde{r}(\boldsymbol{x}_t, \boldsymbol{u}_t) \mid \boldsymbol{x}_0 = \boldsymbol{x}\right],$$

where $\gamma \in [0, 1)$ is called the discount factor. The optimal state value function for the discounted reward setting satisfies the following Bellman optimality equation:

$$V(\boldsymbol{x}) = \max_{\boldsymbol{u}} \left[\tilde{r}(\boldsymbol{x}, \boldsymbol{u}) + \gamma \mathbb{E}_{\boldsymbol{x}' \sim p_T(\cdot \mid \boldsymbol{x}, \boldsymbol{u})}\left[V(\boldsymbol{x}')\right]\right], \tag{1}$$

where $\mathbb{E}_p[\cdot]$ hereafter denotes the expectation with respect to p. The state–action value function for the discounted reward setting is also defined:

$$Q(\boldsymbol{x}, \boldsymbol{u}) = \tilde{r}(\boldsymbol{x}, \boldsymbol{u}) + \gamma \mathbb{E}_{\boldsymbol{x}' \sim p_T(\cdot \mid \boldsymbol{x}, \boldsymbol{u})}\left[V(\boldsymbol{x}')\right].$$

Eq. (1), which is nonlinear due to the max operator, usually struggles to find an action that maximizes its right hand side.

3.2. Entropy-regularized Markov decision process

Next we consider entropy-regularized MDP (Azar et al., 2012; Haarnoja, Zhou, Abbeel et al., 2018; Kozuno et al., 2019; Ziebart et al., 2008), in which the reward function is regularized by the following form:

$$\tilde{r}(\boldsymbol{x}, \boldsymbol{u}) = r(\boldsymbol{x}, \boldsymbol{u}) + \frac{1}{\kappa}\mathcal{H}(\pi(\cdot \mid \boldsymbol{x})) - \frac{1}{\eta}\mathrm{KL}(\pi(\cdot \mid \boldsymbol{x}) \parallel b(\cdot \mid \boldsymbol{x})), \tag{2}$$

where $r(\boldsymbol{x}, \boldsymbol{u})$ is a standard reward function that is unknown in the inverse RL setting. κ and η are the positive hyperparameters determined by the experimenter, $\mathcal{H}(\pi(\cdot \mid \boldsymbol{x}))$ is the (differential) entropy of policy $\pi(\boldsymbol{u} \mid \boldsymbol{x})$, and $\mathrm{KL}(\pi(\cdot \mid \boldsymbol{x}) \parallel b(\cdot \mid \boldsymbol{x}))$ is the relative entropy, which is also known as the Kullback–Leibler (KL) divergence between $\pi(\boldsymbol{u} \mid \boldsymbol{x})$ and baseline policy $b(\boldsymbol{u} \mid \boldsymbol{x})$. When the reward function is regularized by the entropy functions (2), we can analytically maximize the right hand side of Eq. (1) by a method using Lagrange multipliers. Consequently, the optimal state value function can be represented:

$$V(\boldsymbol{x}) = \frac{1}{\beta} \ln \int \exp\left(\beta Q(\boldsymbol{x}, \boldsymbol{u})\right) \mathrm{d}\boldsymbol{u}, \tag{3}$$

where β is a positive hyperparameter defined by

$$\beta \triangleq \frac{\kappa\eta}{\kappa + \eta},$$

and $Q(\boldsymbol{x}, \boldsymbol{u})$ is the optimal soft state–action value function:

$$Q(\boldsymbol{x}, \boldsymbol{u}) = r(\boldsymbol{x}, \boldsymbol{u}) + \frac{1}{\eta} \ln b(\boldsymbol{u} \mid \boldsymbol{x}) + \gamma \mathbb{E}_{\boldsymbol{x}' \sim p_T(\cdot \mid \boldsymbol{x}, \boldsymbol{u})}\left[V(\boldsymbol{x}')\right]. \tag{4}$$

When the action is discrete, the right hand side of Eq. (3) is a log-sum-exp function, also known as a softmax function. The corresponding optimal policy is given:

$$\pi(\boldsymbol{u} \mid \boldsymbol{x}) = \frac{\exp(\beta Q(\boldsymbol{x}, \boldsymbol{u}))}{\exp(\beta V(\boldsymbol{x}))}, \tag{5}$$

where $\exp(\beta V(\boldsymbol{x}))$ represents a normalizing constant of $\pi(\boldsymbol{u} \mid \boldsymbol{x})$.

For later reference, the update rules are described here. The soft state–action value function is trained to minimize the soft Bellman residual. The soft state value function is trained to minimize the squared residual error derived from Eq. (5). The policy is improved by directly minimizing the expected KL divergence in Eq. (5):

$$J_{\mathrm{ER}}(\boldsymbol{w}_\pi) = \mathbb{E}_{\boldsymbol{x} \sim p(\boldsymbol{x})}\left[\mathrm{KL}\left(\pi(\cdot \mid \boldsymbol{x}) \parallel \frac{\exp(\beta Q(\boldsymbol{x}, \cdot))}{\exp(\beta V(\boldsymbol{x}))}\right)\right].$$

The derivative of the KL divergence is given by

$$\nabla \mathrm{KL} = \mathbb{E}_{\boldsymbol{u} \sim \pi(\cdot \mid \boldsymbol{x})}\left[\nabla_{\boldsymbol{w}_\pi} \ln \pi(\cdot \mid \boldsymbol{x}) \left[\ln \pi(\cdot \mid \boldsymbol{x}) - \beta(Q(\boldsymbol{x}, \cdot) - V(\boldsymbol{x})) + B(\boldsymbol{x})\right]\right], \tag{6}$$

where $B(\boldsymbol{x})$ is a baseline function that does not change the gradient (Peters & Schaal, 2008) and is often used to reduce the variance of the gradient estimation.

3.3. Generative adversarial networks

GANs are a class of neural networks that can approximate probability distribution based on a game theoretic scenario (Goodfellow et al., 2014). Standard GANs consist of a generator and a discriminator. Suppose that $p^E(\boldsymbol{z})$ and $p^L(\boldsymbol{z})$ denote the probability distributions over data $\boldsymbol{z}$ of the expert and a generator. A discriminator, which is a function that distinguishes samples from a generator and an expert, is denoted by $D(\boldsymbol{z})$ and minimizes the following negative log-likelihood (NLL):

$$J_{\mathrm{GAN}}^{(D)} = -\mathbb{E}_{\boldsymbol{z} \sim p^L}[\ln D(\boldsymbol{z})] - \mathbb{E}_{\boldsymbol{z} \sim p^E}[\ln(1 - D(\boldsymbol{z}))]. \tag{7}$$

The optimal discriminator has the following shape (Goodfellow et al., 2014):

$$D^*(\boldsymbol{z}) = \frac{p^L(\boldsymbol{z})}{p^L(\boldsymbol{z}) + p^E(\boldsymbol{z})}. \tag{8}$$

The generator minimizes $-J_{\mathrm{GAN}}^{(D)}$:

$$J_{\mathrm{GAN}}^{(G)} = \mathbb{E}_{\boldsymbol{z} \sim p^L}[\ln D(\boldsymbol{z})], \tag{9}$$

where the second term on the right hand side of Eq. (7) is removed because it is constant with respect to the generator. Recently, Prescribed GAN (PresGAN) introduced an entropy regularization term to J_{GAN} to mitigate mode collapse (Dieng, Ruiz, Blei, & Titsias, 2019).

GAIL (Ho & Ermon, 2016) is an extension of GANs for imitation learning, and its objective function is given by

$$J_{\mathrm{GAIL}}(\boldsymbol{w}_G, \boldsymbol{w}_D) = -\mathbb{E}_{(\boldsymbol{x}, \boldsymbol{u}) \sim \pi^L}[\ln D(\boldsymbol{x}, \boldsymbol{u})] - \mathbb{E}_{(\boldsymbol{x}, \boldsymbol{u}) \sim \pi^E}[\ln(1 - D(\boldsymbol{x}, \boldsymbol{u}))] + \lambda_{\mathrm{GAIL}} \mathcal{H}(\pi^L),$$

where λ_{GAIL} is a positive hyperparameter. Adding the entropy term is key for an association with the entropy-regularized MDP. The objective function of the GAIL discriminator is essentially identical as that of the GAN discriminator, which updates its policy by Trust Region Policy Optimization (TRPO) (Schaul, Horgan, Gregor, & Silver, 2015), where the reward function is defined by

$$r_{\mathrm{GAIL}} = -\ln D(\boldsymbol{x}, \boldsymbol{u}).$$

AIRL (Fu et al., 2018) adopts a special structure for the discriminator:

$$D(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}') = \frac{\pi^L(\boldsymbol{u} \mid \boldsymbol{x})}{\exp(f(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}')) + \pi^L(\boldsymbol{u} \mid \boldsymbol{x})}, \tag{10}$$

where $f(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}')$ is defined using two state-dependent functions, $g(\boldsymbol{x})$ and $h(\boldsymbol{x})$:

$$f(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}') \triangleq g(\boldsymbol{x}) + \gamma h(\boldsymbol{x}') - h(\boldsymbol{x}). \tag{11}$$

Note that the AIRL discriminator (10) has no hyperparameters. A similar discriminator was proposed (Uchibe, 2018; Uchibe & Doya, 2014). AIRL's reward function is calculated:

$$\begin{aligned} r_{\mathrm{AIRL}} &= -\ln D(\boldsymbol{x}, \boldsymbol{u}) + \ln(1 - D(\boldsymbol{x}, \boldsymbol{u})) \\ &= f(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}') - \ln \pi(\boldsymbol{u} \mid \boldsymbol{x}). \end{aligned} \tag{12}$$

Note that $-\ln D$ and $\ln(1 - D)$ are monotonically related.

3.4. Behavior cloning

BC is the most widely used type of imitation learning. It minimizes the "forward" KL divergence, which is also known as M-projection (Ghasemipour et al., 2019; Ke et al., 2020):

$$J_{\mathrm{BC}}(\boldsymbol{w}_\pi) = \mathrm{KL}(\pi^E \parallel \pi^L) = -\mathbb{E}_{\pi^E}\left[\ln \pi^L(\boldsymbol{u} \mid \boldsymbol{x})\right] + C, \tag{13}$$

where $\boldsymbol{w}_\pi$ is the policy parameter vector of $\pi^L(\boldsymbol{u} \mid \boldsymbol{x})$ and C is a constant with respect to the policy parameter. Since BC does not need to interact with the environment, finding a policy that minimizes Eq. (13) is simply achieved by supervised learning. Unfortunately, it suffers from covariate shift between the learner and the expert (Ross et al., 2011).

4. Entropy-Regularized Imitation Learning

4.1. Objective function

To formally present our approach, we denote the expert's policy as $\pi^E(\boldsymbol{u} \mid \boldsymbol{x})$ and the learner's policy as $\pi^L(\boldsymbol{u} \mid \boldsymbol{x})$. Suppose a dataset of transitions generated by

$$\mathcal{D}^E = \{(\boldsymbol{x}_i, \boldsymbol{u}_i, \boldsymbol{x}'_i)\}_{i=1}^{N^E}, \quad \boldsymbol{u}_i \sim \pi^E(\cdot \mid \boldsymbol{x}_i), \quad \boldsymbol{x}'_i \sim p_T(\cdot \mid \boldsymbol{x}_i, \boldsymbol{u}_i),$$

where N^E is the number of transitions in the dataset. We consider two joint probability density functions, $\pi^E(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}')$ and $\pi^L(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}')$, where under a Markovian assumption, $\pi^E(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}')$ is decomposed by

$$\pi^E(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}') = p_T(\boldsymbol{x}' \mid \boldsymbol{x}, \boldsymbol{u}) \pi^E(\boldsymbol{u} \mid \boldsymbol{x}) \pi^E(\boldsymbol{x}), \tag{14}$$

and $\pi^L(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}')$ can be decomposed in the same way.

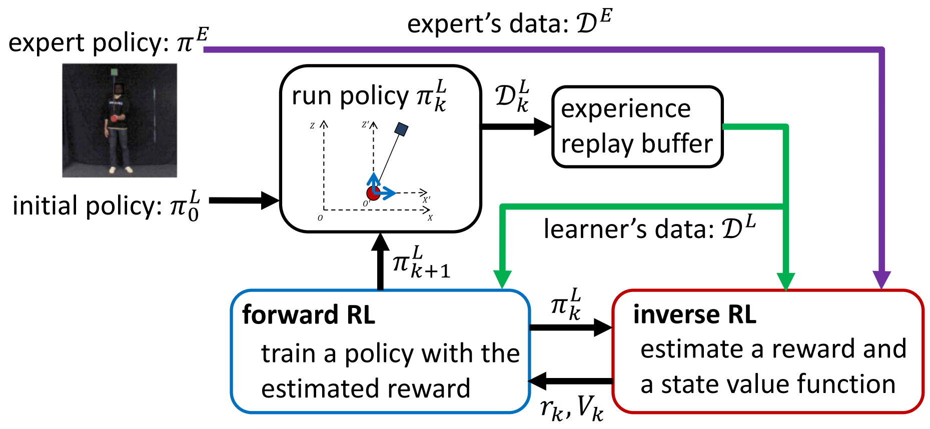

expert policy: π^E
expert's data: $\mathcal{D}^E$
run policy π_k^L
$\mathcal{D}_k^L$
experience replay buffer
initial policy: π_0^L
learner's data: $\mathcal{D}^L$
π_{k+1}^L
π_k^L
forward RL
train a policy with the estimated reward
inverse RL
estimate a reward and a state value function
r_k, V_k

Fig. 1. Overall architecture of Entropy-Regularized Imitation Learning (ERIL).

