Learning Rewards via State Marginal Matching

Harshit Sikchi [*1] Tianwei Ni [*1] Tejus Gupta [*1] Yufei Wang [*1] Benjamin Eysenbach [2] Lisa Lee [2]

Abstract

Imitation learning is a promising approach for robotic tasks where it is difficult to directly program the behaviour or specify a cost for optimal control methods. In this work, we propose a method for learning the reward function (and corresponding policy) by specifying the expert state density. Our main result is the analytic gradient of any f-divergence between the agent and expert state distribution w.r.t. reward parameters. We present a novel algorithm SMM-IRL based on the derived gradient and show that it has competitive performance with previous imitation learning methods on robotic simulation tasks. Unlike our algorithm, previous approaches based on state density matching do not recover the reward function. The recovered reward function can be leveraged to solve downstream tasks efficiently. We empirically show one such example, where the reward function obtained from our algorithm is used to solve hard-to-explore tasks.

1. Introduction

Imitation learning (IL) is a powerful tool to design autonomous behaviours in robotic systems. Though reinforcement learning methods promise to learn such behaviours automatically, they have been most successful in tasks with a clear definition of the reward function. Reward design remains difficult in many robotic tasks such as driving a car (Pomerleau, 1989), tying a knot (Osa et al., 2017), and human-robot cooperation (Hadfield-Menell et al., 2016). Imitation learning is a popular approach to such tasks, since it is easier for an expert teacher to demonstrate the desired behaviour rather than specify the reward (Atkeson & Schaal, 1997; Henderson et al., 2018; Hwangbo et al., 2019).

Methods in IL frameworks are generally split into behavior cloning (BC) (Bain & Sammut, 1995) and inverse reinforcement learning (IRL) (Russell, 1998; Ng et al., 2000). BC is typically based on supervised learning to regress expert actions from expert observations without the need for further interaction with the environment, but suffers from the covariate shift problem (Ross & Bagnell, 2010). On the other hand, IRL methods aim to learn the reward from expert demonstrations, and use it to train the agent policy.

Traditionally, IL methods assume access to expert demonstrations and minimize some divergence between policy and expert's trajectory (or state) distribution. However, in many cases, it is easier to directly specify the state distribution of the desired behaviour rather than provide fully-specified demonstrations of the desired behaviour. For example, consider a safety task where we want to specify zero density on unsafe states. In practice, it is difficult to tweak the reward to penalize safety violations (Dalal et al., 2018). It would also be impossible to exhaustively enumerate expert trajectories that obey the safety constraint. In our problem setting, we directly recover the policy (and reward) by simply specifying the target state marginal. Compared to traditional imitation learning, our problem formulation is well suited to optimize the policy (and reward) parameters to stay within the safety constraints. Similarly, we can specify a uniform density on the whole state space for exploration tasks, or a Gaussian centered at the goal for goal-reaching tasks.

Under this novel and practical setup, we frame IRL as a state-marginal matching (SMM) problem and provide a novel algorithm (SMM-IRL). We work under the MaxEnt RL framework (Haarnoja et al., 2017; 2018) where trajectory distribution is parameterized as an energy-based model, and the agent state density is obtained by marginalizing the trajectory distribution. We derive an analytic expression for the gradient of the state-marginal matching objective with respect to the reward parameters. Our resulting algorithm uses stochastic gradient descent to obtain a reward and policy that match the expert state density.

Previous works (GAIL (Ho & Ermon, 2016), f-MAX (Ghasemipour et al., 2019), SMM (Lee et al., 2019)) have cast the state-marginal matching as an adversarial game between the policy and the discriminator. These methods recover a policy that matches the expert state density, but unlike our method, they do not recover a corresponding reward function. Moreover, since our algorithm directly performs gradient descent on f-divergence objective, the convergence guarantees of our algorithm are stronger.

Our experiments show that (1) SMM-IRL can effectively match expert density on several tasks and (2) the reward recovered from uniform expert density can be leveraged as prior to help hard-to-explore tasks.

2. Related Work

Classical IRL methods (Russell, 1998; Ng et al., 2000) obtain a policy by learning a reward function from sampled trajectories of an expert policy. MaxEntIRL (Ziebart et al., 2008) learns a stationary reward by maximizing likelihood on expert trajectories, i.e., it minimizes forward KL divergence in trajectory space under the maximum entropy RL framework. Similar to MaxEntIRL, Deep MaxEntIRL (Wulfmeier et al., 2015) and GCL (Finn et al., 2016b) optimize the forward KL divergence in trajectory space.

(Finn et al., 2016a) shows that GCL is equivalent to training GAN with a special structure in discriminator. Following this direction, a number of methods (Finn et al., 2016a; Fu et al., 2017; Qureshi et al., 2018) use adversarial training with a special discriminator to learn from demonstrations and extract a reward function. To show that these methods are equivalent to optimizing forward KL divergence in trajectory space, the authors use *incorrect* importance sampling weights which make their gradient *biased*. We provide more information to support this claim in Appendix .6. We also highlight that AIRL does not minimize reverse KL in state-marginal space as was claimed in (Ghasemipour et al., 2019), due to subtle differences in the objective from the original work (Fu et al., 2017).

A set of methods GAIL (Ho & Ermon, 2016), FAIRL (f-MAX-FKL) (Ghasemipour et al., 2019), f-MAX-RKL (Ghasemipour et al., 2019) use a vanilla discriminator to address the issue of running RL in the inner loop. Instead, these methods directly optimize the policy in the outer loop using adversarial training. GAIL, FAIRL, f-MAX-RKL can be showed to optimize the Jensen-Shannon, forward KL and reverse KL divergence respectively, but fail to learn a stationary reward function.

SMM (Lee et al., 2019), perhaps closest to this work, optimizes for reverse KL between expert and policy state marginal but also fails to recover a reward function due to its fictitious play (Brown, 1951).

Contrary to all the methods above, our methods manages to optimize any f-divergence between state-marginal of expert and the agent while also recovering a stationary reward function. Table 1 highlights the contrast more succinctly.

IRL Method	**Space**	**Div**	**Reward**
MaxEntIRL (Ziebart et al., 2008)	τ	FKL	✓
GCL (Finn et al., 2016b)	τ	FKL	✓
GAN-GCL (Finn et al., 2016a)	τ	FKL*	✓
AIRL (Fu et al., 2017)	τ	FKL*	✓
GAIL (Ho & Ermon, 2016)	s, a	JS	×
f-MAX (Ghasemipour et al., 2019)	s, a	f-div	×
SMM (Lee et al., 2019)	s	RKL	×
Our Method (SMM-IRL)	s	f-div	✓

Table 1. Summary of IRL methods. For each algorithm, we list (1) the **space** that their objective operates on: the trajectory τ or state-action marginal s, a or state marginal s, (2) the divergence (**div**) they optimize: f-divergence (f-div), forward KL divergence (FKL), reverse KL divergence (RKL), and Jensen-Shannon divergence (JS), (3) and whether they can learn stationary rewards (**Reward**). Here * means that GAN-GCL and AIRL use incorrect IS weights to approximate FKL, for more details please refer to Appendix .6.

3. Learning Stationary Rewards via State-Marginal Matching

We now present our method, **SMM-IRL** (State-Marginal Matching via Inverse Reinforcement Learning). We first layout the general recipe of SMM-IRL, and then present the details of the algorithm.

3.1. Notation

Consider a Markov Decision Process (MDP) represented as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \rho_0, T)$ with state-space $\mathcal{S}$, action-space $\mathcal{A}$, dynamics $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$, reward function $r(s, a)$, initial state distribution ρ_0, and horizon T.

Under the maximum entropy framework (Ziebart, 2010), the soft-optimal policy π maximizes $\sum_{t=1}^{T} E_{\rho_{\pi,t}(s_t,a_t)}[r(s_t, a_t) + \alpha H(\cdot|s_t)]$. Here $\rho_{\pi,t}$ is the state-action marginal distribution of policy π at timestamp t, and $\alpha > 0$ is the entropy temperature.

We assume access to an expert state marginal $\rho_E(s)$ which is feasible to specify in many tasks. If expert observations are available, we can fit a density model to the samples.

Let $r_\theta(s)$ be a parameterized reward function only dependent on state. The optimal MaxEnt trajectory distribution $\rho_\theta(\tau)$ under reward r_θ can be computed as:

$$\rho_\theta(\tau) = \frac{1}{Z} p(\tau) e^{r_\theta(\tau)/\alpha}$$
$$\text{where } p(\tau) = p(s_1) \prod_{i=1}^{T} p(s_{t+1}|s_t, a_t), \;\; r_\theta(\tau) = \sum_{i=1}^{T} r_\theta(s_t),$$
$$Z = \int p(\tau) e^{r_\theta(\tau)/\alpha} d\tau \tag{1}$$

Slightly overloading the notation, the optimal MaxEnt state marginal distribution $\rho_\theta(s)$ under reward r_θ is obtained by marginalization:

$$\rho_\theta(s) \propto \int p(\tau) e^{r_\theta(\tau)/\alpha} \eta_\tau(s) d\tau, \; \eta_\tau(s) = \sum_{t=1}^{T} \mathbb{1}(\tau_t = s) \tag{2}$$

where $\eta_\tau(s)$ can be interpreted as the visitation count of a state s in a particular trajectory τ.

SMM-IRL can then be formulated as minimizing the following objective in f-divergence D_f:

$$L_f(\theta) = D_f(\rho_E(s) \mid\mid \rho_\theta(s)) \tag{3}$$

The common choices for f-divergence include forward KL divergence, reverse KL divergence, and Jensen-Shannon divergence. In the next section, we derive surrogate objective that can be used to optimize Eq. 3.

