

Highlights

Imitation learning based on entropy-regularized forward and inverse reinforcement learning

Eiji Uchibe, Kenji Doya

- Imitation learning via forward and inverse reinforcement learning is developed
- Forward and inverse reinforcement learning share neural networks and hyperparameters
- Both of forward and inverse reinforcement learning update the state value function
- Discriminator is derived from the soft Bellman optimality equation

Imitation learning based on entropy-regularized forward and inverse reinforcement learning

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*

Abstract

This paper proposes Entropy-Regularized Imitation Learning (ERIL), which is a combination of forward and inverse reinforcement learning under the framework of the entropy-regularized Markov decision process. ERIL minimizes the reverse Kullback-Leibler (KL) divergence between two probability distributions induced by a learner and an expert. Inverse reinforcement learning (RL) in ERIL evaluates the log-ratio between two distributions using the density ratio trick, which is widely used in generative adversarial networks. More specifically, the log-ratio is estimated by building two binary discriminators. The first discriminator is a state-only function, and it tries to distinguish the state generated by the forward RL step from the expert's state. The second discriminator is a function of current state, action, and transitioned state, and it distinguishes the generated experiences from the ones provided by the expert. Since the second discriminator has the same hyperparameters of the forward RL step, it can be used to control the discriminator's ability. The forward RL minimizes the reverse KL estimated by the inverse RL. We show that minimizing the reverse KL divergence is equivalent to finding an optimal policy under entropy regularization. Consequently, a new policy is derived from an algorithm that resembles Dynamic Policy Programming and Soft Actor-Critic. Our experimental results on MuJoCo-simulated environments show that ERIL is more sample-efficient than such previous methods. We further apply the method to human behaviors in performing a pole-balancing task and show that the estimated reward functions show how every subject achieves the goal.

Keywords: reinforcement learning, inverse reinforcement learning, imitation learning, entropy regularization

*Corresponding author.

Email addresses: `uchibe@atr.jp` (Eiji Uchibe), `doya@oist.jp` (Kenji Doya)

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 (Sutton and Barto, 1998; Doya, 2007; Kober et al., 2013). However, there remain several open questions in RL, and one of the critical problems is how to design and prepare an appropriate reward function for a given task. It is easy to design a sparse reward function that gives a positive reward when the task is accomplished and zero otherwise, but that makes it hard to find an optimal policy due to prohibitively long learning time. On the other hand, we can accelerate the speed of learning by using a complicated function that gives a non-zero reward signal most of the time. However, we often observe that optimized behaviors can deviate from an experimenter’s intention if the reward function is too complicated.

In some situations, it is easier to prepare examples of a desired behavior than to hand-craft an appropriate reward function. Several methods of Inverse Reinforcement Learning (IRL) (Ng and Russell, 2000) and apprenticeship learning (Abbeel and Ng, 2004) have been proposed as ways to retrieve a reward function from an expert’s behaviors and to implement imitation learning. Currently available applications include a probabilistic driver route prediction system (Vogel et al., 2012; Liu et al., 2013), modeling risk anticipation and defensive driving of drivers (Shimosaka et al., 2014), investigating human behaviors in table tennis (Muelling et al., 2014), robot navigation tasks (Kretschmar et al., 2016; Xia and El Kamel, 2016), analyzing animal behaviors (Ashida et al., 2019; Hirakawa et al., 2018; Yamaguchi et al., 2018), and parser training (Neu and Szepesvári, 2009). A recent functional magnetic resonance imaging (fMRI) study suggests that the anterior part of dorsomedial prefrontal cortex (dmPFC) is likely to encode the inverse reinforcement learning algorithm (Collette et al., 2017). Combining inverse RL with standard RL is a promising approach to find an optimal policy from expert’s demonstrations. Hereafter, we use the term “forward” reinforcement learning to clarify the difference.

Recently, some works (Fu et al., 2018; Ho and Ermon, 2016) have connected forward and inverse RL and Generative Adversarial Networks (GANs) (Goodfellow et al., 2014), which exhibited remarkable success in such domains as image generation, video prediction, and machine translation. 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 tries to create experiences that are not distinguishable by an inverse RL. Generative Adversarial Imitation Learning (GAIL) (Ho and Ermon, 2016) and Adversarial Inverse Reinforcement Learning (AIRL) (Fu et al., 2018) showed that the iterative process of forward and inverse RL produces policies that outperformed behavior cloning, which learns a policy as a supervised learning problem. However, GAIL and AIRL do not consider the sample efficiency in the forward RL step, although they are sample-efficient with respect to the number of demonstrations from the expert. GAIL and AIRL adopt Trust Region Policy Optimization (TRPO) (Schulman

et al., 2015) as an on-policy forward RL algorithm, but the relation between TRPO and inverse reinforcement learning remains unclear. Therefore, utilizing the result of an inverse RL step to a forward RL step and vice versa is difficult.

To further improve the sample-efficiency, this paper proposes Entropy-Regularized Imitation Learning (ERIL) that minimizes an information projection or an I-projection, which is also known as the reverse Kullback-Leibler (KL) divergence between two probability distributions induced by a learner and an expert. Figure 1 shows the overall architecture of ERIL. The difficulty is that the reverse KL divergence cannot be computed because the expert’s distribution is unknown. Our idea is to apply the density ratio trick (Sugiyama et al., 2012) to evaluate the log-ratio between the two distributions from samples drawn from the distributions. 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’s policy and the expert’s policy. Consequently, computing the log-ratio can be achieved by training two binary discriminators. One is a state-only discriminator, which tries to distinguish the state generated by the learner from the expert’s state. The second discriminator is a function of a tuple of state, action, and next-state, and it also tries to distinguish the learner’s experience and the expert’s one. The second discriminator is represented by reward, state value function, and the log-ratio of the first discriminator. We show that the AIRL and LogReg-IRL discriminators (Uchibe and Doya, 2014; Uchibe, 2018) are a special case of that of ERIL. The loss function is essentially the same as that of GAN for training the discriminators, which are efficiently trained by logistic regression.

After evaluating the log-ratio, the forward RL in ERIL minimizes the 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 et al., 2012), Soft Actor-Critic (Haarnoja et al., 2018), and conservative value iteration (Kozuno et al., 2019). One important feature of ERIL is sample efficiency in terms of the number of environmental interactions because the state value function is updated in both the inverse RL and forward RL steps.

We perform experiments with the MuJoCo benchmark tasks (Todorov et al., 2012) in the OpenAI gym (Brockman et al., 2016). The experimental results demonstrate that ERIL is comparable with some modern imitation learning algorithms in terms of the number of trajectories from expert data and outperformed the sample efficiency in terms of the number of trajectories in the forward RL step. Ablation study shows that entropy regularization plays a critical role in improving sample efficiency. Then, we apply ERIL to human behaviors in performing a pole-balancing task. Since the actions of the human subjects are not observable, the task is an example of realistic situations. ERIL recovers the subjects’ policy better than baselines. We also show that the estimated reward functions show how every subject achieves the goal.

2. Related work

2.1. Behavior cloning

Behavior Cloning (BC) directly maximizes the log-likelihood of the expert’s action. Pomerleau (1989) realizes Autonomous Land Vehicle In a Neural Network (ALVINN), which is viewed as the ancestor of autonomous driving cars. The policy of ALVINN is implemented by a 3-layer neural network, and it learns a mapping from video and range finder inputs to steering direction via supervised learning. However, it is known that ALVINN suffers from the covariate shift problem. To reduce covariate shift, Ross et al. (2011) propose an iterative method called Dataset Aggregation (DAGGER), where the learner runs its policy while the expert is asked to provide the correct action for the states visited by the learner. Laskey et al. (2017) propose Disturbances for Augmented Robot Trajectories (DART) that collects the expert’s dataset with injected noise.