ERIL minimizes the "reverse" KL divergence given by

$$J_{\mathrm{ERIL}}(\boldsymbol{w}_\pi) = \mathrm{KL}(\pi^L \parallel \pi^E), \tag{15}$$

where the reverse KL divergence is often called information projection (I-projection), defined by

$$\mathrm{KL}(\pi^L \parallel \pi^E) = \mathbb{E}_{\pi^L}\left[\ln \frac{\pi^L(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}')}{\pi^E(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}')}\right].$$

The difficulty is how to evaluate the log-ratio, $\ln \pi^L(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}')/\pi^E(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}')$, because $\pi^E(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}')$ is unknown. The basic idea for resolving the problem is to adopt the density ratio trick (Sugiyama et al., 2012), which can be efficiently achieved by solving binary classification tasks. To derive the algorithm, we assume that the expert reward function is given by

$$\tilde{r}(\boldsymbol{x}, \boldsymbol{u}) = r_k(\boldsymbol{x}) + \frac{1}{\kappa}\mathcal{H}(\pi^E(\cdot \mid \boldsymbol{x})) - \frac{1}{\eta}\mathrm{KL}(\pi^E(\cdot \mid \boldsymbol{x}) \parallel \pi_k^L(\cdot \mid \boldsymbol{x})), \tag{16}$$

where k is the iteration index. $r_k(\boldsymbol{x})$ is a state-only reward function parameterized by $\boldsymbol{w}_r$. Although using a state–action reward function is possible, the learned reward function will be a shaped advantage function, and transferability to a new environment is restricted (Fu et al., 2018). When the expert reward function is given by Eq. (16), the expert state–action value function satisfies the following soft Bellman optimality equation:

$$Q_k(\boldsymbol{x}, \boldsymbol{u}) = r_k(\boldsymbol{x}) + \frac{1}{\eta}\ln \pi_k^L(\boldsymbol{u} \mid \boldsymbol{x}) + \gamma \mathbb{E}_{\boldsymbol{x}' \sim p_T(\cdot \mid \boldsymbol{x}, \boldsymbol{u})}\left[V_k(\boldsymbol{x}')\right], \tag{17}$$

where $V_k(\boldsymbol{x}')$ is the corresponding state value function at the kth iteration. Fig. 1 shows ERIL's overall architecture. The inverse RL step estimates reward r_k and state value function V_k from the expert's and learner's datasets. The forward RL step improves learner's policy π_k^L based on r_k and V_k. The state value function and the expert policy are expressed by Eq. (3) and Eq. (5).

4.2. Inverse reinforcement learning based on density ratio estimation

Arranging Eqs. (5) and (17) yields the Bellman optimality equation in a different form:

$$\frac{1}{\beta}\ln \frac{\pi^E(\boldsymbol{u} \mid \boldsymbol{x})}{\pi_k^L(\boldsymbol{u} \mid \boldsymbol{x})} = r_k(\boldsymbol{x}) - \frac{1}{\kappa}\ln \pi_k^L(\boldsymbol{u} \mid \boldsymbol{x}) + \gamma \mathbb{E}_{\boldsymbol{x}' \sim p_T(\cdot \mid \boldsymbol{x}, \boldsymbol{u})}\left[V_k(\boldsymbol{x}')\right] - V_k(\boldsymbol{x}). \tag{18}$$

With the density ratio trick (Sugiyama et al., 2012), Eq. (18) is re-written:

$$\frac{1}{\beta}\ln \frac{D_k^{(2)}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}')}{1 - D_k^{(2)}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}')} = \frac{1}{\beta}\ln \frac{D_k^{(1)}(\boldsymbol{x})}{1 - D_k^{(1)}(\boldsymbol{x})} - r_k(\boldsymbol{x}) + \frac{1}{\kappa}\ln \pi_k^L(\boldsymbol{u} \mid \boldsymbol{x}) - \gamma V_k(\boldsymbol{x}') + V_k(\boldsymbol{x}), \tag{19}$$

where $D_k^{(1)}(\boldsymbol{x})$ and $D_k^{(2)}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}')$ denote a discriminator that classifies the expert data from those of the generator. See Appendix A.1 for the derivation. Suppose that $\ln D_k^{(1)}(\boldsymbol{x})/(1 - D_k^{(1)}(\boldsymbol{x}))$ is approximated by $g_k(\boldsymbol{x})$ parameterized by $\boldsymbol{w}_g$. Then the first discriminator is constructed:

$$D_k^{(1)}(\boldsymbol{x}) = \frac{1}{1 + \exp(-g_k(\boldsymbol{x}))},$$

and $\boldsymbol{w}_g$ is obtained by logistic regression. In the same way, Eq. (19) is used to design the second discriminator:

$$D_k^{(2)}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}') = \frac{\exp(\beta\kappa^{-1}\ln \pi_k^L(\boldsymbol{u} \mid \boldsymbol{x}))}{\exp(\beta f_k(\boldsymbol{x}, \boldsymbol{x}')) + \exp(\beta\kappa^{-1}\ln \pi_k^L(\boldsymbol{u} \mid \boldsymbol{x}))}, \tag{20}$$

where

$$f_k(\boldsymbol{x}, \boldsymbol{x}') \triangleq r_k(\boldsymbol{x}) - \beta^{-1}g_k(\boldsymbol{x}) + \gamma V_k(\boldsymbol{x}') - V_k(\boldsymbol{x}).$$

Note that $V(\boldsymbol{x}) = 0$ is required for absorbing states $\boldsymbol{x} \in \mathcal{X}$ so that the process can continue indefinitely without incurring extra rewards. If $\boldsymbol{x}'$ is an absorbing state, then $f_k(\boldsymbol{x}, \boldsymbol{x}') = r_k(\boldsymbol{x}) + g_k(\boldsymbol{x}) - V_k(\boldsymbol{x})$. When $V_k(\boldsymbol{x})$ is parameterized by $\boldsymbol{w}_V$, the parameters of $D^{(2)}$ are $\boldsymbol{w}_r$ and $\boldsymbol{w}_V$ because $\boldsymbol{w}_g$ and the policy are fixed during training. Fig. 2 shows the architecture of the ERIL discriminator. The AIRL discriminator (10) is a special case of Eq. (20) that sets $g_k(\boldsymbol{x}) = 0$ and $\beta\kappa^{-1} = 1$, where LogReg-IRL (Uchibe, 2018) is obtained by $\kappa \to \infty$. Note that the discriminator (20) remains unchanged even if $r_k(\boldsymbol{x})$ and $V_k(\boldsymbol{x})$ are modified by $r_k(\boldsymbol{x}) + C$ and $V_k(\boldsymbol{x}) + C/(1-\gamma)$, where C is a constant value. Therefore, we can recover them up to the constant.

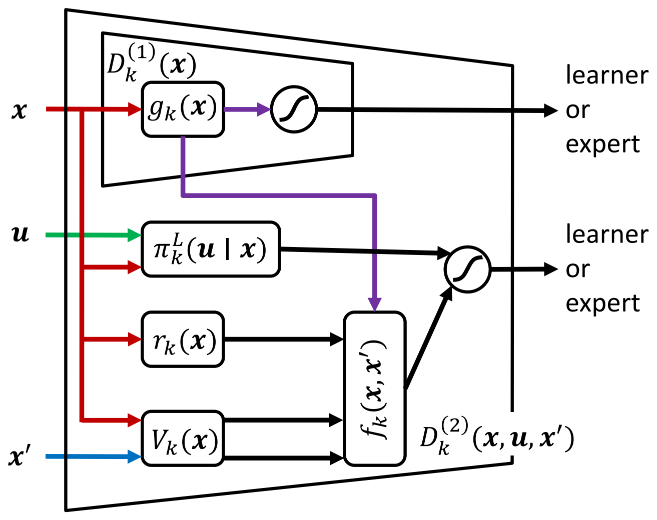
Our inverse RL step consists of two parts. First, $g_k(\boldsymbol{x})$ is evaluated by maximizing the following log-likelihood:

$$J_D^{(1)}(\boldsymbol{w}_g) = \mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}_k^L}\left[\ln D_k^{(1)}(\boldsymbol{x})\right] + \mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}^E}\left[\ln\left(1 - D_k^{(1)}(\boldsymbol{x})\right)\right], \tag{21}$$

where $\boldsymbol{x} \sim \mathcal{D}$ denotes that the transition data are drawn from $\mathcal{D}$ and $\mathcal{D}_k^L$ is the dataset of the transitions provided by learner's policy $\pi_k^L(\boldsymbol{u} \mid \boldsymbol{x})$ at the kth iteration. $D_k^{(2)}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}')$ is trained in the same way, in which $g_k(\boldsymbol{x})$ and $\pi_k^L(\boldsymbol{u} \mid \boldsymbol{x})$ are fixed during training. The goal of $D_k^{(2)}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}')$ is to maximize the following log-likelihood:

$$J_D^{(2)}(\boldsymbol{w}_r, \boldsymbol{w}_V) = \mathbb{E}_{(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}') \sim \mathcal{D}_k^L}\left[\ln D_k^{(2)}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}')\right] + \mathbb{E}_{(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}') \sim \mathcal{D}^E}\left[\ln\left(1 - D_k^{(2)}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}')\right)\right]. \tag{22}$$

This simply minimizes the cross entropy function of binary classifier $D_k^{(2)}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}')$ that separates the generated data from those



$D_k^{(1)}(\boldsymbol{x})$
$\boldsymbol{x}$
$g_k(\boldsymbol{x})$
learner or expert
$\boldsymbol{u}$
$\pi_k^L(\boldsymbol{u} \mid \boldsymbol{x})$
learner or expert
$r_k(\boldsymbol{x})$
$f_k(\boldsymbol{x}, \boldsymbol{x}')$
$\boldsymbol{x}'$
$V_k(\boldsymbol{x})$
$D_k^{(2)}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}')$

Fig. 2. Architecture of two ERIL discriminators: $D_k^{(1)}(\boldsymbol{x})$ and $D_k^{(2)}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}')$.

of the expert. The formulation of the inverse RL algorithm in ERIL is identical to that of the GAN discriminator. Note that β is not estimated as an independent parameter.

In practice, $\mathcal{D}_k^L$ is replaced with $\mathcal{D}^L = \cup_k \mathcal{D}_k^L$, which means that the discriminators are trained with all the generated transitions by the learner. Theoretically, we have to use importance sampling to correctly evaluate the expectation, but Kostrikov et al. (2019) showed that it works well without it.

4.3. Forward reinforcement learning based on KL minimization

After estimating the log-ratio by the inverse RL described in Section 4.2, we minimize the reverse KL divergence (15). Then we rewrite Eq. (19) using Eq. (17) and obtain the following equation:

$$\ln \frac{D_k^{(2)}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}')}{1 - D_k^{(2)}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}')} = \ln \pi_k^L(\boldsymbol{u} \mid \boldsymbol{x}) - \beta\left(Q_k(\boldsymbol{x}, \boldsymbol{u}) - V_k(\boldsymbol{x})\right) + g_k(\boldsymbol{x}). \tag{23}$$

ERIL's objective function is expressed by

$$J_{\mathrm{ERIL}}(\boldsymbol{w}_\pi) = \mathbb{E}_{\pi^L}\left[\ln \pi_k^L(\boldsymbol{u} \mid \boldsymbol{x}) - \beta\left(Q_k(\boldsymbol{x}, \boldsymbol{u}) - V_k(\boldsymbol{x})\right) + g_k(\boldsymbol{x})\right] \tag{24}$$

and its derivative by

$$\nabla J_{\mathrm{ERIL}}(\boldsymbol{w}_\pi) = \mathbb{E}_{\pi^L}\left[\nabla_{\boldsymbol{w}_\pi} \ln \pi_k^L(\boldsymbol{u} \mid \boldsymbol{x}) \times \left[\ln \pi_k^L(\boldsymbol{u} \mid \boldsymbol{x}) - \beta(Q_k(\boldsymbol{x}, \boldsymbol{u}) - V_k(\boldsymbol{x})) + g_k(\boldsymbol{x})\right]\right]. \tag{25}$$

When $g_k(\boldsymbol{x}) = B(\boldsymbol{x})$, Eq. (25) is essentially identical to Eq. (6).