3.2. Objectives for State-Marginal Matching in f-Divergence

We now present the main result of our method in a form of surrogate objective that has same gradient w.r.t. reward parameter θ as the original objective Eq. 3.

Theorem 3.1 (**f-divergence surrogate objective**). *Given any f-divergence, the surrogate objective that minimizes the f-divergence between state-marginals of expert (E) and soft-optimal agent under reward ($\hat{\theta}$) in the maximum entropy framework is given by:*

$$\tilde{L}_f(\theta) = \frac{1}{\alpha T}\text{cov}_{\tau\sim\rho_{\hat{\theta}}(\tau)}\left(\sum_{t=1}^{T} h_f\left(\frac{\rho_E(s_t)}{\rho_{\hat{\theta}}(s_t)}\right), \sum_{t=1}^{T} r_\theta(s_t)\right) \tag{4}$$

where $h_f(t) \triangleq f(t) - f'(t)t$, $\rho_E(s)$ is the expert state marginal and $\rho_{\hat{\theta}}(s)$ is the state marginal of the soft-optimal agent under reward function r_θ, and the covariance is taken under the agent's trajectory distribution $\rho_{\hat{\theta}}(\tau)$.

Proof. Refer to Appendices A and B. □

We distinguish $\hat{\theta}$ from θ to specify that we do not differentiate through $\hat{\theta}$. By the classic theory of gradient descent, if the f-divergence objective is Lipschitz continuous, then our method has sublinear convergence to a stationary point.

Choosing f-divergence to be FKL, RKL or JS allows to derive useful practical algorithms, so we form the following corollaries. The notations used in the corollaries are the same as in Theorem 3.1. JS objective is showed in Appendix .5.

Corollary 3.1.1 (**Forward KL surrogate objective**). *Choosing f-divergence to be Forward KL. divergence, the state marginal matching objective reduces to optimizing the following surrogate objective:*

$$\tilde{L}_{\text{FKL}}(\theta) = -\frac{1}{\alpha T}\text{cov}_{\tau\sim\rho_{\hat{\theta}}(\tau)}\left(\sum_{t=1}^{T} \frac{\rho_E(s_t)}{\rho_{\hat{\theta}}(s_t)}, \sum_{t=1}^{T} r_\theta(s_t)\right)$$

Proof. Refer to Appendix .3. □

Corollary 3.1.2 (**Reverse KL surrogate objective**). *Choosing f-divergence to be Reverse KL divergence, the state marginal matching objective reduces to optimizing the following surrogate objective:*

$$\tilde{L}_{\text{RKL}}(\theta) = -\frac{1}{\alpha T}\text{cov}_{\tau\sim\rho_{\hat{\theta}}(\tau)}\left(\sum_{t=1}^{T} \log\frac{\rho_E(s_t)}{\rho_{\hat{\theta}}(s_t)}, \sum_{t=1}^{T} r_\theta(s_t)\right)$$

Proof. Refer to Appendix .4. □

RKL surrogate objective has a special property that the expert density can be specified in *unnormalized* form, since the normalizing factor does not change the surrogate objective. This makes density specification easier in a number of scenarios.

Corollary 3.1.3 (**Jensen-Shannon (JS) surrogate objective**). *Choosing f-divergence to be Jensen-Shannon Divergence, the state marginal matching objective reduces to optimizing the following surrogate objective:*

$$\tilde{L}_{\text{JS}}(\theta) = -\tfrac{1}{\alpha T}\text{cov}_{\tau\sim\rho_{\hat{\theta}}(\tau)}\left(\textstyle\sum_{t=1}^{T}\log\left(1+\frac{\rho_E(s_t)}{\rho_{\hat{\theta}}(s_t)}\right), \sum_{t=1}^{T} r_\theta(s_t)\right)$$

Proof. Refer to Appendix .5. □

Intuitively, the surrogate objectives encourage trajectories which have higher sum of (1) density ratios (FKL) or (2) log density ratios (RKL) or (3) mixture of them (JS) between the expert and the agent to have higher return under the reward function r_θ. FKL is also known as a mode-covering objective while RKL is a mode-seeking objective. In safety-critical applications, RKL is more preferable, while FKL is preferable in uniform exploration task.

3.3. Algorithm

Using the corollaries presented above, we can propose a practical algorithm for learning the reward function r_θ. Given expert state marginal density $\rho_E(s)$ and one instantiation of f-divergence, the algorithm *alternates* between solving the optimal MaxEnt policy under current reward, and updating the reward parameter using SGD on the sample covariance. In practice, we need to fit a density $\hat{\rho}_{\hat{\theta}}(s)$ to estimate $\rho_{\hat{\theta}}(s)$. The full algorithm is shown in Algorithm 1.

4. Experiments

In our experiments, we seek answers to the following questions: 1. How does our method compare to previous IRL/IL methods in terms of state-marginal matching? 2. How can learning a stationary reward help in downstream tasks?

Algorithm 1 Inverse RL via State Marginal Matching

```
Input   : $\rho_E(s)$, initialized $r_\theta$, $f$-divergence
Output : Learned reward $r_\theta$, Policy $\pi_\theta$
for $i \leftarrow 1$ to $Iter$ do
    $\pi_\theta \leftarrow$ MaxEntRL($r_\theta$)
    Fit the density $\hat{\rho}_{\hat{\theta}}(s)$ to collected samples from $\pi_\theta$
    Compute estimated gradient $\hat{\nabla}_\theta \tilde{L}_f(\theta)$ for Eq. 4 over $\rho_{\hat{\theta}}(\tau)$
    $\theta \leftarrow \theta - \lambda \hat{\nabla}_\theta \tilde{L}_f(\theta)$
end
```

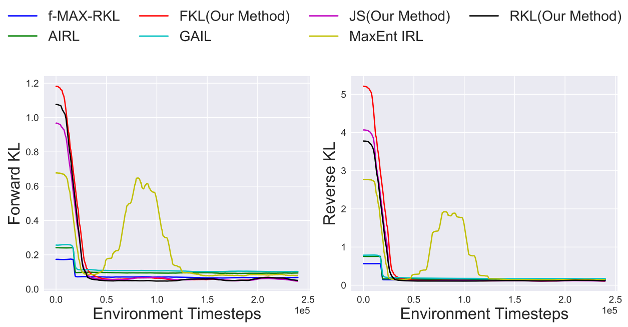


Figure 1. Forward (left) and Reverse (right) KL curves in pointmass environment for Gaussian target density for all the methods during training. Smoothed in a window of 120 evaluations.

4.1. Environments and Expert State Densities Setup

We test our methods and baselines on a pointmass environment and the OpenAI Gym environment `Reacher-v2` (Reacher). The pointmass environment is a 2D 4 x 4 room, with a pointmass which operates under linear dynamics. Reacher has non-linear dynamics to control a 2D arm of 2 DOF and is a more difficult task than pointmass.

We consider three tasks by specification of three different expert state marginals in each environment: Gaussian and Mixture of two Gaussians for single/bi-goal reaching, and uniform distribution for exploration and downstream tasks. More details of environment and expert state marginals are discussed in Appendix .7.1 and .8.

4.2. Training Details

We benchmark the following baselines: MaxEnt IRL (Ziebart et al., 2008), GAIL (Ho & Ermon, 2016), AIRL (Fu et al., 2017), and f-MAX-RKL (Ghasemipour et al., 2019) which shares the same objective with AIRL except that it uses a vanilla discriminator unlike the discriminator with a special structure in AIRL. Since we assume access to expert densities only, we use importance sampling for the expert samples needed in baselines.

We use SAC (Haarnoja et al., 2018) as the MaxEnt RL algorithm for all the methods in continuous state-action spaces. For our method and MaxEnt IRL, we fit a Kernel Density (KDE) model to estimate agent state marginal, and use a MLP for reward parameterization. For the baselines GAIL, AIRL, f-MAX-RKL, we use the f-MAX (Ghasemipour et al., 2019) authors' official implementation[1]. The training details are in Appendix .9.

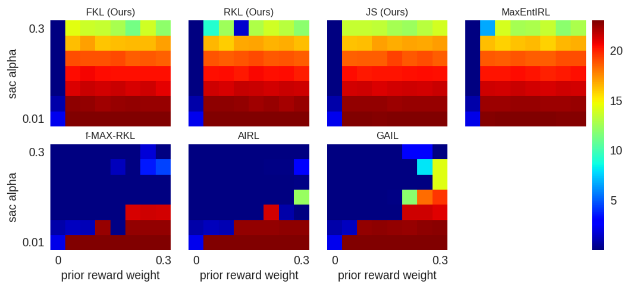


Figure 2. The task return (in terms of r_{task}) with different α and prior reward weight λ. The performance of vanilla SAC is shown in the leftmost column with $\lambda = 0$ in each subplot.

4.3. Comparison to baselines

In general, it is not possible to evaluate real returns without access to ground truth reward. Given expert density, we estimate Forward and Reverse KL divergence in state marginals between the expert and the agent during training, to evaluate the state-marginal matching objectives for all the methods. More details for the evaluation are discussed in Appendix .10.