It is often difficult to provide the expert’s data in the form of state-action pairs in a realistic situation such as analyzing animal behaviors and learning from videos. Torabi et al. (2018) consider the situation in which the expert’s action is not available. They propose Behavioral Cloning from Observation (BCO) that estimates the action from the inverse dynamics model. Soft Q Imitation Learning (SQIL) (Reddy et al., 2020) is the BC with the regularization term that penalizes large squared soft Bellman error. SQIL is implemented by SAC, which assigns the reward of the expert’s data to 1 and that of generated data to 0. SQIL can learn from both of the expert’s and the learner’s samples because SAC is an off-policy algorithm.

2.2. Generative adversarial imitation learning

GAIL (Ho and Ermon, 2016) is one of the most popular imitation learning algorithms, and it formulates the objective of imitation learning as the training objective of GANs. Its discriminator tries to discriminate the generated state-action pairs from the expert’s ones while the generator acts as a forward reinforcement learning algorithm, and it maximizes the sum of rewards computed by the discriminator. There are several extensions of GAN. AIRL (Fu et al., 2018) uses the optimal discriminator of which the expert’s distribution approximated by a disentangled reward function. A similar discriminator is independently proposed by (Uchibe and Doya, 2014; Uchibe, 2018), which is derived from the framework of entropy-regularized reinforcement learning and density ratio estimation. AIRL experimentally shows that the learned reward function can transfer to new, unseen environments. Situated GAIL (Kobayashi et al., 2019) extends GAIL to learn multiple reward functions and multiple policies by introducing a task variable to both the discriminator and the generator. Kinose and Taniguchi (2019) integrates the GAIL discriminator with reinforcement learning, in which the policy is trained with both the original reward and the additional rewards calculated by the discriminator.

As discussed in the previous section, the expert’s action is not sometimes available. IRLGAN (Henderson et al., 2018) is the special case of GAN of which the discriminator is given as a function of state. Torabi et al. (2019)

propose Generative Adversarial Imitation from Observation (GAIfO), of which the functions are characterized by the state transitions. Sun and Ma (2014) propose Action Guided Adversarial Imitation Learning (AGAIL) that can deal with expert’s demonstrations with incomplete action sequences. AGAIL uses mutual information between the expert’s action and the generated one as an additional regularizer for the training objective.

To reduce the number of interactions in the forward RL step, Blondé and Kalousis (2019) propose Sample-efficient Adversarial Mimic (SAM) that adopts the off-policy method so-called the Deep Deterministic Policy Gradients (DDPG) algorithm (Lillicrap et al., 2016). SAM maintains three different neural networks, and they approximate a reward function, a state-action value function, and a policy. The reward function is estimated in the same way as in GAN, while the state-action value function and the policy are trained by DDPG. Kostrikov et al. (2019) show that the reward function of GAIL is biased and absorbing states are not treated appropriately. They propose a preprocessing technique on the expert’s data before learning and develop the Discriminator Actor-Critic (DAC) algorithm. DAC utilizes the Twin Delayed Deep Deterministic policy gradient (TD3) algorithm (Fujimoto et al., 2018), which is an extension of DDPG. Sasaki et al. (2019) exploit the Bellman equation to represent the reward function, and the exponential transformation of the reward is trained as a kind of the discriminator. They use the off-policy actor-critic (Degris et al., 2012) to improve the policy. Zuo et al. (2020) propose Deterministic GAIL that adopts the modified DDPG algorithm that incorporates the behavior cloning loss in the forward RL step. Discriminator Soft Actor-Critic (Nishio et al., 2020) extends SQIL by estimating the reward function via the AIRL-like discriminator. Ghasemipour et al. (2019) and Ke et al. (2020) show that the relationship between several imitation learning algorithms from the viewpoint of objective functions. Since our study mainly focuses on the sample efficiency with respect to the number of interactions with the environment, the most related works are SAM, DAC, and Sasaki’s method. Our method is compared with these three methods in experiments.

3. Preliminaries

3.1. Markov decision process

Here, we give a brief introduction to the Markov Decision Process (MDP) for a discrete-time domain. Let \mathcal{X} and \mathcal{U} be continuous or discrete state and action spaces, respectively. At a time step t , a learning agent observes an environmental current state $\mathbf{x}_t \in \mathcal{X}$ and executes an action $\mathbf{u}_t \in \mathcal{U}$ sampled from a stochastic policy $\pi(\mathbf{u}_t | \mathbf{x}_t)$. Consequently, an immediate reward $\tilde{r}(\mathbf{x}_t, \mathbf{u}_t)$ is given from the environment, and the environment makes a state transition according to a state transition probability $p_T(\mathbf{x}'_t | \mathbf{x}_t, \mathbf{u}_t)$ from \mathbf{x}_t to $\mathbf{x}'_t = \mathbf{x}_{t+1} \in \mathcal{X}$ by executing the action \mathbf{u}_t .

The goal of forward reinforcement learning is to construct an optimal policy $\pi(\mathbf{u} | \mathbf{x})$ that maximizes the given objective function. Among the several

available objective functions, the most widely used one is a discounted sum of rewards defined by

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

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

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

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

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

Equation (1) is a nonlinear equation due to the max operator, and it is usually difficult to find the action that maximizes the right hand side of Eq.(1).

3.2. Entropy-regularized Markov decision process

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

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

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

$$V(\mathbf{x}) = \frac{1}{\beta} \ln \int \exp(\beta Q(\mathbf{x}, \mathbf{u})) d\mathbf{u}, \quad (3)$$

where β is the positive hyperparameter defined by

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

and $Q(\mathbf{x}, \mathbf{u})$ is the optimal soft state-action value function, defined by

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

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

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

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

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):

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

The derivative of the KL divergence is given by

$$\nabla \text{KL} = \mathbb{E}_{\mathbf{u} \sim \pi(\cdot \mid \mathbf{x})} [\nabla_{\mathbf{w}_\pi} \ln \pi(\cdot \mid \mathbf{x}) [\ln \pi(\cdot \mid \mathbf{x}) - \beta(Q(\mathbf{x}, \cdot) - V(\mathbf{x})) + B(\mathbf{x})]], \quad (6)$$

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

3.3. Generative Adversarial Networks

Generative Adversarial Networks (GANs) are a class of neural networks that can approximate the probability distribution based on a game theoretic scenario (Goodfellow et al., 2014). The standard GANs consist of a generator and a discriminator. Here, the generator is given by a probability distribution $p^L(\mathbf{z})$ for simplicity. The discriminator is a function whose role is to distinguish samples from the generator and the expert, and it is denoted by $D(\mathbf{z})$. The discriminator minimizes the following negative log likelihood:

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

One can show that the optimal discriminator has the following shape (Goodfellow et al., 2014):

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

The generator tries to minimize $-J_{\text{GAN}}^{(D)}$, and it is given by

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

where the second term of the right hand side of Eq. (7) is removed because it is constant with respect to the generator. Recently, Prescribed GAN (PresGAN) introduces the entropy regularization term to J_{GAN} to mitigate mode collapse (Dieng et al., 2019).