Our forward RL step updates V_k and Q_k like the standard SAC algorithm (Haarnoja, Zhou, Abbeel et al., 2018). The loss function of the state value function is given:

$$J_{\mathrm{ERIL}}^V(\boldsymbol{w}_V) = \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}} \times \left[\left(V(\boldsymbol{x}) - \mathbb{E}_{\boldsymbol{u}\sim\pi_k^L(\cdot\mid\boldsymbol{x})}\left[Q_k(\boldsymbol{x}, \cdot) - \frac{1}{\beta}\ln \pi_k^L(\cdot \mid \boldsymbol{x})\right]\right)^2\right]. \tag{26}$$

The state–action value function is trained to minimize the soft Bellman residual

$$J_{\mathrm{ERIL}}^Q(\boldsymbol{w}_Q) = \frac{1}{2}\mathbb{E}_{(\boldsymbol{x},\boldsymbol{u},\boldsymbol{x}')\sim\mathcal{D}^L}\left[\left(Q(\boldsymbol{x}, \boldsymbol{u}) - \bar{Q}_k(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}')\right)^2\right], \tag{27}$$

with

$$\bar{Q}_k(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}') = r_k(\boldsymbol{x}) + \frac{1}{\eta}\ln \pi_k^L(\boldsymbol{u} \mid \boldsymbol{x}) + \gamma \bar{V}_k(\boldsymbol{x}'),$$

Algorithm 1 Entropy-Regularized Imitation Learning

```
Input: Expert dataset $\mathcal{D}^E$ and hyperparameters $\kappa$ and $\eta$
Output: Learner's policy $\pi^L$ and estimated reward $r(\boldsymbol{x})$.
1: Initialize all parameters of networks and replay buffer $\mathcal{D}^L$.
2: for $k = 0, 1, 2, \ldots$ do
3:     Sample $\tau_i = \{(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{x}_{t+1})\}_{t=0}^T$ with $\pi_k^L$ and store it in $\mathcal{D}^L$.
4:     Update $\boldsymbol{w}_g$ with the gradient of Eq. (21).
5:     Update $\boldsymbol{w}_r$ and $\boldsymbol{w}_V$ with the gradient of Eq. (22).
6:     Update $\boldsymbol{w}_Q$ with the gradient of Eq. (27).
7:     Update $\boldsymbol{w}_V$ with the gradient of Eq. (26).
8:     Update $\boldsymbol{w}_\pi$ with the gradient of Eq. (24).
9:     Update $\bar{\boldsymbol{w}}_V$ by Eq. (28).
10: end for
```

where $\tilde{V}_k(\boldsymbol{x})$ is the target state value function parameterized by $\bar{\boldsymbol{w}}_V$. Since our method is a model-free approach, we cannot compute the expected value of Eq. (4). Instead, we use $\bar{Q}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}')$, which is an approximation of $Q(\boldsymbol{x}, \boldsymbol{u})$, and $Q(\boldsymbol{x}, \boldsymbol{u}) = \mathbb{E}_{\boldsymbol{x}'\sim p_T(\cdot\mid\boldsymbol{x},\boldsymbol{u})}[\bar{Q}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}')]$. Two alternatives can be chosen to update $\bar{\boldsymbol{w}}_V$. The first is a periodic update, i.e., where the target network is synchronized with current $\boldsymbol{w}_g$ at regular intervals (Mnih et al., 2015). We use the second alternative, which is a Polyak averaging update, where $\bar{\boldsymbol{w}}_g$ is updated by a weighted average over the past parameters (Lillicrap et al., 2016):

$$\bar{\boldsymbol{w}}_V \leftarrow \tau\boldsymbol{w}_V + (1-\tau)\bar{\boldsymbol{w}}_V, \tag{28}$$

where $\tau \ll 1$.

Algorithm 1 shows an overview of Entropy-Regularized Imitation Learning. Lines 4–5 and 6–8 represent the inverse RL and forward RL steps. Note that $\boldsymbol{w}_V$ is updated twice in each iteration. Since the second discriminator depends on the first one, update $\boldsymbol{w}_g$ first, followed by $\boldsymbol{w}_r$ and $\boldsymbol{w}_V$. The order of Lines 6–8 is exchangeable in practice.

4.4. Interpretation of second discriminator

We show the connection between $D^{(2)}$ and the optimal discriminator of GAN shown in Eq. (8). Arranging Eq. (23) and assuming $|\mathcal{D}^E| = |\mathcal{D}^L|$ yield the second discriminator:

$$D^{(2)}(\boldsymbol{x}, \boldsymbol{u}) = \frac{\pi^L(\boldsymbol{x})\pi^L(\boldsymbol{u} \mid \boldsymbol{x})}{\pi^E(\boldsymbol{x})\tilde{\pi}^E(\boldsymbol{u} \mid \boldsymbol{x}) + \pi^L(\boldsymbol{x})\pi^L(\boldsymbol{u} \mid \boldsymbol{x})}, \tag{29}$$

where $\tilde{\pi}^E(\boldsymbol{u} \mid \boldsymbol{x}) \triangleq \exp(\beta(Q(\boldsymbol{x}, \boldsymbol{u}) - V(\boldsymbol{x})))$. We omit $\boldsymbol{x}'$ from the input to $D^{(2)}$ here because it does not depend on the next state. See A.2 for the derivation. By comparing Eqs. (8) and (29), $D^{(2)}$ represents the form of the optimal discriminator, and the expert policy is approximated by the value functions.

4.5. Extension

Here we describe two extensions to deal with more realistic situations. One learns multiple policies from multiple experts. The dataset of experts is augmented by adding conditioning variable c that represents the index of experts:

$$\mathcal{D}^E = \{(\boldsymbol{x}_i, \boldsymbol{u}_i, \boldsymbol{x}_i', c)\}_{i=1}^{N^E}, \quad \boldsymbol{u}_i \sim \pi^E(\cdot \mid \boldsymbol{x}_i, c), \quad \boldsymbol{x}_i' \sim p_T(\cdot \mid \boldsymbol{x}_i, \boldsymbol{u}_i),$$

where c is often encoded as a one-hot vector. Then we introduce universal value function approximators (Schaul et al., 2015) to extend the value functions to be conditioned on the subject index. For example, the second discriminator is extended:

$$D_k^{(2)}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}', c) = \frac{\exp(\beta f_k(\boldsymbol{x}, \boldsymbol{x}', c))}{\exp(\beta f_k(\boldsymbol{x}, \boldsymbol{x}', c)) + \exp(\beta\kappa^{-1}\ln \pi_k^L(\boldsymbol{u} \mid \boldsymbol{x}, c))},$$

where $f_k(\boldsymbol{x}, \boldsymbol{x}', c) \triangleq r_k(\boldsymbol{x}, c) + \beta^{-1}g_k(\boldsymbol{x}, c) + \gamma V_k(\boldsymbol{x}', c) - V_k(\boldsymbol{x}, c)$.

The other extension deals with the case where actions are not observed. ERIL needs to observe the expert action because $\mathcal{D}^{(2)}$ explicitly depends on the action. One simple solution is to set $\kappa^{-1} = 0$. This is the special case of ERIL in which the inverse RL step is LogReg-IRL (Uchibe, 2018; Uchibe & Doya, 2014). An alternative is to exploit an inverse dynamics model (Torabi et al., 2018), which is formulated as a maximum-likelihood estimation problem by maximizing the following function:

$$J_{\mathrm{IDM}}(\boldsymbol{w}_a) = \sum_{(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{x}_{t+1}) \in \mathcal{D}^L} \ln p(\boldsymbol{u}_t \mid \boldsymbol{x}_t, \boldsymbol{x}_{t+1}),$$

where $\boldsymbol{w}_a$ is a parameter of the conditional probability density over actions given a specific state transition. Then the expert dataset is augmented:

$$\tilde{\mathcal{D}}^E = \{(\boldsymbol{x}_i, \tilde{\boldsymbol{u}}_i, \boldsymbol{x}'_i, c)\}_{i=1}^{N^E}, \quad \tilde{\boldsymbol{u}}_i \sim p(\cdot \mid \boldsymbol{x}_i, \boldsymbol{x}'_i).$$

Using these inferred actions, we apply ERIL as usual.

5. MuJoCo benchmark control tasks

5.1. Task description

We evaluated ERIL with six MuJoCo-simulated (Todorov et al., 2012) environments, provided by the OpenAI gym (Brockman et al., 2016): Hopper, Walker, Reacher, Half-Cheetah, Ant, and Humanoid. The goal for all the tasks was to move forward as quickly as possible. First, optimal policy $\pi^E(\boldsymbol{u} \mid \boldsymbol{x})$ for every task was trained by TRPO with a reward function provided by the OpenAI gym, and expert dataset $\mathcal{D}^E$ was created by executing π^E. Then we evaluated the imitation performance with respect to the sample complexity of the expert data by changing the number of samples in $\mathcal{D}^E$ and the number of interactions with the environment. Based on Ho and Ermon (2016), the trajectories constituting each dataset consisted of about 50 transitions $(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}')$. The same demonstration data were used to train all the algorithms for a fair comparison. We compared ERIL with BC, GAIL, Sasaki, DAC, and SAM. The network architectures that approximate the functions are shown in Table 1. We used a Rectified Linear Unit (ReLU) activation function in the hidden layers. The output nodes used a linear activation function, except $\mu(\boldsymbol{x})$ and $\sigma(\boldsymbol{x})$. Functions $\mu(\boldsymbol{x})$ and $\sigma(\boldsymbol{x})$ represent the learner's policy by a Gaussian distribution:

$$\pi^L(\boldsymbol{u} \mid \boldsymbol{x}) = \mathcal{N}(\boldsymbol{u} \mid \mu(\boldsymbol{x}), \sigma(\boldsymbol{x})),$$

where $\mu(\boldsymbol{x})$ and $\sigma(\boldsymbol{x})$ denote the mean and the diagonal covariance matrix. The output nodes of $\mu(\boldsymbol{x})$ and $\sigma(\boldsymbol{x})$ use tanh and sigmoid functions.

The number of trajectories generated by $\pi_k^L(\boldsymbol{u} \mid \boldsymbol{x})$ was set to 100. In all of our experiments, the hyperparameters for regularizing the rewards were $\kappa = 1$ and $\eta = 10$, and the discount factor was $\gamma = 0.99$. We trained all the networks with the Adam optimizer (Kingma & Ba, 2015) and a decay learning rate. Following Fujimoto et al. (2018), we performed evaluations using ten different random seeds.

Table 1
Neural network architectures used in MuJoCo benchmark control tasks: $V(\boldsymbol{x})$ is approximated by a two-layer fully-connected neural network consisting of (256, 256) hidden units in HalfCheetah and Humanoid tasks.

Function	Number of nodes	Task
$\mu(\boldsymbol{x})$	$(\dim\mathcal{X}$, 256, 256, $\dim\mathcal{U})$	HalfCheetah and Humanoid
	$(\dim\mathcal{X}$, 100, 100, $\dim\mathcal{U})$	Other tasks
$\sigma(\boldsymbol{x})$	$(\dim\mathcal{X}$, 100, $\dim\mathcal{U})$	All tasks
$r(\boldsymbol{x})$	$(\dim\mathcal{X}$, 100, 100, 1)	All tasks
$V(\boldsymbol{x})$	$(\dim\mathcal{X}$, 256, 256, 1)	HalfCheetah and Humanoid
	$(\dim\mathcal{X}$, 100, 100, 1)	Other tasks
$Q(\boldsymbol{x}, \boldsymbol{u})$	$(\dim\mathcal{X} + \dim\mathcal{U}$, 256, 256, 1)	HalfCheetah and Humanoid
	$(\dim\mathcal{X} + \dim\mathcal{U}$, 100, 100, 1)	Other tasks
$g(\boldsymbol{x})$	$(\dim\mathcal{X}$, 100, 100, 1)	All tasks
$D(\boldsymbol{x}, \boldsymbol{u})$	$(\dim\mathcal{X} + \dim\mathcal{U}$, 256, 256, 1)	HalfCheetah and Humanoid
	$(\dim\mathcal{X} + \dim\mathcal{U}$, 100, 100, 1)	Other tasks

5.2. Comparative evaluations

Fig. 3 shows the normalized return of the evaluation rollouts during training for ERIL, BC, GAIL, Sasaki, DAC, and SAM. A normalized return is defined by

$$\bar{R} = \frac{R - R_0}{R^E - R_0},$$

where R, R_0, and R^E respectively denote the total returns of the learner, a randomly initialized Gaussian policy, and an expert trained by TRPO. The horizontal axis depicts the number of interactions with the environment in the logarithmic scale. ERIL, Sasaki, DAC, and SAM have considerably better sample efficiency than BC and GAIL in all the MuJoCo control tasks. In particular, ERIL performed consistently across all the tasks and outperformed the baseline methods in each one. DAC outperformed Sasaki and SAM in Walker2d, HalfCheetah, and Ant, and its performance was competitive with that of Hopper, Reacher, and Humanoid. Sasaki achieved better performance than SAM in Walker2D, and there was no significant difference between Sasaki and SAM in our implementation. GAIL was not very competitive because the on-policy TRPO algorithm used in its forward RL step could not use the technique of experience replay.

Fig. 4 shows the normalized return of the evaluation rollout of the six algorithms with respect to the number of demonstrations. ERIL, Sasaki, DAC, and SAM were more sample-efficient than BC and GAIL. ERIL was consistently one of the most sample-efficient algorithms when the number of demonstrations was small. DAC's performance was comparable to that of ERIL. SAM showed comparable performance to ERIL in Hopper, but its normalized return was worse than ERIL when the number of demonstrations was limited. Although the BC implemented in this paper did not consider the covariate shift problem, the normalized return approached 1.0 with 25 demonstrations. Note that BC is an M-projection that avoids $\pi^L = 0$ whenever $\pi^E > 0$. Therefore, the policy trained by BC is averaged over several modes, even if it is approximated by a Gaussian distribution. On the contrary, ERIL is an I-projection that forces π^L to be zero even if $\pi^E > 0$. Although the policy trained by ERIL concentrates on a single mode, BC obtained a comparable policy to ERIL because the expert policy obtained by TRPO was also approximated by the Gaussian distribution.

