

A Divergence Minimization Perspective on Imitation Learning Methods

Seyed Kamyar Seyed Ghasemipour
University of Toronto, Vector Institute
kamyar@cs.toronto.edu

Richard Zemel
University of Toronto, Vector Institute
zemel@cs.toronto.edu

Shixiang Gu
Google Brain
shanegu@google.com

Abstract: In many settings, it is desirable to learn decision-making and control policies through learning or bootstrapping from expert demonstrations. The most common approaches under this Imitation Learning (IL) framework are Behavioural Cloning (BC), and Inverse Reinforcement Learning (IRL). Recent methods for IRL have demonstrated the capacity to learn effective policies with access to a very limited set of demonstrations, a scenario in which BC methods often fail. Unfortunately, due to multiple factors of variation, directly comparing these methods does not provide adequate intuition for understanding this difference in performance. In this work, we present a unified probabilistic perspective on IL algorithms based on divergence minimization. We present f -MAX, an f -divergence generalization of AIRL [1], a state-of-the-art IRL method. f -MAX enables us to relate prior IRL methods such as GAIL [2] and AIRL [1], and understand their algorithmic properties. Through the lens of divergence minimization we tease apart the differences between BC and successful IRL approaches, and empirically evaluate these nuances on simulated high-dimensional continuous control domains. Our findings conclusively identify that IRL’s state-marginal matching objective contributes most to its superior performance. Lastly, we apply our new understanding of IL method to the problem of state-marginal matching, where we demonstrate that in simulated arm pushing environments we can teach agents a diverse range of behaviours using simply hand-specified state distributions and no reward functions or expert demonstrations. For datasets and reproducing results please refer to https://github.com/KamyarGh/rl_swiss/blob/master/reproducing/fmax_paper.md.

Keywords: Imitation Learning, State-Marginal Matching

1 Introduction

Modern advances in reinforcement learning (RL) aim to alleviate the need for hand-engineered decision-making and control algorithms by designing general purpose methods that learn to optimize provided reward functions. In many cases however, it is either too challenging to optimize a given reward (e.g. due to sparsity of signal), or it is simply impossible to design a reward function that captures the intricate details of desired outcomes [3, 4, 5, 6, 7, 8]. One approach to overcoming such hurdles is Imitation Learning (IL) or Learning from Demonstrations (LfD) [3, 4, 9] where algorithms are provided with expert demonstrations of how to accomplish desired tasks.

The most common approaches in IL framework are Behavioural Cloning (BC) and Inverse Reinforcement Learning (IRL) [10, 11]. In standard BC, learning from demonstrations is treated as a supervised learning problem and policies are trained to regress expert actions from a dataset of expert demonstrations. On the other hand, in IRL the aim is to infer the reward function of the expert, and subsequently train a policy to optimize this reward. The motivation for IRL stems from the intuition that the reward function is the most concise and portable representation of a task [11, 9].

Recent “adversarial” IRL methods [2, 12, 1] have shown tremendous success in benchmarks for **continuous control** [13]; these methods outperform Behaviour Cloning by a wide margin, particularly in the low data regime where a very limited number of expert trajectories are available. However, it is not immediately clear why adversarial IRL methods outperform BC, since at optimality both methods exactly recover the expert policy. This question motivates the work presented here.



The contribution of this work are as follows. Drawing upon the literature on f -divergences [14, 15], we begin by presenting f -MAX, an algorithm for Max-Ent IRL. We demonstrate how f -MAX generalizes AIRL [1], and provides new intuition for what similar algorithms accomplish. From our findings we generate hypotheses for why Max-Ent IRL methods outperform BC, and empirically evaluate them in continuous control benchmarks. Through this process we gain a unified understanding of Imitation Learning methods from a divergence minimization perspective. To demonstrate the versatility of this perspective, we present a new approach to the recently proposed problem of state-marginal matching [16] and show that we can train policies for a diverse range of behaviours using no reward functions or expert demonstrations.

2 Related Work

The connections between RL and divergence minimization have long been studied in the rich prior literature of control as probabilistic inference [17, 18, 19, 20, 21, 22, 23]. Specifically, they have shown that optimal control under entropy regularization can be viewed as approximate inference on a graphical model, or equivalently minimizing reverse KL divergence between reward-weighted trajectory and policy trajectory distributions [20, 24]. Building on such intuitions, a number of work extended RL algorithms based on picking another divergence metric, such as forward KL [25, 26], and demonstrated substantially improved empirical performances in certain situations. Our work draws significant inspirations from these prior works in RL and aims to provide a probabilistic perspective in Imitation Learning (IL).

In the field of robotics, imitation learning (IL), or bootstrapping from IL, has often been the method of choice over RL due to difficulty in exploration and scarcity of data [3, 4, 5, 6]. While Behavioural Cloning (BC) is the most widely used IL algorithm due to the simplicity of its objective, **it suffers from the problem of covariate shift between train and test time**. Methods such as DAgger [27] and Dart [28] aim to relieve this mismatch, yet assume interactive access to expert policies.

Inverse Reinforcement Learning (IRL) algorithms have shown promising results in challenging continuous control problems [29, 30, 2, 1, 7], outperforming BC. Similarly to RL, the connections between IRL and divergence minimization have long been alluded. Early works in IRL operated by matching feature expectations or moments [9] between policies and experts, a popular approach in distribution matching [31, 32]. Furthermore, Maximum Entropy (Max-Ent) IRL [21, 22] — an IRL framework that addresses degeneracies of the original IRL formulation [10, 11] — is explicitly formulated as an energy-based modeling problem. Recent scalable approaches to Max-Ent IRL [2, 1, 12], motivated by adversarial approaches to generative modeling [33], demonstrate additional connections to distribution matching. Our work generalizes the objective proposed in [1, 12] based on recent insights from generative modeling [15], and further provides a unified perspective for viewing common IL algorithms. Concurrent to our work, Ke et al. [34] also present a unifying probabilistic perspective on IL; however, their empirical experiments solely focus on grid world domains, while our work provides comparative results on high-dimensional continuous control environments and also evaluates the effectiveness of IL algorithms for state marginal matching [16].

3 Background

Consider a Markov Decision Process (MDP) represented as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \rho_0, \gamma)$ with state-space \mathcal{S} , action-space \mathcal{A} , dynamics $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$, reward function $r(s, a)$, initial state distribution ρ_0 , and discount factor $\gamma \in (0, 1)$. Throughout this work we will denote the marginal state-action and marginal state distribution of a policy by $\rho^\pi(s, a)$ and $\rho^\pi(s)$ respectively.¹

¹Intuitively, the marginal distributions of a policy are obtained by generating infinitely many trajectories (i.e. finite or infinite horizon episodes) with a the given policy and computing the frequency of (s, a) or (s) .

Adversarial Methods for IRL Instead of recovering the reward function and policy, recent successful methods in Maximum-Entropy IRL (Max-Ent IRL) aim to directly recover the policy resulting from the full process.

GAIL: Generative Adversarial Imitation Learning Before describing the work of [2], we establish the definition of causal entropy $\mathcal{H}^{\text{causal}}(\pi) := \mathbb{E}_{\rho^\pi(s)} [\log \pi(a|s)]$ [22, 35]. Intuitively, causal entropy can be thought of as the “amount of options” the policy has in each state, in expectation.

Let \mathcal{C} denote a class of cost functions (negative reward functions). Furthermore, let $\rho^{\text{exp}}(s, a), \rho^\pi(s, a)$ denote the state-action marginal distributions of the expert and student policy respectively. Ho and Ermon [2] begin with a regularized Max-Ent IRL objective,

$$\text{IRL}_\psi(\pi^{\text{exp}}) := \arg \max_{c \in \mathcal{C}} -\psi(c) + \left(\min_{\pi} -\mathcal{H}^{\text{causal}}(\pi) + \mathbb{E}_{\rho^\pi(s,a)} [c(s, a)] \right) - \mathbb{E}_{\rho^{\text{exp}}(s,a)} [c(s, a)] \quad (1)$$

where $\psi : \mathcal{C} \rightarrow \mathbb{R}$ is a convex regularization function on the space of cost functions, and $\text{IRL}_\psi(\pi^{\text{exp}})$ returns the optimal cost function given the expert and choice of regularization. Let $\text{RL}(c) := \arg \min_{\pi} -\mathcal{H}^{\text{causal}}(\pi) + \mathbb{E}_{\pi} [c(s, a)]$, be a function that returns the optimal Max-Ent policy given cost $c(s, a)$. Ho and Ermon [2] show that

$$\text{RL} \circ \text{IRL}_\psi(\pi^{\text{exp}}) = \arg \min_{\pi} -\mathcal{H}^{\text{causal}}(\pi) + \psi^*(\rho^\pi(s, a) - \rho^{\text{exp}}(s, a)) \quad (2)$$

where ψ^* denotes the convex conjugate of ψ . This tells us that if we were to find the cost function $c(s, a)$ using the regularized Max-Ent IRL objective 1, and subsequently find the optimal Max-Ent policy for this cost, we would arrive at the same policy had we directly optimized objective 2.