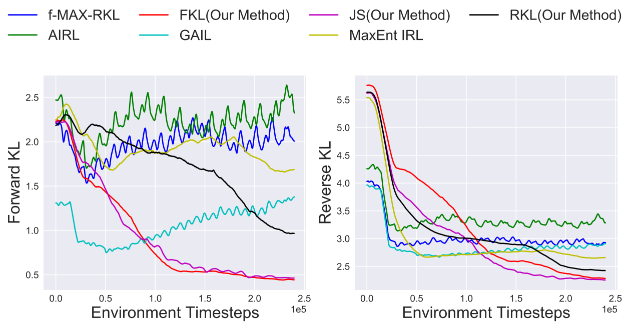
Figure 1 compares our algorithm with baselines on the pointmass environment. We observe that all methods converge (Max-Ent IRL is slightly unstable) and are able reduce the FKL and RKL to near zero. Figure 3 compares these algorithms on harder Reacher tasks. We observe that the baselines give unstable convergence, which can be attributed to optimizing a different objective than SMM. Our methods FKL, JS outperform the baselines in the forward KL and the reverse KL metric.

4.4. Utility of learning a stationary reward

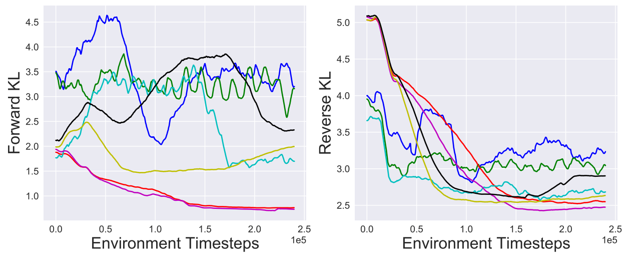
In this subsection, we demonstrate the utility of the learned stationary reward by using it as a prior reward for the downstream task. Specifically, we consider a prior reward obtained from a uniform expert density, and use it to make the hard-exploration task in pointmass learnable. We use a difficult goal-reaching task with distraction rewards, with full details presented in appendix .11.

We use the learned prior reward r_{prior} to augment the task reward r_{task} by the following rule: $r(s) = r_{\text{task}}(s) +$

[1] https://github.com/KamyarGh/rl_swiss



(a) Expert Density: Gaussian



(b) Expert Density: Mixture of two Gaussians

Figure 3. Forward (left) and Reverse (right) KL curves in Reacher environment for the goal-reaching tasks for all the methods during training. Smoothed in a window of 120 evaluations.

$\lambda(\gamma r_{\text{prior}}(s') - r_{\text{prior}}(s))$. The main theoretical result of (Ng et al., 1999) dictates that adding a potential-based reward in this form will not change the optimal policy. For GAIL and f-MAX-RKL, they do not extract a reward function but rather a discriminator, so we derive a prior reward from the discriminator the same way as in (Ghasemipour et al., 2019; Ho & Ermon, 2016).

The leftmost column of all subplots in Figure 2 show the poor performance of SAC training without reward augmentation ($\lambda = 0$). This verifies the difficulty in exploration for solving the task. We change λ in the x-axis, and α in SAC in the y-axis, and plot the final task return (in terms of r_{task}) as a heatmap in Figure 2. The presence of larger red region in the heatmap shows that our method is able to extract a prior reward that is more robust and effective in helping the downstream task attain better final performance with its original reward. The learned reward functions are visualized in Figure 5 of Appendix .11.

5. Conclusion

We present a new method to learn stationary rewards for imitation via the state-marginal matching objective. This method works under the setting where expert density can be specified, making it ideal for exploration and safety tasks. The rewards and the policies learned can be further used in downstream tasks as task-agnostic priors. In future work, we aim to test the capabilities for complex high-dimensional continuous tasks and extend it to meta-learning setting.

References

Ali, S. M. and Silvey, S. D. A general class of coefficients of divergence of one distribution from another. *Journal of the Royal Statistical Society: Series B (Methodological)*, 28(1):131–142, 1966.

Atkeson, C. G. and Schaal, S. Robot learning from demonstration. In *ICML*, volume 97, pp. 12–20. Citeseer, 1997.

Bain, M. and Sammut, C. A framework for behavioural cloning. In *Machine Intelligence 15*, pp. 103–129, 1995.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Brown, G. W. Iterative solution of games by fictitious play. *Activity analysis of production and allocation*, 13(1):374–376, 1951.

Dalal, G., Dvijotham, K., Vecerik, M., Hester, T., Paduraru, C., and Tassa, Y. Safe exploration in continuous action spaces. *arXiv preprint arXiv:1801.08757*, 2018.

Finn, C., Christiano, P., Abbeel, P., and Levine, S. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *arXiv preprint arXiv:1611.03852*, 2016a.

Finn, C., Levine, S., and Abbeel, P. Guided cost learning: Deep inverse optimal control via policy optimization. In *International conference on machine learning*, pp. 49–58, 2016b.

Fu, J., Luo, K., and Levine, S. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*, 2017.

Ghasemipour, S. K. S., Zemel, R., and Gu, S. A divergence minimization perspective on imitation learning methods. *arXiv preprint arXiv:1911.02256*, 2019.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Haarnoja, T., Tang, H., Abbeel, P., and Levine, S. Reinforcement learning with deep energy-based policies. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1352–1361. JMLR. org, 2017.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

Hadfield-Menell, D., Russell, S. J., Abbeel, P., and Dragan, A. Cooperative inverse reinforcement learning. In *Advances in neural information processing systems*, pp. 3909–3917, 2016.

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., and Meger, D. Deep reinforcement learning that matters. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Ho, J. and Ermon, S. Generative adversarial imitation learning. In *Advances in neural information processing systems*, pp. 4565–4573, 2016.

Hwangbo, J., Lee, J., Dosovitskiy, A., Bellicoso, D., Tsounis, V., Koltun, V., and Hutter, M. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26):eaau5872, 2019.

Ke, L., Barnes, M., Sun, W., Lee, G., Choudhury, S., and Srinivasa, S. Imitation learning as f-divergence minimization. *arXiv preprint arXiv:1905.12888*, 2019.

Kozachenko, L. and Leonenko, N. N. Sample estimate of the entropy of a random vector. *Problemy Peredachi Informatsii*, 23(2):9–16, 1987.

Kraskov, A., Stögbauer, H., and Grassberger, P. Estimating mutual information. *Physical review E*, 69(6):066138, 2004.

Lee, L., Eysenbach, B., Parisotto, E., Xing, E., Levine, S., and Salakhutdinov, R. Efficient exploration via state marginal matching. *arXiv preprint arXiv:1906.05274*, 2019.

Ng, A. Y., Harada, D., and Russell, S. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, volume 99, pp. 278–287, 1999.

Ng, A. Y., Russell, S. J., et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, pp. 663–670, 2000.

Nowozin, S., Cseke, B., and Tomioka, R. f-gan: Training generative neural samplers using variational divergence minimization. In *Advances in neural information processing systems*, pp. 271–279, 2016.

Osa, T., Sugita, N., and Mitsuishi, M. Online trajectory planning and force control for automation of surgical tasks. *IEEE Transactions on Automation Science and Engineering*, 15(2):675–691, 2017.

Pomerleau, D. A. Alvinn: An autonomous land vehicle in a neural network. In *Advances in neural information processing systems*, pp. 305–313, 1989.

Qureshi, A. H., Boots, B., and Yip, M. C. Adversarial imitation via variational inverse reinforcement learning. *arXiv preprint arXiv:1809.06404*, 2018.

Ross, S. and Bagnell, D. Efficient reductions for imitation learning. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 661–668, 2010.

Russell, S. Learning agents for uncertain environments. In *Proceedings of the eleventh annual conference on Computational learning theory*, pp. 101–103, 1998.

Singh, S. and Póczos, B. Analysis of k-nearest neighbor distances with application to entropy estimation. *arXiv preprint arXiv:1603.08578*, 2016.

Ver Steeg, G. Non-parametric entropy estimation toolbox (npeet). 2000.

Wulfmeier, M., Ondruska, P., and Posner, I. Maximum entropy deep inverse reinforcement learning. *arXiv preprint arXiv:1507.04888*, 2015.

Ziebart, B. D. Modeling purposeful adaptive behavior with the principle of maximum causal entropy. 2010.

Ziebart, B. D., Maas, A. L., Bagnell, J. A., and Dey, A. K. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pp. 1433–1438. Chicago, IL, USA, 2008.

.1. Analytical Gradient of State Marginal Distribution

In this section we compute the gradient of state marginal distribution w.r.t. parameters of the reward function. We use this gradient for the gradient of f-divergence objective in following section .2.

We start by writing the probability of trajectory τ of fixed horizon T under the optimal MaxEnt trajectory distribution for $r_\theta(s)$.

$$\rho_\theta(\tau) \propto \rho_0(s_0) \prod_{t=0}^{T-1} p(s_{t+1}|s_t, a_t) e^{\sum_{t=1}^{T} r_\theta(s_t)/\alpha} \tag{5}$$

Let $p(\tau) = \rho_0(s_0) \prod_{t=0}^{T-1} p(s_{t+1}|s_t, a_t)$, which is the probability of the trajectory under the dynamics of the environment.

Explicitly computing the normalizing factor, we can write the distribution over trajectories as follows:

$$\rho_\theta(\tau) = \frac{p(\tau) e^{\sum_{t=1}^{T} r_\theta(s_t)/\alpha}}{\int p(\tau) e^{\sum_{t=1}^{T} r_\theta(s_t)/\alpha} d\tau} \tag{6}$$

Let $\eta_\tau(s)$ denote the number of times a state occurs in a trajectory τ. We now compute the marginal distribution of all states in the trajectory:

$$\rho_\theta(s) \propto \int p(\tau) e^{\sum_{t=1}^{T} r_\theta(s_t)/\alpha} \eta_\tau(s) d\tau \tag{7}$$

where

$$\eta_\tau(s) = \sum_{t=1}^{T} \mathbb{1}(\tau_t = s) \tag{8}$$

is the empirical frequency of state s in trajectory τ (omitting the starting state τ_0 as the policy cannot change the initial state distribution).