5.3. Ablation study

Next we investigate which ERIL component contributed most to its performance by an ablation study on the Ant environment. We tested five different components:

1. ERIL without the first discriminator, i.e., we set $g_k(\boldsymbol{x}) = 0$ for all $\boldsymbol{x}$.
2. ERIL without sharing the state value function between the forward and inverse RLs. The forward RL step updates the policy with reward $r_k(\boldsymbol{x})$.
3. ERIL without sharing the state value function between the forward and inverse RL. The forward RL step updates the policy with shaping reward $r_k(\boldsymbol{x}) + \gamma V_k(\boldsymbol{x}') - V_k(\boldsymbol{x})$ for state transition $(\boldsymbol{x}, \boldsymbol{x}')$.

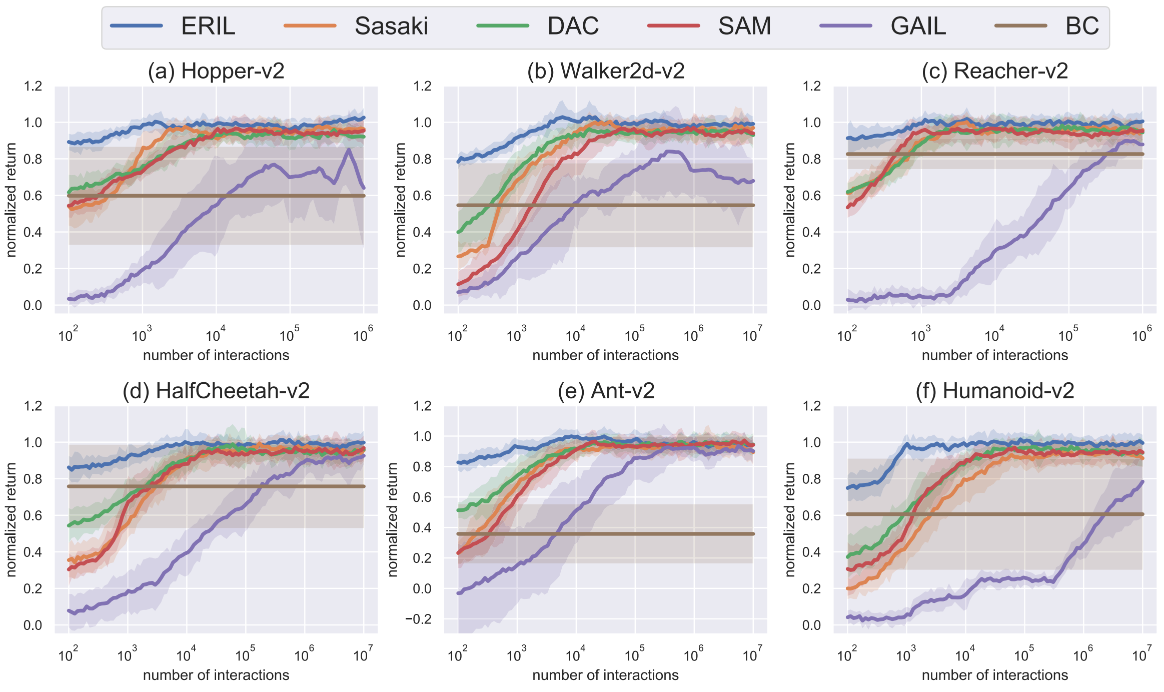


Fig. 3. Performance with respect to number of interactions in MuJoCo tasks: Solid lines represent average values, and shaded areas correspond to ±1 standard deviation region.

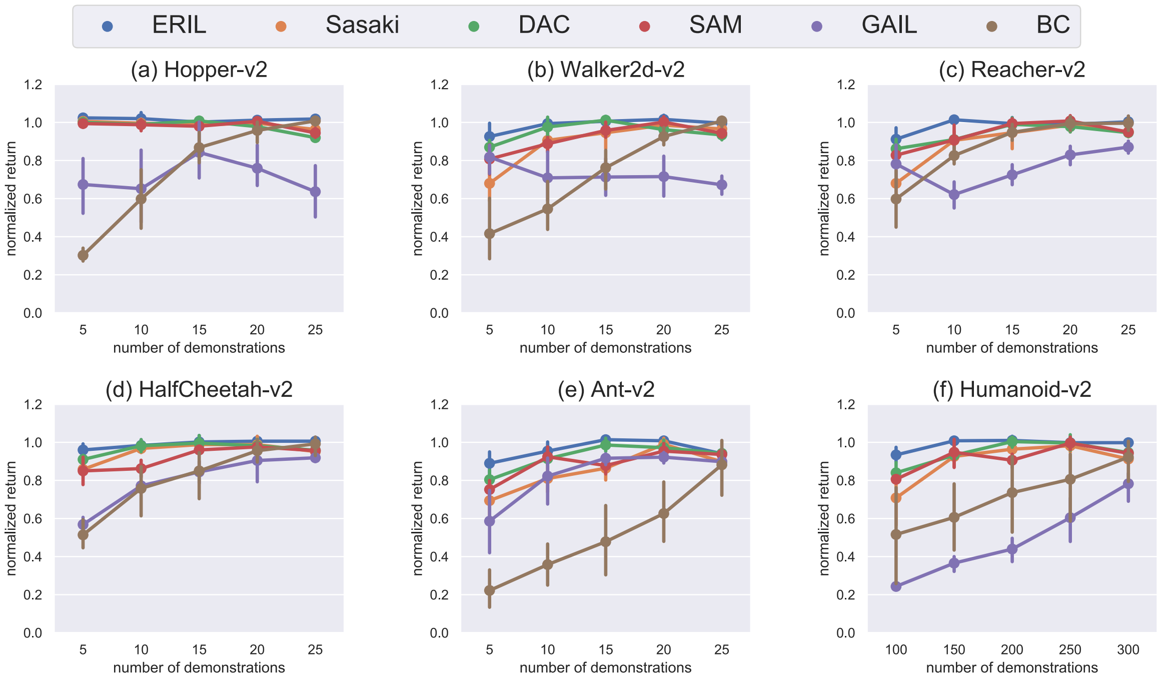


Fig. 4. Performance with respect to number of trajectories provided by expert in MuJoCo tasks: Solid lines represent average values, and shaded areas correspond to ±1 standard deviation region.

4. ERIL without sharing the hyperparameters. We set $\beta = 0$ and $\kappa = 0$ in Eq. (20), and the β and η of the forward RL step are used as they are.
5. ERIL without the soft Bellman equation. Discriminator $D_k^{(3)}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}')$ is defined by

$$D_k^{(3)}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}') = \frac{1}{1 + \exp(-h_k(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}'))},$$

where $h_k(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}')$ is directly implemented by a neural network. Then the reward function is computed by $r = -\ln D_k^{(3)}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}') + \ln(1 - D_k^{(3)}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}'))$ like AIRL.

Fig. 5 compares the learning curves. In the benchmark tasks, we found no significant difference between the original ERIL and the ERIL without the first discriminator. One possible reason is that $g_k(\boldsymbol{x})$ did not change the policy gradient, as we explained in

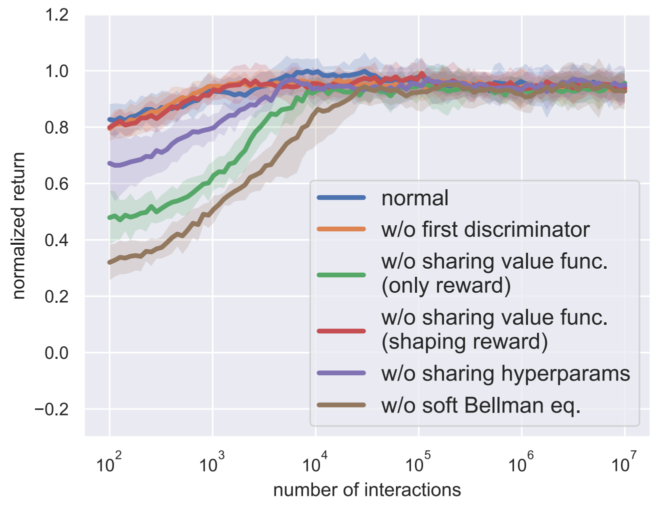


Fig. 5. Comparison of learning curves in ablation study.

Section 4.3. The other reason might be that $g_k(\boldsymbol{x})$ was close to zero because the state distributions of the experts and learners did not differ significantly due to the small variation of the initial states. Regarding the property of sharing the state value function, the results depended on how the reward was calculated from the results of the inverse RL step. When the policy was trained with $r_k(\boldsymbol{x})$, it took longer steps to reach the performance of the original ERIL. On the other hand, if the reward is calculated by the form of the shaping reward, there was no significant difference compared with the original ERIL. When we removed the hyperparameters from the second discriminator, slower learning resulted. The ERIL without the soft Bellman equation was the most inefficient sample in the early stage of learning for several reasons. One is that the discriminator is larger than the original ERIL because it was defined in the joint space of the state, the action, and the next state. As a result, it required more samples for training the discriminator. The second reason is that there were no hyperparameters.

6. Real robot experiment

We further investigated the role of the first discriminator by conducting a robot experiment in which the learner's initial distribution differs from the expert's initial distribution.

6.1. Task description

We performed a reaching task defined as controlling the end-effector to a target position. We used an upper-body robot called Nextage developed and manufactured by Kawada Industries, Inc.(Fig. 6(a)). Nextage has a head with two cameras, a torso, two 6-axis manipulators, and two cameras attached to its end-effectors. We used the left arm, a camera mounted on it, and the left camera on the head in this task. The head pose was fixed during the experiments. There were two colored blocks as obstacles and one metal can that indicated the goal position in the workspace. We prepared several environmental configurations by changing the arm's initial pose, the can's location, and the height of the blocks. To evaluate the effect of the first discriminator, we considered the six initial poses shown in Fig. 6(b). The expert agent controlled the arm with three different initial poses, and the learning agent controlled it with five different initial poses, including the expert's initial poses. One pose was used to evaluate the learned policies. We prepared blocks of two different heights. Three goal positions were set: two for training and one for testing. The environmental configurations were constructed as follows:

- Expert configuration: An initial pose and a target pose were selected from three possible poses (black-filled circle) and two possible poses (black-filled squares), shown in Fig. 6(b). We randomly chose the short blocks and the tall ones. Consequently, there were $3 \times 2 \times 2 = 12$ configurations.
- Learner's configuration: An initial pose and a target pose were selected from five possible poses (black-filled circle and black-empty circle) and two possible poses (black-filled square). We randomly chose short blocks and tall ones. Consequently, there were $5\times2\times2 = 20$ configurations. Note that eight configurations were not included in the expert configuration.
- Test configuration: An initial pose was the red-filled circle shown in Fig. 6(b). The red-filled square represents the target position. There were $1 \times 2 \times 2 = 4$ configurations.

We used MoveIt! (Chitta, Sucan, & Cousins, 2012), which is the most widely used software for motion planning, to create expert demonstrations using the geometric information of the metal can and the colored blocks. Such information was not available for learning the algorithms. MoveIt! generated 20 trajectories for every expert configuration. We recorded the RGB images and joint angles. The sequence of the joint angles was almost deterministic, but that of the RGB images was stochastic and noisy due to the lighting conditions.

The state of this task consists of six joint angles, $\{\theta_i\}_{i=1}^6$, of the left arm and two 84×84 RGB images, I_{head} and I_{hand}, captured by Nextage's cameras. The action is given by the changes in the joint angle from previous position $\{\Delta\theta_i\}_{i=1}^6$, where $\Delta\theta_i$ is the change in the joint angle from the previous position of the ith joint. We provided a deep convolutional encoder to find an appropriate representation from the pixels based on a previous study (Yarats et al., 2020). Fig. 7(a) shows the relationship among the networks used in the experiment. The policy, the reward, the state value, the state–action value, and the first discriminator shared the encoder network. The encoder maps captured images I_{head} and I_{hand} to latent variable $\boldsymbol{z}$: $\text{enc}(I_{\text{head}}, I_{\text{hand}}) = \boldsymbol{z}$. Then it was concatenated with the joint angles for the input of the first discriminator, the reward, and the state value function. The action vector is also added to the policy and the state–action value function. The decoder is introduced to train the encoder. Fig. 7(b) shows the network architecture of the encoder network.

Based on a suggestion inferred from research by Yarats et al. (2020), we used a deterministic Regularized Autoencoder (RAE) (Ghosh, Sajjadi, Vergari, Black, & Scholkopf, 2020), which imposed an L2 penalty on learned representation $\boldsymbol{z}$ and a weight-decay on the decoder parameters. We prevented the gradients of the policy and the state–action value function from updating the convolutional encoder, another idea borrowed from Yarats et al. (2020). See Appendix B for details.