Directly optimizing Equation 2 is challenging for many choices of ψ . A special case leads to the successful method dubbed Generative Adversarial Imitation Learning (GAIL). As before, let $\rho^{\text{exp}}(s, a), \rho^\pi(s, a)$ denote the state-action marginal distributions of the expert and student policy respectively. Let $D(s, a) : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$ be a binary classifier - often referred to as the discriminator - for identifying positive samples (sampled from $\rho^{\text{exp}}(s, a)$) from negative samples (sampled from $\rho^\pi(s, a)$). Using RL, the student policy is trained to maximize $\mathbb{E}_{\tau \sim \pi} [\sum_t \log D(s_t, a_t)] - \lambda \mathcal{H}^{\text{causal}}(\pi)$, where λ is a hyperparameter. **The training procedure alternates between optimizing the discriminator and updating the policy;** this procedure minimizes the Jensen-Shannon divergence between $\rho^{\text{exp}}(s, a)$ and $\rho^\pi(s, a)$ [2].

AIRL: Adversarial Inverse Reinforcement Learning Subsequent to the advent of GAIL [2], Finn et al. [12] present a theoretical discussion relating Generative Adversarial Networks (GANs) [33], IRL, and energy-based models. They demonstrate how an adversarial training approach could recover the Max-Ent reward function and simultaneously train the Max-Ent policy corresponding to that reward. Building on this discussion, Fu et al. [1] present a practical implementation of this method, named Adversarial IRL (AIRL). The main difference between AIRL [1] and GAIL [2] arises from the **objective used to train the policy** in AIRL, the policy optimizes $\mathbb{E}_{\tau \sim \pi} [\sum_t \log D(s_t, a_t) - \log (1 - D(s_t, a_t))]$. Fu et al. [1] also present additional contributions regarding the recovery of the expert reward function which is not directly relevant to this work.

Performance With Respect to BC Methods such as GAIL and AIRL have demonstrated significant performance gains compared to Behavioural Cloning. In particular, in standard Mujoco benchmarks [36, 13], adversarial methods for Max-Ent IRL achieve strong performance using a *very limited* amount of expert demonstrations, an important failure scenario for standard BC.

f -GAN Let P, Q be two distributions with density functions p, q . Motivated by variational lower bounds of f -divergences [37], as well as Generative Adversarial Networks (GANs) [33], Nowozin et al. [15] present an iterative optimization scheme for matching an implicit distribution² Q to a fixed distribution P using any f -divergence. For a given f -divergence, the corresponding minimax optimization is,

$$\min_Q \max_{T_\omega} \mathbb{E}_{x \sim P} [T_\omega(x)] - \mathbb{E}_{x \sim Q} [f^*(T_\omega(x))] \quad (3)$$

where $T_\omega : X \rightarrow \mathbb{R}$, and f^* is the convex conjugate of f . Further discussion in Appendix B.

²We use the term “implicit distributions” to refer to distributions we can efficiently sample from (e.g. GAN [38] generators) but do not necessarily have the densities of

4 f -MAX: f -Divergence Max-Ent IRL

We begin by presenting f -MAX, a generalization of AIRL [1] which provides a more intuitive interpretation of what similar algorithms accomplish. Imagine for some f -divergence we aim minimize $D_f(\rho^{\text{exp}}(s, a) || \rho^\pi(s, a))$. Using the f -GAN [15] formulation this objective can be written as,

$$\min_{\pi} \max_{T_\omega} \mathbb{E}_{(s,a) \sim \rho^{\text{exp}}(s,a)} [T_\omega(s, a)] - \mathbb{E}_{(s,a) \sim \rho^\pi(s,a)} [f^*(T_\omega(s, a))] \quad (4)$$

To optimize this objective with propose the following iterative optimization procedure,

$$\max_{T_\omega} \mathbb{E}_{(s,a) \sim \rho^{\text{exp}}(s,a)} [T_\omega(s, a)] - \mathbb{E}_{(s,a) \sim \rho^\pi(s,a)} [f^*(T_\omega(s, a))] , \quad (5)$$

$$\max_{\pi} \mathbb{E}_{\tau \sim \pi} \left[\sum_t f^*(T_\omega(s_t, a_t)) \right] \quad (6)$$

Equation 5 is the same as the inner maximization in Equation 4; this objective optimizes T_ω so that Equation 5 best approximates $D_f(\rho^{\text{exp}}(s, a) || \rho^\pi(s, a))$. On the other hand, using the identities in appendix A we have that up to a multiplicative constant, $\mathbb{E}_{\tau \sim \pi} [\sum_t f^*(T_\omega(s_t, a_t))] \propto \mathbb{E}_{(s,a) \sim \rho^\pi(s,a)} [f^*(T_\omega(s, a))]$. This implies that the policy objective (Equation 6) is equivalent to minimizing Equation 5 with respect to π . With an identical proof as in Goodfellow et al. [33, Proposition 2], if in each iteration the optimal T_ω is found, the described optimization procedure converges to the global optimum where the policy’s state-action distribution matches that of the expert.

4.1 Corollary: A Simple Derivation and Intuition for AIRL

Choosing $f(u) := -\log u$ leads to $D_f(\rho^{\text{exp}}(s, a) || \rho^\pi(s, a)) = \text{KL}(\rho^\pi(s, a) || \rho^{\text{exp}}(s, a))$. This divergence is commonly referred to as the “reverse” KL divergence. In this setting we have, $f^*(t) = -1 - \log(-t)$, and $T_\omega^\pi(s, a) = -\frac{\rho^\pi(s, a)}{\rho^{\text{exp}}(s, a)}$ [15]. As we demonstrate in Appendix C, in this setting f -MAX is equivalent to AIRL [1], meaning that AIRL is solving the Max-Ent IRL problem by minimizing the reverse KL divergence.

4.2 Relation to Cost-Regularized Max-Ent IRL

As discussed above, Ho and Ermon [2] present a class of methods for Max-Ent IRL that directly retrieve the expert policy without explicitly finding the reward function of the expert (sec. 3). Additionally, they present practical approaches for minimizing any *symmetric*³ f -divergence between $\rho^{\text{exp}}(s, a)$ and $\rho^\pi(s, a)$. Choosing the symmetric f -divergence to be the Jensen-Shannon divergence leads to the successful special case, GAIL (sec 3).

We can show that f -MAX is a subset of the cost-regularized Max-Ent IRL framework of Ho and Ermon [2]. For a given f -divergence, choosing $\psi(c) := \mathbb{E}_{\rho^{\text{exp}}(s,a)} [f^*(c(s, a)) - c(s, a)]$ we obtain⁴,

$$\psi_f^*(\rho^\pi(s, a) - \rho^{\text{exp}}(s, a)) = D_f(\rho^\pi(s, a) || \rho^{\text{exp}}(s, a)) \quad (7)$$

$$\text{RL} \circ \text{IRL}_\psi(\pi^{\text{exp}}) = \arg \min_{\pi} -\mathcal{H}^{\text{causal}}(\pi) + D_f(\rho^\pi(s, a) || \rho^{\text{exp}}(s, a)) \quad (8)$$

Typically, the causal entropy term is considered a policy regularizer, and is weighted by $0 \leq \lambda \leq 1$. Therefore, modulo the term $\mathcal{H}^{\text{causal}}(\pi)$, our derivations show that f -MAX, and by inheritance AIRL [1], all fall under the cost-regularized Max-Ent IRL framework of Ho and Ermon [2]!