The marginal distribution over states can now be written as:

$$\rho_\theta(s) \propto \int p(\tau) e^{\sum_{t=1}^{T} r_\theta(s_t)/\alpha} \eta_\tau(s) d\tau \tag{9}$$

In the following derivation we will use s_t to denote states in trajectory τ and s'_t to denote states from trajectory τ'. Explicitly computing the normalizing factor, the marginal distribution can be written as follows:

$$\begin{aligned} \rho_\theta(s) &= \frac{\int p(\tau) e^{\sum_{t=1}^{T} r_\theta(s_t)/\alpha} \eta_\tau(s) d\tau}{\int \int p(\tau') e^{\sum_{t=1}^{T} r_\theta(s'_t)/\alpha} \eta_{\tau'}(s') d\tau' ds'} \\ &= \frac{\int p(\tau) e^{\sum_{t=1}^{T} r_\theta(s_t)/\alpha} \eta_\tau(s) d\tau}{\int p(\tau') e^{\sum_{t=1}^{T} r_\theta(s'_t)/\alpha} \int \eta_{\tau'}(s') ds' d\tau'} \\ &= \frac{\int p(\tau) e^{\sum_{t=1}^{T} r_\theta(s_t)/\alpha} \eta_\tau(s) d\tau}{T \int p(\tau') e^{\sum_{t=1}^{T} r_\theta(s'_t)/\alpha} d\tau'} \end{aligned} \tag{10}$$

In the second step we swap the order of integration in the denominator. The last line follows because only the T states in τ satisfy $s \in \tau$. Finally, we define $f(s)$ and Z to denote the numerator (dependent on s) and denominator (normalizing constant), to simplify notation in further calculations.

$$\begin{aligned} f(s) &= \int p(\tau) e^{\sum_{t=1}^{T} r_\theta(s_t)/\alpha} \eta_\tau(s) d\tau \\ Z &= T \int p(\tau) e^{\sum_{t=1}^{T} r_\theta(s_t)/\alpha} d\tau \\ \rho_\theta(s) &= \frac{f(s)}{Z} \end{aligned} \tag{11}$$

As an initial step, we compute the derivatives of $f(s)$ and Z w.r.t reward function at some state $r_\theta(s^*)$.

$$\frac{df(s)}{dr_\theta(s^*)} = \frac{1}{\alpha}\int p(\tau)e^{\sum_{t=1}^T r_\theta(s_t)/\alpha}\eta_\tau(s)\eta_\tau(s^*)d\tau \tag{12}$$

$$\frac{dZ}{dr_\theta(s^*)} = \frac{T}{\alpha}\int p(\tau)e^{\sum_{t=1}^T r_\theta(s_t)/\alpha}\eta_\tau(s^*)d\tau = \frac{T}{\alpha}f(s^*) \tag{13}$$

We can then apply the quotient rule to compute the derivative of policy marginal distribution w.r.t the reward function.

$$\begin{aligned}\frac{d\rho_\theta(s)}{dr_\theta(s^*)} &= \frac{Z\frac{df(s)}{dr_\theta(s^*)} - f(s)\frac{dZ}{dr_\theta(s^*)}}{Z^2}\\ &= \frac{\int p(\tau)e^{\sum_{t=1}^T r_\theta(s_t)/\alpha}\eta_\tau(s)\eta_\tau(s^*)d\tau}{\alpha Z} - \frac{f(s)}{Z}\frac{Tf(s^*)}{\alpha Z}\\ &= \frac{\int p(\tau)e^{\sum_{t=1}^T r_\theta(s_t)/\alpha}\eta_\tau(s)\eta_\tau(s^*)d\tau}{\alpha Z} - \frac{T}{\alpha}\rho_\theta(s)\rho_\theta(s^*)\end{aligned} \tag{14}$$

Now we have all the tools needed to get the derivative of ρ_θ w.r.t θ by the chain rule.

$$\begin{aligned}\frac{d\rho_\theta(s)}{d\theta} &= \int \frac{d\rho_\theta(s)}{dr_\theta(s^*)}\frac{dr_\theta(s^*)}{d\theta}ds^*\\ &= \frac{1}{\alpha}\int\left(\frac{\int p(\tau)e^{\sum_{t=1}^T r_\theta(s_t)/\alpha}\eta_\tau(s)\eta_\tau(s^*)d\tau}{Z} - T\rho_\theta(s)\rho_\theta(s^*)\right)\frac{dr_\theta(s^*)}{d\theta}ds^*\\ &= \frac{1}{\alpha Z}\int\int p(\tau)e^{\sum_{t=1}^T r_\theta(s_t)/\alpha}\eta_\tau(s)\eta_\tau(s^*)\frac{dr_\theta(s^*)}{d\theta}ds^*d\tau - \frac{T}{\alpha}\rho_\theta(s)\int\rho_\theta(s^*)\frac{dr_\theta(s^*)}{d\theta}ds^*\\ &= \frac{1}{\alpha Z}\int p(\tau)e^{\sum_{t=1}^T r_\theta(s_t)/\alpha}\eta_\tau(s)\sum_{t=1}^T\frac{dr_\theta(s_t)}{d\theta}d\tau - \frac{T}{\alpha}\rho_\theta(s)\int\rho_\theta(s^*)\frac{dr_\theta(s^*)}{d\theta}ds^*\end{aligned} \tag{15}$$

.2. Derivation of f-divergence surrogate objective

f-divergence (Ali & Silvey, 1966) is a family of divergence, which generalizes forward/reverse KL divergence. Formally, let P and Q be two probability distributions over a space Ω, then for a convex function f such that $f(1) = 0$, the f-divergence of P from Q is defined as:

$$D_f(P \,||\, Q) := \int_\Omega f\left(\frac{dP}{dQ}\right)dQ \tag{16}$$

Applied to state marginal matching between expert density $\rho_E(s)$ and agent density $\rho_\theta(s)$ over state space $\mathcal{S}$, the f-divergence objective is:

$$\min_\theta L_f(\theta) := D_f(\rho_E \,||\, \rho_\theta) = \int_{\mathcal{S}} f\left(\frac{\rho_E(s)}{\rho_\theta(s)}\right)\rho_\theta(s)ds \tag{17}$$

Now we show the proof of **Theorem 3.1** on f-divergence surrogate objective:

Proof. The gradient of the original objective can be derived as:

$$\begin{aligned}\frac{dL_f(\theta)}{d\theta} &= \int\nabla_\theta\left(f\left(\frac{\rho_E(s)}{\rho_\theta(s)}\right)\rho_\theta(s)\right)ds\\ &= \int\left(f\left(\frac{\rho_E(s)}{\rho_\theta(s)}\right) - f'\left(\frac{\rho_E(s)}{\rho_\theta(s)}\right)\frac{\rho_E(s)}{\rho_\theta(s)}\right)\frac{d\rho_\theta(s)}{d\theta}ds\\ &\triangleq \int h_f\left(\frac{\rho_E(s)}{\rho_\theta(s)}\right)\frac{d\rho_\theta(s)}{d\theta}ds\end{aligned} \tag{18}$$

, where $h_f(t) \triangleq f(t) - f'(t)t$ for convenience. Note that if $f(t)$ is non-differentiable at some points, such as $f(t) = |t-1|/2$ at $t = 1$ for Total Variation distance, we take one of its subderivative.

Substituting the gradient of state marginal distribution w.r.t θ in Eq. 15, we have:

$$\begin{aligned}
\frac{dL_f(\theta)}{d\theta} &= \int h_f\left(\frac{\rho_E(s)}{\rho_\theta(s)}\right)\left(\frac{1}{\alpha Z}\int p(\tau)e^{\sum_{t=1}^T r_\theta(s_t)/\alpha}\eta_\tau(s)\sum_{t=1}^T \frac{dr_\theta(s_t)}{d\theta}d\tau - \frac{T}{\alpha}\rho_\theta(s)\int \rho_\theta(s^*)\frac{dr_\theta(s^*)}{d\theta}ds^*\right)ds \\
&= \frac{1}{\alpha Z}\int p(\tau)e^{\sum_{t=1}^T r_\theta(s_t)/\alpha}\sum_{t=1}^T h_f\left(\frac{\rho_E(s_t)}{\rho_\theta(s_t)}\right)\sum_{t=1}^T \frac{dr_\theta(s_t)}{d\theta}d\tau - \frac{T}{\alpha}\int h_f\left(\frac{\rho_E(s)}{\rho_\theta(s)}\right)\rho_\theta(s)\left(\int \rho_\theta(s^*)\frac{dr_\theta(s^*)}{d\theta}ds^*\right)ds \\
&= \frac{1}{\alpha T}\int \rho_\theta(\tau)\sum_{t=1}^T h_f\left(\frac{\rho_E(s_t)}{\rho_\theta(s_t)}\right)\sum_{t=1}^T \frac{dr_\theta(s_t)}{d\theta}d\tau - \frac{T}{\alpha}\left(\int h_f\left(\frac{\rho_E(s)}{\rho_\theta(s)}\right)\rho_\theta(s)ds\right)\left(\int \rho_\theta(s^*)\frac{dr_\theta(s^*)}{d\theta}ds^*\right) \\
&= \frac{1}{\alpha T}\mathbb{E}_{\tau\sim\rho_\theta(\tau)}\left[\sum_{t=1}^T h_f\left(\frac{\rho_E(s_t)}{\rho_\theta(s_t)}\right)\sum_{t=1}^T \frac{dr_\theta(s_t)}{d\theta}\right] - \frac{T}{\alpha}\mathbb{E}_{s\sim\rho_\theta(s)}\left[h_f\left(\frac{\rho_E(s)}{\rho_\theta(s)}\right)\right]\mathbb{E}_{s\sim\rho_\theta(s)}\left[\frac{dr_\theta(s)}{d\theta}\right]
\end{aligned} \tag{19}$$