6.2. Experimental results

We compared the normal ERIL with (1) the ERIL without the first discriminator, (2) DAC, (3) AIRL, and (4) BC. To measure the performances, we considered a synthetic reward function given by

$$r = \exp\left(-\frac{\|\boldsymbol{p}_t - \boldsymbol{p}_g\|_2^2}{\sigma^2}\right),$$

where $\boldsymbol{p}_t$ and $\boldsymbol{p}_g$ denote the end-effector's current and the target position. Parameter σ was set to 5. Note that TRPO could not find

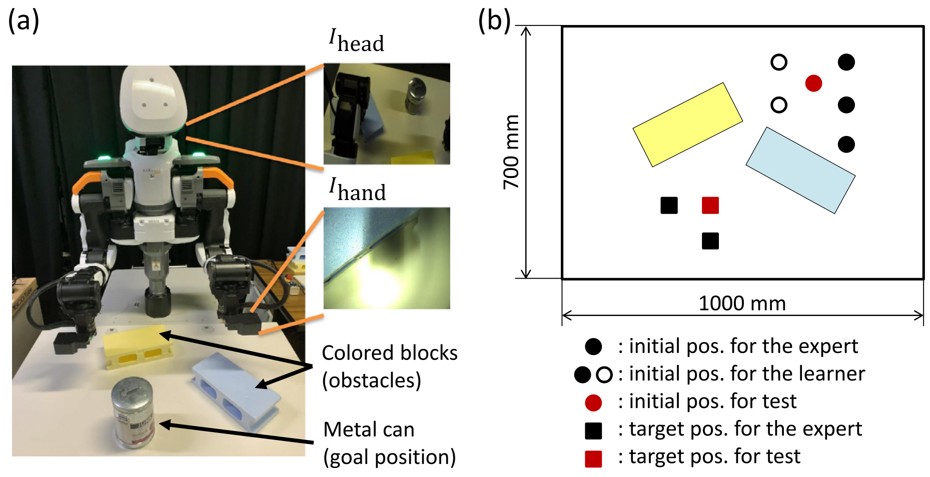

(a)
I_{head}
I_{hand}
Colored blocks (obstacles)
Metal can (goal position)
(b)
700 mm
1000 mm
● : initial pos. for the expert
●○ : initial pos. for the learner
● : initial pos. for test
■ : target pos. for the expert
■ : target pos. for test

Fig. 6. Real robot setting: (a) environment of reaching task, which moves end-effector of left arm to target position; (b) possible environmental configuration.

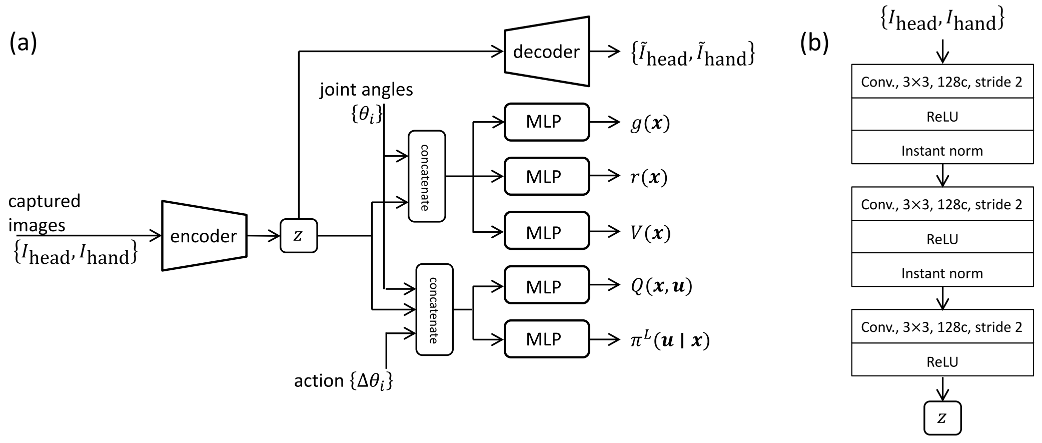

(a)
decoder
$\{\tilde{I}_{\text{head}}, \tilde{I}_{\text{hand}}\}$
joint angles $\{\theta_i\}$
concatenate
MLP
$g(\boldsymbol{x})$
$r(\boldsymbol{x})$
$V(\boldsymbol{x})$
$Q(\boldsymbol{x}, \boldsymbol{u})$
$\pi^L(\boldsymbol{u} \mid \boldsymbol{x})$
captured images $\{I_{\text{head}}, I_{\text{hand}}\}$
encoder
z
action $\{\Delta\theta_i\}$
(b)
$\{I_{\text{head}}, I_{\text{hand}}\}$
Conv., 3×3, 128c, stride 2
ReLU
Instant norm
Conv., 3×3, 128c, stride 2
ReLU
Instant norm
Conv., 3×3, 128c, stride 2
ReLU
z

Fig. 7. Neural networks used in robot experiment: (a) architecture for policy, reward, state value, state–action value, and first discriminator, sharing the encoder; (b) architecture for encoder: Conv. denotes a convolutional neural network. "nc" denotes "n channels".

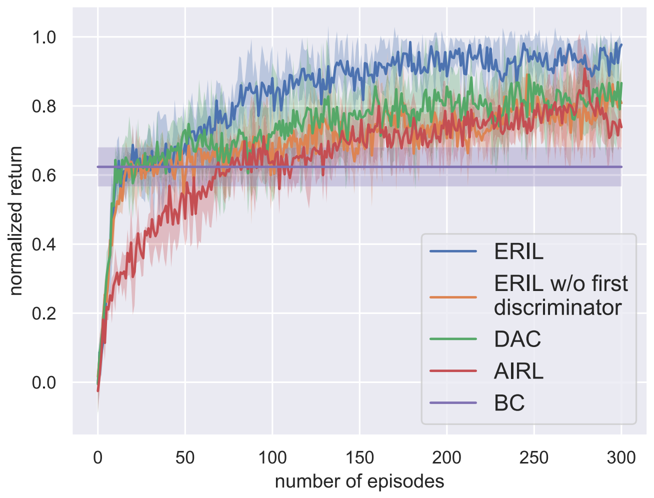


Fig. 8. Performance with respect to number of interactions in real robot experiment.

an optimal policy from the synthetic reward due to the sparseness of the reward.

The averaged learning results based on the three experiments are shown in Fig. 8. The most sample-efficient method was ERIL, which achieved the best asymptotic performance. ERIL without the first discriminator and DAC learned efficiently at the early stage of learning, but their asymptotic performances were worse than that of ERIL. AIRL also achieved the same performance as ERIL without the first discriminator and DAC at the end of learning, and BC achieved the worse performance.

To evaluate the learned policies, we computed NLL for the trajectories generated by MoveIt!. Fig. 9 compares the NLL for the learning configuration (starting from two positions not included in the expert configuration) and those for the test configuration. Note that $g(\boldsymbol{x}) \neq 0$ for these initial positions in the learning configuration. ERIL obtained a smaller NLL than other methods for both the configurations, suggesting that the policy obtained by ERIL was closer to that of MoveIt!. We found that ERIL assigned a small reward value close to zero around these positions because the first discriminator was more dominant than the second one. These results suggest that the first discriminator plays a role when the learner's initial distribution differs from the expert one.

7. Human behavior analysis

7.1. Task description

Next we evaluated ERIL in a realistic situation by conducting a dynamic motor control experiment in which a human subject solved a planar pole-balancing problem. Fig. 10(a) shows the experimental setup. The subject can move the base in the left,

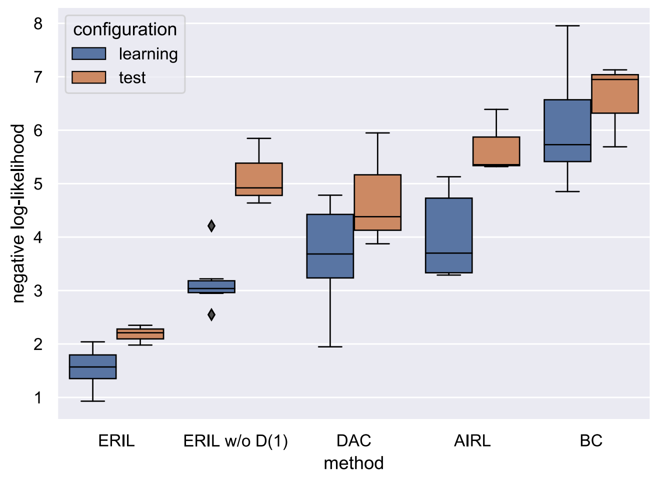


Fig. 9. Comparison of NLL in real robot experiment: "ERIL w/o D(1)" denotes ERIL without first discriminator. Note that smaller NLL values indicate a better fit.

right, top, and bottom directions to swing the pole several times and decelerate it to balance it in the upright position. As shown in Fig. 10(b), the dynamics are described by a six-dimensional state vector, $\boldsymbol{x} = [\theta, \dot{\theta}, x, \dot{x}, z, \dot{z}]^\top$, where θ and $\dot{\theta}$ are the angle and the angular velocity of the pole, x and z are the horizontal and vertical positions of the base, and $\dot{x}$ and $\dot{z}$ are their time derivatives. The state variables are defined in the following ranges: $\theta \in [-\pi, \pi]$ (in [rad]), $\dot{\theta} \in [-4\pi, 4\pi]$ (in [rad/s]), $x \in [0, 639]$ (in [pixel]), $\dot{x} \in [-5, 5]$ (in [pixel/s]), $z \in [0, 319]$ (in [pixel]), and $\dot{z} \in [-5, 5]$ (in [pixel/s]). Note that no applied forces F_x and F_z were observed on the pendulum in Fig. 10 in this experiment.

The task was performed under two pole conditions: long (73 cm) and short (29 cm). Each subject had 15 trials to balance the pole in each condition after some practice. Each trial ended when the subject could keep the pole upright for three seconds or after 40 s had elapsed. We collected data from seven subjects (five right-handed and two left-handed). We used a trajectory-based sampling method to construct the following three expert datasets: $\mathcal{D}^E_{i,j,\mathrm{tr}}$ for training, $\mathcal{D}^E_{i,j,\mathrm{va}}$ for validation, and $\mathcal{D}^E_{i,j,\mathrm{te}}$ for testing the ith subject. Subscript j indicates 1 for the long-pole conditions and 2 for the short-pole conditions.

Since we had multiple experts whose actions were unavailable, we used the extended ERIL described in Section 4.5. We augmented the ERIL functions by a seven-dimensional conditional one-hot vector $\boldsymbol{c}$ and evaluated two ERIL variations. One is where the second discriminator was replaced by the LogReg-IRL by setting $\kappa^{-1} = 0$. This method is called ERIL($\kappa^{-1} = 0$). The other is the ERIL with IDM, in which the expert action is estimated by the inverse dynamics model. The two ERILs were compared with

Table 2
Neural network architectures used in pole-balancing task: For example, $V(\boldsymbol{x})$ is approximated by a two-layer fully-connected neural network consisting of (256, 256) hidden units in HalfCheetah and Humanoid tasks.

Function	Number of nodes
$\mu(\boldsymbol{x}, \boldsymbol{c})$	$(\dim \mathcal{X} + \dim \boldsymbol{c},\ 100,\ \dim \mathcal{U})$
$\sigma(\boldsymbol{x}, \boldsymbol{c})$	$(\dim \mathcal{X} + \dim \boldsymbol{c},\ 50,\ \dim \mathcal{U})$
$\boldsymbol{m}(\boldsymbol{x}, \boldsymbol{c})$	$(\dim \mathcal{X} + \dim \boldsymbol{c},\ 50,\ \dim \mathcal{X})$
$V(\boldsymbol{x}, \boldsymbol{c})$	$(\dim \mathcal{X} + \dim \boldsymbol{c},\ 100,\ 1)$
$Q(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{c})$	$(\dim \mathcal{X} + \dim \mathcal{U} + \dim \boldsymbol{c},\ 256,\ 1)$
$g(\boldsymbol{x}, \boldsymbol{c})$	$(\dim \mathcal{X} + \dim \boldsymbol{c},\ 50,\ 1)$
$D(\boldsymbol{x}, \boldsymbol{c})$	$(\dim \mathcal{X} + \dim \boldsymbol{c},\ 100,\ 1)$
$D(\boldsymbol{x}, \boldsymbol{x}', \boldsymbol{c})$	$(2 \times \dim \mathcal{X} + \dim \boldsymbol{c},\ 256,\ 1)$

GAIfO, IRLGAN, and BCO. Since the two ERILs, GAIfO, and IRLGAN improved the policy by forward RL, they require a simulator of the environment. We modeled the dynamics shown in Fig. 10(b) as an X–Z inverted pendulum (Wang, 2012). The physical parameters and the motion of the equations are provided in Appendix C.1.

We parameterized the log of the reward function as a quadratic function of the nonlinear features of the state:

$$\ln r_k(\boldsymbol{x}, c) = -\frac{1}{2}(\boldsymbol{x} - \boldsymbol{m}(\boldsymbol{x}, \boldsymbol{c}))^\top \boldsymbol{P}(\boldsymbol{x}, \boldsymbol{c})(\boldsymbol{x} - \boldsymbol{m}(\boldsymbol{x}, \boldsymbol{c})),$$

where $\boldsymbol{c}$ is a vector encoding the subject index and condition. $\boldsymbol{c}$ is a concatenation of $\boldsymbol{c}^s$ and $\boldsymbol{c}^c$ that denotes the one-hot vector to encode the subject and the condition. Since we had seven subjects and two experimental conditions, $\boldsymbol{c}$ was a seven-dimensional vector. $\boldsymbol{P}(\boldsymbol{x}, \boldsymbol{c})$ is a positive definite square matrix, parameterized by $\boldsymbol{P}(\boldsymbol{x}, \boldsymbol{c}) = \boldsymbol{L}(\boldsymbol{x}, \boldsymbol{c})\boldsymbol{L}(\boldsymbol{x}, \boldsymbol{c})^\top$, where $\boldsymbol{L}(\boldsymbol{x}, \boldsymbol{c})$ is a lower triangular matrix. The $\boldsymbol{L}(\boldsymbol{x}, \boldsymbol{c})$ element is a linear output layer of the neural network with exponentially transformed diagonal terms. Table 2 shows the network architecture. As discriminators, note that IRLGAN uses $D(\boldsymbol{x}, \boldsymbol{c})$, and GAIfO uses $D(\boldsymbol{x}, \boldsymbol{x}', \boldsymbol{c})$.