5 Understanding the Relation Among Imitation Learning Methods

Given results derived in the prior section we can now begin to populate Table 1, writing various IL algorithms in a common form, as the minimization of some statistical divergence between $\rho^{\text{exp}}(s, a)$ and $\rho^\pi(s, a)$. In BC we minimize $\mathbb{E}_{\rho^{\text{exp}}(s)} [\text{KL}(\pi^{\text{exp}}(a|s) || \pi(a|s))]$ ⁵. On the other hand, the corollary in section 4.1 demonstrates that AIRL [1] minimizes $\text{KL}(\rho^\pi(s, a) || \rho^{\text{exp}}(s, a))$, while GAIL [2]

³We call an f -divergence symmetric if for any P,Q we have $D_f(P||Q) = D_f(Q||P)$

⁴Full derivations can be found in Appendix D

⁵Since $\mathbb{E}_{\rho^{\text{exp}}(s)} [\text{KL}(\rho^{\text{exp}}(a|s) || \rho^\pi(a|s))] = -\mathbb{E}_{\rho^{\text{exp}}(s,a)} [\log \rho^\pi(a|s)] - \mathcal{H}^{\text{exp}}(s, a)$ and $\mathcal{H}^{\text{exp}}(s, a)$ is constant w.r.t. the policy ($\mathcal{H}^{\text{exp}}(s, a)$ is the entropy of $\rho^{\text{exp}}(s, a)$)

Method	Optimized Objective (Minimization)
Standard Behavioural Cloning	$\mathbb{E}_{\rho^{\text{exp}}(s)} [\text{KL}(\pi^{\text{exp}}(a s) \pi(a s))] = -\mathbb{E}_{\rho^{\text{exp}}(s,a)} [\log \pi(a s)] + C$
Dagger [27]	$\mathbb{E}_{\rho^{\text{agg}_{1:n}}(s)} [\text{KL}(\pi^{\text{exp}}(a s) \pi(a s))] \text{ at iteration } n + 1$
AIRL [1]	$\text{KL}(\rho^\pi(s, a) \rho^{\text{exp}}(s, a)) = -\mathbb{E}_{\rho^\pi(s,a)} [\log \rho^{\text{exp}}(s, a)] - \mathcal{H}(\rho^\pi(s, a))$
GAIL [2]	$D_{\text{JS}}(\rho^\pi(s, a) \rho^{\text{exp}}(s, a)) - \lambda \mathcal{H}^{\text{causal}}(\pi)$
FAIRL (this work, section 6)	$\text{KL}(\rho^{\text{exp}}(s, a) \rho^\pi(s, a)) = -\mathbb{E}_{\rho^{\text{exp}}(s,a)} [\log \rho^\pi(s, a)] - \mathcal{H}(\rho^{\text{exp}}(s, a))$
symmetric f -div [2]	$D_{f\text{-symm}}(\rho^\pi(s, a) \rho^{\text{exp}}(s, a)) - \lambda \mathcal{H}^{\text{causal}}(\pi)$
f -MAX (this work, section 4)	$D_f(\rho^\pi(s, a) \rho^{\text{exp}}(s, a))$

Table 1: The objective function for various imitation learning algorithms, written in a common form as the minimization of statistical divergences. $\mathcal{H}(\cdot)$ denotes entropy, $\mathcal{H}^{\text{causal}}(\pi)$ denotes the causal entropy of the policy [22, 2], and λ is a hyperparameter. JS denotes the Jensen-Shannon divergence and D_f indicates any f -divergence. For DAgger, we are showing the objective for the simplest form of the algorithm, where $\pi^{(i)}$ is the policy obtained at iteration i , $\pi^{(1)}$ is the expert, and $\rho^{\text{agg}_{1:n}}(s) = \frac{1}{n} \sum_{i=1}^n \rho^{\pi^{(i)}}(s)$.

optimizes $D_{\text{JS}}(\rho^{\text{exp}}(s, a) || \rho^\pi(s, a)) - \lambda \mathcal{H}^{\text{causal}}(\pi)$. Hence, there are two ways in which adversarial IRL methods differ from Behavioural Cloning. First, in standard BC the policy is optimized to match the conditional distribution $\pi^{\text{exp}}(a|s)$, whereas in the other two the policy is explicitly encouraged to match the marginal state distributions as well. Second, in BC we make use of the forward KL divergence, whereas AIRL and GAIL use divergences that exhibit more mode-seeking behaviour. These observations allow us to generate the following two hypotheses about why IRL methods outperform BC, particularly in the low-data regime,

Hypothesis 1 *In common MDPs of interest, the reward function depends more on the state than the action. Hence encouraging policies to explicitly match expert state marginals is an important learning criterion.*

Hypothesis 2 *It is known that optimization using the forward KL divergence results in distributions with a mode-covering behaviour, whereas using the reverse KL results in mode-seeking behaviour [39]. In RL we care about the “quality of trajectories”, as measured by the likelihood under the expert distribution. Therefore, being mode-seeking is more beneficial than mode-covering, particularly in the low-data regime.*

In what follows, we seek to experimentally evaluate our hypotheses. To tease apart the differences between Max-Ent IRL methods and BC, we present an algorithm that optimizes the forward $\text{KL}(\rho^{\text{exp}}(s, a) || \rho^\pi(s, a))$. We then compare its performance to Behaviour Cloning and the standard AIRL algorithm using varying amounts of expert demonstrations.

6 FAIRL: An Alternative Method for Forward KL

While f -MAX is a general algorithm, useful for most choices of f , it unfortunately cannot be used for the special case of forward KL, i.e. $\text{KL}(\rho^{\text{exp}}(s, a) || \rho^\pi(s, a))$. We identify the problem in Appendix E and present a separate method that optimizes this divergence.

Similar to AIRL [1], let us have a discriminator, $D(s, a)$ whose objective is to discriminate between expert and policy state-action pairs. We now define the reward in Equation 6 for the policy to be,

$$h(s, a) := \log D(s, a) - \log(1 - D(s, a)), \quad r(s, a) := \exp(h(s, a)) \cdot (-h(s, a)) \quad (9)$$

In appendix F we show that up to a multiplicative constant, $\mathbb{E}_{\tau \sim \pi} [\sum_t r(s_t, a_t)] \propto -\text{KL}(\rho^{\text{exp}}(s, a) || \rho^\pi(s, a))$. This is a refreshing result since it demonstrates that we can convert the AIRL algorithm [1] into its forward KL counterpart by simply modifying the reward function used. We refer to this forward KL version of AIRL as FAIRL.

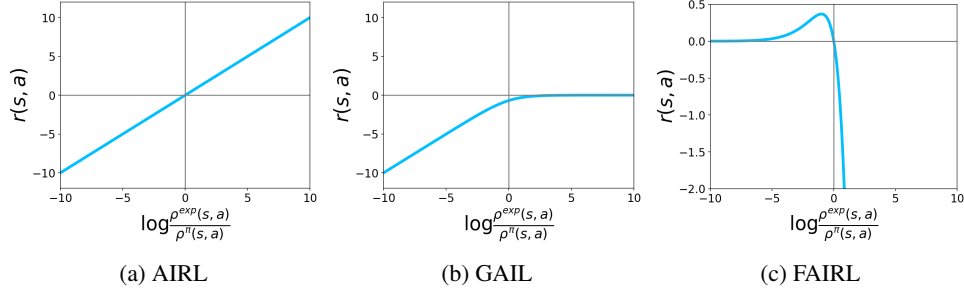


Figure 1: $r(s, a)$ as the function of the logits of the optimal discriminator, $\ell^{\text{opt}}(s, a) = \log \frac{\rho^{\text{exp}}(s, a)}{\rho^{\pi}(s, a)}$. As a reminder, AIRL, GAIL, and FAIRL respectively correspond to the reverse KL, JS, and forward KL divergences.

7 Intuition About Different Divergence Rewards

Let us parameterize the discriminator as $D(s, a) := \sigma(\ell(s, a))$, where σ represents the sigmoid activation function and $\ell(s, a)$ is the logit. Rearranging equations we can write $\ell(s, a) = \log D(s, a) - \log(1 - D(s, a))$. Hence, given a policy π , for an optimal discriminator we have that the logits are equal to the log density ratio: $\ell^{\text{opt}}(s, a) = \log \frac{\rho^{\text{exp}}(s, a)}{\rho^{\pi}(s, a)}$. It is instructive to plot the reward functions of AIRL [1], GAIL [2], and FAIRL as a function of the log density ratio; Figure 1 presents these plots.

As can be seen, in AIRL (reverse KL), the policy is encouraged to place more probability mass on regions where the expert puts more mass than the policy, and less where the expert puts less. On the other hand, the GAIL (Jensen-Shannon) reward *only* discourages the policy from placing more mass than the expert has.

Lastly, but very interestingly, the reward structure in FAIRL (forward KL) drastically differs from the previous two scenarios and has three distinct characteristics: (1) It encourages the policy to visit regions of the $\mathcal{S} \times \mathcal{A}$ space where the expert has put *slightly less* mass than the policy; (2) It does not care if the policy places a lot more mass than the expert in some regions; (3) It *severely* punishes regions where the expert has put more mass than the policy. We interpret the net effect to be that the student policy covers the expert distribution from “outwards to inwards”, meaning that it begins by placing mass in low probability regions of the space and gradually moves towards the modes of the expert’s state-action distribution.

8 Experiments

8.1 Evaluating Hypotheses

In this section we seek to empirically evaluate Hypotheses 1 and 2. For each of the HalfCheetah, Ant, Walker, and Hopper simulated environments [13] we train expert policies using Soft-Actor-Critic (SAC) [23]. Using the trained expert policies, we generated 3 sets of expert demonstrations that contained $\{4, 16, 32\}$ trajectories. Starting from a random offset, each trajectory is subsampled by a factor of 20.⁶ To compare the various learning-from-demonstration algorithms we train each method at each amount of expert demonstrations using 3 random seeds. For each seed, we checkpoint the model at its best validation loss, averaged on 10 test episodes, throughout training. At the end of training, the resulting checkpoints are evaluated on 50 test episodes. We defer additional implementation and experimental details to Appendix G.