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

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

where λ_{GAIL} is a positive hyperparameter. Adding the entropy term is the key to associate with entropy-regularized MDP. The objective function of the GAIL discriminator is essentially the same as that of the GANs discriminator while the GAIL discriminator update the policy by TRPO with the reward function defined by

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

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

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

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

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

Note that the AIRL discriminator (10) has no hyperparameters. A similar discriminator is proposed by (Uchibe and Doya, 2014; Uchibe, 2018). Besides, the reward function of AIRL is calculated by

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

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

3.4. Behavior cloning

Behavior Cloning (BC) is the most widely used imitation learning. BC minimizes the “forward” KL divergence which is also known as M-projection (Ghasemipour et al., 2019; Ke et al., 2020):

$$J_{\text{BC}}(\mathbf{w}_\pi) = \text{KL}(\pi^E \parallel \pi^L) = -\mathbb{E}_{\pi^E} [\ln \pi^L(\mathbf{u} \mid \mathbf{x})] + C, \quad (13)$$

where \mathbf{w}_π is the policy parameter vector of $\pi^L(\mathbf{u} \mid \mathbf{x})$, 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. However, it suffers from the problem of 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 $\pi^E(\mathbf{u} \mid \mathbf{x})$ and the learner’s policy as $\pi^L(\mathbf{u} \mid \mathbf{x})$. Suppose that we have a dataset of transitions generated by

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

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

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

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

Entropy-Regularized Imitation Learning (ERIL) minimizes the “reverse” KL divergence given by

$$J_{\text{ERIL}}(\mathbf{w}_\pi) = \text{KL}(\pi^L \parallel \pi^E), \quad (15)$$

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

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

The difficulty is how one evaluates the the log-ratio, $\ln \pi^L(\mathbf{x}, \mathbf{u}, \mathbf{x}') / \pi^E(\mathbf{x}, \mathbf{u}, \mathbf{x}')$, because $\pi^E(\mathbf{x}, \mathbf{u}, \mathbf{x}')$ is not known. The basic idea to resolve 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’s reward function is given by

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

where k is the index of iteration. $r(\mathbf{x})$ is a state-only reward function parameterized by \mathbf{w}_r . It is possible to use a state-action reward function, but 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’s reward function is given by Eq. (16), the expert’s state-action value function satisfies the following soft Bellman optimality equation:

$$Q_k(\mathbf{x}, \mathbf{u}) = r_k(\mathbf{x}) + \frac{1}{\eta} \ln \pi_k^L(\mathbf{u} \mid \mathbf{x}) + \gamma \mathbb{E}_{\mathbf{x}' \sim p_T(\cdot \mid \mathbf{x}, \mathbf{u})} [V(\mathbf{x}')]. \quad (17)$$

The state value function and the expert policy are expressed by Eq. (3) and Eq. (5), respectively.

4.2. Inverse reinforcement learning based on density ratio estimation

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

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

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

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

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

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

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

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

where

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

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

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

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

where $\mathbf{x} \sim \mathcal{D}$ means that the transition data are drawn from \mathcal{D} and \mathcal{D}_k^L is the dataset of transitions provided by the learner's policy $\pi_k^L(\mathbf{u} | \mathbf{x})$ at the

k -th iteration. $D_k^{(2)}(\mathbf{x}, \mathbf{u}, \mathbf{x}')$ is trained in the same way, in which $g_k(\mathbf{x})$ and $\pi_k^L(\mathbf{u} \mid \mathbf{x})$ are fixed during training. The goal of $D_k^{(2)}(\mathbf{x}, \mathbf{u}, \mathbf{x}')$ is to maximize the following log likelihood:

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

This simply minimizes the cross entropy function of binary classifier $D_k^{(2)}(\mathbf{x}, \mathbf{u}, \mathbf{x}')$ that tries to separate the generated data from the expert's one. Apparently, 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 evaluate the expectation correctly, but Kostrikov et al. (2019) showed that it works well without using importance sampling.

4.3. Forward reinforcement learning based on KL minimization

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

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

Then, the ERIL's objective function is expressed by

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

Its derivative is expressed by

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

When $g_k(\mathbf{x}) = B(\mathbf{x})$, Eq. (25) is essentially the same as Eq. (6).

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

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

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

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

with

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

Algorithm 1 Entropy-Regularized Imitation Learning

Input: Expert’s dataset \mathcal{D}^E and hyperparameters κ and η

Output: Learner’s policy π^L and estimated reward $r(\mathbf{x})$.

- 1: Initialize all parameters of the networks and replay buffer \mathcal{D}^L .
 - 2: **for** $k = 0, 1, 2, \dots$ **do**
 - 3: Sample $\tau_i = \{(\mathbf{x}_t, \mathbf{u}_t, \mathbf{x}_{t+1})\}_{t=0}^T$ with π_k^L and store it in \mathcal{D}^L .
 - 4: Update \mathbf{w}_g with the gradient of Eq. (21).
 - 5: Update \mathbf{w}_r and \mathbf{w}_V with the gradient of Eq. (22).
 - 6: Update \mathbf{w}_Q with the gradient of Eq. (27).
 - 7: Update \mathbf{w}_V with the gradient of Eq. (26).
 - 8: Update \mathbf{w}_π with the gradient of Eq. (24).
 - 9: Update $\bar{\mathbf{w}}_V$ by Eq. (28).
 - 10: **end for**
-

where $\tilde{V}_k(\mathbf{x})$ is the target state value function parameterized by $\bar{\mathbf{w}}_V$. Two alternatives can be chosen to update $\bar{\mathbf{w}}_V$. The first one is periodic update, i.e., the target network is synchronized with the current \mathbf{w}_g at regular interval (Mnih et al., 2015). We use the second alternative, which is Polyak averaging update, where $\bar{\mathbf{w}}_g$ is updated by weighted average over the past parameters (Lillicrap et al., 2016). Specifically,

$$\bar{\mathbf{w}}_V \leftarrow \tau \mathbf{w}_V + (1 - \tau) \bar{\mathbf{w}}_V \quad (28)$$

with $\tau \ll 1$.

Algorithm 1 shows the overview of Entropy-Regularized Imitation Learning. The lines 4-5 and 6-8 represent the inverse RL step and forward RL step, respectively. Note that \mathbf{w}_V is updated twice in each iteration.

4.4. Interpretation of the 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:

$$D^{(2)}(\mathbf{x}, \mathbf{u}) = \frac{\pi^L(\mathbf{x})\pi^L(\mathbf{u} | \mathbf{x})}{\pi^E(\mathbf{x})\tilde{\pi}^E(\mathbf{u} | \mathbf{x}) + \pi^L(\mathbf{x})\pi^L(\mathbf{u} | \mathbf{x})}, \quad (29)$$

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

4.5. Extension

We describe two extensions to deal with more realistic situations. One is to learn multiple policies from multiple experts. The dataset of experts is augmented by adding a conditioning variable c that represent the index of the

expert:

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

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 by

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

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

The other extension is to deal with the case where actions are not observed. ERIL needs to observe the expert’s 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 and Doya, 2014; Uchibe, 2018). Alternative is to exploit an inverse dynamics model (Torabi et al., 2018). It is formulated as a maximum-likelihood estimation problem, i.e., we maximize the following function:

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

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

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

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, Hopper, Walker, Reacher, Half-Cheetah, Ant, and Humanoid, provided by the OpenAI gym (Brockman et al., 2016). The goal for all the tasks was to move forward as quickly as possible. First, an optimal policy $\pi^E(\mathbf{u} | \mathbf{x})$ 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 $(\mathbf{x}, \mathbf{u}, \mathbf{x}')$. The same demonstration data were used to train all the algorithms to make a fair comparison. We compared ERIL with BC, GAIL, Sasaki, DAC, and SAM. The network architectures to approximate functions were shown in Table 1. We use