7.2. Experimental results

Fig. 11 shows the learning curves of the seven subjects, indicating quite different learning processes. Subject 7 achieved the best performance for both conditions, and subjects 1 and 3 failed to accomplish the task. Subjects 1 and 5 performed well in the long-pole condition, but they failed to balance the pole in the upright position. Although the trajectories generated by Subjects 1 and 3 were unsuccessful, we used their data as the expert trajectories.

To evaluate the algorithms, we computed the NLL for test datasets $\mathcal{D}^E_{i,\mathrm{te}}$:

$$\mathrm{NLL}(i, j) = -\frac{1}{N^E_{i,\mathrm{te}}} \sum_{\ell=1}^{N^E_{i,j,\mathrm{te}}} \ln \pi^L(\boldsymbol{x}'_\ell \mid \boldsymbol{x}_\ell, \boldsymbol{c}_{i,j}),$$

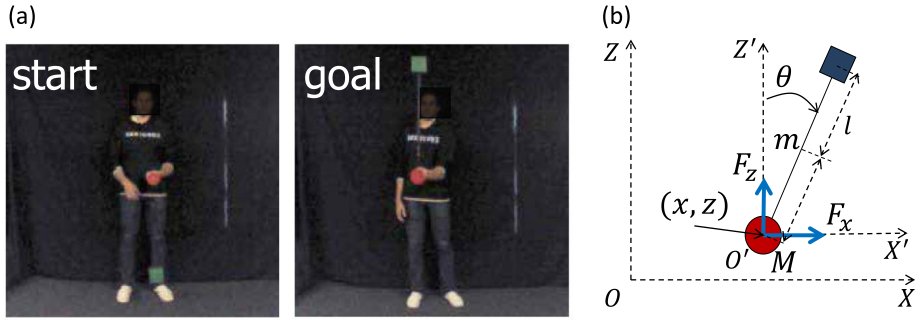


Fig. 10. Inverted pendulum task solved by a human subject: (a) start and goal positions; (b) state representation. Notations are explained in Appendix C.1.

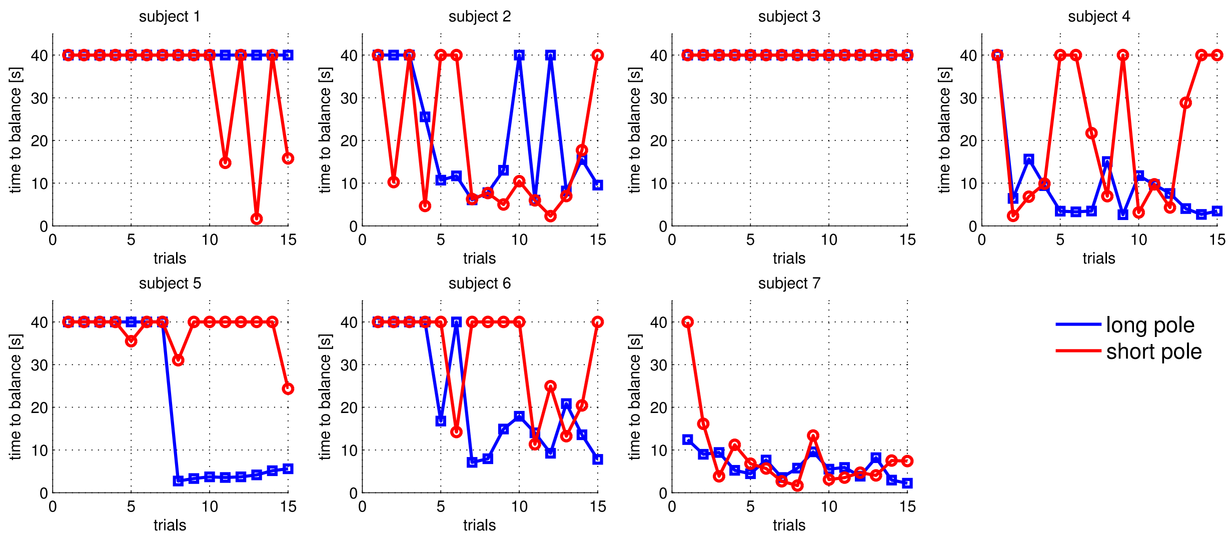


Fig. 11. Learning curves of seven subjects: blue: long pole; red: short pole. Trial is considered a failure if subject cannot balance pole in upright position within 40 [s]. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

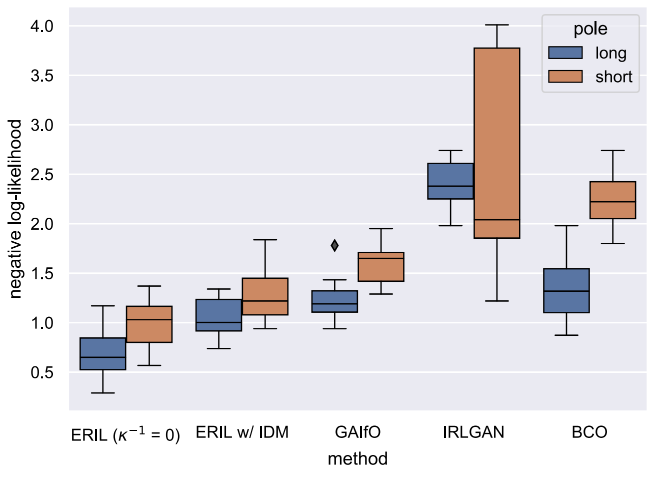


Fig. 12. Comparison of NLL among imitation learning algorithms: Note that smaller NLL values indicate a better fit.

where $N^E_{i,\text{te}}$ is the number of samples in $\mathcal{D}^E_{i,\text{te}}$. $\boldsymbol{c}_{i,j} = [\boldsymbol{c}^s_i, \boldsymbol{c}^c_j]$ is the conditioning vector, where $\boldsymbol{c}_i$ represents that the ith element is 1 and otherwise 0. See Appendix C.2 for an evaluation of the state transition probability. Fig. 12 shows that ERIL($\kappa^{-1} = 0$) obtained a smaller NLL than the other baselines for both conditions. Note that the data in the expert dataset were not even sub-optimal because Subjects 3 and 5 did not fully accomplish the balancing task (Fig. 11). The ERIL with IDM achieved comparable performance to ERIL ($\kappa^{-1} = 0$) in the long-pole condition, but a lower performance in the short-pole condition. This result suggests that data augmentation by IDM did not help the discriminator's training. BCO also augmented the data by IDM, but its NLL was higher than ERIL with IDM. Compared to the IRLGAN and GAIfO results, our methods efficiently represented the policy of the experts, indicating that the structure of the ERIL's discriminator was critical even if expert actions were unavailable.

Fig. 13 shows the reward function of Subjects 2, 4, and 7, estimated by ERIL($\kappa^{-1} = 0$), which was projected to subspace $(\theta, \dot{\theta})$; $x, z, \dot{x}$ and $\dot{z}$ were set to zero for visualization. Although they balanced the pole in the long-pole condition, their estimated rewards differed. In the case of Subject 7, the reward function of the long-pole condition was about the same as the short-pole condition, although there was a significant difference in the results of Subject 4, who did not perform well in the short-pole condition (Fig. 11).

Finally, we computed NLL where the policy of the ith subject was evaluated by the test dataset of the jth subject. Fig. 14 shows the ERIL ($\kappa^{-1} = 0$), GAIfO, and BCO results. In the upper row, we used the test dataset of Subject 4 in the long-pole condition. For instance, the upper left figure shows the set of NLL computed by

$$\mathrm{NLL}(i,j) = -\frac{1}{N^E_{4,1,\text{te}}} \sum_{\ell=1}^{N^E_{4,1,\text{te}}} \ln \pi^L(\boldsymbol{x}'_\ell \mid \boldsymbol{x}_\ell, \boldsymbol{c}_{i,j}).$$

Note that the test dataset and the training data were inconsistent when $i \neq 4$ and $j \neq 1$. The minimum NLL was achieved when the conditioning vector was set properly (i.e., $i = 4$ and $j = 1$). On the other hand, the policy of Subject 4 in the short-pole condition ($i = 4$ and $j = 2$) did not fit the test data well in the long-pole condition, suggesting that Subject 4 used different reward functions for different poles.

The bottom row in the figures shows the results when we used the test dataset of Subject 7 in the long-pole condition. The minimum NLL was achieved by a proper conditioning vector. Unlike the case of Subject 4, ERIL ($\kappa^{-1} = 0$) found that the policy for the short-pole condition ($i = 7$ and $j = 2$) also achieved better performance with no significant differences. These results suggest that Subject 7, who was the best performer, used similar reward functions for both poles. No such property was found in GAIfO and BCO.

8. Discussion

8.1. Hyperparameter settings

ERIL has two hyperparameters, κ and η. As with a standard RL, efficiency and performance depend on how the hyperparameters are tuned during learning (Henderson, Islam et al., 2018; Zhang et al., 2021). Kozuno et al. (2019) showed that one hyperparameter derived from κ and η controlled the tradeoff between the noise tolerance and the convergence rate, and beta controls the quality of the asymptotic performance from the viewpoint of the forward RL setting. We adopted a simple grid

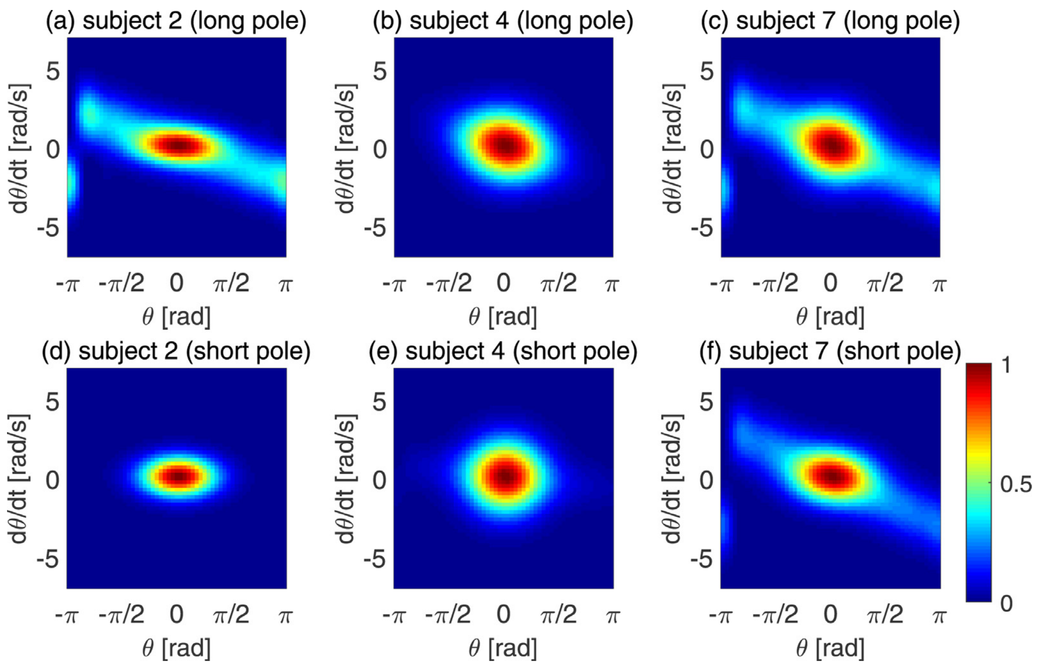


Fig. 13. Estimated reward function of Subjects 2, 4, and 7 projected to subspace (θ, ω), $\omega = d\theta/dt$.