Table 2 demonstrates that both AIRL and FAIRL outperform BC by a large margin, especially in the low data regime. The fact that FAIRL outperforms BC conclusively supports that the major performance gain of IRL methods is not due to the direction of KL divergence used, but is the result of their objectives explicitly encouraging the policy to match the marginal state distribution of the expert in addition to the matching of conditional action distribution. The fact that F/AIRL obtain

⁶This is standard protocol employed in prior adversarial methods for Max-Ent IRL [2, 1]

Method	Halfcheetah		Ant		Walker		Hopper	
	Det	Stoch	Det	Stoch	Det	Stoch	Det	Stoch
BC	-62 ± 182	-126 ± 218	82 ± 124	19 ± 70	1804 ± 1286	1293 ± 480	1435 ± 78	764 ± 129
AIRL	8043 ± 237	7377 ± 482	6024 ± 155	4598 ± 65	3979 ± 323	3846 ± 319	3393 ± 7	2561 ± 331
FAIRL	7924 ± 318	7453 ± 640	6607 ± 139	5525 ± 287	4297 ± 71	4225 ± 34	3379 ± 10	3061 ± 170
BC	641 ± 70	285 ± 166	258 ± 292	23 ± 69	656.4 ± 72	594 ± 37	2543 ± 328	1673 ± 375
AIRL	8132 ± 143	6914 ± 313	5811 ± 208	5027 ± 287	4499 ± 68	4355 ± 92	3417 ± 4	2530 ± 260
FAIRL	8275 ± 24	7900 ± 25	6267 ± 312	5473 ± 73	4824 ± 3	4778 ± 13	3429 ± 27	3335 ± 38
BC	872 ± 640	302 ± 288	147 ± 59	94 ± 11	726 ± 33	578 ± 25	2253 ± 433	1135 ± 407
AIRL	8347 ± 37	7061 ± 324	5984 ± 58	4406 ± 506	4433 ± 166	4284 ± 218	3425 ± 14	2524 ± 363
FAIRL	8302 ± 15	7522 ± 406	6365 ± 128	5442 ± 147	4807 ± 6	4764 ± 28	3428 ± 27	3415 ± 21
Dagger	8418 ± 14	6646 ± 1209	6978 ± 11	6011 ± 201	4874 ± 34	4071 ± 1073	3460 ± 5	2962 ± 157

Table 2: The performance of BC, AIRL, and FAIRL on a series of standard continuous control benchmarks [13, 36]. From top to bottom, the collection of rows present results for the settings where we use 4, 16, and 32 demonstrations trajectories (with subsampling factor of 20). “Det” and “Stoch” respectively refer to whether we evaluate using the mode of the policy’s action distribution or we sample from it. The policy architectures are held constant throughout. Hyperparameters for all models were tuned using similar budgets and values in the table report the mean and standard deviation of returns for 3 random seeds. The key trend is that in all scenarios both IRL approaches significantly outperform BC, supporting Hypothesis 1. We also observe that FAIRL trained with 32 demonstrations obtain a similar results to Dagger trained with the same expert data.

similar results to Dagger [27] further supports this claim since the Dagger algorithm was designed to mitigate the state distribution mismatch problem of BC. In conclusion, our results support Hypothesis 1 while not supporting — nor declining — Hypothesis 2.⁷ Concurrent to our work, [34] present evidence that imitation learning with mode-seeking divergences may be preferable. Further investigations into 2 would require evaluations in domains with multi-modal expert behaviours.

8.2 f -MAX for State Marginal Matching

The core intuition we have built in our work is that adversarial IRL approaches additionally match state marginal distributions, rather than only action distributions as in BC. We can thus trivially apply f -MAX formulation to state-only marginal matching⁸, i.e. minimizing $D_f(\rho^{\text{target}}(s) || \rho^\pi(s))$, by following an iterative optimization procedure similar to f -MAX in Equations 5 and 6,

$$\max_{T_\omega} \mathbb{E}_{s \sim \rho^{\text{target}}(s)} [T_\omega(s)] - \mathbb{E}_{s \sim \rho^\pi(s)} [f^*(T_\omega(s))], \quad \max_{\pi} \mathbb{E}_{\tau \sim \pi} \left[\sum_t f^*(T_\omega(s_t)) \right] \quad (10)$$

whose justification follows identically to that of f -MAX with the omission of actions. This problem was also concurrently explored in Lee et al. [16] as an approach for learning effective exploration strategies. In this work we are instead motivated from the direction of Imitation Learning, as an alternative approach for guiding policies that removes the need for expert demonstrations. Crucially, unlike in traditional IL, the target distribution need not even be a realizable state-marginal distribution; rather than gathering expensive expert demonstrations, we could rely on heuristically-designed interpretable distributions which policies will try to best match. The main difference between our setup and that of Lee et al. [16] is that in this work we assume access to *samples* from the target state-marginal distribution, whereas Lee et al. [16] operate under the setting where they have access to the target’s density function. Similar to the IRL setting, it is not necessary to use the full state either; due to prior knowledge, or simply convenience, we may wish to match distributions over features of the state. For example, in a locomotion task we can train a policy such that its distribution of x - y coordinates has a particular desired form.

Results Using various environments we design interesting target distributions and train policies using a reverse KL variant of our proposed method. In the interest of space we defer experimental details to Appendix H and use this section to focus on main results. Videos of trained policies can be found at <https://sites.google.com/view/corl2019fmaxvideos/home>.

Point-Mass and Pusher Draw Beginning with a simple point-mass domain we observe that we can train a policy to match multi-modal, complex distributions as depicted in Figure 2b. Figure 2d

⁷In our experience, in the presented benchmark settings, with sufficient hyperparameter tuning AIRL and FAIRL can outperform one-another. Additional hyperparameter tuning details discussed in Appendix I.

⁸It is important to note that state-marginal matching is not an imitation learning algorithm as is it fairly simple to design scenarios where two policies have identical state marginals yet are not the same policy.

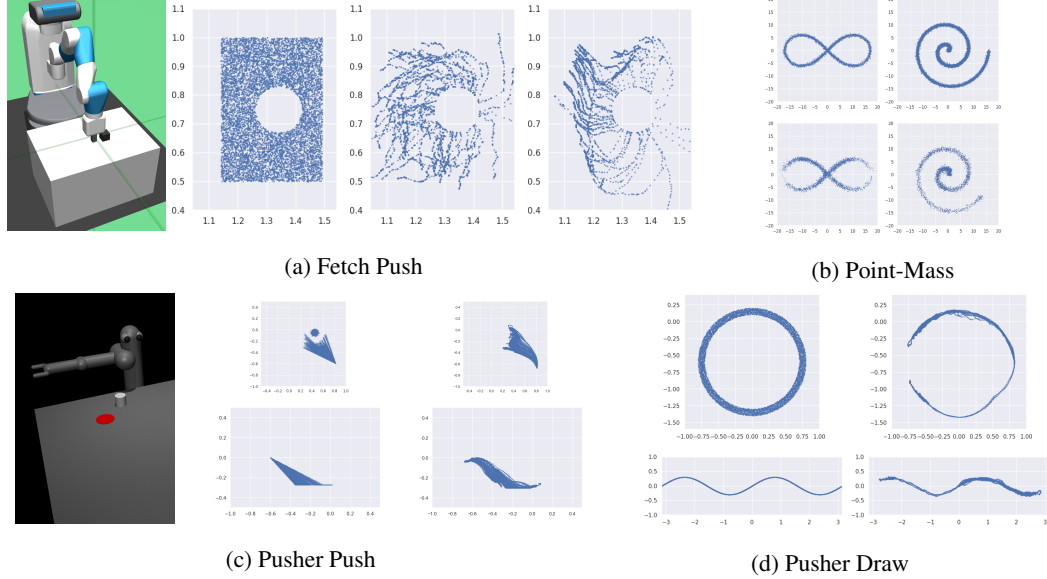


Figure 2: (a) Using the Fetch robot we demonstrate that we can train exploration policies through our approach to state-marginal matching. Figures in order are: Fetch environment, target, and two policies’ state marginals. Full image region depicts the extent of the table. (b) In the point-mass domain we train policies that exhibit complex and multi-modal trajectories. (c) Using state-marginal matching we train policies for solving the Pusher Push task. Left image is Pusher environment. The next two columns correspond to the target and policy distribution of the arm tip position. Top images are bird’s eye view (x-y) and bottom images are side view (y-z) of these distributions. (d) Pusher Draw target and policy distributions. Top images are top-down view of arm tip distribution and bottom images visualize the z coordinate as a function of angle of rotation around the circle.

demonstrates we can also train a Pusher agent [13] — a joint-velocity controlled 7-DoF simulated arm — to draw a sinusoidal function on the surface of an imaginary cylinder in 3D space.