Table 1: Neural network architectures used in the MuJoCo benchmark control tasks. For example, $V(\mathbf{x})$ 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
$\mu(\mathbf{x})$	(dim \mathcal{X} , 256, 256, dim \mathcal{U})	HalfCheetah and Humanoid
	(dim \mathcal{X} , 100, 100, dim \mathcal{U})	other tasks
$\sigma(\mathbf{x})$	(dim \mathcal{X} , 100, dim \mathcal{U})	all the tasks
$r(\mathbf{x})$	(dim \mathcal{X} , 100, 100, 1)	all the tasks
$V(\mathbf{x})$	(dim \mathcal{X} , 256, 256, 1)	HalfCheetah and Humanoid
	(dim \mathcal{X} , 100, 100, 1)	other tasks
$Q(\mathbf{x}, \mathbf{u})$	(dim $\mathcal{X} + \dim \mathcal{U}$, 256, 256, 1)	HalfCheetah and Humanoid
	(dim $\mathcal{X} + \dim \mathcal{U}$, 100, 100, 1)	other tasks
$g(\mathbf{x})$	(dim \mathcal{X} , 100, 100, 1)	all the tasks
$D(\mathbf{x}, \mathbf{u})$	(dim $\mathcal{X} + \dim \mathcal{U}$, 256, 256, 1)	HalfCheetah and Humanoid
	(dim $\mathcal{X} + \dim \mathcal{U}$, 100, 100, 1)	other tasks

a Rectified Linear Unit (ReLU) activation function in the hidden layers. The output nodes except $\mu(\mathbf{x})$ and $\sigma(\mathbf{x})$ use a linear activation function. The functions $\mu(\mathbf{x})$ and $\sigma(\mathbf{x})$ were used to represent the learner’s policy by a Gaussian distribution:

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

where $\mu(\mathbf{x})$ and $\sigma(\mathbf{x})$ denote the mean and the diagonal covariance matrix, respectively. The output nodes of $\mu(\mathbf{x})$ and $\sigma(\mathbf{x})$ use a tanh and a sigmoid function, respectively.

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

5.2. Comparative evaluation

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

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

where R , R_0 , and R^E denote the total return of the learner, that of a randomly initialized Gaussian policy, and that of the expert trained by TRPO, respectively. The horizontal axis depicts the number of interactions with the environment in the logarithmic scale. The figure shows that ERIL, Sasaki, DAC, and SAM have a considerably better sample efficiency than BC and GAIL in all the MuJoCo control tasks. In particular, ERIL performed consistently across all

tasks and outperforming the baseline methods in all the tasks. DAC showed better performance than Sasaki and SAM in Walker2d, HalfCheetah, and Ant, and competitive performance in Hopper, Reacher, and Humanoid. Sasaki achieved better performance than SAM in Walker2D, but there was no significant difference between Sasaki and SAM in our implementation. GAIL was apparently not very competitive because the on-policy TRPO algorithm used in the GAIL’s forward RL step could not use the technique of experience replay.

Figure 4 shows the normalized return of evaluation rollout of the six algorithms with respect to the number of demonstrations. The figure shows that ERIL, Sasaki, DAC, and SAM were more sample efficient than BC and GAIL. In particular, ERIL was consistently one of the most sample efficient algorithms when the number of demonstrations is small. The performance of DAC 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 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 the M-projection that avoids $\pi^L = 0$ whenever $\pi^E > 0$. Therefore, the policy trained by BC averages over several modes, even if the policy is approximated by a Gaussian distribution. On the contrary, ERIL is the I-projection that forces π^L to be zero even if $\pi^E > 0$. The policy trained by ERIL concentrates on a single mode. Nevertheless, 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 components of ERIL contribute most to its performance via an ablation study on the Ant environment. Specifically, we tested five different components:

1. ERIL without the first discriminator, i.e., we set $g_k(\mathbf{x}) = 0$ for all \mathbf{x} .
2. ERIL without sharing the state value function between forward and inverse RL. The forward RL step updates the policy with the reward $r_k(\mathbf{x})$.
3. ERIL without sharing the state value function between forward and inverse RL. The forward RL step updates the policy with the shaping reward $r_k(\mathbf{x}) + \gamma V_k(\mathbf{x}') - V_k(\mathbf{x})$ for the state transition $(\mathbf{x}, \mathbf{x}')$.
4. ERIL without sharing the hyperparameters. We set $\beta = 0$ and $\kappa = 0$ in Eq. (20), while β and η of the forward RL step are used as they are.
5. ERIL without the soft Bellman equation. The discriminator $D_k^{(3)}(\mathbf{x}, \mathbf{u}, \mathbf{x}')$ is defined by

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

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

Figure 5 compares the learning curves. In the benchmark tasks, there was no significant difference between the original ERIL and ERIL without the first discriminator. One possible reason is that $g_k(\mathbf{x})$ does not change the policy gradient, as we explained in Section 4.3. The other reason is that $g_k(\mathbf{x})$ was close to zero because the state distributions of the expert and the learner did not differ significantly due to the small variation of initial states. Regarding the property of sharing the state value function, the results depended on how the reward was calculated from the result of the inverse RL step. When the policy was trained with $r_k(\mathbf{x})$, 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 shaping reward, there was not any significant difference as compared with the original ERIL. When we removed the hyperparameters from the second discriminator, it resulted in slower learning. ERIL without the soft Bellman equation was the most sample inefficient at the early stage of learning. There are several reasons. One is that the size of the discriminator is larger than that of the original ERIL because the discriminator is defined in the joint space of state, action, and next state. As a result, it required a large number of samples for training the discriminator. The second reason is that there were no hyperparameters.

6. Human Behavior Analysis

6.1. Task description

Next, in order to evaluate ERIL in a realistic situation, we conducted a dynamic motor control experiment in which a human subject solved a planar pole-balancing problem. Figure 6(a) shows the experimental setup. A subject can move the base in the left, right, top and bottom directions to swing the pole several times and then decelerate the pole to balance it at the upright position. As shown in Figure 6(b), the dynamics is described by the six-dimensional state vector $\mathbf{x} = [\theta, \dot{\theta}, x, \dot{x}, z, \dot{z}]^\top$, where θ and $\dot{\theta}$ are the angle and 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, respectively. 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 the applied forces F_x and F_z to the pendulum in Figure 6 are not observed in this experiment.

The task was performed under two conditions: long pole (73 cm) and short pole (29 cm). Each subject had 15 trials to balance the pole in each condition after he or she did some practice. Each trial ended when the subject could keep the pole upright for 3 seconds or when 40 seconds elapsed. We collected data from seven subjects (five right-handed and two left-handed), and the trajectory-based sampling method was used to construct the following three datasets of the experts: $\mathcal{D}_{i,j,\text{tr}}^E$ for training, $\mathcal{D}_{i,j,\text{va}}^E$ for validation, and $\mathcal{D}_{i,j,\text{te}}^E$ for testing of the i -th subject. The subscript j indicates the condition (1 for the long-pole and 2 for the short-pole conditions).

Table 2: Neural network architectures used in the pole-balancing task. For example, $V(\mathbf{x})$ 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
$\mu(\mathbf{x}, \mathbf{c})$	$(\dim \mathcal{X} + \dim \mathbf{c}, 100, \dim \mathcal{U})$
$\sigma(\mathbf{x}, \mathbf{c})$	$(\dim \mathcal{X} + \dim \mathbf{c}, 50, \dim \mathcal{U})$
$\mathbf{m}(\mathbf{x}, \mathbf{c})$	$(\dim \mathcal{X} + \dim \mathbf{c}, 50, \dim \mathcal{X})$
$V(\mathbf{x}, \mathbf{c})$	$(\dim \mathcal{X} + \dim \mathbf{c}, 100, 1)$
$Q(\mathbf{x}, \mathbf{u}, \mathbf{c})$	$(\dim \mathcal{X} + \dim \mathcal{U} + \dim \mathbf{c}, 256, 1)$
$g(\mathbf{x}, \mathbf{c})$	$(\dim \mathcal{X} + \dim \mathbf{c}, 50, 1)$
$D(\mathbf{x}, \mathbf{c})$	$(\dim \mathcal{X} + \dim \mathbf{c}, 100, 1)$
$D(\mathbf{x}, \mathbf{x}', \mathbf{c})$	$(2 \times \dim \mathcal{X} + \dim \mathbf{c}, 256, 1)$

Since we had multiple experts and the experts’ actions were not available, we had to use the extended ERIL described in Section 4.5. Specifically, the functions of ERIL was augmented by the seven-dimensional conditional one-hot vector \mathbf{c} . Then, we evaluated two variations of ERIL. One is that 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 ERIL with IDM, in which the expert’s action is estimated by the inverse dynamics model. Two ERILs were compared with GAIfO, IRLGAN, and BCO. Since two ERILs, GAIfO, and IRLGAN improve the policy by forward RL, they require the simulator of the environment. We modeled the dynamics shown in Figure 6(b) as the X-Z inverted pendulum (Wang, 2012). See the equations of motion in Appendix B.1.