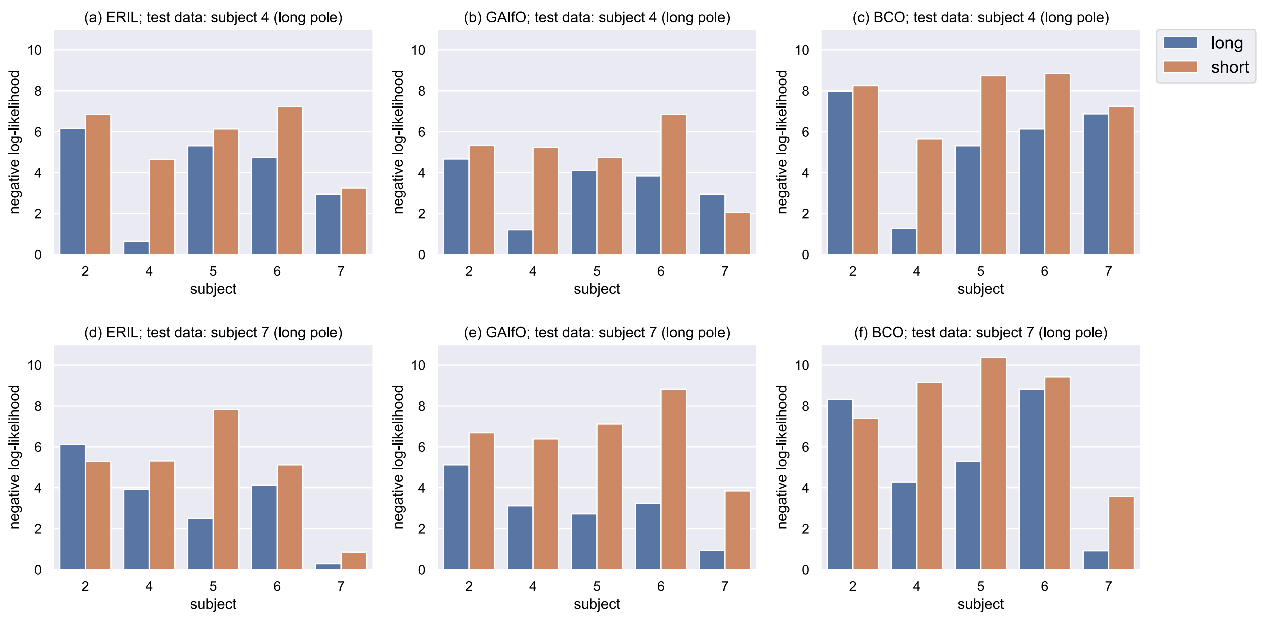


Fig. 14. Comparison of NLL when condition of test dataset was different from training one in human inverted pendulum task. Figs. in upper row (a, b, and c) show results when test dataset is $\mathcal{D}^E_{4,1,\mathrm{te}}$. Figs. in lower row (d, e, and f) show results when test dataset is $\mathcal{D}^E_{7,1,\mathrm{te}}$.

search method to tune the hyperparameters for every task, although naive grid search methods are sample inefficient and computationally expensive.

One possible way to tune the hyperparameters is to evaluate multiple hyperparameters and update them using a genetic algorithm-like method (Elfwing, Uchibe, & Doya, 2018; Jaderberg et al., 2017). A later version of SAC (Haarnoja, Zhou, Hartikainen et al., 2018) updates the hyperparameter that corresponds to the κ of ERIL by a simple gradient descent algorithm. Lee, Lee, Vrancx, Kim, and Kim (2020) optimized the state-dependent hyperparameter that corresponds to η by the hypergradient on the validation data. Their methods will be helpful for the forward RL step.

However, the ERIL hyperparameters play a different role. The κ of ERIL, which is the coefficient of the entropy term of the expert policy, should be determined by the properties of the expert policy. On the other hand, the κ of entropy-regularized RL, which is the coefficient of the learner's policy, determines the softness of the max operator of the Bellman optimality equation. The η of ERIL is the coefficient of the KL divergence between the expert and the learner policies, and the entropy-regularized RL is the coefficient of the KL divergence between the learner's current policy and the previous one. Tuning hyperparameters is future research.

8.2. Kinesthetic teaching

ERIL assumes that experts and learners have identical state transition probabilities. Therefore, the log-ratios of joint distributions can be decomposed into the sum of the policies and the state distributions. However, this assumption limits ERIL's applicability. For example, for a humanoid robot to imitate human expert behavior, the expert must demonstrate by kinesthetic teaching. To do so, the robot's gravity compensation mode is activated, and the expert must guide the robot's body to execute the task. Since this might not be a natural behavior for the expert, kinesthetic teaching is neither a practical nor a viable option to provide demonstrations.

To overcome this problem, we should consider the log-ratio between the state transitions of experts and learners. The log-ratio can be estimated by the density ratio estimation methods, as we did for the first discriminator. We plan to modify ERIL to incorporate the third discriminator.

9. Conclusion

This paper presented ERIL, which is entropy-regularized imitation learning based on forward and inverse reinforcement learning. Unlike previous methods, the update rules of the forward and inverse RL steps are derived from the same soft Bellman equation, and the state value function and hyperparameters are shared between the inverse and forward RL steps. The inverse RL step is done by training two binary classifiers, one of which is constructed by the reward function, the state value function, and the learner's policy. Therefore, the state value function estimated by the inverse RL step is used to initialize the state value function of the forward RL step. The state value function updated by the forward RL step also provides an initial state value function of the inverse RL step.

Experimental results of the MuJoCo control benchmarks show that ERIL was more sample-efficient than the modern off-policy imitation learning algorithms in terms of environmental interactions in the forward RL step. We also showed that ERIL got better asymptotic performance in a vision-based, target-reaching task whose experimental results demonstrated the importance of the first discriminator, which does not appear in other imitation learning methods. ERIL also showed comparable performance in terms of the number of demonstrations provided by the expert. In pole-balancing experiments, we showed how ERIL can be applied to the analysis of human behaviors.

Declaration of competing interest

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

Acknowledgments

This work was supported by Innovative Science and Technology Initiative for Security, Japan Grant No. JPJ004596, ATLA, Japan and JST-Mirai Program Grant No. JPMJMI18B8, Japan to EU. This work was also supported by the Japan Society for the Promotion of Science, KAKENHI Grant Nos. JP16K21738, JP16H06561, and JP16H06563 to KD and JP19H05001 to EU. We thank Shoko Igarashi who collected the data of the human pole-balancing task.

Appendix A. Derivation

A.1. Derivation of Eq. (19)

Since we assume that agent-environment interaction is modeled as a Markov decision process, joint distribution is decomposed by Eq. (14). The log of the density ratio is given by

$$\ln \frac{\pi_k^L(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}')}{\pi^E(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}')} = \ln \frac{\pi_k^L(\boldsymbol{u} \mid \boldsymbol{x})}{\pi^E(\boldsymbol{u} \mid \boldsymbol{x})} + \ln \frac{\pi_k^L(\boldsymbol{x})}{\pi^E(\boldsymbol{x})}. \tag{A.1}$$

Assign selector variable $L = 1$ to the samples from the learner at the kth iteration and $L = -1$ to the samples from the expert:

$$\pi_k^L(\boldsymbol{x}) \triangleq \Pr(\boldsymbol{x} \mid L = 1),$$
$$\pi^E(\boldsymbol{x}) \triangleq \Pr(\boldsymbol{x} \mid L = -1) = 1 - \pi_k^L(\boldsymbol{x}).$$

Then the first discriminator is represented by

$$D_k^{(1)}(\boldsymbol{x}) \triangleq \frac{\Pr(L = 1 \mid \boldsymbol{x}) \Pr(L = 1)}{\Pr(\boldsymbol{x})}.$$

We obtain the following log of the density ratio:

$$\ln \frac{\pi_k^L(\boldsymbol{x})}{\pi^E(\boldsymbol{x})} = \ln \frac{D_k^{(1)}(\boldsymbol{x})}{1 - D_k^{(1)}(\boldsymbol{x})} - \ln \frac{\Pr(L = 1)}{\Pr(L = -1)}. \tag{A.2}$$

The log of density ratio $\ln \pi_k^L(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}')/\pi^L(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}')$ is represented by second discriminator $D_k^{(2)}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}')$ in the same way. Substituting the above results and Eq. (18) into Eq. (A.1) yields Eq. (19).

A.2. Derivation of Eq. (29)

The probability ratio of the learner and expert samples is simply estimated by the ratio of the number of samples (Sugiyama et al., 2012):

$$\ln \frac{\Pr(L = 1)}{\Pr(L = -1)} \approx \ln \frac{|\mathcal{D}^L|}{|\mathcal{D}^E|}.$$

When $|\mathcal{D}^L| = |\mathcal{D}^E|$, $\exp(g_k(\boldsymbol{x}))$ represents the following density ratio:

$$\exp(g_k(\boldsymbol{x})) = \pi_k^L(\boldsymbol{x})/\pi^E(\boldsymbol{x}).$$

Arranging Eq. (23) yields

$$D^{(2)}(\boldsymbol{x}, \boldsymbol{u}) = \frac{\pi_k^L(\boldsymbol{u} \mid \boldsymbol{x})}{\exp(-g_k(\boldsymbol{x}))\tilde{\pi}^E(\boldsymbol{u} \mid \boldsymbol{x}) + \pi_k^L(\boldsymbol{u} \mid \boldsymbol{x})},$$

and we immediately obtain Eq. (29). When term $\exp(-g_k(\boldsymbol{x}))$ is ignored, $D^{(2)}(\boldsymbol{x}, \boldsymbol{u})$ represents the optimal discriminator conditioned on the state.

Appendix B. Deterministic regularized autoencoders

This appendix briefly explains the deterministic RAE (Ghosh et al., 2020). For notational brevity, captured images and latent variables are denoted by $\boldsymbol{x}$ and $\boldsymbol{z}$. Suppose that the encoder network is given by mean $\boldsymbol{\mu}_e$ and covariance parameters $\boldsymbol{\sigma}_e$:

$$E(\boldsymbol{x}) = \boldsymbol{\mu}_e(\boldsymbol{x}) + \boldsymbol{\sigma}_e(\boldsymbol{x}) \odot \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}),$$

where $\odot$ denotes the Hadamard product. $\boldsymbol{I}$ is the identity matrix. The decoder network is also given by $\boldsymbol{\mu}_d$ and $\boldsymbol{\sigma}_d$ in the same way. The loss function of RAE is given by

$$J_{\mathrm{RAE}}(\boldsymbol{w}_e, \boldsymbol{w}_d) = \mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}} \left[\|\boldsymbol{x} - \boldsymbol{\mu}_d(E(\boldsymbol{x}))\|_2^2 + \lambda_2 \|\boldsymbol{z}\|_2^2 + \lambda_d \|\boldsymbol{w}_d\|_2^2 \right],$$

where λ_e and λ_d are hyperparameters. We set $\lambda_e = 10^{-6}$ and $\lambda_d = 10^{-7}$ according to a previous work (Yarats et al., 2020). Variables $\boldsymbol{w}_e$ and $\boldsymbol{w}_d$ represent the network weights of the encoder and the decoder.

Appendix C. Notes on the pole-balancing problem

C.1. Equations of motion

Fig. 10(b) shows the X–Z inverted pendulum on a pivot driven by horizontal and vertical forces. We used the equations of motion given in a previous work (Wang, 2012):

$$\begin{aligned} M(M+m)\ddot{x} &= Ml m\dot{\theta}^2 \sin\theta + (M + m\cos^2\theta)F_x - mF_z \sin\theta\cos\theta, \\ M(M+m)\ddot{z} &= Mml\dot{\theta}^2\cos\theta - (M + m\sin^2\theta)F_z - g, \qquad \text{(C.1)} \\ Ml\ddot{\theta} &= -F_x\cos\theta + F_z\sin\theta, \end{aligned}$$

where (x, z) is the position of the pivot in the xoz coordinate and $(\dot{x}, \dot{z})$ and $(\ddot{x}, \ddot{z})$ are the speed and acceleration. M and m are the mass of the pivot and the pendulum. l is the distance from the pivot to the center of the pendulum's mass. g denotes the gravitational acceleration. F_x and F_z are the horizontal and vertical forces.

The state vector is given by a three-dimensional vector, $\boldsymbol{x} = [x, \dot{x}, z, \dot{z}, \theta, \dot{\theta}]^\top$, and the action is represented by a two-dimensional vector, $\boldsymbol{u} = [F_x, F_z]^\top$. To implement the simulation, the time axis is discretized by $h = 0.01$ [s]. The parameters of the X–Z inverted pendulum are described below: $M = 0.85$ [kg], $m = 0.30$ [kg] (long pole) or 0.12 [kg] (short pole), and $l = 0.73$ [m] (long pole) or 0.29 [m] (short pole). $g = 9.81$ [m/s^2]. The inertia of the pendulum is negligible.

C.2. Evaluation of state transition probability

Since no action is available in the pole-balancing task, the learner's policy cannot be used for evaluation. We exploit the property where a linear transformation of a multivariate Gaussian random vector has a multivariate Gaussian distribution because the policy in this study is also represented by a Gaussian distribution. Using a first-order Taylor expansion, the time-discretized version of Eq. (C.1) can be expressed by $\boldsymbol{x}_{t+1} = \boldsymbol{A}_t\boldsymbol{x}_t + \boldsymbol{B}_t\boldsymbol{u}_t$.

References

Abbeel, P., & Ng, A. Y. (2004). Apprenticeship learning via inverse reinforcement learning. In *Proc. of the 21st International Conference on Machine Learning*.

Ahmed, Z., Le Roux, M. N. N., & Schuurmans, D. (2019). Understanding the impact of entropy on policy optimization. In *Proc. of the 36th International Conference on Machine Learning* pp.151–160.

Amit, R., Meir, R., & Ciosek, K. (2020). Discount Factor as a Regularizer in Reinforcement Learning. In *Proc. of the 37th International Conference on Machine Learning*.

Ashida, K., Kato, T., Hotta, K., & Oka, K. (2019). Multiple tracking and machine learning reveal dopamine modulation for area-restricted foraging behaviors via velocity change in caenorhabditis elegans. *Neuroscience Letters*, *706*, 68–74.

Azar, M. G., Gómez, V., & Kappen, H. J. (2012). Dynamic policy programming. *Journal of Machine Learning Research*, *13*, 3207–3245.

Belousov, B., & Peters, J. (2019). Entropic regularization of Markov decision processes. *Entropy*, *21*(7), 3207–3245.

Blondé, L., & Kalousis, A. (2019). Sample-Efficient Imitation Learning via Generative Adversarial Nets. In *Proc. of the 22nd International Conference on Artificial Intelligence and Statistics* pp.3138–3148.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., et al. (2016). Openai gym. *ArXiv Preprint*.