Pusher Push Beyond path-tracing, we attempt to train the Pusher agent to solve the pushing task it was originally designed for [13]. The target and trained policy state distributions are presented in Figure 2c. While this result is far from what can be achieved using RL algorithms and the provided reward function for this task, we believe this an intriguing result as our policies were merely guided by lines and points drawn in 3D space using a short python script.

Fetch Push Lastly, in the spirit of the original motivation of [16] we examine whether our method can lead to successful exploration strategies. To this end we use the position-controlled Fetch robot in the pick and place environment [13]. We fix the grippers height slightly above the table and train policies to uniformly explore the region depicted in Figure 2a. We observe that our training procedure gives rise to a diverse range of policies which learn to push the block around the target region, yet exhibit sufficient control to prevent the block from moving outside the boundary.

9 Conclusion

The central motivation for this work stemmed from the superior performance of recent adversarial IRL methods [2, 1] compared to BC in the low-data regime, and the desire to understand the relation among various approaches for Imitation Learning. We presented f -MAX, a generalization of AIRL [1] based on f -divergence [14]. This enabled us to form a unified view of Imitation Learning and interpret various IL methods as different forms of divergence minimization. From these findings, we generated hypotheses for why IRL methods outperformed BC, and empirically evaluated them in high-dimensional continuous control benchmarks. To tease apart the differences between prior IRL methods and BC, we addressed the degeneracy of f -MAX in a special case, and provided a one-line modification of AIRL, named FAIRL, which optimizes the forward KL divergence. Our experiments conclusively disambiguated that the factor contributing most to IRL’s gain over BC is the additional state marginal matching objective. Lastly, we demonstrated the efficacy of applying f -MAX to the problem of state-marginal matching, suggesting future directions where we could replace the need for expert demonstrations in IRL with simple hand-designed state distributions.

References

- [1] J. Fu, K. Luo, and S. Levine. Learning robust rewards with adversarial inverse reinforcement learning. In *International Conference on Learning Representations*, 2018. URL <https://openreview.net/forum?id=rkHwyl-A->.
- [2] J. Ho and S. Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, pages 4565–4573, 2016.
- [3] C. G. Atkeson and S. Schaal. Robot learning from demonstration. In *ICML*, volume 97, pages 12–20. Citeseer, 1997.
- [4] S. Schaal. Learning from demonstration. In *Advances in neural information processing systems*, pages 1040–1046, 1997.
- [5] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.
- [6] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal. Learning and generalization of motor skills by learning from demonstration. In *2009 IEEE International Conference on Robotics and Automation*, pages 763–768. IEEE, 2009.
- [7] Y. Zhu, Z. Wang, J. Merel, A. Rusu, T. Erez, S. Cabi, S. Tunyasuvunakool, J. Kramár, R. Hadsell, N. de Freitas, et al. Reinforcement and imitation learning for diverse visuomotor skills. *arXiv preprint arXiv:1802.09564*, 2018.
- [8] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.
- [9] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1. ACM, 2004.
- [10] S. J. Russell. Learning agents for uncertain environments. In *COLT*, volume 98, pages 101–103, 1998.
- [11] A. Y. Ng, S. J. Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, page 2, 2000.
- [12] C. Finn, P. Christiano, P. Abbeel, and S. Levine. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *arXiv preprint arXiv:1611.03852*, 2016.
- [13] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.
- [14] J. Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information theory*, 37(1):145–151, 1991.
- [15] S. Nowozin, B. Cseke, and R. Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. In *Advances in Neural Information Processing Systems*, pages 271–279, 2016.
- [16] L. Lee, B. Eysenbach, E. Parisotto, E. Xing, S. Levine, and R. Salakhutdinov. Efficient exploration via state marginal matching. *arXiv preprint arXiv:1906.05274*, 2019.
- [17] E. Todorov. Linearly-solvable markov decision problems. In *Advances in neural information processing systems*, pages 1369–1376, 2007.
- [18] M. Toussaint. Robot trajectory optimization using approximate inference. In *Proceedings of the 26th annual international conference on machine learning*, pages 1049–1056. ACM, 2009.
- [19] J. Peters, K. Mulling, and Y. Altun. Relative entropy policy search. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.
- [20] H. J. Kappen, V. Gómez, and M. Opper. Optimal control as a graphical model inference problem. *Machine learning*, 87(2):159–182, 2012.
- [21] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.
- [22] B. D. Ziebart. Modeling purposeful adaptive behavior with the principle of maximum causal entropy. 2010.

- [23] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.
- [24] S. Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.
- [25] J. Peters and S. Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th international conference on Machine learning*, pages 745–750. ACM, 2007.
- [26] M. Norouzi, S. Bengio, N. Jaitly, M. Schuster, Y. Wu, D. Schuurmans, et al. Reward augmented maximum likelihood for neural structured prediction. In *Advances In Neural Information Processing Systems*, pages 1723–1731, 2016.
- [27] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635, 2011.
- [28] M. Laskey, J. Lee, W. Hsieh, R. Liaw, J. Mahler, R. Fox, and K. Goldberg. Iterative noise injection for scalable imitation learning. In *1st conference on robot learning (CoRL)*, (ed., Sergey Levine and Vincent Vanhoucke and Ken Goldberg), Mountain View, CA, USA, pages 13–15, 2017.
- [29] A. Coates, P. Abbeel, and A. Y. Ng. Learning for control from multiple demonstrations. In *Proceedings of the 25th international conference on Machine learning*, pages 144–151. ACM, 2008.
- [30] P. Abbeel, A. Coates, and A. Y. Ng. Autonomous helicopter aerobatics through apprenticeship learning. *The International Journal of Robotics Research*, 29(13):1608–1639, 2010.
- [31] G. K. Dziugaite, D. M. Roy, and Z. Ghahramani. Training generative neural networks via maximum mean discrepancy optimization. *arXiv preprint arXiv:1505.03906*, 2015.
- [32] Y. Li, K. Swersky, and R. Zemel. Generative moment matching networks. In *International Conference on Machine Learning*, pages 1718–1727, 2015.
- [33] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.
- [34] L. Ke, M. Barnes, W. Sun, G. Lee, S. Choudhury, and S. Srinivasa. Imitation learning as f -divergence minimization. *arXiv preprint arXiv:1905.12888*, 2019.
- [35] M. Bloem and N. Bambos. Infinite time horizon maximum causal entropy inverse reinforcement learning. In *53rd IEEE Conference on Decision and Control*, pages 4911–4916. IEEE, 2014.
- [36] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.
- [37] X. Nguyen, M. J. Wainwright, and M. I. Jordan. Estimating divergence functionals and the likelihood ratio by convex risk minimization. *IEEE Transactions on Information Theory*, 56(11):5847–5861, 2010.
- [38] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio. *Deep learning*, volume 1. MIT Press, 2016.
- [39] C. M. Bishop. *Pattern recognition and machine learning*. springer, 2006.
- [40] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. In *Advances in neural information processing systems*, pages 5767–5777, 2017.
- [41] D. McAllester and K. Statos. Formal limitations on the measurement of mutual information. *arXiv preprint arXiv:1811.04251*, 2018.

A Some Useful Identities

Let $h : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ be an arbitrary function. If all episodes have the same length T , we have,

$$\mathbb{E}_{\tau \sim \pi} \left[\sum_t h(s_t, a_t) \right] = \sum_t \mathbb{E}_{(s_t, a_t) \sim \rho^\pi(s_t, a_t)} [h(s_t, a_t)] \quad (11)$$

$$= \sum_t \int_{S, A} \rho^\pi(s_t, a_t) h(s_t, a_t) \quad (12)$$

$$= \int_{S, A} \left[\sum_t \rho^\pi(s_t, a_t) \right] h(s, a) \quad (13)$$

$$= T \cdot \int_{S, A} \rho^\pi(s, a) h(s, a) \quad (14)$$

$$= T \cdot \mathbb{E}_{(s, a) \sim \rho^\pi(s, a)} [h(s, a)] \quad (15)$$

In a somewhat similar fashion, in the infinite horizon case with fixed probability $\gamma \in (0, 1)$ of transitioning to a terminal state, for the discounted sum below we have,

$$\mathbb{E}_{\tau \sim \pi} \left[\sum_t \gamma^t h(s_t, a_t) \right] = \sum_t \mathbb{E}_{(s_t, a_t) \sim \rho^\pi(s_t, a_t)} [\gamma^t h(s_t, a_t)] \quad (16)$$

$$= \sum_t \int_{S, A} \gamma^t \rho^\pi(s_t, a_t) h(s_t, a_t) \quad (17)$$

$$= \int_{S, A} \left[\sum_t \gamma^t \rho^\pi(s_t, a_t) \right] h(s, a) \quad (18)$$

$$= \Gamma \cdot \int_{S, A} \rho^\pi(s, a) h(s, a) \quad (19)$$

$$= \Gamma \cdot \mathbb{E}_{(s, a) \sim \rho^\pi(s, a)} [h(s, a)] \quad (20)$$

where $\Gamma := \frac{1}{1-\gamma}$ is the normalizer of the sum $\sum_t \gamma^t$. Since the integral of an infinite series is not always equal to the infinite series of integrals, some analytic considerations must be made to go from equation 17 to 18. But, one simple case in which it holds is when the ranges of h and all $\rho^\pi(s_t, a_t)$ are bounded.

B More on f -divergences and f -GAN

Let P, Q be two distributions with density functions p, q . For any convex, lower-semicontinuous function $f : \mathbb{R}^+ \rightarrow \mathbb{R}$ a statistical divergence can be defined as: $D_f(P||Q) = \int_{\mathcal{X}} q(x) f\left(\frac{p(x)}{q(x)}\right)$. Divergences derived in this manner are called f -divergences and amongst many interesting divergences include the forward and reverse KL. Nguyen et al. [37] present a variational estimation method for f -divergences between arbitrary distributions P, Q . Using the notation of Nowozin et al. [15] we can write, $D_f(P||Q) \geq \sup_{T_\omega \in \mathcal{T}} (\mathbb{E}_{x \sim P} [T_\omega(x)] - \mathbb{E}_{x \sim Q} [f^*(T_\omega(x))])$, where \mathcal{T} is an arbitrary class of functions $T_\omega : \mathcal{X} \rightarrow \mathbb{R}$, and f^* is the convex conjugate of f . Under mild conditions equality holds between the two sides [37]. Motivated by this variational approximation as well as Generative Adversarial Networks (GANs) [33], Nowozin et al. [15] present an iterative optimization scheme for matching an implicit distribution⁹ Q to a fixed distribution P using any f -divergence. For a given f -divergence, the corresponding minimax optimization is,

$$\min_Q \max_{T_\omega} \mathbb{E}_{x \sim P} [T_\omega(x)] - \mathbb{E}_{x \sim Q} [f^*(T_\omega(x))] \quad (21)$$

⁹We use the term ‘‘implicit distributions’’ to refer to distributions we can efficiently sample from (e.g. GAN [38] generators) but do not necessarily have the densities of

C Corollary: A Simple Derivation and Intuition for AIRL

Choosing $f(u) := -\log u$ leads to $D_f(\rho^{\text{exp}}(s, a) || \rho^\pi(s, a)) = \text{KL}(\rho^\pi(s, a) || \rho^{\text{exp}}(s, a))$. This divergence is commonly referred to as the “reverse” KL divergence. In this setting we have, $f^*(t) = -1 - \log(-t)$, and $T_\omega^\pi(s, a) = -\frac{\rho^\pi(s, a)}{\rho^{\text{exp}}(s, a)}$ [15]. Hence, given T_ω^π , the policy objective in equation 6 takes the form,

$$\begin{aligned} & \max_{\pi} \mathbb{E}_{\tau \sim \pi} \left[\sum_t f^*(T_\omega^\pi(s_t, a_t)) \right] \\ &= \max_{\pi} \mathbb{E}_{\tau \sim \pi} \left[\sum_t \log \rho^{\text{exp}}(s_t, a_t) - \log \rho^\pi(s_t, a_t) - 1 \right] \end{aligned} \quad (22)$$

On the other hand, plugging the optimal discriminator $D^\pi(s, a) = \frac{\rho^{\text{exp}}(s, a)}{\rho^{\text{exp}}(s, a) + \rho^\pi(s, a)}$ [33] into the AIRL [1] policy objective, we get,

$$\begin{aligned} & \max_{\pi} \mathbb{E}_{\tau \sim \pi} \left[\sum_t \log D^\pi(s_t, a_t) - \log(1 - D^\pi(s_t, a_t)) \right] \\ &= \mathbb{E}_{\tau \sim \pi} \left[\sum_t \log \rho^{\text{exp}}(s_t, a_t) - \log \rho^\pi(s_t, a_t) \right] \end{aligned} \quad (23)$$

As can be seen, the right hand side of equation 23 matches that of equation 22 up to a constant ¹⁰, meaning that AIRL is solving the Max-Ent IRL problem by minimizing the reverse KL divergence, $\text{KL}(\rho^\pi(s, a) || \rho^{\text{exp}}(s, a))$!

D Simple Algebraic Manipulation

For our proof we will operate in the finite state-action space, as in the original work [2]. In this setting, cost functions can be represented as vectors in $\mathbb{R}^{S \times A}$, and joint state-action distributions can be represented as vectors in $[0, 1]^{S \times A}$. Let f be the function defining some f -divergence. Given the expert for the task, we can define the following cost function regularizer,

$$\psi_f(c) := \mathbb{E}_{\rho^{\text{exp}}(s, a)} [f^*(c(s, a)) - c(s, a)] \quad (24)$$

where f^* is the convex conjugate of f . Given this choice, with simple algebraic manipulation done below in Section D.1 we have,

$$\psi_f^*(\rho^\pi(s, a) - \rho^{\text{exp}}(s, a)) = D_f(\rho^\pi(s, a) || \rho^{\text{exp}}(s, a)) \quad (25)$$

$$\text{RL} \circ \text{IRL}_{\psi}(\pi^{\text{exp}}) = \arg \min_{\pi} -\mathcal{H}^{\text{causal}}(\pi) + D_f(\rho^\pi(s, a) || \rho^{\text{exp}}(s, a)) \quad (26)$$

Typically, the causal entropy term is considered a policy regularizer, and is weighted by $0 \leq \lambda \leq 1$. Therefore, modulo the term $\mathcal{H}^{\text{causal}}(\pi)$, our derivations show that f -MAX, and by inheritance AIRL [1], all fall under the cost-regularized Max-Ent IRL framework of [2]!

¹⁰In both settings of fixed finite horizon, and infinite horizon with constant probability of termination, the additional term resulting from the -1 is a constant.

D.1 Algebraic Manipulation

$$\psi_f^*(\rho^\pi(s, a) - \rho^{\text{exp}}(s, a)) \quad (27)$$

$$= \sup_{c \in \mathbb{R}^{S \times \mathcal{A}}} \left[(\rho^\pi(s, a) - \rho^{\text{exp}}(s, a))^T c - \psi_f(c) \right] \quad (28)$$

$$= \sup_{c \in \mathbb{R}^{S \times \mathcal{A}}} \left[\sum_{S \times \mathcal{A}} (\rho^\pi(s, a) - \rho^{\text{exp}}(s, a)) \cdot c(s, a) \right] \quad (29)$$

$$- \sum_{S \times \mathcal{A}} \rho^{\text{exp}}(s, a) \cdot (f^*(c(s, a)) - c(s, a)) \quad (30)$$

$$= \sup_{c \in \mathbb{R}^{S \times \mathcal{A}}} \left[\sum_{S \times \mathcal{A}} [\rho^\pi(s, a) \cdot c(s, a) - \rho^{\text{exp}}(s, a) \cdot f^*(c(s, a))] \right] \quad (31)$$

$$= \sup_{c \in \mathbb{R}^{S \times \mathcal{A}}} [\mathbb{E}_{\rho^\pi(s, a)} [c(s, a)] - \mathbb{E}_{\rho^{\text{exp}}(s, a)} [f^*(c(s, a))]] \quad (32)$$

$$= \sup_{T_\omega \in \mathbb{R}^{S \times \mathcal{A}}} [\mathbb{E}_{\rho^\pi(s, a)} [T_\omega(s, a)] - \mathbb{E}_{\rho^{\text{exp}}(s, a)} [f^*(T_\omega(s, a))]] \quad (33)$$

$$= D_f(\rho^\pi(s, a) || \rho^{\text{exp}}(s, a)) \quad (34)$$

To go from 32 to 33 we simply changed notation $T_\omega(s, a) := c(s, a)$, and we can go from 33 to 34 because it is the exact same form as the variational characterization of f -divergences shown in equation B. Note that equation 33 suggests the same training procedure as described for f -MAX.

E The Problem with Forward KL

Let T_ω^π denote the maximizer of equation 5 for a given policy π . For the case of forward KL, drawing upon equations from [15] we have,

$$\begin{aligned} u &:= \frac{\rho^{\text{exp}}(s, a)}{\rho^\pi(s, a)} & f(u) &:= u \log u \\ f^*(t) &= \exp(t - 1) & T_\omega^\pi &= 1 + \log \frac{\rho^{\text{exp}}(s, a)}{\rho^\pi(s, a)} \end{aligned} \quad (35)$$

Given this, the objective for the policy (equation 6) under the optimal T_ω^π becomes,

$$\max_{\pi} \mathbb{E}_{T \sim \pi} \left[\sum_t f^*(T_\omega^\pi(s_t, a_t)) \right] \quad (36)$$

$$\propto \mathbb{E}_{(s, a) \sim \rho^\pi(s, a)} [f^*(T_\omega^\pi(s, a))] \quad (37)$$

$$= \mathbb{E}_{(s, a) \sim \rho^\pi(s, a)} \left[\exp \left(\left(1 + \log \frac{\rho^{\text{exp}}(s, a)}{\rho^\pi(s, a)} \right) - 1 \right) \right] \quad (38)$$

$$= \mathbb{E}_{(s, a) \sim \rho^\pi(s, a)} \left[\frac{\rho^{\text{exp}}(s, a)}{\rho^\pi(s, a)} \right] \quad (39)$$

$$= 1 \quad (40)$$

Hence, there is no signal to train the policy! ¹¹

¹¹A similar results holds for the standard f -GAN formulation [15].

F Derivation for FAIRL

Below we present the derivation for FAIRL. Recalling definitions,

$$h(s, a) := \log D(s, a) - \log (1 - D(s, a)) \quad (41)$$

$$r(s, a) := \exp(h(s, a)) \cdot (-h(s, a)) \quad (42)$$

and assuming the discriminator is optimal¹², we have,

$$\mathbb{E}_{\tau \sim \pi} \left[\sum_t r(s_t, a_t) \right] = \mathbb{E}_{\tau \sim \pi} \left[\sum_t \exp(h(s_t, a_t)) \cdot (-h(s_t, a_t)) \right] \quad (43)$$

$$= \mathbb{E}_{\tau \sim \pi} \left[\sum_t \frac{\rho^{\exp}(s_t, a_t)}{\rho^{\pi}(s_t, a_t)} \cdot \log \frac{\rho^{\pi}(s_t, a_t)}{\rho^{\exp}(s_t, a_t)} \right] \quad (44)$$

$$\propto \mathbb{E}_{(s, a) \sim \rho^{\pi}(s, a)} \left[\frac{\rho^{\exp}(s_t, a_t)}{\rho^{\pi}(s_t, a_t)} \cdot \log \frac{\rho^{\pi}(s_t, a_t)}{\rho^{\exp}(s_t, a_t)} \right] \quad (45)$$

$$= \mathbb{E}_{(s, a) \sim \rho^{\exp}(s, a)} \left[\log \frac{\rho^{\pi}(s_t, a_t)}{\rho^{\exp}(s_t, a_t)} \right] \quad (46)$$

$$= -\text{KL}(\rho^{\exp}(s, a) || \rho^{\pi}(s, a)) \quad (47)$$

G Experimental Details for Hypotheses Evaluation

Expert Policy To simulate access to expert demonstrations we train an expert policy using Soft-Actor-Critic (SAC) [23], a state-of-the-art reinforcement learning algorithm for **continuous control**. The expert policy consists of a 2-layer MLP with 256-dim layers, ReLU activations, and two output streams for the mean and the diagonal covariance of a $\text{Tanh}(\text{Normal}(\mu, \sigma))$ distribution¹³. We use the default hyperparameter settings for training the expert.

Evaluation Setup Using a trained expert policy, we generated 3 sets of expert demonstrations of that contain $\{4, 16, 32\}$ trajectories. Starting from a random offset, each trajectory is subsampled by a factor of 20. This is standard protocol employed in prior direct methods for Max-Ent IRL [2, 1]. Also note that when generating demonstrations we *sample* from the expert’s action distribution rather than taking the mode. This way, since the expert was trained using Soft-Actor-Critic, the expert should correspond to the Max-Ent optimal policy for the reward function $\frac{1}{\tau} r_g(s, a)$, where τ is the SAC temperature used and $r_g(s, a)$ is the ground-truth reward function. To compare the various learning-from-demonstration algorithms we train each method at each amount of expert demonstrations using 3 random seeds. For each seed, we checkpoint the model at its best validation loss¹⁴ throughout training. At the end of training, the resulting checkpoints are evaluated on 50 test episodes.

Details for AIRL & FAIRL For AIRL and FAIRL, the student policy has an identical architecture to that of the expert, and the discriminator is a 2-layer MLP with 256-dim layers and Tanh activations. The logits of the discriminator is clipped to be with the range $[-10, 10]$. Gradient penalty is also used in the discriminator [40]. We normalize the observations from the environment by computing the mean and standard deviations of the expert demonstrations. The RL algorithm used for the student policies is SAC [23], and the temperature parameter is tuned separately for AIRL & FAIRL. We believe the combination of clipped discriminator logits, and tuning the gradient penalty as well as the SAC temperature allowed for effective training of IRL policies.

Details for BC For BC, we use an identical architecture as the expert. The model was fit using Maximum Likelihood Estimation¹⁵. As before, the observations from the environment are normalized using the mean and standard deviation of the expert demonstrations.

¹²As a reminder, the optimal discriminator has the form, $D(s, a) = \frac{\rho^{\exp}(s, a)}{\rho^{\exp}(s, a) + \rho^{\pi}(s, a)}$. A simple proof of which can be found in [33].

¹³This is the architecture presented in SAC [23]

¹⁴Average return on 10 test episodes

¹⁵Recall that given a state, the output of the policy is a $\text{Tanh}(\text{Normal}(\mu, \sigma))$ distribution

Details for DAgger The DAgger algorithm was trained for 200 epochs. At the end of each epoch the policy at that point is run for 8000 timesteps. The states observed through these rollouts are labelled by the expert policy and added to the aggregate dataset.

H State-Marginal Matching Experimental Details

H.1 Environments

Point-Mass The point-mass environment we experiment with is a very simple environment implemented in NumPy with 2-dimensional observation and action space. Each episode starts with the agent initialized near the origin. At each timestep, the agent’s action is a 2-dimensional vector (with magnitude of at most 1) and the agent’s location is displaced by that vector. For Inifinity sign task the horizon is 120 timesteps, and for the Spiral task the horizon is 480 steps.

Pusher Draw For this experiment we use the Pusher environment found in OpenAI gym [13] and remove the table and object. The agent is controlled by setting velocities at various joints. We also increase the range of motion of the central joint to increase range of motion. Episode horizon is 500 timesteps.

Pusher Push For this experiment we use the Pusher environment as found in OpenAI gym [13]. The agent is controlled by setting velocities at various joints. Episode horizon is the default 100 timesteps.

Fetch Push For this experiment we use the Fetch Pick and Place environment found in OpenAI gym [13]. The Fetch robot is a position controlled robot, meaning that at each timestep, the agent outputs a bounded displacement vector for the gripper position. We modify the initial state distribution to start the gripper at about the same height as the block at the center of the table. In each episode the block is randomly initialized at 0.08 radius from the center of the table. We fix the gripper height and make the action space be only operate on the x - y coordinate. Full model and environment files will be released with code release. Episode horizon is 200 timesteps.

H.2 Models

As a note, we likely used excessively large models and our choices were somewhat arbitrary with no attempt at tuning the architecture. It is very feasible that one can obtain similar performance with much more compact model architectures.

Discriminator In all experiments, the discriminator architecture is as follows. 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 $[-10, 10]$.

Policy and SAC Models For any given experiment the policy, Q, and V function had the same architecture, with the only difference being 1) the input-output dimensions, and 2) the policy outputs $\text{Tanh}(\text{Normal}(\mu, \sigma))$ given a state. All models used relu activations. The dimensionality and number of layer for each experiment were: Point-Mass (4 layers, 64 dim), Pusher (3 layers 128 dim), Fetch (3 layers, 256 dim).

H.3 Target Distributions

In this section we include the code (including parameters) used to generate the target distributions for the various experiments.