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

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

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

6.2. Experimental Results

Figure 7 shows the learning curves of the seven subjects, indicating that the learning processes were quite different among the subjects. Subject 7 achieved the best performance for both conditions, while two subjects (1 and 3) could not accomplish the task. Subjects 1 and 5 performed well in the long-pole

condition, but they could not balance the pole at the upright position. Although the trajectories generated by Subject 1 and 3 were not successful, we used their data as the experts trajectories.

In order to evaluate the algorithms, we computed the negative log-likelihood for the test datasets $\mathcal{D}_{i,\text{te}}^E$

$$\text{NLL}(i, j) = -\frac{1}{N_{i,\text{te}}^E} \sum_{\ell=1}^{N_{i,j,\text{te}}^E} \ln \pi^L(\mathbf{x}'_{\ell} \mid \mathbf{x}_{\ell}, \mathbf{c}_{i,j}),$$

where $N_{i,\text{te}}^E$ is the number of samples in $\mathcal{D}_{i,\text{te}}^E$. $\mathbf{c}_{i,j} = [\mathbf{c}_i^s, \mathbf{c}_j^c]$ is the conditioning vector, where \mathbf{c}_i represents that the i -th element is 1 and otherwise 0. See Appendix B.2 for the evaluation of the state transition probability. Figure 8 shows that $\text{ERIL}(\kappa^{-1} = 0)$ obtained smaller negative log-likelihood than the other baselines for both conditions. Note that the data in the experts' dataset was not even sub-optimal because Subjects 3 and 5 did not fully accomplish the balancing task as shown in Figure 7. ERIL with IDM achieved comparable performance to $\text{ERIL}(\kappa^{-1} = 0)$ in the long-pole condition, but resulted in lower performance in the short-pole condition. It suggests that data augmentation by IDM did not help the training of the discriminator. BCO also augmented the data by IDM, but its negative log-likelihood was higher than ERIL with IDM. As compared with the results of IRLGAN and GAIfO, our methods represented the experts' policy efficiently. It indicates that the structure of the ERIL's discriminator was critical even if the experts' actions were not available.

Figure 9 shows the reward function of subjects 2, 4, and 7 estimated by $\text{ERIL}(\kappa^{-1} = 0)$, which was projected to the subspace $(\theta, \dot{\theta})$, while x, z, \dot{x} and \dot{z} were set to zero for visualization. Although they balanced the pole in the long-pole condition, the estimated rewards differed from each other. In the case of subject 7, the reward function of the long-pole condition was not so different from that of the short-pole condition, while there was a significant difference in those results of subject 4, who did not perform well in the short-pole condition as shown in Figure 7.

Finally, we computed the negative log-likelihood of which the policy of the i -th subject was evaluated by the test dataset of the j -th subject. Figure 10 shows the results of $\text{ERIL}(\kappa^{-1} = 0)$, GAIfO, and BCO. In the figures of 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 the negative log-likelihood computed by

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

Note that the test dataset and the training data were inconsistent when $i \neq 4$ and $j \neq 1$. The minimum negative log-likelihood 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. This suggests that Subject 4 used different reward functions for the different poles.

The lower row figures show the results when we used the test dataset of Subject 7 in the long pole condition. The minimum negative log-likelihood was achieved by the proper conditioning vector. As opposed to 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, and there were no significant differences. These results suggest that Subject 7 who was the best performer used similar reward functions for both poles. GAIfO and BCO could not find such property.

7. Discussion

7.1. Hyperparameter settings

ERIL has two hyperparameters, kappa and eta. As with standard RL, efficiency and performance depend on how the hyperparameters are tuned during learning. The later version of SAC (Haarnoja et al., 2018) updates the hyperparameter corresponding κ of ERIL by a simple gradient descent algorithm. Kozuno et al. (2019) show that one hyperparameter derived from κ and η control the tradeoff between noise-tolerance and convergence rate, and beta controls the quality of asymptotic performance from the viewpoint of forward RL setting.

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

7.2. Kinesthetic teaching

ERIL assumes that the expert and the learner has the same state transition probability. Therefore, the log-ratio of the joint distributions can be decomposed into the sum of that of the policies and that of the state distributions. However, this assumption limits the applicability of ERIL. For example, in order for the humanoid robot to imitate the human expert’s behavior, the expert should provide the demonstrations via kinesthetic teaching. To do so, the gravity compensation mode of the robot is activated, and the expert should guide the robot’s body to execute the task. Apparently, it is not a natural behavior for the expert, and therefore, kinesthetic teaching is not practically a viable option to provide demonstrations.

To overcome the problem, we should consider the log-ratio between the expert’s and the learner’s state transitions, and it will be estimated by the density ratio estimation methods as we do for the first discriminator. We plan to modify ERIL to incorporate the third discriminator.

8. Conclusion

This paper presented ERIL, which is entropy-regularized imitation learning based on forward and inverse reinforcement learning. Unlike the 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 the 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, state value function, and 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. Besides, 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 the environmental interactions in the forward RL step. ERIL also showed comparable performance in terms of the number of demonstrations provided by the expert. In the pole-balancing experiments, we showed how ERIL could be applied to the analysis of human behaviors.

Appendix A. Derivation

Appendix A.1. Derivation of Eq. (19)

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

$$\ln \frac{\pi_k^L(\mathbf{x}, \mathbf{u}, \mathbf{x}')}{\pi^E(\mathbf{x}, \mathbf{u}, b\mathbf{x}')} = \ln \frac{\pi_k^L(\mathbf{u} | \mathbf{x})}{\pi^E(\mathbf{u} | \mathbf{x})} + \ln \frac{\pi_k^L(\mathbf{x})}{\pi^E(\mathbf{x})}. \quad (\text{A.1})$$

Let us assign a selector variable $L = 1$ to the samples from the learner at the k -th iteration and $L = -1$ to the samples from the expert:

$$\begin{aligned} \pi_k^L(\mathbf{x}) &\triangleq \Pr(\mathbf{x} | L = 1), \\ \pi^E(\mathbf{x}) &\triangleq \Pr(\mathbf{x} | L = -1) = 1 - \pi_k^L(\mathbf{x}). \end{aligned}$$

Then, the first discriminator is represented by

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

We obtain the following log of the density ratio:

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

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

Appendix A.2. Derivation of Eq. (29)

The probability ratio of learner's and expert's 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 $|\mathcal{D}^L| = |\mathcal{D}^E|$, $\exp(g_k(\mathbf{x}))$ represents the following density ratio:

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

Arranging Eq. (23) yields

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

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

Appendix B. Notes on the pole-balancing problem

Appendix B.1. Equations of motion

Figure 6(b) shows the X-Z inverted pendulum on a pivot driven by horizontal and vertical forces. The equations of motion are given in (Wang, 2012) that are described by

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

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, respectively. M and m are the mass of the pivot and the pendulum, respectively. l is the distance from the pivot to the center of the mass of the pendulum. g denotes the gravitational acceleration. F_x and F_z are the horizontal and vertical force, respectively.

The state vector is given by a three-dimensional vector $\mathbf{x} = [x, \dot{x}, z, \dot{z}, \theta, \dot{\theta}]^\top$, while the action is represented by a two-dimensional vector $\mathbf{u} = [F_x, F_z]^\top$. 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/s²]. The inertia of the pendulum is negligible.