Chitta, S., Sucan, I., & Cousins, S. (2012). Moveit! [ROS topics]. *IEEE Robotics & Automation Magazine*, *19*(1), 18–19.

Collette, S., Pauli, W. M., Bossaerts, P., & O'Doherty, J. (2017). Neural computations underlying inverse reinforcement learning in the human brain. *ELife*, *6*.

Degris, T., White, M., & Sutton, R. S. (2012). Off-Policy Actor-Critic. In *Proc. of the 29th International Conference on Machine Learning*.

Dieng, A. B., Ruiz, F. J. R., Blei, D. M., & Titsias, M. K. (2019). Prescribed generative adversarial networks. *ArXiv Preprint*.

Doya, K. (2007). Reinforcement learning: Computational theory and biological mechanisms. *HFSP Journal*, *1*(1), 30–40.

Doya, K., & Uchibe, E. (2005). The Cyber Rodent Project: Exploration of adaptive mechanisms for self-preservation and self-reproduction. *Adaptive Behavior*, *13*, 149–160.

Elfwing, S., Uchibe, E., & Doya, K. (2018). Online Meta-Learning by Parallel Algorithm Competition. In *Proc. of the Genetic and Evolutionary Computation Conference* pp.426-433.

Fu, J., Luo, K., & Levine, S. (2018). Learning robust rewards with Adversarial Inverse Reinforcement Learning. In *Proc. of the 6th International Conference on Learning Representations*.

Fujimoto, S., van Hoof, H., & Meger, D. (2018). Addressing Function Approximation Error in Actor-Critic Methods. In *Proc. of the 35th International Conference on Machine Learning*.

Ghasemipour, S. K. S., Zemel, R., & Gu, S. (2019). A Divergence Minimization Perspective on Imitation Learning Methods. In *Proc. of the 3rd Conference on Robot Learning* pp.1259–1277.

Ghosh, P., Sajjadi, M. S. M., Vergari, A., Black, M., & Scholkopf, B. (2019). From Variational to Deterministic Autoencoders. In *Proc. of the 7th International Conference on Learning Representations*.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., et al. (2014). Generative adversarial nets. In *Advances in Neural Information Processing Systems 27* (pp. 2672–2680).

Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018). Soft Actor-Critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proc. of the 35th International Conference on Machine Learning* pp.1856–1865.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., et al. (2018). Soft actor-critic algorithms and applications. *ArXiv Preprint*.

Henderson, P., Chang, W.-D., Bacon, P.-L., Meger, D., Pineau, J., & Precup, D. (2018). OptionGAN: Learning Joint Reward-Policy Options using Generative Adversarial Inverse Reinforcement Learning. In *Proc. of the 32nd AAAI Conference on Artificial Intelligence*.

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., & Meger, D. (2018). Deep Reinforcement Learning that Matters. In *Proc. of the 32nd AAAI Conference on Artificial Intelligence*.

Hirakawa, T., Yamashita, T., Tamaki, T., Fujiyoshi, H., Umezu, Y., Takeuchi, I., et al. (2018). Can AI predict animal movements? Filling gaps in animal trajectories using inverse reinforcement learning. *Ecosphere*, (10).

Ho, J., & Ermon, S. (2016). Generative Adversarial Imitation Learning. In *Advances in Neural Information Processing Systems (vol.29)* pp.4565-4573.

Jaderberg, M., Dalibard, V., Osindero, S., Czarnecki, W. M., Donahue, J., Razavi, A., et al. (2017). Population based training of neural networks. *ArXiv Preprint*.

Jena, R., Liu, C., & Sycara, K. (2020). Augmenting GAIL with BC for sample efficient imitation learning. In *Proc. of the 3rd Conference on Robot Learning*.

Ke, L., Barnes, M., Sun, W., Lee, G., Choudhury, S., & Srinivasa, S. Imitation Learning as f-Divergence Minimization In *Proc. of the 14th International Workshop on the Algorithmic Foundations of Robotics (WAFR)*.

Kingma, D., & Ba, J. (2015). ADAM: A Method for Stochastic Optimization. In *Proc. of the 3rd International Conference for Learning Representations*.

Kinose, A., & Taniguchi, T. (2020). Integration of imitation learning using GAIL and reinforcement learning using task-achievement rewards via probabilistic graphical model. *Advanced Robotics*, (16), 1055–1067.

Kobayashi, K., Horii, T., Iwaki, R., Nagai, Y., & Asada, M. (2019). Situated GAIL: Multitask imitation using task-conditioned adversarial inverse reinforcement learning. *ArXiv Preprint*.

Kober, J., Bagnell, J. A., & Peters, J. (2013). Reinforcement learning in robotics: A survey. *International Journal of Robotics Research*, *32*(11), 1238–1274.

Kostrikov, I., Agrawal, K. K., Dwibedi, D., Levine, S., & Tompson, J. (2019). Discriminator-Actor-Critic: Addressing Sample Inefficiency and Reward Bias in Adversarial Imitation Learning. In *Proc. of the 7th International Conference on Learning Representations*.

Kozuno, T., Uchibe, E., & Doya, K. (2019). Theoretical Analysis of Efficiency and Robustness of Softmax and Gap-Increasing Operators in Reinforcement Learning. In *Proc. of the 22nd International Conference on Artificial Intelligence and Statistics.* pp.2995–3003.

Kretzschmar, H., Spies, M., Sprunk, C., & Burgard, W. (2016). Socially compliant mobile robot navigation via inverse reinforcement learning. *The International Journal of Robotics Research*.

Laskey, M., Lee, J., Fox, R., Dragan, A., & Goldberg, K. (2017). DART: Noise Injection for Robust Imitation Learning. In *Proc. of the 1st Conference on Robot Learning*.

Lee, B.-J., Lee, J., Vrancx, P., Kim, D., & Kim, K.-E. (2020). Batch Reinforcement Learning with Hyperparameter Gradients. In *Proc. of the 37th International Conference on Machine Learning*.

Li, H., Liu, D., & Wang, D. (2018). Manifold regularized reinforcement learning. *IEEE Transactions on Neural Networks and Learning Systems*, *29*(4), 932–943.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., et al. (2016). Continuous control with deep reinforcement learning. In *Proc. of the 4th International Conference on Learning Representations*.

Liu, S., Araujo, M., Brunskill, E., Rossetti, R., Barros, J., & Krishnan, R. (2013). Understanding sequential decisions via inverse reinforcement learning. In *Proc. of the 14th IEEE International Conference on Mobile Data Management* (pp. 177–186). IEEE.

Liu, Z., Li, X., Kang, B., & Darrell, T. (2021). Regularization Matters in Policy Optimization – An Empirical Study on Continuous Control. In *Proc. of the 9th International Conference on Learning Representations*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533.

Muelling, K., Boularias, A., Mohler, B., Schölkopf, B., & Peters, J. (2014). Learning strategies in table tennis using inverse reinforcement learning.. *Biological Cybernetics*, *108*(5), 603–619.

Neu, G., & Szepesvári, C. (2009). Training parsers by inverse reinforcement learning. *Machine Learning*, *77*(2–3), 303–337.

Ng, A. Y., & Russell, S. (2000). Algorithms for inverse reinforcement learning. In *Proc. of the 17th International Conference on Machine Learning*.

Nishio, D., Kuyoshi, D., Tsuneda, T., & Yamane, S. (2020). Discriminator soft actor critic without extrinsic rewards. *ArXiv Preprint*.

Odekunle, A., Gao, M. D. W., & Jiang, Z.-P. (2020). Reinforcement learning and non-zero-sum game output regulation for multi-player linear uncertain systems. *Automatica*, *112*.

Ohnishi, S., Uchibe, E., Yamaguchi, Y., Nakanishi, K., Yasui, Y., & Ishii, S. (2019). Constrained deep Q-learning gradually approaching ordinary Q-learning. *Frontiers in Neurorobotics*, *13*(103).

OpenAI, Akkaya, I., Andrychowicz, M., Chociej, M., Litwin, M., McGrew, B., et al. (2019). Solving rubik's cube with a robot hand. *ArXiv Preprint*.

OpenAI, Berner, C., Brockman, G., Chan, B., Cheung, V., Debiak, P., et al. (2019). Dota 2 with large scale deep reinforcement learning. *ArXiv Preprint*.

Parisi, S., Tangkaratt, V., Peters, J., & Khan, M. E. (2019). TD-regularized actor-critic methods. *Machine Learning*, (8), 1467–1501.

Peters, J., & Schaal, S. (2008). Reinforcement learning of motor skills with policy gradients. *Neural Networks*, (4), 1–13.

Pomerleau, D. A. (1989). ALVINN: An autonomous land vehicle in a neural network. In *Advances in Neural Information Processing Systems 1* (pp. 305–313).

Reddy, S., Dragan, A. D., & Levine, S. (2020). SQIL: Imitation Learning via Regularized Behavioral Cloning.In *Proc. of the 8th International Conference on Learning Representations*.

Ross, S., Gordon, G., & Bagnell, D. (2011). A reduction of imitation learning and structured prediction to no-regret online learning. In *Proc. of the 14th International Conference on Artificial Intelligence and Statistics* pp.627-635.

Sasaki, F., Yohira, T., & Kawaguchi, A. (2019). Sample Efficient Imitation Learning for Continuous Control. In *Proc. of the 7th International Conference on Learning Representations*.

Schaul, T., Horgan, D., Gregor, K., & Silver, D. (2015). Universal Value Function Approximators. In *Proc. of the 32nd International Conference on Machine Learning* pp.1312–1320.

Shimosaka, M., Kaneko, T., & Nishi, K. (2014). Modeling risk anticipation and defensive driving on residential roads with inverse reinforcement learning. In *Proc. of the 17th International IEEE Conference on Intelligent Transportation Systems* pp.1694–1700.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., et al. (2017). Mastering the game of go without human knowledge. *Nature*, *550*(7676), 354–359.

Sugiyama, M., Suzuki, T., & Kanamori, T. (2012). *Density Ratio Estimation in Machine Learning*. Cambridge University Press.

Sun, M., & Ma, X. (2014). Adversarial Imitation Learning from Incomplete Demonstrations. *Proc. of the 28th International Joint Conference on Artificial Intelligence*.

Sutton, R., & Barto, A. (1998). *Reinforcement learning*. MIT Press.

Todorov, E., Erez, T., & Tassa, Y. (2012). MuJoCo: A physics engine for model-based control. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems.* pp.5026–5033.

Torabi, F., Warnell, G., & Stone, P. (2018). Behavioral Cloning from Observation. In *Proc. of the 27th International Joint Conference on Artificial Intelligence and the 23rd European Conference on Artificial Intelligence* pp.4950-4957.

Torabi, F., Warnell, G., & Stone, P. (2019). Generative adversarial imitation from observation. In *ICML 2019 Workshop on Imitation, Intent, and Interaction*.

Tsurumine, Y., Cui, Y., Uchibe, E., & Matsubara, T. (2019). Deep reinforcement learning with smooth policy update: Application to robotic cloth manipulation. *Robotics and Autonomous Systems*, *112*, 72–83.

Uchibe, E. (2018). Model-free deep inverse reinforcement learning by logistic regression. *Neural Processing Letters*, *47*(3), 891–905.

Uchibe, E., & Doya, K. (2014). Inverse Reinforcement Learning Using Dynamic Policy Programming. In *Proc. of IEEE International Conference on Development and Learning and Epigenetic Robotics* pp.222–228.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., et al. (2019). Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, *575*(7782), 350–354.

Vogel, A., Ramachandran, D., Gupta, R., & Raux, A. (2012). Improving hybrid vehicle fuel efficiency using inverse reinforcement learning. In *Proc. of the 26th AAAI Conference on Artificial Intelligence*.

Wang, J.-J. (2012). Stabilization and tracking control of X-Z inverted pendulum with sliding-mode control. *ISA Transactions*, *51*(6), 763–770.

Wang, D., & Qiao, J. (2019). Approximate neural optimal control with reinforcement learning for a torsional pendulum device. *Neural Networks*, *117*(6), 1–7.

Xia, C., & El Kamel, A. (2016). Neural inverse reinforcement learning in autonomous navigation. *Robotics and Autonomous Systems*, *84*, 1–14.

Yamaguchi, S., Honda, N., Ikeda, Y., Nakano, S., Mori, I., & Ishii, S. (2018). Identification of animal behavioral strategies by inverse reinforcement learning. *PLoS Computational Biology*.

Yarats, D., Zhang, A., Kostrikov, I., Amos, B., Pineau, J., & Fergus, R. (2020). Improving sample efficiency in model-free reinforcement learning from images. *ArXiv Preprint*.

Zhang, B., Rajan, R., Pineda, L., Lambert, N., Biedenkapp, A., Chua, K., et al. On the Importance of Hyperparameter Optimization for Model-based Reinforcement Learning. In *Proc. of the 24th International Conference on Artificial Intelligence and Statistics* pp.4015–4023.

Ziebart, B. D., Maas, A., Bagnell, J. A., & Dey, A. K. (2008). Maximum Entropy Inverse Reinforcement Learning. In *Proc. of the 23rd AAAI Conference on Artificial Intelligence*.

Zuo, G., Chen, K., Lu, J., & Huang, X. (2020). Deterministic generative adversarial imitation learning. *Neurocomputing*, 60–69.