To gain more intuition about this equation, we can convert all the expectations to be over the trajectories:

$$\frac{dL_f(\theta)}{d\theta} = \frac{1}{\alpha T}\mathbb{E}_{\tau\sim\rho_\theta(\tau)}\left[\sum_{t=1}^T h_f\left(\frac{\rho_E(s_t)}{\rho_\theta(s_t)}\right)\sum_{t=1}^T \frac{dr_\theta(s_t)}{d\theta}\right] - \frac{1}{\alpha T}\mathbb{E}_{\tau\sim\rho_\theta(\tau)}\left[\sum_{t=1}^T h_f\left(\frac{\rho_E(s_t)}{\rho_\theta(s_t)}\right)\right]\mathbb{E}_{\tau\sim\rho_\theta(\tau)}\left[\sum_{t=1}^T \frac{dr_\theta(s_t)}{d\theta}\right] \tag{20}$$

To compute this gradient we can form a surrogate objective $\tilde{L}_f$ which is not the true objective but has the correct gradient:

$$\tilde{L}_f(\theta) = \frac{1}{\alpha T}\mathbb{E}_{\tau\sim\rho_{\hat\theta}(\tau)}\left[\sum_{t=1}^T h_f\left(\frac{\rho_E(s_t)}{\rho_{\hat\theta}(s_t)}\right)\sum_{t=1}^T r_\theta(s_t)\right] - \frac{1}{\alpha T}\mathbb{E}_{\tau\sim\rho_{\hat\theta}(\tau)}\left[\sum_{t=1}^T h_f\left(\frac{\rho_E(s_t)}{\rho_{\hat\theta}(s_t)}\right)\right]\mathbb{E}_{\tau\sim\rho_{\hat\theta}(\tau)}\left[\sum_{t=1}^T r_\theta(s_t)\right] \tag{21}$$

where parameter $\hat\theta$ does not involve optimization for surrogate objective w.r.t. θ.

It is **covariance** over agent trajectories between function of density ratios and the sum of rewards:

$$\begin{aligned}
\tilde{L}_f(\theta) &= \frac{1}{\alpha T}\mathrm{cov}_{\tau\sim\rho_{\hat\theta}(\tau)}\left(\sum_{t=1}^T h_f\left(\frac{\rho_E(s_t)}{\rho_{\hat\theta}(s_t)}\right), \sum_{t=1}^T r_\theta(s_t)\right) \\
&= \frac{1}{\alpha T}\mathrm{cov}_{\tau\sim\rho_{\hat\theta}(\tau)}\left(\sum_{t=1}^T f\left(\frac{\rho_E(s_t)}{\rho_{\hat\theta}(s_t)}\right) - f'\left(\frac{\rho_E(s_t)}{\rho_{\hat\theta}(s_t)}\right)\left(\frac{\rho_E(s_t)}{\rho_{\hat\theta}(s_t)}\right), \sum_{t=1}^T r_\theta(s_t)\right)
\end{aligned} \tag{22}$$

Thus we have derived the general case of state-marginal matching surrogate objective in Eq. 22. □

Now we instantiate f-divergence as follows:

.3. Forward KL Divergence

Proof. Forward KL divergence (FKL) is defined as $f(t) = t\log t$, thus $h_f(t) = -t$, then FKL surrogate objective is:

$$\tilde{L}_{\mathrm{FKL}}(\theta) = -\frac{1}{\alpha T}\mathrm{cov}_{\tau\sim\rho_{\hat\theta}(\tau)}\left(\sum_{t=1}^T \frac{\rho_E(s_t)}{\rho_{\hat\theta}(s_t)}, \sum_{t=1}^T r_\theta(s_t)\right) \tag{23}$$

□

FKL is *mode covering*, so the agent will try to cover all the modes of the state marginal distribution potentially also visiting regions where the expert does not visit.

.4. Reverse KL Divergence

Proof. Reverse KL divergence (RKL) is defined as $f(t) = -\log t$, thus $h_f(t) = 1 - \log t$, then RKL surrogate objective is:

$$\begin{aligned}\tilde{L}_{\mathrm{RKL}}(\theta) &= \frac{1}{\alpha T}\mathrm{cov}_{\tau\sim\rho_{\hat{\theta}}(\tau)}\left(\sum_{t=1}^{T}\left(1-\log\frac{\rho_E(s_t)}{\rho_{\hat{\theta}}(s_t)}\right), \sum_{t=1}^{T} r_\theta(s_t)\right) \\ &= -\frac{1}{\alpha T}\mathrm{cov}_{\tau\sim\rho_{\hat{\theta}}(\tau)}\left(\sum_{t=1}^{T}\log\frac{\rho_E(s_t)}{\rho_{\hat{\theta}}(s_t)}, \sum_{t=1}^{T} r_\theta(s_t)\right) \\ &= -\frac{1}{\alpha T}\mathrm{cov}_{\tau\sim\rho_{\hat{\theta}}(\tau)}\left(\sum_{t=1}^{T}\log\frac{\tilde{\rho}_E(s_t)}{\rho_{\hat{\theta}}(s_t)}, \sum_{t=1}^{T} r_\theta(s_t)\right)\end{aligned} \tag{24}$$

□

Note RKL surrogate objective is special that the expert density can be in an *unnormalized* form $\tilde{\rho}_E(s) = Z\rho_E(s)$ as the normalizing constant Z can be omitted by taking the logarithm, bypassing the use of complicated sampling methods.

RKL is *mode seeking*. In this case the agent will likely adhere to modes of the state marginal distribution and will have very little probability mass where the expert has low probability mass. This makes RKL an ideal candidate in risk-averse applications.

.5. Jensen-Shannon Divergence

Proof. Jensen-Shannon divergence (JS) is defined as $f(t) = t\log t - (1+t)\log\frac{1+t}{2}$, thus $h_f(t) = -\log(1+t)$, then JS surrogate objective is:

$$\tilde{L}_{\mathrm{JS}}(\theta) = -\frac{1}{\alpha T}\mathrm{cov}_{\tau\sim\rho_{\hat{\theta}}(\tau)}\left(\sum_{t=1}^{T}\log\left(1+\frac{\rho_E(s_t)}{\rho_{\hat{\theta}}(s_t)}\right), \sum_{t=1}^{T} r_\theta(s_t)\right) \tag{25}$$

□

JS is bounded and lies between Forward KL and Reverse KL.

.6. What objective do the previous IRL algorithms optimize?

In this section, we discuss previous IRL methods and analyze which objectives they truly optimize. Our analysis shows that AIRL and GAN-GCL methods optimize for a different objective than they claim, due to their usage of incorrect importance sampling weights.

.6.1. MaxEntIRL (Ziebart et al., 2008), Deep MaxEntIRL (Wulfmeier et al., 2015), GCL (Finn et al., 2016b)

Classical IRL methods (Russell, 1998; Ng et al., 2000) obtain a policy by learning a reward function from sampled trajectories of an expert policy. MaxEntIRL (Ziebart et al., 2008) learns a stationary reward by maximizing likelihood on expert trajectories, i.e., it minimizes forward KL divergence in trajectory space under the maximum entropy RL framework. A trajectory is a temporal collection of state-action pairs, and this makes the trajectory distribution different from state-action marginal and state marginal distribution. Each objective - minimizing divergence in trajectory space τ, in state-action marginal space (s, a) and state marginal s is a different method in its own sense.

MaxEntIRL derives a surrogate objective w.r.t. reward parameter as the difference in cumulative rewards of the trajectories between the expert and the soft-optimal policy under current reward function. To train the soft-optimal policy, it requires running MaxEnt RL in an inner loop after every reward update. This algorithm has been successfully applied for predicting behaviors of taxi drivers with a linear parameterization of reward. Wulfmeier et al. (2015) shows that MaxEntIRL reward function can be parameterized as deep neural networks as well.

Guided cost learning (GCL) (Finn et al., 2016b) is one of the first methods to train rewards using neural network directly through experiences from real robots. They achieve this result by leveraging sample-efficient guided policy search for policy optimization, employing importance sampling to correct for distribution shift when the policy has not converged, and using novel regularizations in reward network. GCL optimizes for the same objective as MaxEntIRL and Deep MaxEntIRL. To summary this three work, we have the following observation: MaxEntIRL, Deep MaxEntIRL, GCL all optimize for the forward KL divergence in trajectory space, i.e. $D_{\mathrm{KL}}(\rho_E(\tau) \,||\, \rho_\theta(\tau))$.

.6.2. GAN-GCL (Finn et al., 2016a), AIRL (Fu et al., 2017), EAIRL (Qureshi et al., 2018)

Finn et al. (2016a) shows that GCL is equivalent to training GANs with a special structure in the discriminator (GAN-GCL). Note that this result uses an approximation in importance sampling, and hence the gradient estimator is *biased*. Fu et al. (2017) shows that GAN-GCL does not perform well in practice since its discriminator models density ratio over trajectories which lead to high-variance. They propose an algorithm AIRL in which the discriminator estimates the density ratio of state-action marginal, and shows that AIRL empirically performs better than GAN-GCL. AIRL also uses approximate importance sampling in its derivation, and therefore its gradient is also *biased*. GAN-GCL and AIRL claim to be able to recover a reward function due to the special structure in the discriminator. EAIRL (Qureshi et al., 2018) uses empowerment regularization on policy objective based on AIRL.

All the above algorithm optimize for an objective that they claim to be equivalent to MaxEntIRL. However, there is an approximation involved in the procedure and let us analyze what that is, by going through the derivation for *equivalence of AIRL to MaxEntIRL* as shown in Fu et al. (2017) (Appendix A of that paper).

The authors start from writing down the objective for MaxEntIRL: $\max_\theta L_{\mathrm{MaxEntIRL}}(\theta) = \mathbb{E}_{\tau\sim D}[\log p_\theta(\tau)]$.

When the trajectory distribution is induced by the soft-optimal policy under reward r_θ, it can be parameterized as $p_\theta(\tau) \propto p(s_0)\prod_{t=0}^{T-1} p(s_{t+1}|s_t,a_t)e^{r_\theta(s_t,a_t)}$, then its gradient is derived as follows:

$$\begin{aligned}\frac{d}{d\theta}L_{\mathrm{MaxEntIRL}}(\theta) &= \mathbb{E}_D\left[\frac{d}{d\theta}r_\theta(s_t,a_t)\right] - \frac{d}{d\theta}\log(Z_\theta)\\ &= \mathbb{E}_D\left[\sum_{t=1}^{T}\frac{d}{d\theta}r_\theta(s_t,a_t)\right] - \mathbb{E}_{p_\theta}\left[\sum_{t=1}^{T}\frac{d}{d\theta}r_\theta(s_t,a_t)\right]\\ &= \sum_{t=1}^{T}\mathbb{E}_D\left[\frac{d}{d\theta}r_\theta(s_t,a_t)\right] - \mathbb{E}_{p_{\theta,t}}\left[\frac{d}{d\theta}r_\theta(s_t,a_t)\right]\end{aligned} \tag{26}$$

where $p_{\theta,t}(s_t,a_t) = \int_{s_{t'!=t},a_{t'!=t}} p_\theta(\tau)$ denote the state action marginal at time t.

As it is difficult to draw samples from p_θ, the authors instead train a separate importance sampling distribution $\mu(\tau)$. For the choice of distribution they follow (Finn et al., 2016b) and use a mixture policy $\mu(a|s) = 0.5\pi(a|s) + 0.5\hat{q}(a|s)$ where $\hat{q}(a|s)$ is the rough density estimate trained on the demonstrations. This is justified as reducing the variance of the importance sampling distribution. Thus the new gradient becomes:

$$\frac{d}{d\theta}L_{\mathrm{MaxEntIRL}}(\theta) = \sum_{t=1}^{T}\mathbb{E}_D\left[\frac{d}{d\theta}r_\theta(s_t,a_t)\right] - \mathbb{E}_{\mu_t}\left[\frac{p_{\theta,t}(s_t,a_t)}{\mu_t(s_t,a_t)}\frac{d}{d\theta}r_\theta(s_t,a_t)\right] \tag{27}$$

We emphasize here $\hat{q}(a|s)$ is the density estimate trained on the demonstrations.

They additionally aim to adapt the importance sampling distribution to reduce variance by minimizing $D_{\mathrm{KL}}(\pi(\tau) \,||\, p_\theta(\tau))$, and this KL objective can be simplified to the following MaxEnt RL objective:

$$\max_{\pi}\mathbb{E}_{\pi}\left[\sum_{t=1}^{T} r_\theta(s_t, a_t) - \log\pi(a_t|s_t)\right] \tag{28}$$

This ends the derivation of gradient of MaxEntIRL. Now AIRL authors tried to show that the gradient of AIRL matches the gradient for MaxEntIRL objective shown above, i.e. $\frac{d}{d\theta}L_{\text{MaxEntIRL}}(\theta) = \frac{d}{d\theta}L_{\text{AIRL}}(\theta)$, then AIRL is equivalent to MaxEntIRL to a constant, i.e. $L_{\text{MaxEntIRL}}(\theta) = L_{\text{AIRL}}(\theta) + C$.

In AIRL, the cost learning objective is replaced by training a discriminator of the following form:

$$D_\theta(s,a) = \frac{e^{f_\theta(s,a)}}{e^{f_\theta(s,a)} + \pi(a|s)} \tag{29}$$

The objective of the discriminator is to maximize the cross-entropy between the expert demonstrations and the generated samples:

$$\begin{aligned}\max_\theta L_{\text{AIRL}}(\theta) &= \sum_{t=1}^{T}\mathbb{E}_D[\log D_\theta(s_t,a_t)] + \mathbb{E}_{\pi_t}[\log(1-D_\theta(s_t,a_t))]\\ &= \sum_{t=1}^{T}\mathbb{E}_D\left[\log\frac{e^{f_\theta(s_t,a_t)}}{e^{f_\theta(s_t,a_t)}+\pi(a_t|s_t)}\right] + \mathbb{E}_{\pi_t}\left[\log\frac{\pi(a_t|s_t)}{\pi(a_t|s_t)+e^{f_\theta(s_t,a_t)}}\right]\\ &= \sum_{t=1}^{T}\mathbb{E}_D[f_\theta(s_t,a_t)] + \mathbb{E}_{\pi_t}[\log\pi(a_t|s_t)] - 2\mathbb{E}_{\mu_t}\left[\log(\pi(a_t|s_t)) + e^{f_\theta(s_t,a_t)}\right]\end{aligned} \tag{30}$$

where μ_t is the mixture of state-action marginal from expert demonstrations and from state-action marginal induced by current policy π at time t.

In AIRL, the policy π is optimized with the following reward:

$$\begin{aligned}\hat{r}(s,a) &= \log(D_\theta(s,a)) - \log(1-D_\theta(s,a))\\ &= f_\theta(s,a) - \log\pi(a|s)\end{aligned} \tag{31}$$

Taking the derivative with respect to θ,

$$\frac{d}{d\theta}L_{\text{AIRL}}(\theta) = \sum_{t=1}^{T}\mathbb{E}_D\left[\frac{d}{d\theta}f_\theta(s_t,a_t)\right] - \mathbb{E}_{\mu_t}\left[\frac{e^{f_\theta(s_t,a_t)}}{(e^{f_\theta(s_t,a_t)}+\pi(a_t|s_t))/2}\frac{d}{d\theta}f_\theta(s_t,a_t)\right] \tag{32}$$

The authors multiply state marginal $\pi(s_t) = \int_a \pi_t(s_t,a_t)$ to the fraction term in the second expectation, and denote that $\hat{p}_{\theta,t}(s_t,a_t) \triangleq e^{f_\theta(s_t,a_t)}\pi(s_t)$ and $\hat{\mu}_t(s_t,a_t) \triangleq (e^{f_\theta(s_t,a_t)} + \pi(a_t|s_t))\pi(s_t)/2$.

Thus the gradient of the discriminator becomes:

$$\frac{d}{d\theta}L_{\text{AIRL}}(\theta) = \sum_{t=1}^{T}\mathbb{E}_D\left[\frac{d}{d\theta}f_\theta(s_t,a_t)\right] - \mathbb{E}_{\mu_t}\left[\frac{\hat{p}_{\theta,t}(s_t,a_t)}{\hat{\mu}_t(s_t,a_t)}\frac{d}{d\theta}f_\theta(s_t,a_t)\right] \tag{33}$$

The issues occurs when the authors tried to match Eq. 33 with Eq. 44, i.e. $\frac{d}{d\theta}L_{\text{MaxEntIRL}}(\theta) \overset{?}{=} \frac{d}{d\theta}L_{\text{AIRL}}(\theta)$ with same reward parameterization $f_\theta = r_\theta$.

If they are equivalent, then we have the importance weights equality:

$$\boxed{\hat{p}_{\theta,t}(s_t,a_t) = p_{\theta,t}(s_t,a_t), \hat{\mu}_t(s_t,a_t) = \mu_t(s_t,a_t)} \tag{34}$$

Then given the definitions, we have:

$$\begin{aligned} \hat{p}_{\theta,t}(s_t,a_t) &\triangleq e^{f_\theta(s_t,a_t)}\pi(s_t) = \pi^*_\theta(s_t)\pi^*_\theta(a_t|s_t) \triangleq p_{\theta,t}(s,a) \\ \hat{\mu}_t(s_t,a_t) &\triangleq (e^{f_\theta(s_t,a_t)} + \pi(a_t|s_t))\pi(s_t)/2 = (\pi(a_t|s_t) + \hat{q}(a_t|s_t))\pi(s_t)/2 \triangleq \mu_t(s_t,a_t) \end{aligned} \tag{35}$$

where π^*_θ is soft-optimal policy under reward $r_\theta = f_\theta$ (assumption), thus $\log \pi^*_\theta(a_t|s_t) = f_\theta(s_t,a_t)$. Then equivalently,

$$\boxed{e^{f_\theta(s_t,a_t)} = \hat{q}(a_t|s_t) = \pi^*_\theta(a_t|s_t) = \pi(a_t|s_t)} \tag{36}$$

Unfortunately these equivalences hold only at the global optimum of the algorithm, when the policy π reaches the expert policy $\pi_E \approx \hat{q}$ and the discriminator is also optimal. This issue also applies to GAN-GCL and EAIRL. Therefore, we have the following critical conclusion:

GAN-GCL, AIRL, EAIRL are **not** equivalent to MaxEntIRL, i.e. **not** minimizing forward KL in trajectory space.

Interestingly f-MAX Ghasemipour et al. (2019) (refer to their Appendix C) authors further go one step to prove that AIRL minimizes the reverse KL between state action marginal $D_{\mathrm{KL}}(\rho_\theta(s,a) \mid\mid \rho_E(s,a))$. This result is surprising because if both the proofs in AIRL and f-MAX paper are *correct*, it means that:

$$\text{AIRL} \equiv \text{Forward KL between trajectory distributions} \equiv \text{Reverse KL between state-action marginal distribution.} \tag{37}$$

where the equivalence between first and second is shown in AIRL paper and equivalence between first and third is given in f-MAX paper.

A bandit counter-example: Consider a one-timestep bandit MDP. Here the state-action marginal is equivalent to trajectory distribution. Applying Equation 37 on this simple MDP gives that forward KL between state-action marginal is equivalent to reverse KL between state-action marginal distribution, which is clearly wrong.

To show the equivalence of AIRL to reverse KL matching objective, Ghasemipour et al. (2019) considers that the AIRL discriminator can be trained till convergence:

$$\frac{e^{f_\theta(s,a)}}{e^{f_\theta(s,a)} + \pi(a|s)} \equiv \frac{\rho_E(s,a)}{\rho_E(s,a) + \rho_\theta(s,a)} \text{ (at convergence)} \tag{38}$$

but if this is true then $f_\theta(s,a)$ can no longer be interpreted as the stationary reward function as it is a function of current policy:

$$f_\theta(s,a) = \frac{\rho_E(s,a)}{\rho_\theta(s,a)}\pi(a|s) \tag{39}$$

This is quite different from the AIRL setup as AIRL uses $f_\theta(s,a)$ as a a policy independent reward in the policy objective step.

AIRL is **not** optimizing reverse KL in state-action marginal space.

.6.3. GAIL (Ho & Ermon, 2016), FAIRL, f-MAX-RKL (Ghasemipour et al., 2019)

Generative Adversarial Imitation Learning (GAIL) (Ho & Ermon, 2016) addresses the issue of running RL in an inner step by adversarial training (Goodfellow et al., 2014). A discriminator learns to differentiate over state-action marginal and a policy learns to maximize the rewards acquired from the discriminator in an alternating fashion. It can be further shown that the GAIL is minimizing the Jensen-Shannon divergence over state-action marginal given optimal discriminator.

Recently the idea of minimizing the divergence between expert and policy's marginal distribution is further comprehensively studied and summarized in Ke et al. (2019) and Ghasemipour et al. (2019), where the authors show that any f-divergence can be minimized for imitation through f-GAN framework (Nowozin et al., 2016). f-MAX proposes several instantiations of f-divergence: forward KL for f-MAX-FKL (FAIRL), reverse KL for f-MAX-RKL, and Jensen-Shannon for original

GAIL. Their objectives are summarized as below, where θ is policy parameter, f^* is the convex conjugate of f and T_ω is the discriminator.

$$\min_\theta D_f(\rho_E(s,a) \mid\mid \rho_\theta(s,a)) = \min_\theta \max_\omega \mathbb{E}_{(s,a)\sim\rho_E(s,a)}[T_\omega(s,a)] - \mathbb{E}_{(s,a)\sim\rho_\theta(s,a)}[f^*(T_\omega(s,a))] \tag{40}$$

These adversarial IRL methods cannot recover a reward function because they do minimax optimization with discriminator in the inner-loop (when optimal, the discriminator predicts $\frac{1}{2}$ everywhere), and have poorer convergence guarantees opposed to using an analytical gradient.

GAIL, FAIRL, f-MAX-RKL are optimizing JS, forward KL, and reverse KL in state-action marginal space, respectively.

.6.4. SMM (Lee et al., 2019)

Lee et al. (2019) presents state marginal matching (SMM) for efficient exploration by minimizing reverse KL between expert and policy's state marginals (Eq 41). However, their method cannot recover the stationary reward function because it uses fictitious play between policy π_θ and variational density q, and requires storing a historical average of policies and densities over previous iterations.

$$\max_\theta -D_{\mathrm{KL}}(\rho_\theta(s) \mid\mid \rho_E(s)) = \max_\theta \mathbb{E}_{\rho_\theta(s)}\left[\log \frac{\rho_E(s)}{\rho_\theta(s)}\right] = \max_\theta \min_q \mathbb{E}_{\rho_\theta(s)}\left[\log \frac{\rho_E(s)}{q(s)}\right] \tag{41}$$

.6.5. Summary of IRL Methods: Two Classes of Bilevel Optimization

We can generalize the related works including our method into reward-dependent and policy-dependent classes from the view of optimization objective.

For the **reward-dependent** method such as MaxEntIRL, AIRL, and our method, the objective of reward/discriminator r_θ and policy π_ϕ can be viewed as a bilevel optimization:

$$\begin{aligned} &\min_{\theta,\phi} L(r_\theta, \pi_\phi) \\ &\text{s.t. } \pi_\phi \in \arg\max_\pi g(r_\theta, \pi) \end{aligned} \tag{42}$$

Thus the policy is dependent on current reward $\pi_\phi = f(r_\theta)$, thus training on the final reward does produce optimal policy, i.e. recovering the reward.

For the **policy-dependent** method such as f-MAX, GAIL, and SMM, the objective of reward/discriminator r_θ and policy π_ϕ can be viewed as:

$$\max_\phi \min_\theta L(r_\theta, \pi_\phi) \tag{43}$$

This is a special case of bilevel optimization, minimax game. The reward is dependent on current policy $r_\theta = h(\pi_\phi)$, thus it's non-stationary and cannot guarantee to recover the reward.

.7. Implementation Details

.7.1. Environment Details

- *Pointmass*: The pointmass environment has 2D square state space with range $[0, l]^2$, and 2D actions that linearly move the agent upto a unit distance in each dimension. The agent starts from the bottom left corner at coordinate $(0, 0)$. The time horizon is $T = 30$. For the goal-reaching tasks, l is set as 4 and for the uniform exploration task, $l = 6$.
- *Reacher*: The `Reacher-v2` environment (Brockman et al., 2016) has a robotic arm with 2 DOF on a 2D arena. The state space is 8-dimensional: sine and cosine of both joint angles, and the position and velocity of the arm fingertip in x and y direction. The action controls the torques for both joints. The lengths of two bodies are $r_1 = 0.1, r_2 = 0.11$, thus the trace space of the fingertip is an annulus with $R = r_1 + r_2 = 0.21$ and $r = r_2 - r_1 = 0.01$. Since r is very small, it can be approximated as a disc with radius $R = 0.21$. The time horizon is $T = 30$. We remove the object in original reacher environment as we only focus on the fingertip trajectories. Figure 4 visualizes one frame of the environment.[2]

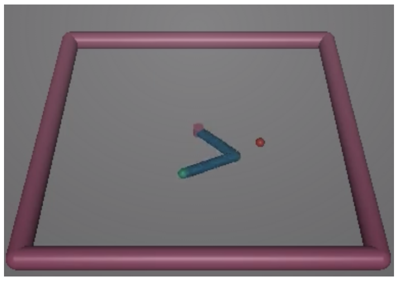


Figure 4. `Reacher-v2` environment rendering. The arm is shown in blue.

.8. Expert State Density Details

For pointmass, the state marginal domain is 2D coordinate of pointmass position; for reacher, the domain is x-y coordinate of fingertip position. We experiment with the following expert densities:

- *Single Gaussian:* for pointmass, $\mu = (l/2, l/2) = (2, 2), \sigma = 0.5$; for reacher: $\mu = (-R, 0) = (-0.21, 0), \sigma = 0.05$.
- *Mixture of two equally-weighted Gaussians:* for pointmass: $\mu_1 = (l/4, 3l/4) = (1, 3), \mu_2 = (3l/4, l/4) = (3, 1), \sigma_1 = \sigma_2 = 0.3$. reacher: $\mu_1 = (-R/\sqrt{2}, -R/\sqrt{2}), \mu_2 = (-R/\sqrt{2}, R/\sqrt{2}), \sigma_1 = \sigma_2 = 0.05$.
- *Uniform Distribution:* for pointmass, uniform over whote space $[-l, l]^2 = [-6, 6]^2$; we didn't experiment for reacher as a random policy can explore well enough.

.9. Training Details

Hyperparameters: We use SAC as the underlying RL algorithm for all compared methods. The policy network is a tanh squashed Gaussian, where the mean and std is parameterized by a (64, 64) ReLU MLP with two output heads. The Q-network is a (64, 64) ReLU MLP. We use Adam to optmize both the policy and the Q-network with a learning rate of 0.003. α is fixed to be 1. The replay buffer has a size of 12000, and we use a batch size of 256.

For our method and MaxEnt IRL, the reward function is a (64, 64) ReLU MLP. We clamp the output of the network to be within the range [-10, 10]. We also use Adam to optimize the reward network with a learning rate of 0.001.

For other baselines including AIRL, f-MAX-RKL, GAIL, a discriminator is used. We use the default discriminator architecture as in (Ghasemipour et al., 2019). In detail, first the input is linearly embedded into a 128-dim vector. This hidden state then passes through 6 resnet blocks of 128-dimensions; the residual path uses batch normalization and Tanh activation. The last hidden state is then linearly embedded into a single-dimensional output, which is the logits of the discriminator. The logit is clipped to be within the range $[-10, 10]$. The discriminator is optimized using Adam with a learning rate of 0.0003 and a batch size of 128.

[2]Figure credit: `https://gym.openai.com/envs/Reacher-v2/`

At each epoch, for all methods, we train the sac for 10 episodes using the current reward/discriminator. We do not reinitialize SAC after the reward is updated, nor do we reinitialize the replay buffer. This proves not to affect the overall training, while saving lots of computation time. For our methods and MaxEnt IRL, we update the reward for 2 gradient steps in Reacher, and 5 gradient steps in pointmass. For AIRL, f-MAX-RKL and GAIL, the discrminator takes 60 gradient steps per epoch in pointmass, and 20 gradient steps in Reacher. We train all methods for 800 epochs.

Our method and MaxEnt IRL requires an estimation of the agent density. We use kernel density estimation to fit the agent's density, using epanechnikov kernel with a bandwidth of 0.2 for pointmass, and a bandwidth of 0.02 for Reacher. At each epoch, we sample 1000 trajectories (30000 states) from the trained SAC to fit the kernel density model.

Importance Sampling for Baselines: Since we assume only access to expert density instead of expert trajectories in traditional IL framework, we use *importance sampling* for the expert term in the objectives of baselines:

- *For MaxEntIRL*: given the reward is only dependent on state, its surrogate objective can be transformed into **covariance in state marginal space** using importance sampling from agent states:

$$\begin{aligned}\tilde{L}_{\text{MaxEntIRL}}(\theta) &= \frac{1}{\alpha}\sum_{t=1}^{T}\left(\mathbb{E}_{s_t\sim\rho_{E,t}}[r_\theta(s_t)] - \mathbb{E}_{s_t\sim\rho_{\hat{\theta},t}}[r_\theta(s_t)]\right)\\ &= \frac{T}{\alpha}\left(\mathbb{E}_{s\sim\rho_E}[r_\theta(s)] - \mathbb{E}_{s\sim\hat{\rho}_\theta}[r_\theta(s)]\right)\\ &= \frac{T}{\alpha}\text{cov}_{s\sim\rho_{\hat{\theta}}}\left(\frac{\rho_E(s)}{\rho_{\hat{\theta}}(s)}, r_\theta(s)\right)\end{aligned} \tag{44}$$

  where $\rho_t(s)$ is state marginal at timestamp t, and $\rho(s) = \sum_{t=1}^{T}\rho_t(s)/T$ is state marginal averaged over all timestamps.

- *For GAIL, AIRL, f-MAX-RKL*: The discriminator needs to be trained using expert samples. Since we have only samples from the agent and the expert density, we first fit a density model to the agent distribution as $\hat{\rho}_\theta$ (using the same density model as described above), and then use importance sampling to compute the discriminator objective:

$$\min_D L(D) = -\mathbb{E}_{s\sim\hat{\rho}_\theta}\left[\frac{\rho_E(s)}{\hat{\rho}_\theta(s)}\log D(s)\right] - \mathbb{E}_{s\sim\rho_\theta}[\log(1-D(s))] \tag{45}$$

.10. Evaluation Metric Details

For the approximation of both forward and reverse KL divergence, we use non-parametric Kozachenko-Leonenko estimator (Kozachenko & Leonenko, 1987; Kraskov et al., 2004) with lower error (Singh & Póczos, 2016) compared to plug-in estimators using density models. Suggested by (Ver Steeg, 2000)[3], we choose $k = 3$ in k-nearest neighbor for Kozachenko-Leonenko estimator. Thus for each evaluation, we need to collect agent state samples and expert samples for computing the estimators.

Since in practice we use Gaussian distributions for expert density, we can easily obtain the expert samples through direct sampling before training. In addition, to avoid numeric overflow in KL estimation for two disjoint distributions, we discard the "illegal" expert samples outside the observation space of environment, and this is showed to be a *constant* offset from original KL value. Formally, we have the following statement:

Proposition .0.1 (Constant offset in f-divergence estimation)**.** *Given two probability distributions P and Q on the same domain Ω, assume that the support of P covers the whole domain, whereas the support of Q is a finite proper subset of the domain, i.e.* $\text{supp}(P) = \Omega$ *and* $\text{supp}(Q) = \Psi \subset \Omega$*. For robust estimation for KL divergence to avoid numeric overflow, here we manually assign $Q(x) = \epsilon > 0$ for $x \in \Omega - \Psi$ in the disjoint space. Then the difference in computing f-divergence $D_f(P \,||\, Q)$ on between the domain Ω and the support Ψ is constant w.r.t. Q.*

[3]https://github.com/gregversteeg/NPEET

Proof.

$$\begin{aligned} D_f(P \,||\, Q) &= \int_\Omega Q(x) f\left(\frac{P(x)}{Q(x)}\right) dx \\ &= \int_\Psi Q(x) f\left(\frac{P(x)}{Q(x)}\right) dx + \int_{\Omega-\Psi} Q(x) f\left(\frac{P(x)}{Q(x)}\right) dx \\ &= \int_\Psi Q(x) f\left(\frac{P(x)}{Q(x)}\right) dx + \int_{\Omega-\Psi} \epsilon f\left(\frac{\epsilon}{P(x)}\right) dx \\ &= \int_\Psi Q(x) f\left(\frac{P(x)}{Q(x)}\right) dx + C \end{aligned} \tag{46}$$

Specifically, for reverse KL, $D_{\mathrm{KL}}(Q \,||\, P) \to \int_\Psi Q(x) \log \frac{Q(x)}{P(x)} dx$ as $0 \log 0 \to 0$. □

Thus in practice, P is expert Gaussian density over the whole domain, and Q is agent density within environment space, according to the proposition above, we can discard expert samples outside the environment space with a constant decrease in approximate KL value across different Q.

In our experiments, before training we sample $M = 10000$ expert samples and keep the valid ones within observation space. For agent, we collect 1000 trajectories of $N = 1000 * T = 30000$ state samples. Then we use these two batches of samples to estimate KL divergence for every epoch during training.

.11. Hard-to-explore Downstream Task

Task Details: We designed a hard-to-explore task for the pointmass. The grid size is 6×6, the agent is always born at $[0, 0]$, and the goal is to reach the region $[5.95, 6] \times [5.95, 6]$. The agent only receives a reward of 1 if it reaches the goal region. To make the task more difficult, we add two distraction goals: one is at $[5.95, 6] \times [0, 0.05]$, and the other at $[0, 0.05] \times [5.95, 6]$. The agent receives a reward of 0.1 if it reaches one of these distraction goals.

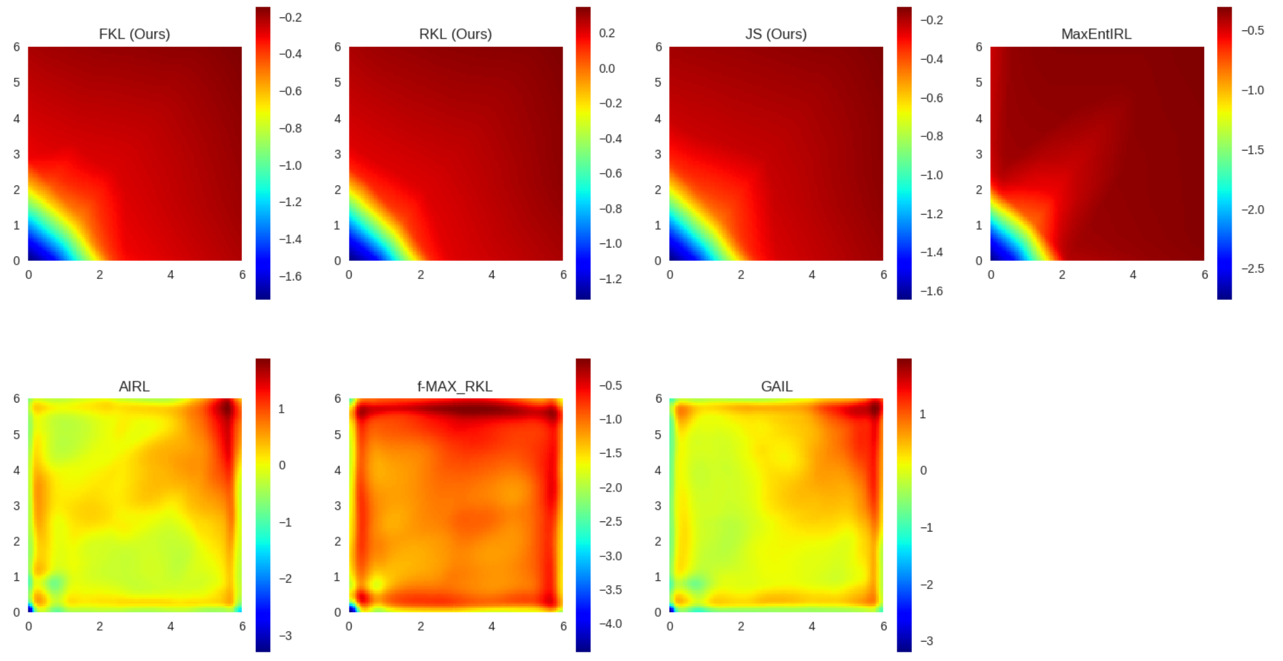


Figure 5. **Extracted final reward of all compared methods for the uniform expert density in pointmass.** We can see that the reward recovered by FKL, RKL, JS and the baseline MaxEntIRL are very similar: the reward increases as the distance to the agent's born place, the bottom left corner, increases. This is very intuitive for achieving the target unfirom density: states that takes more time to reach should have higher rewards. AIRL, f-MAX-RKL, and GAIL demonstrate a different pattern that the rewards change very smoothly, suggesting that they cannot learn the stationary reward exactly.