Appendix B.2. Evaluation of the state transition probability

Since the action is not available in the pole-balancing task, the learner’s policy cannot be used for evaluation. We exploit the property that a linear transformation of a multivariate Gaussian random vector has a multivariate Gaussian distribution because the policy in this study is also represented by the Gaussian distribution. Using a first-order Taylor expansion, the time-discretized version of Eq. (B.1) can be expressed by $\mathbf{x}_{t+1} = \mathbf{A}_t \mathbf{x}_t + \mathbf{B}_t \mathbf{u}_t$.

Acknowledgments

This work was supported by Innovative Science and Technology Initiative for Security Grant Numbr JPJ004596, ATLA, Japan and JST-Mirai Program Grant Number JPMJMI18B8, Japan to EU. This work was also supported by Japan Society for the Promotion of Science KAKENHI Grant Numbers JP16K21738, JP16H06561 and JP16H06563 to KD and JP19H05001 to EU. We would like to thank Shoko Igarashi for collecting the data of the human pole-balancing task.

Figure captions

Figure 1: The overall architecture of Entropy-Regularized Imitation Learning (ERIL).

Figure 2: The architecture of the ERIL discriminators: It consists of two discriminators, $D_k^{(1)}(\mathbf{x})$ and $D_k^{(2)}(\mathbf{x}, \mathbf{u}, \mathbf{x}')$.

Figure 3: Performance with respect to number of trajectories provided by learning policy at every iteration: Solid lines represent average values, while corresponding shaded areas correspond to ± 1 standard deviation region.

Figure 4: Performance with respect to number of trajectories provided by expert: Solid lines represent average values, while corresponding shaded areas correspond to ± 1 standard deviation region.

Figure 5: Comparison of the learning curves in the ablation study.

Figure 6: Inverted pendulum task solved by a human subject. (a) Start and goal positions. (b) State representation.

Figure 7: Learning curves of seven subjects. Blue: long pole. Red: short pole. A trial is considered a failure if a subject cannot balance the pole at the upright position within 40 [s].

Figure 8: Comparison of negative log-likelihood among the imitation learning algorithms. Note that the smaller values of NLL indicate a better fit.

Figure 9: Estimated reward function of subjects 2, 4, and 7 projected to the subspace (θ, ω) , $\omega = d\theta/dt$.

Figure 10: Comparison of negative log-likelihood when the condition of the test dataset was different from that of the training one. in human inverted pendulum task. Figures in the upper row (a, b, and c) show the results when the test dataset is $\mathcal{D}_{4,1,te}^E$. Figures in the lower row (d, e, and f) show the results when the test dataset is $\mathcal{D}_{7,1,te}^E$.

References

- R. Sutton, A. Barto, Reinforcement Learning, MIT Press, 1998.
- K. Doya, Reinforcement learning: Computational theory and biological mechanisms., *HFSP Journal* 1 (2007) 30–40.
- J. Kober, J. A. Bagnell, J. Peters, Reinforcement Learning in Robotics: A Survey, *International Journal of Robotics Research* 32 (2013) 1238–1274.
- A. Y. Ng, S. Russell, Algorithms for inverse reinforcement learning, in: *Proc. of the 17th International Conference on Machine Learning*, 2000.
- P. Abbeel, A. Y. Ng, Apprenticeship learning via inverse reinforcement learning, in: *Proc. of the 21st International Conference on Machine Learning*, 2004.
- A. Vogel, D. Ramachandran, R. Gupta, A. Raux, Improving Hybrid Vehicle Fuel Efficiency using Inverse Reinforcement Learning, in: *Proc. of the 26th AAAI Conference on Artificial Intelligence*, 2012.
- S. Liu, M. Araujo, E. Brunskill, R. Rossetti, J. Barros, R. Krishnan, Understanding Sequential Decisions via Inverse Reinforcement Learning, in: *Proc. of the 14th IEEE International Conference on Mobile Data Management*, IEEE, 2013, pp. 177–186.
- M. Shimosaka, T. Kaneko, K. Nishi, 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*, 2014, pp. 1694–1700.
- K. Muelling, A. Boularias, B. Mohler, B. Schölkopf, J. Peters, Learning strategies in table tennis using inverse reinforcement learning., *Biological Cybernetics* 108 (2014) 603–619.
- H. Kretzschmar, M. Spies, C. Sprunk, W. Burgard, Socially compliant mobile robot navigation via inverse reinforcement learning, *The International Journal of Robotics Research* (2016).
- C. Xia, A. El Kamel, Neural inverse reinforcement learning in autonomous navigation, *Robotics and Autonomous Systems* 84 (2016) 1–14.
- K. Ashida, T. Kato, K. Hotta, K. Oka, Multiple tracking and machine learning reveal dopamine modulation for area-restricted foraging behaviors via velocity change in *Caenorhabditis elegans*, *Neuroscience Letters* 706 (2019) 68–74.
- T. Hirakawa, T. Yamashita, T. Tamaki, H. Fujiyoshi, Y. Umezū, I. Takeuchi, S. Matsumoto, K. Yoda, Can ai predict animal movements? filling gaps in animal trajectories using inverse reinforcement learning, *Ecosphere* (2018).

- S. Yamaguchi, N. Honda, Y. Ikeda, M. Tsukada, S. Nakano, I. Mori, S. Ishii, Identification of animal behavioral strategies by inverse reinforcement learning, *PLoS Computational Biology* (2018).
- G. Neu, C. Szepesvári, Training parsers by inverse reinforcement learning, *Machine Learning* 77 (2009) 303–337.
- S. Collette, W. M. Pauli, P. Bossaerts, J. O’Doherty, Neural computations underlying inverse reinforcement learning in the human brain, *eLife* 6 (2017).
- J. Fu, K. Luo, S. Levine, Learning robust rewards with adversarial inverse reinforcement learning, in: *Proc. of the 6th International Conference on Learning Representations*, 2018.
- J. Ho, S. Ermon, Generative adversarial imitation learning, in: *Advances in Neural Information Processing Systems* 29, 2016, pp. ??–??
- I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: *Advances in Neural Information Processing Systems* 27, 2014, pp. 2672–2680.
- J. Schulman, S. Levine, P. Abbeel, M. Jordan, P. Moritz, Trust region policy optimization, in: *Proc. of the 32nd International Conference on Machine Learning*, 2015, pp. 1889–1897.
- M. Sugiyama, T. Suzuki, T. Kanamori, Density ratio estimation in machine learning, Cambridge University Press, 2012.
- E. Uchibe, K. Doya, Inverse Reinforcement Learning Using Dynamic Policy Programming, in: *Proc. of IEEE International Conference on Development and Learning and Epigenetic Robotics*, 2014, pp. 222–228.
- E. Uchibe, Model-free deep inverse reinforcement learning by logistic regression, *Neural Processing Letters* 47 (2018) 891–905.
- M. G. Azar, V. Gómez, H. J. Kappen, Dynamic policy programming, *Journal of Machine Learning Research* 13 (2012) 3207–3245.
- T. Haarnoja, A. Zhou, P. Abbeel, S. Levine, Soft Actor-Critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, in: *Proc. of the 35th International Conference on Machine Learning*, 2018, pp. 1856–1865.
- T. Kozuno, E. Uchibe, K. Doya, 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*, Okinawa, Japan, 2019, pp. 2995–3003.
- E. Todorov, T. Erez, Y. Tassa, MuJoCo: A physics engine for model-based control, in: *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 5026–5033.