Point-Mass For the Point-Mass domain, target distributions are parameteric curves.

```
import numpy as np
```

```

def infty(r, noise_scale, num_points):
    a = np.linspace(0.0, 2*np.pi, num=num_points, endpoint=True)
    X = r*(2**0.5)*np.cos(a) / (np.sin(a)**2 + 1)
    Y = X * np.sin(a)
    X += np.random.normal(scale=noise_scale, size=X.shape)
    Y += np.random.normal(scale=noise_scale, size=Y.shape)
    return X, Y

def spiral(num_rotations, radius, noise_scale, num_points):
    a = np.linspace(0.0, 2*np.pi*num_rotations, num=num_points, endpoint=True)
    r = np.linspace(0.0, radius, num=num_points, endpoint=True)
    X = r*np.cos(a)
    Y = r*np.sin(a)
    X += np.random.normal(scale=noise_scale, size=X.shape)
    Y += np.random.normal(scale=noise_scale, size=Y.shape)
    return X, Y

if __name__ == '__main__':
    X, Y = infty(12.0, 0.3, 4000)
    X, Y = spiral(2.0, 16.0, 0.3, 16000)

```

Pusher Draw For the Pusher Draw task we ask the agent to trace a sinusoidal function on the surface of an imaginary cylinder.

```

import numpy as np

def pusher_sin_trace(noise, num_points):
    center = np.array([0.0, -0.6])
    amp = 0.3
    a = np.linspace(-np.pi, np.pi, num=num_points, endpoint=False)
    Z = amp*np.sin(2*(a+np.pi))
    r = np.random.uniform(0.7, 0.8, size=num_points)
    X = r*np.cos(a) + center[0]
    Y = r*np.sin(a) + center[1]
    return np.stack([X,Y,Z], axis=1)

if __name__ == '__main__':
    pusher_points = pusher_sin_trace(0.2, 8000)

```

Pusher Push In the Pusher Push task — i.e. the original task designed in this environment Brockman et al. [13] — the agent must move its arm to the cylinder and push it to the target region. To examine whether we can guide policies using state-marginal matching we design the following target distribution over the joint arm tip xyz position and the object xy position (5-dimensional distribution). For a desired number of imaginary episodes we sample a position to initialize the cylinder at. We then sample points along the long connecting the initial arm tip coordinates to the object. Additionally, to encourage the policy to push the object to the target region. We sample many points from a Gaussian centered at the goal position.

```

import numpy as np

def pusher_to_obj_to_goal_gaussian_target(num_episodes):
    obj_Z = -0.275
    init_arm_pos = np.array([8.20999983e-01, -5.99903808e-01, -1.25506088e-04])
    target_pos = np.array([0.45, -0.05, obj_Z])

    all_samples = []
    for _ in range(num_episodes):
        arm_traj = []
        obj_traj = []

```

```

init_obj_pos = np.array([0.45, -0.05, obj_Z])
while True:
    cylinder_pos = np.concatenate([
        np.random.uniform(low=-0.2, high=0.2, size=1),
        np.random.uniform(low=-0.3, high=0, size=1)
    ])
    if np.linalg.norm(cylinder_pos - np.zeros(2)) > 0.17:
        break
init_obj_pos[:2] = cylinder_pos + target_pos[:2]

# move arm to object
w = np.linspace(0, 1, num=50, endpoint=True)[: ,None]
arm_traj.append(
    w * init_obj_pos[None,:] + (1-w) * init_arm_pos[None,:]
)
obj_traj.append(
    np.repeat(init_obj_pos[None,:], 50, axis=0)
)

# gaussian from target center
target_region_pos = np.repeat(target_pos[None,:], 50, axis=0)
target_region_pos[:, :2] += np.random.normal(scale=0.02, size=(target_region_pos.shape[0]
arm_traj.append(
    target_region_pos
)
obj_traj.append(
    target_region_pos
)

# put the samples together
arm_traj = np.concatenate(arm_traj, axis=0)
obj_traj = np.concatenate(obj_traj, axis=0)
target_traj = np.repeat(target_pos[None,:], arm_traj.shape[0], axis=0)

all_samples.append(
    np.concatenate([arm_traj, obj_traj[:, :2]], axis=1)
)

all_samples = np.concatenate(all_samples, axis=0)
return all_samples

if __name__ == '__main__':
    pusher_points = pusher_to_obj_to_goal_gaussian_target(200)

```

Fetch Push In this experiment our goal is to examine whether we can effectively explore a desired target region. To this end we draw a rectangular region contained within the table surface and uniformly sample xy coordinates in this box and subtract a circular region about the initial state distribution. This forms the desired distribution over block positions. The full target distribution is a joint distribution between the gripper xy coordinate and that of the object. The gripper positions is obtained by sampling from a circular region around object positions.

```

import numpy as np

def uniform_box_minus_middle(num_points):
    center = np.array([1.34196849, 0.74910081])

    X = np.random.uniform(center[0] - 0.2, center[0] + 0.15, size=num_points)
    Y = np.random.uniform(center[1] - 0.25, center[1] + 0.25, size=num_points)

```

```

obj_pos = np.stack([X, Y], axis=1)
dist = np.linalg.norm(obj_pos - center[None, :], axis=1)
obj_pos = obj_pos[dist > 0.08, :]

noise = np.random.normal(size=obj_pos.shape)
noise /= np.linalg.norm(noise, axis=1, keepdims=True)
noise *= np.random.uniform(low=0.04, high=0.06, size=noise.shape)
grip_pos = obj_pos + noise

data = np.concatenate([obj_pos, grip_pos], axis=1)
return data

if __name__ == '__main__':
    data = uniform_box_minus_middle(10000)

```

I Hyperparameter Tuning

In this section we clarify our hyperparameter tuning process for AIRL and FAIRL. For both algorithms the hyperparameters to tune were 1) the Soft-Actor-Critic reward scale, and 2) the weight of the gradient penalty loss term. Our goal was to cover a wide range of benchmark domains (HalfCheetah, Ant, Walker, Hopper), with varying amounts of demonstrations (4, 16, 32), and using multiple random seeds (3 seeds). As a result, despite there being only two hyperparameters, performing a very fine hyperparameter search was unfortunately not possible for us. However, we would also like to emphasize that the gap between AIRL and FAIRL is substantially smaller than the consistent gap between BC and (A/FA)IRL.

Our tuning process for the numbers reported in Table 2 was as follows. In initial experiments with the HalfCheetah domain we discovered that the optimal range of hyperparameters for AIRL and FAIRL differed greatly. As a result, for each domain (HalfCheetah, Ant, Walker, Hopper), at each amount of expert demonstrations (4, 16, 32), we performed a 4x4 grid search using 3 random seeds. The search grid for AIRL was {reward_scale: [2.0, 4.0, 8.0, 16.0], grad_pen_weight: [2.0, 4.0, 8.0, 16.0]}, and the search grid for FAIRL was {reward_scale: [64.0, 128.0, 196.0, 256.0], grad_pen_weight: [0.01, 0.05, 0.1, 0.5]}. These grids were chosen based on initial experiments with the HalfCheetah domain. In each setting, the best hyperparameters found were used to evaluate performance.

In our experience, it can be more challenging to find the optimal range of hyperparameters for FAIRL, likely due to the discussions of Section 7. In the specific case of the Ant environment we performed a grid search over the two main hyperparameters: 1) the Soft-Actor-Critic reward scale, 2) The discriminator gradient penalty weight. As can be seen in 3, AIRL is less sensitive to these hyperparameters and obtains good performance for a wider range of hyperparameters. Note that due to the computational demand, each hyperparameter setting was run with a single random seed.

J Miscellaneous Notes

Sample-Based KL Estimates Recent investigations have demonstrated that sample-based estimates of lower bounds of KL divergence may significantly under-estimate the KL divergence [41]. In this work we instead took the approach of estimating density ratios using a discriminator, and plugging this estimate into desired divergence. To our knowledge formal bounds on the quality of these estimates have not been discussed in the literature.

On-Policy Training While State-Marginal-Matching (SMM) can be an interesting alternative to providing expert demonstrations, similar to IRL, SMM can be quite sample-inefficient due to the need for on-policy training. Investigations into off-policy IRL and SMM methods could lead to fruitful algorithms that may be applicable in a real-world setting.

Disconnected Modes in SMM In SMM, if the target distribution has disconnected modes it may be very hard for a policy to discover these modes and cover the full desired target distribution. In

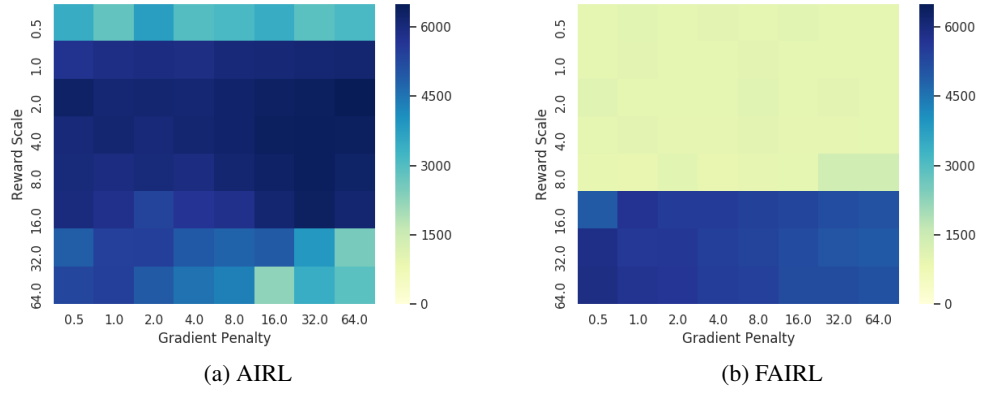


Figure 3: Hyperparameter grid search for AIRL and FAIRL in the Ant environment.

IRL this problem does not occur since the target distribution is defined by a realizable policy (i.e. the expert policy).