- G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, W. Zaremba, OpenAI Gym, arXiv preprint (2016).
- D. A. Pomerleau, ALVINN: An autonomous land vehicle in a neural network, in: *Advances in Neural Information Processing Systems* 1, 1989, pp. 305–313.
- S. Ross, G. Gordon, D. Bagnell, 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*, 2011, pp. 627–35.
- M. Laskey, J. Lee, R. Fox, A. Dragan, K. Goldberg, Dart: Noise injection for robust imitation learning, in: *Proc. of the 1st Conference on Robot Learning*, 2017.
- F. Torabi, G. Warnell, P. Stone, Behavioral cloning from observation, in: *Proc. of the 27th International Joint Conference on Artificial Intelligence and the 23rd European Conference on Artificial Intelligence*, 2018, pp. 4950–57.
- S. Reddy, A. D. Dragan, S. Levine, Sqil: Imitation learning via regularized behavioral cloning, in: *Proc. of the 8th International Conference on Learning Representations*, 2020, pp. 627–35.
- K. Kobayashi, T. Horii, R. Iwaki, Y. Nagai, M. Asada, Situated GAIL: Multi-task imitation using task-conditioned adversarial inverse reinforcement learning, arXiv preprint (2019).
- A. Kinose, T. Taniguchi, Integration of imitation learning using gail and reinforcement learning using task-achievement rewards via probabilistic graphical model, arXiv preprint (2019).
- P. Henderson, W.-D. Chang, P.-L. Bacon, D. Meger, J. Pineau, D. Precup, OptionGAN: Learning joint reward-policy options using generative adversarial inverse reinforcement learning, in: *Proc. of the 32nd AAAI Conference on Artificial Intelligence*, 2018.
- F. Torabi, G. Warnell, P. Stone, Generative adversarial imitation from observation, in: *ICML 2019 Workshop on Imitation, Intent, and Interaction*, 2019, pp. 4950–57.
- M. Sun, X. Ma, Adversarial imitation learning from incomplete demonstrations, in: *Proc. of the 28th International Joint Conference on Artificial Intelligence*, 2014.
- L. Blondé, A. Kalousis, Sample-efficient imitation learning via generative adversarial nets, in: *Proc. of the 22nd International Conference on Artificial Intelligence and Statistics*, 2019, pp. 3138–3148.
- T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, in: *Proc. of the 4th International Conference on Learning Representations*, San Diego, 2016.

- I. Kostrikov, K. K. Agrawal, D. Dwibedi, S. Levine, J. Tompson, Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning, in: Proc. of the 7th International Conference on Learning Representations, 2019.
- S. Fujimoto, H. van Hoof, D. Meger, Addressing function approximation error in actor-critic methods, in: Proc. of the 35th International Conference on Machine Learning, 2018.
- F. Sasaki, T. Yohira, A. Kawaguchi, Sample efficient imitation learning for continuous control, in: Proc. of the 7th International Conference on Learning Representations, 2019.
- T. Degris, M. White, R. S. Sutton, Off-policy actor-critic, in: Proc. of the 29th International Conference on Machine Learning, 2012.
- G. Zuo, K. Chen, J. Lu, X. Huang, Deterministic generative adversarial imitation learning, *Neurocomputing* (2020) 60–69.
- D. Nishio, D. Kuyoshi, T. Tsuneda, S. Yamane, Discriminator soft actor critic without extrinsic rewards, *arXiv preprint* (2020).
- S. K. S. Ghasemipour, R. Zemel, S. Gu, A divergence minimization perspective on imitation learning methods, in: Proc. of the 3rd Conference on Robot Learning, 2019.
- L. Ke, M. Barnes, W. Sun, G. Lee, S. Choudhury, S. Srinivasa, Imitation learning as f -divergence minimization, in: Proc. of the 14th International Workshop on the Algorithmic Foundations of Robotics (WAFR), 2020.
- B. D. Ziebart, A. Maas, J. A. Bagnell, A. K. Dey, Maximum entropy inverse reinforcement learning, in: Proc. of the 23rd AAAI Conference on Artificial Intelligence (AAAI-08), 2008.
- J. Peters, S. Schaal, Reinforcement learning of motor skills with policy gradients, *Neural Networks* (2008) 1–13.
- A. B. Dieng, F. J. R. Ruiz, D. M. Blei, M. K. Titsias, Prescribed generative adversarial networks, *arXiv preprint arXiv:1910.04302* (2019).
- V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al., Human-level control through deep reinforcement learning, *Nature* 518 (2015) 529–533.
- T. Schaul, D. Horgan, K. Gregor, D. Silver, Universal value function approximators, in: Proc. of the 32nd International Conference on Machine Learning, 2015, pp. 1312–1320.
- D. Kingma, J. Ba, ADAM: A method for stochastic optimization, in: Proc. of the 3rd International Conference for Learning Representations, 2015.

- J.-J. Wang, Stabilization and tracking control of x-z inverted pendulum with sliding-mode control, ISA Transactions 51 (2012) 763–70.
- T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, S. Levine, Soft actor-critic algorithms and applications, arXiv preprint (2018).

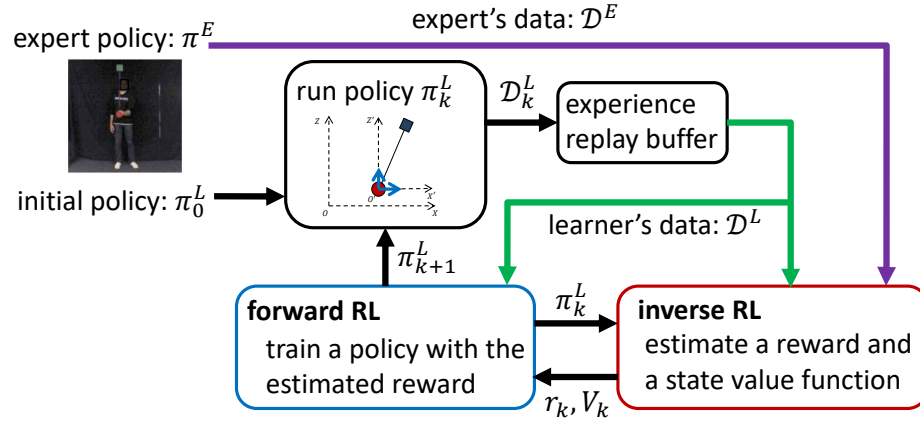


Figure 1:

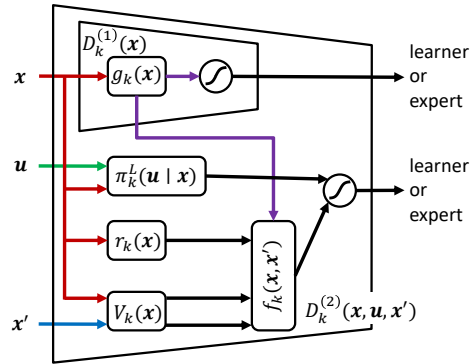


Figure 2:

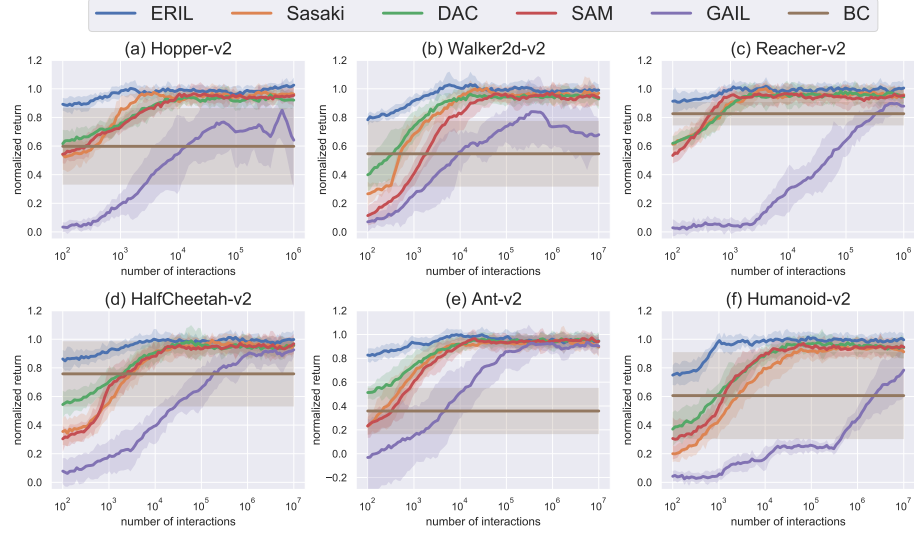


Figure 3:

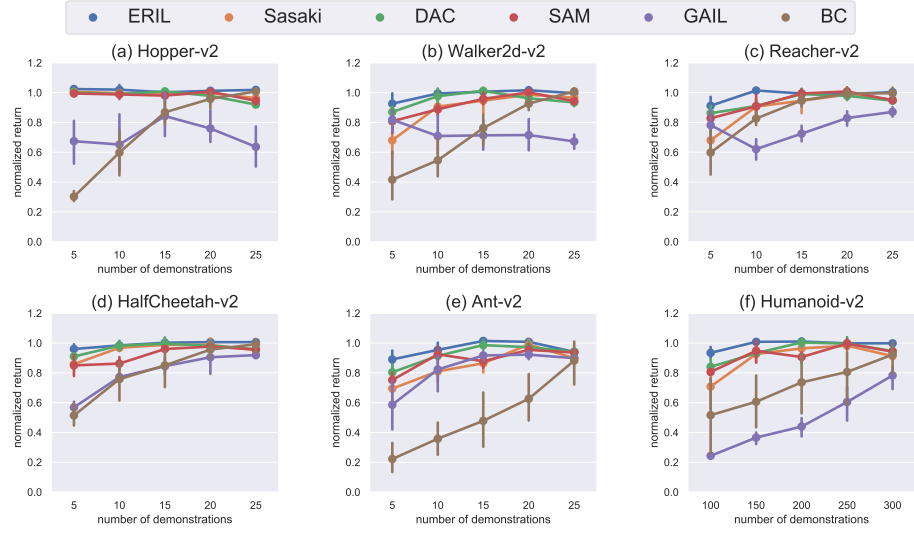


Figure 4:

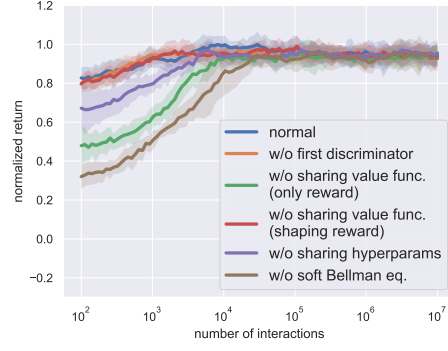


Figure 5:

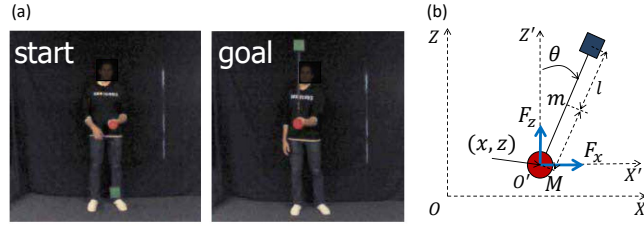


Figure 6:

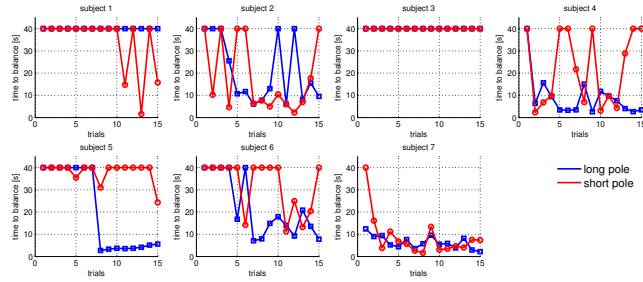


Figure 7:

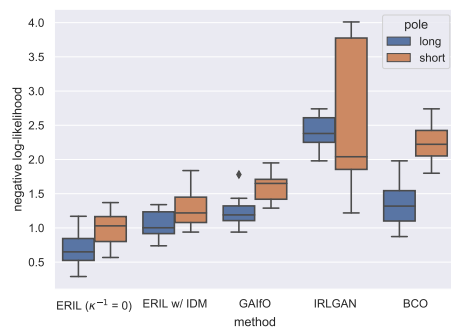


Figure 8:

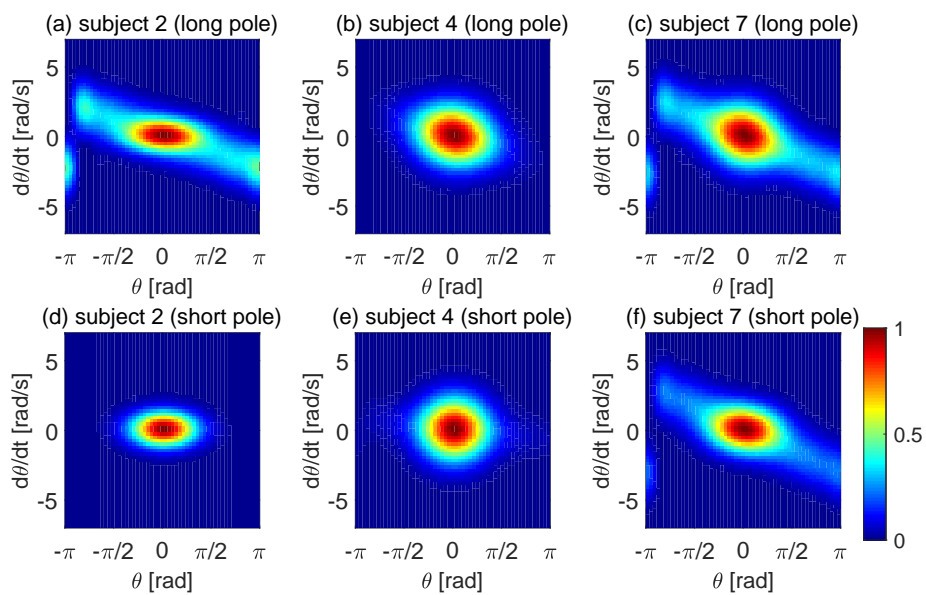


Figure 9:

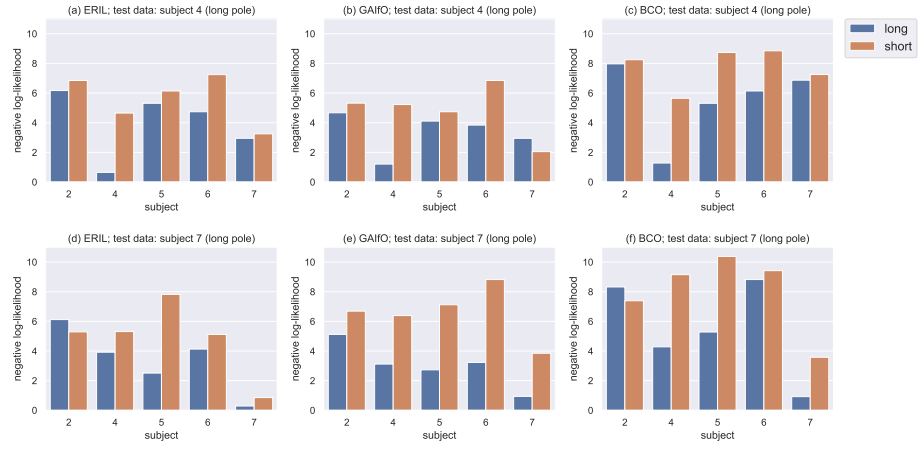


Figure 10: