A Primer on Maximum Causal Entropy Inverse Reinforcement Learning

Adam Gleave* gleave@berkeley.edu

Sam Toyer* sdt@berkeley.edu

Abstract

Inverse Reinforcement Learning (IRL) [14, 1] algorithms infer a reward function that explains *demonstrations* provided by an expert acting in the environment. Maximum Causal Entropy (MCE) IRL [26, 25] is currently the most popular formulation of IRL, with numerous extensions [4, 6, 18]. In this tutorial, we present a compressed derivation of MCE IRL and the key results from contemporary implementations of MCE IRL algorithms. We hope this will serve both as an introductory resource for those new to the field, and as a concise reference for those already familiar with these topics.

Contents

*Equal contribution.

1 Introduction

The most direct approach to automating a task is to manually specify a *policy* that completes the task. However, it is often easier to specify a *reward function* defining the task, and then use reinforcement learning (RL) to train a policy [19]. Unfortunately, procedurally specifying a reward function can also be challenging. Even a task as simple as peg insertion from pixels has a non-trivial reward function [23, IV.A]. Most real-world tasks have far more complex reward functions than this, especially when they involve human interaction.

A natural resolution is to learn the reward function itself. A common approach is *Inverse Reinforcement Learning* (IRL) [14, 1]: inferring a reward function from a set of demonstrations of a particular task. This is well-suited to tasks that humans can easily perform but are difficult to manually specify a reward function for, such as walking or driving. An additional benefit is that demonstrations can be cheaply collected at scale: for example, vehicle manufacturers can learn from their customer's driving behaviour [20, 12].

A key challenge for IRL is that the problem is *under-constrained*: many different reward functions are consistent with the observed expert behaviour. Some of these differences, such as scale or potential shaping, will *never* change the optimal policy and so may be ignored [15, 7]. However, many differences *do* change the optimal policy—yet perhaps only in states that were never observed in the (finite, and often small) set of expert demonstrations. By contrast, alternative modalities such as actively querying the user for preference comparisons [17] can avoid this ambiguity, at the cost of a higher cognitive workload for the user.

Maximum Causal Entropy (MCE) IRL is by far the most popular IRL framework. Introduced by Brian Ziebart [25], MCE IRL breaks ties by favouring reward functions that lead to higher entropy policies. Informally, if the expert demonstrations are consistent either with wanting x or wanting not x, then MCE IRL will favour a reward function that is indifferent to x.

An alternative framework, Bayesian IRL [16], embraces the ambiguity and infers a *posterior distribution* over reward functions rather than choosing a point estimate. It therefore assigns probability mass to *all* reward functions compatible with the demonstrations (so long as they have support in the prior). Unfortunately, Bayesian IRL is difficult to scale, and has to date only been demonstrated in relatively simple environments such as small, discrete MDPs.

By contrast, algorithms based on MCE IRL have scaled to high-dimensional environments. Maximum Entropy Deep IRL [24] was one of the first extensions, and is able to learn rewards in gridworlds from pixel observations. More recently, Guided Cost Learning [5] and Adversarial IRL [6] have scaled to MuJoCo continuous control tasks. Given its popularity and accomplishments we focus on MCE IRL in the remainder of this document; we refer the reader to Jeon et al. [11] for a broader overview of reward learning.

2 Background

Before introducing the MCE IRL algorithm itself, we first need to introduce some notation and concepts, such as *Markov Decision Processes* (MDPs), IRL based on *feature matching*, and the notion of *causal entropy*.

2.1 Markov decision processes

A *Markov Decision Process (MDP)* $M = (\mathcal{S}, \mathcal{A}, \gamma, T, D, \mathcal{T}, r)$ consists of a set of states $\mathcal{S}$ and a set of actions $\mathcal{A}$; a discount factor $\gamma \in [0, 1]$; a horizon $T \in \mathbb{N} \cup \{\infty\}$; an initial state distribution

$D(s)$; a transition distribution $\mathcal{T}(s' \mid s, a)$ specifying the probability of transitioning to s' from s after taking action a; and a reward function $r(s, a, s')$ specifying the reward upon taking action a in state s and transitioning to state s'. To ensure convergence, MDPs must either be discounted ($\gamma < 1$) or finite-horizon ($T \in \mathbb{N}$). In the IRL problem, the true reward r is unknown and must be learned.

In practice, benchmark environments often have an indefinite horizon: episodes end when some *termination condition* is reached, which may not be related to time. Formally, this is modelled as an infinite-horizon MDP that transitions into a zero-reward absorbing state when the termination condition is satisfied. Critically, the termination condition often measures "failure" or "success": did the robot fall over? Did the vehicle reach its destination? Consequently, above-random performance can be achieved by giving the agent a reward of the correct sign up until the termination condition, then zero thereafter [13]. Implementations of IRL algorithms are often biased towards providing either positive or negative reward, so this creates a significant confounder for evaluation. We would accordingly recommend testing algorithms in fixed-horizon environments [8].

A *stochastic policy* $\pi_t(a_t \mid s_t)$ assigns probabilities to taking action $a_t \in \mathcal{A}$ in state $s_t \in \mathcal{S}$ at time step t. The probability of a policy acting in the MDP producing a *trajectory* $\tau = (s_0, a_0, s_1, a_1, \ldots, s_{T-1}, a_{T-1}, s_T)$ is given by:

$$p(\tau) = D(s_0) \prod_{t=0}^{T-1} \mathcal{T}(s_{t+1} \mid s_t, a_t)\ \pi_t(a_t \mid s_t)\,. \tag{1}$$

Note in a finite-horizon MDP ($T \in \mathbb{N}$) the policy may be *non-stationary*, i.e. it can depend on the time step t. In the infinite-horizon case ($T = \mathbb{N}$), the MDP is symmetric over time, and so we assume the policy is stationary. We drop the subscript and write only π when the policy is acting across multiple timesteps or is known to be stationary.

The objective of an agent is to maximise the expected return:

$$G(\pi) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{T-1} \gamma^t r(S_t, A_t, S_{t+1})\right]. \tag{2}$$

An optimal policy π^* maximises the expected return: $\pi^* \in \arg\max_\pi G(\pi)$.

2.2 Imitation as feature expectation matching

In IRL, our objective is to recover a reward function that—when maximised by a reinforcement learner—will lead to similar behaviour to the demonstrations. One way to formalise "similar behaviour" is by *feature expectation matching*. Suppose the demonstrator is optimising some unknown linear reward function $r_{\theta_\star}(s_t, a_t) = \theta_\star^T \phi(s_t, a_t)$, where $\phi(s_t, a_t) \in \mathbb{R}^d$ is some fixed feature mapping. In this case, the expected reward enjoyed by the demonstrator under its behaviour distribution $\mathcal{D}$ will be linear in the expected sum of discounted feature vectors observed by the agent:

$$\mathop{\mathbb{E}}_{\tau\sim\mathcal{D}}\left[\sum_{t=0}^{T-1} \gamma^t r_{\theta_\star}(S_t, A_t)\right] = \theta_\star^T \mathop{\mathbb{E}}_{\tau\sim\mathcal{D}}\left[\sum_{t=0}^{T-1} \gamma^t \phi(S_t, A_t)\right]. \tag{3}$$

Say that we recover some imitation policy π with identical expected feature counts to the

demonstrator; that is,

$$\mathop{\mathbb{E}}_{\tau\sim\pi}\left[\sum_{t=0}^{T-1}\gamma^t\phi(S_t,A_t)\right]=\mathop{\mathbb{E}}_{\tau\sim\mathcal{D}}\left[\sum_{t=0}^{T-1}\gamma^t\phi(S_t,A_t)\right]. \tag{4}$$

Because expected reward is linear in the (matched) feature expectations above, the reward enjoyed by π under the unknown true reward function $r_{\theta_\star}(s_t,a_t)$ must be the same as the reward experienced by the demonstrator $\mathcal{D}$ under that reward function [1].

If our imitation policy π is optimal under reward function parameters $\hat{\theta}$, then it is reasonable to say that $\hat{\theta}$ is an estimate of the demonstrator's true reward parameters. However, in general there will be many choices of $\hat{\theta}$ that lead to the same expected feature counts. In the next section, we will see how we can apply the *principle of maximum entropy* to break ties between these reward functions.

2.3 Maximum causal entropy

The *principle of maximum entropy* holds that when choosing between many probability distributions that are consistent with the data, one should pick the distribution that has *highest entropy*. Intuitively, such a distribution is the "most uncertain" among those that meet the data-fitting constraints. This principle can also be formally justified with an appeal to games: choosing the maximum entropy distribution minimises one's expected log loss in the setting where an adversary is able to choose the true distribution from those consistent with the data [21].

In an IRL setting, this principle leads to a simple and effective algorithm for simultaneously recovering a reward function and corresponding imitation policy from demonstrations. In particular, in MCE IRL we choose the reward function whose corresponding policy has maximal *causal entropy*. Let $S_{0:T}$ and $A_{0:T}$ be random variables of states and actions induced by following a policy π in an MDP and sampled according to eq. 1. Then the causal entropy[1] $H(S_{0:T-1}\|A_{0:T-1})$ is the sum of the entropies of the policy action selection conditioned on the state at that timestep:

$$H(A_{0:T-1}\|S_{0:T-1})=\mathbb{E}_{\tau\sim\pi}\left[-\sum_{t=0}^{T-1}\gamma^t\log\pi_t(A_t|S_t)\right]=\sum_{t=0}^{T-1}\gamma^t H(A_t|S_{0:t},A_{0:t-1}). \tag{5}$$

Note the sum is discounted, effectively valuing entropy of later actions less. This is needed for consistency with discounted returns, and for convergence in infinite-horizon problems; see Haarnoja et al. [10, appendix A] for more information.

Crucially, the causal entropy $H(A_{0:T-1}\|S_{0:T-1})$ depends only on the policy's choice at each timestep. By contrast, the conditional entropy of actions given states $H(A_{0:T-1}\mid S_{0:T-1})$ depends on states *after* each action was taken. Moreover, conventional Shannon entropy $H(S_{0:T-1},A_{0:T-1})$ calculates the entropy over the entire trajectory distribution, introducing

[1]The definition of causal entropy can also be generalised to non-Markovian contexts [25, section 4.2].

an unwanted dependency on transition dynamics via terms $H(S_t \mid S_{t-1}, A_{t-1})$:

$$H(S_{0:T-1}, A_{0:T-1}) = \sum_{t=0}^{T-1} \gamma^t H(S_t, A_t \mid S_{0:t-1}, A_{0:t-1}) \qquad \text{chain rule} \tag{6}$$

$$= \sum_{t=0}^{T-1} \gamma^t H(S_t \mid S_{0:t-1}, A_{0:t-1}) + \sum_{t=0}^{T-1} \gamma^t H(A_t \mid S_{0:t}, A_{0:t-1}) \qquad \text{chain rule} \tag{7}$$

$$= \sum_{t=0}^{T-1} \gamma^t H(S_t \mid S_{t-1}, A_{t-1}) + \sum_{t=0}^{T-1} \gamma^t H(A_t \mid S_t) \qquad \text{Markovian} \tag{8}$$

$$= \sum_{t=0}^{T-1} \gamma^t H(S_t \mid S_{t-1}, A_{t-1}) + H(A_{0:T-1} \| S_{0:T-1})\,. \tag{9}$$

Maximising Shannon entropy therefore introduces a bias towards taking actions with uncertain (and possibly risky) outcomes. For this reason, maximum causal entropy should be used rather than maximum (Shannon) entropy in stochastic MDPs. In deterministic MDPs, $H(S_t \mid S_{t-1}, A_{t-1}) = 0$ and the methods are equivalent (section 3.3).

3 Maximum Causal Entropy (MCE) IRL

In this section, we start by reviewing two complementary ways of formalising the MCE IRL objective as an optimisation problem. First, we will consider MCE IRL as a dual ascent problem. Second, we will describe MCE IRL in terms of maximum likelihood estimation, which will allow us to replace the linear reward function with a non-linear one. Finally, we will discuss how MCE IRL simplifies to Maximum Entropy (ME) IRL under deterministic dynamics.

3.1 MCE IRL as dual ascent

MCE IRL was originally introduced in Ziebart's PhD thesis [25], which considered the general setting of non-Markovian dynamics and trajectory-centric features. The thesis then derived simplifications for the special case of Markovian dynamics and policies, as well as feature functions that decompose across state–action transitions. This primer only considers the simpler case (decomposed, Markovian), and is thus substantially shorter.

The optimisation problem of finding a tabular, time-dependent policy $\pi_t(s_t, a_t)$ that matches feature expectations while maximising causal entropy can be written as

$$\max_{\{\pi_t(a_t|s_t)\geq 0\}} H(A_{0:T-1} \| S_{0:T-1}) \tag{10}$$

$$\text{Subject to} \quad \mathop{\mathbb{E}}_{\tau\sim\pi}\left[\sum_{t=0}^{T-1} \gamma^t \phi(S_t, A_t)\right] = \mathop{\mathbb{E}}_{\tau\sim\mathcal{D}}\left[\sum_{t=0}^{T-1} \gamma^t \phi(S_t, A_t)\right] \tag{11}$$

$$\sum_{a_t\in\mathcal{A}} \pi_t(a_t|s_t) = 1 \qquad (\forall t, \forall s_t)\,, \tag{12}$$

where $S_{0:T-1}$ and $A_{0:T-1}$ are random sequences of states and actions induced by π_t and sampled according to Eq. (1). It suffices to treat the non-negativity constraint, $\pi(a_t|s_t) \geq 0$, as implicit, since the causal entropy objective is undefined outside this domain.

Notice that this problem has a convex objective (minimising negative entropy) subject to affine equality constraints, and is thus convex. Further, if a solution exists, then it must necessarily be strictly feasible (i.e. have $\pi_t(a_t|s_t) > 0$ always), or else the $\log \pi_t(a_t|s_t)$ terms in the entropy will be undefined. Thus Slater's condition holds, and so we are justified in solving this problem using dual ascent. As we will see, the dual ascent perspective also allows us to recover a linear reward function $r_\theta(s_t, a_t)$ in addition to the policy π_t. Further, the dual perspective will serve as justification for a probabilistic view of IRL in the next section that can be extended to non-linear reward functions.

In order to perform dual ascent, we must first form the Lagrangian [3, Section 5.1.1] of the primal problem given above. Our Lagrangian—which we'll refer to as Λ to avoid confusion with the log likelihood $\mathcal{L}$ used in later sections—is

$$\Lambda(\pi, \theta, \mu) = -H(A_{0:T-1} \| S_{0:T-1}) \tag{13a}$$

$$+ \theta^T \left(\mathop{\mathbb{E}}_{\tau \sim \mathcal{D}} \left[\sum_{t=0}^{T-1} \gamma^t \phi(S_t, A_t) \right] - \mathop{\mathbb{E}}_{\tau \sim \pi} \left[\sum_{t=0}^{T-1} \gamma^t \phi(S_t, A_t) \right] \right) \tag{13b}$$

$$+ \sum_{t=0}^{T-1} \sum_{s_t \in \mathcal{S}} \mu_{t,s_t} \left(\sum_{a_t \in \mathcal{A}} \pi_t(a_t|s_t) - 1 \right), \tag{13c}$$

where $\theta \in \mathbb{R}^d$ and $\{\mu_{t,s} \in \mathbb{R}\}_{0 \le t \le T-1, s \in \mathcal{S}}$ are dual variables. The dual variable vector $\theta \in \mathbb{R}^d$ has been introduced to enforce the feature expectation matching constraint; we will later see that θ can be interpreted as the parameters of a linear reward function. The dual variables $\{\mu_{t,s} \in \mathbb{R}\}_{0 \le t \le T-1, s \in \mathcal{S}}$ have been introduced to enforce the policy normalisation constraint at each time step t and for each state $s_t \in \mathcal{S}$. These variables do not have a simple intuitive interpretation, but we will show that they influence the "soft value" $V^{\text{soft}}(s_t)$ of a state. As a shorthand, we will treat this set of dual variables as a matrix $\mu \in \mathbb{R}^{T-1 \times |\mathcal{S}|}$. To perform dual ascent, we will alternate between finding a minimum of $\Lambda(\pi, \theta, \mu)$ with respect to the primal variables π, then taking a single step to maximise Λ with respect to the dual variables θ and μ.

First, consider how we can minimise the Lagrangian with respect to the primal variables $\{\pi_t(a_t|s_t)\}_{0 \le t \le T-1}$. Taking the derivative of the normalisation term in Eq. (13c) with respect to some arbitrary $\pi_t(a_t|s_t)$ yields

$$\nabla_{\pi_t(a_t|s_t)} \sum_{t'=0}^{T-1} \sum_{s_{t'} \in \mathcal{S}} \mu_{t',s_{t'}} \left(\sum_{a_{t'} \in \mathcal{A}} \pi_t(a_{t'}|s_{t'}) - 1 \right) = \mu_{t,s_t} . \tag{14}$$

Note that we are differentiating with respect to the action selection probability $\pi_t(a_t \mid s_t)$, which is a variable specific to time t; the policy need not be stationary, so we may have $\pi_t(a \mid s) \neq \pi_{t'}(a|s)$.

The derivative of the feature expectation term in Eq. (13b) is

$$\nabla_{\pi_t(a_t|s_t)} \theta^T \left(\mathop{\mathbb{E}}_{\tau \sim \mathcal{D}} \left[\sum_{t=0}^{T-1} \gamma^t \phi_t(S_t, A_t) \right] - \mathop{\mathbb{E}}_{\tau \sim \pi} \left[\sum_{t=0}^{T-1} \gamma^t \phi_t(S_t, A_t) \right] \right)$$

$$= -\rho_{\pi,t}(s_t)\, \theta^T \left[\phi(s_t, a_t) + \mathop{\mathbb{E}}_{\tau \sim \pi} \left[\sum_{t'=t+1}^{T-1} \gamma^{t'-t} \phi(S_{t'}, A_{t'}) \middle| s_t, a_t \right] \right], \tag{15}$$

where

$$\rho_{\pi,t}(s_t) = \gamma^t \mathbb{E}_{\tau \sim \pi} \left[\mathcal{T}\left(s_t \mid S_{t-1}, A_{t-1}\right) \right] \tag{16}$$

is the discounted probability that the agent will be in state s_t at time t if it follows policy π. Finally, the derivative of the causal entropy term Eq. (13a) is

$$\nabla_{\pi_t(a_t|s_t)}(-H(A_{0:T-1}\|S_{0:T-1})) = \nabla_{\pi_t(a_t|s_t)} \mathop{\mathbb{E}}_{\tau\sim\pi}\left[\sum_{t'=0}^{T-1}\gamma^t \log \pi_t(A_{t'}|S_{t'})\right] \tag{17}$$

$$= \mathop{\mathbb{E}}_{\tau_{0:t-1}\sim\pi}\Bigg[\nabla_{\pi_t(a_t|s_t)}\mathcal{T}(s_t|S_{t-1},A_{t-1})\gamma^t\pi_t(a_t|s_t) \left(\log \pi_t(a_t|s_t) + \mathop{\mathbb{E}}_{\tau_{t+1:T-1}\sim\pi}\left[\sum_{t'=t+1}^{T-1}\gamma^{t'-t}\log\pi_{t'}(A_{t'}|S_{t'})\middle| s_t,a_t\right]\right)\Bigg] \tag{18}$$

$$= \rho_{\pi,t}(s_t)\left[1+\log\pi_t(a_t|s_t) + \mathop{\mathbb{E}}_{\tau_{t+1:T-1}\sim\pi}\left[\sum_{t'=t+1}^{T-1}\gamma^{t'-t}\log\pi_{t'}(A_{t'}|S_{t'})\middle| s_t,a_t\right]\right]. \tag{19}$$

Putting it all together, the derivative of the Lagrangian with respect to our policy is

$$\nabla_{\pi_t(a_t|s_t)}\left(\Lambda(\pi,\theta,\mu)\right) = \mu_{s_t} + \rho_{\pi,t}(s_t)\Bigg(1+\log\pi_t(a_t|s_t) - \theta^T\phi(s_t,a_t) - \mathop{\mathbb{E}}_{\tau\sim\pi}\left[\sum_{t'=t+1}^{T-1}\gamma^{t'-t}\left(\theta^T\phi(S_{t'},A_{t'}) - \log\pi_{t'}(A_{t'}|S_{t'})\right)\middle| s_t,a_t\right]\Bigg). \tag{20}$$

After setting the Lagrangian to zero and rearranging, we find that at optimality the policy (i.e. primal variables) must take the form

$$\pi_t(a_t|s_t) = \exp\Bigg(\theta^T\phi(s_t,a_t) - 1 - \frac{\mu_{s_t}}{\rho_{\pi,t}(s_t)} + \mathop{\mathbb{E}}_{\tau_{t+1:T-1}\sim\pi}\left[\sum_{t'=t+1}^{T-1}\gamma^{t'-t}\left(\theta^T\phi(S_{t'},A_{t'}) - \log\pi_{t'}(A_{t'}|S_{t'})\right)\middle| s_t,a_t\right]\Bigg). \tag{21}$$

Naively calculating the optimal policy from Eq. (21) would require enumeration of exponentially many trajectories to obtain the inner expectation. Fortunately, π can be recovered by dynamic programming. To see why this is the case, first decompose $\pi_{\theta,t}(a_t|s_t)$ into $\pi_{\theta,t}(a_t|s_t) = \exp(Q^{\text{soft}}_{\theta,t}(s_t,a_t) - V^{\text{soft}}_{\theta,t}(s_t))$, where the (suggestively-named) Q^{soft} and V^{soft} functions are defined as:

$$V^{\text{soft}}_{\theta,t}(s_t) \triangleq \frac{\mu_{s_t}}{\rho_{\pi,t}(s_t)} + 1\,, \tag{22}$$

$$Q^{\text{soft}}_{\theta,t}(s_t,a_t) \triangleq \theta^T\phi(s_t,a_t) + \mathop{\mathbb{E}}_{\tau\sim\pi}\left[\sum_{t'=t+1}^{T-1}\gamma^{t'-t}(\theta^T\phi(S_{t'},A_{t'}) - \log\pi_{\theta,t'}(A_{t'}|S_{t'}))\middle| s_t,a_t\right], \tag{23}$$

$$Q^{\text{soft}}_{\theta,t}(s_{T-1},a_{T-1}) \triangleq \theta^T\phi(s_{T-1},a_{T-1})\,. \tag{24}$$

$$\tag{25}$$

We can show that $Q^{\text{soft}}_{\theta,t}(s_t, a_t)$ and $V^{\text{soft}}_{\theta,t}(s_t)$ satisfy a "softened" version of the Bellman equations. By exploiting the fact that $\sum_{a_t \in \mathcal{A}}(a_t|s_t) = 1$ for all $s_t \in \mathcal{S}$ at optimality, we can show that $V^{\text{soft}}_{\theta,t}(s_t)$ must be a soft maximum over $Q^{\text{soft}}_{\theta,t}(s_t, a_t)$ values in s_t:

$$1 = \sum_{a_t \in \mathcal{A}} \pi_{\theta,t}(a_t|s_t) = \sum_{a_t \in \mathcal{A}} \exp(Q^{\text{soft}}_{\theta,t}(s_t, a_t) - V^{\text{soft}}_{\theta,t}(s_t)) \tag{26}$$

$$\exp(V^{\text{soft}}_{\theta,t}(s_t)) = \sum_{a_t \in \mathcal{A}} \exp\left(Q^{\text{soft}}_{\theta,t}(s_t, a_t)\right) \tag{27}$$

$$V^{\text{soft}}_{\theta,t}(s_t) = \log \sum_{a_t \in \mathcal{A}} \exp\left(Q^{\text{soft}}_{\theta,t}(s_t, a_t)\right) . \tag{28}$$

Likewise, we can use the definitions of $Q^{\text{soft}}_{\theta,t}$ and $V^{\text{soft}}_{\theta,t}$ to show that for $t < T$, we have

$$Q^{\text{soft}}_{\theta,t}(s_t, a_t) = \theta^T \phi(s_t, a_t) + \mathop{\mathbb{E}}_{\tau \sim \pi_\theta}\left[\sum_{t'=t+1}^{T-1} \gamma^{t'-t}(\theta^T \phi(S_{t'}, A_{t'}) - \log \pi_{\theta,t'}(A_{t'}|S_{t'}))\middle| s_t, a_t\right] \tag{29}$$

$$= \theta^T \phi(s_t, a_t) + \gamma \mathop{\mathbb{E}}_{S_{t+1} \sim \mathcal{T}}\left[\mathop{\mathbb{E}}_{A_{t+1} \sim \pi_\theta}\left[Q^{\text{soft}}_{\theta,t+1} - \log \pi_{\theta,t+1}\middle| S_{t+1}\right]\middle| s_t, a_t\right] \tag{30}$$

$$= \theta^T \phi(s_t, a_t) + \gamma \mathop{\mathbb{E}}_{\mathcal{T}}\left[\mathop{\mathbb{E}}_{\pi_\theta}\left[Q^{\text{soft}}_{\theta,t+1}\left(Q^{\text{soft}}_{\theta,t+1} - V^{\text{soft}}_{\theta,t+1}(s_{t+1})\right)\middle| S_{t+1}\right]\middle| s_t, a_t\right]$$

$$= \theta^T \phi(s_t, a_t) + \gamma \mathop{\mathbb{E}}_{S_{t+1} \sim \mathcal{T}}\left[V^{\text{soft}}_{\theta,t+1}(S_{t+1})\middle| s_t, a_t\right] , \tag{31}$$

where the penultimate step follows from substituting $\pi_{\theta,t}(a_t|s_t) = \exp\left(Q^{\text{soft}}_{\theta,t}(s_t, a_t) - V^{\text{soft}}_{\theta,t}(s_t)\right)$.

Putting it all together, we recover a set of *soft value iteration* equations, which are analogous to the finite-horizon discrete Bellman equations, but with the hard maximum over actions replaced with a soft maximum (i.e. log-sum-exp):

$$\left.\begin{aligned} V^{\text{soft}}_{\theta,t}(s_t) &= \log \sum_{a_t \in \mathcal{A}} \exp\left(Q^{\text{soft}}_{\theta,t}(s_t, a_t)\right) \\ Q^{\text{soft}}_{\theta,t}(s_t, a_t) &= \theta^T \phi(s_t, a_t) + \gamma \mathop{\mathbb{E}}_{S_{t+1} \sim \mathcal{T}}\left[V^{\text{soft}}_{\theta,t+1}(S_{t+1}) \mid s_t, a_t\right] \\ Q^{\text{soft}}_{\theta,T}(s_T, a_T) &= \theta^T \phi(s_T, a_T) \end{aligned}\right\} \text{ Soft VI.} \tag{32}$$

In the finite-horizon case, these equations can be applied recursively from time $t = T - 1$ down to $t = 0$, yielding an action distribution $\pi_{\theta,t}(a_t|s_t) = \exp(Q^{\text{soft}}_{\theta,t}(s_t, a_t) - V^{\text{soft}}_{\theta,t}(s_t))$ at each time step t and state s_t. In the infinite-horizon case, we drop the subscripts t and search for a fixed point V^{soft}_θ and Q^{soft}_θ to the soft VI equations with corresponding stationary policy π_θ.

In both cases, the agent chooses actions with probability exponential in the soft "optimal advantage" $A^{\text{soft}}_{\theta,t}(s_t, a_t) \triangleq Q^{\text{soft}}_{\theta,t}(s_t, a_t) - V^{\text{soft}}_{\theta,t}(s_t)$. Thus, minimising the Lagrangian with respect to the primal (policy) variables reduces to solving a planning problem. The similarity between the soft VI equations and the ordinary Bellman equations means that we are (somewhat) justified in interpreting the dual variable vector θ as a set of parameters for a reward function $r_\theta(s_t, a_t) = \theta^T \phi(s_t, a_t)$.

The log-sum-exp is *not* a non-expansion. In an undiscounted finite-horizon setting, $V^{\text{soft}}_{\theta,t}$ may grow at each timestep t, so implementations may need to rescale $V^{\text{soft}}_{\theta,t}$ to avoid numerical

Algorithm 1 Maximum causal entropy IRL on demonstrator $\mathcal{D}$

1: Initialise some reward parameter estimate θ_0, and set $k \leftarrow 0$
2: **repeat**
3: Apply soft value iteration to obtain optimal policy π_{θ_k} w.r.t θ_k (Eq. (32))
4: $\theta_{k+1} \leftarrow \theta_k + \alpha_k \left(\mathop{\mathbb{E}}_{\tau \sim \pi_{\theta_k}} \left[\sum_{t=0}^{T-1} \gamma^t \nabla r_\theta(S_t, A_t) \right] - \mathop{\mathbb{E}}_{\tau \sim \mathcal{D}} \left[\sum_{t=0}^{T-1} \gamma^t \nabla r_\theta(S_t, A_t) \right] \right)$ (Eq. (34))
5: $k \leftarrow k + 1$
6: **until** Stopping condition satisfied

instability. More seriously, in infinite-horizon settings there may not be a (unique) fixed point. A solution to this is to use *mellowmax*, which takes a "log-mean-exp" and *is* a non-expansion [2]:

$$V_{\theta,t}^{\text{mellow}} = \log \sum_{a_t \in \mathcal{A}} \frac{1}{|\mathcal{A}|} \exp\left(Q_{\theta,t}^{\text{soft}}(s_t, a_t)\right) . \tag{33}$$

In practice, however, log-sum-exp is widely used even for infinite-horizon algorithms (see section 4) and typically works reasonably well.

Above, we determined how to minimise the Lagrangian $\Lambda(\pi, \theta, \mu)$ with respect to the primal variables π, and found an expression for the dual variables μ in terms of π. All that remains is to determine the step that we need to take on the dual variables θ (which we are interpreting as reward parameters). The gradient of the dual w.r.t θ is clearly given by

$$\nabla_\theta \Lambda(\pi, \theta, \mu) = \mathop{\mathbb{E}}_{\tau \sim \mathcal{D}} \left[\sum_{t=0}^{T-1} \gamma^t \phi(S_t, A_t) \right] - \mathop{\mathbb{E}}_{\tau \sim \pi} \left[\sum_{t=0}^{T-1} \gamma^t \phi(S_t, A_t) \right] . \tag{34}$$

The first (demonstrator) term can be computed directly from demonstration trajectories. The second term can be computed by applying soft VI to θ and then rolling the optimal policy forward to obtain occupancy measures. This cycle of performing soft VI followed by a gradient step on θ is illustrated in Algorithm 1. In the next section, we'll see an alternative perspective on the same algorithm that does not appeal to dual ascent, and can be extended to non-linear reward functions.

3.2 MCE IRL as maximum likelihood estimation

In the previous section, we saw that IRL can be reduced to dual ascent on a maximum entropy imitation learning objective. We can also interpret this method as maximum likelihood estimation of reward parameters θ subject to the distribution over trajectories induced by the policy $\pi_{\theta,t}(a_t|s_t) = \exp(Q_{\theta,t}^{\text{soft}}(s_t, a_t) - V_{\theta,t}^{\text{soft}}(s_t))$ under the soft VI recursion in Eq. (32). This perspective has the advantage of allowing us to replace the linear reward function $r_\theta(s_t, a_t) = \theta^T \phi(s_t, a_t)$ with a general non-linear reward function $r_\theta(s_t, a_t)$. If $r_\theta(s_t, a_t)$ is non-linear, then the resulting problem no longer has the same maximum-entropy feature-matching justification as before, although it does still appear to work well in practice.

To see why the maximum-log-likelihood view is equivalent to the dual ascent view, consider that

the log discounted likelihood of trajectory τ under parameters θ is

$$\log p_\theta(\tau) = \sum_{t=0}^{T-1} \gamma^t \left[\log \mathcal{T}(s_t|s_{t-1}, a_{t-1}) + \log \pi_{\theta,t}(a_t|s_t)\right] \tag{35}$$

$$= \sum_{t=0}^{T-1} \gamma^t \left(Q^{\text{soft}}_{\theta,t}(s_t, a_t) - V^{\text{soft}}_{\theta,t}(s_t)\right) + f(\tau)\,, \tag{36}$$

where $f(\tau)$ consists of dynamics terms that are constant in θ. The expected log likelihood of a demonstrator's trajectories, sampled from demonstration distribution $\mathcal{D}$, is then

$$\mathcal{L}(\mathcal{D};\theta) = \mathop{\mathbb{E}}_{\tau\sim\mathcal{D}}\left[\log p_\theta(\tau)\right] \tag{37}$$

$$= \mathop{\mathbb{E}}_{\tau\sim\mathcal{D}}\left[\sum_{t=0}^{T-1} \gamma^t \left(Q^{\text{soft}}_{\theta,t}(S_t, A_t) - V^{\text{soft}}_{\theta,t}(S_t)\right)\right] + c \tag{38}$$

$$= \mathop{\mathbb{E}}_{\tau\sim\mathcal{D}}\left[\sum_{t=0}^{T-2} \gamma^t \left(r_\theta(S_t, A_t) + \gamma \mathop{\mathbb{E}}_{S_{t+1}}\left[V^{\text{soft}}_{\theta,t+1}(S_{t+1})|S_t, A_t\right] - V^{\text{soft}}_{\theta,t}(S_t)\right)\right. \tag{39}$$

$$\left. + \gamma^{T-1}\left(r_\theta(S_{T-1}, A_{T-1}) - V^{\text{soft}}_{\theta,T-1}(S_{T-1})\right)\right] + c$$

$$= \mathop{\mathbb{E}}_{\tau\sim\mathcal{D}}\left[\sum_{t=0}^{T-1} \gamma^t r_\theta(S_t, A_t)\right] + \mathop{\mathbb{E}}_{\tau\sim\mathcal{D}}\left[\sum_{t=0}^{T-2} \gamma^{t+1} \mathop{\mathbb{E}}_{S_{t+1}}\left[V^{\text{soft}}_{\theta,t+1}(S_{t+1})|S_t, A_t\right]\right] \tag{40}$$

$$- \mathop{\mathbb{E}}_{\tau\sim\mathcal{D}}\left[\sum_{t=0}^{T-1} \gamma^t V^{\text{soft}}_{\theta,t}(s_t)\right] + c$$

$$= \mathop{\mathbb{E}}_{\tau\sim\mathcal{D}}\left[\sum_{t=0}^{T-1} \gamma^t r_\theta(s_t, a_t)\right] - V^{\text{soft}}_{\theta,0}(s_0) + c\,, \tag{41}$$

where c is a constant that does not depend on θ. Computing the gradient of the first term with respect to θ is trivial—we just average $\nabla r_\theta(s_t, a_t)$ over the states and actions in our dataset of samples from $\mathcal{D}$. The second term is slightly more involved, but can be derived recursively as

$$\nabla_\theta V^{\text{soft}}_{\theta,t}(s_t) = \nabla_\theta \log \sum_{a_t\in\mathcal{A}} \exp Q^{\text{soft}}_{\theta,t}(s_t, a_t) \tag{42}$$

$$= \frac{1}{\sum_{a_t\in\mathcal{A}} \exp Q^{\text{soft}}_{\theta,t}(s_t, a_t)} \sum_{a_t\in\mathcal{A}} \nabla_\theta \exp Q^{\text{soft}}_{\theta,t}(s_t, a_t) \tag{43}$$

$$= \frac{\sum_{a_t\in\mathcal{A}} \exp\left(Q^{\text{soft}}_{\theta,t}(s_t, a_t)\right) \nabla_\theta Q^{\text{soft}}_{\theta,t}(s_t, a_t)}{\exp V^{\text{soft}}_{\theta,t}(s_t)} \tag{44}$$

$$= \sum_{a_t\in\mathcal{A}} \pi_{\theta,t}(a_t|s_t) \nabla_\theta Q^{\text{soft}}_{\theta,t}(s_t, a_t) \tag{45}$$

$$= \mathop{\mathbb{E}}_{A_t\sim\pi_{\theta,t}}\left[\nabla_\theta r_\theta(s_t, A_t) + \gamma \mathop{\mathbb{E}}_{S_{t+1}\sim\mathcal{T}}[\nabla_\theta V^{\text{soft}}_{\theta,t+1}(S_{t+1})|s_t, a_t])\middle| s_t\right] \tag{46}$$

$$= \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t'=t}^{T-1} \gamma^{t'-t} \nabla_\theta r_\theta(S_{t'}, A_{t'})\middle| s_t\right]\,, \tag{47}$$

where the last step unrolled the recursion to time T. Armed with this derivative, the gradient of our log likelihood is

$$\nabla_\theta \mathcal{L}(\mathcal{D};\theta) = \mathop{\mathbb{E}}_{\tau\sim\mathcal{D}}\left[\sum_{t=0}^{T-1}\gamma^t \nabla_\theta r_\theta(S_t,A_t)\right] - \mathop{\mathbb{E}}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T-1}\gamma^t \nabla_\theta r_\theta(S_t,A_t)\right]. \tag{48}$$

In the special case of a linear reward $r_\theta(s,a) = \theta^T\phi(s,a)$, this gradient is equal to the one derived in the previous section, showing the equivalence of the dual ascent and maximum likelihood views.

3.3 Maximum Entropy IRL: MCE IRL for deterministic MDPs

So far we've considered the general setting of stochastic transition dynamics, where for any given state–action pair $(s_t, a_t) \in \mathcal{S}\times\mathcal{A}$ there may be many possible successor states $s_{t+1}\in\mathcal{S}$ for which $\mathcal{T}(s_{t+1}|s_t,a_t) > 0$. Consider what happens if we act under the MCE policy $\pi_{\theta,t}(a_t|s_t) = \exp(Q_{\theta,t}(s_t,a_t) - V_{\theta,t}(s_t))$ for some θ, but restrict ourselves to the case of deterministic dynamics, where $\mathcal{T}(s_{t+1}|s_t,a_t) = 1$ for one successor state s_{t+1} and zero for all others. Here the discounted probability of some *feasible* trajectory τ—in which the state transitions agree with the dynamics—is

$$\begin{aligned}
p_\theta(\tau) &= D(s_0)\prod_{t=0}^{T-1}\gamma^t \mathcal{T}(s_{t+1}\mid s_t,a_t)\pi_t(a_t\mid s_t) && (49)\\
&= D(s_0)\prod_{t=0}^{T-1}\gamma^t \pi_t(a_t\mid s_t) && (50)\\
&= D(s_0)\exp\left(\sum_{t=0}^{T-1}\gamma^t\left(Q_{\theta,t}(s_t,a_t) - V_{\theta,t}(s_t)\right)\right) && (51)\\
&= D(s_0)\exp\Bigg(\gamma^{T-1}\left(r_\theta(s_{T-1},a_{T-1}) - V_{\theta,T-1}(s_{T-1})\right) && (52)\\
&\qquad + \sum_{t=0}^{T-2}\gamma^t\left(r_\theta(s_t,a_t) + \gamma \mathop{\mathbb{E}}_{S_{t+1}\sim\mathcal{T}}[V_{\theta,t+1}(S_{t+1})|s_t,a_t] - V_{\theta,t}(s_t)\right)\Bigg)\\
&= D(s_0)\exp\Bigg(\gamma^{T-1}\left(r_\theta(s_{T-1},a_{T-1}) - V_{\theta,T-1}(s_{T-1})\right) && (53)\\
&\qquad + \sum_{t=0}^{T-2}\gamma^t\left(r_\theta(s_t,a_t) + \gamma V_{\theta,t+1}(s_{t+1}) - V_{\theta,t}(s_t)\right)\Bigg)\\
&= \frac{D(s_0)}{\exp V_{\theta,0}(s_0)}\exp\left(\sum_{t=0}^{T-1}\gamma^t r_\theta(s_t,a_t)\right), && (54)
\end{aligned}$$

where we have made use of deterministic dynamics to drop the dynamics factors in Eq. (50), and to collapse the expectation over successor states that appeared in Eq. (52). In the literature, it is common to assume that s_0 is the same for all trajectories (i.e. that the initial state distribution is deterministic) and use the shorthand $Z_\theta = \exp V_{\theta,0}(s_0)$ for the normalising constant, in which

case the discounted probability of a feasible trajectory is

$$p_\theta(\tau) = \frac{1}{Z_\theta} \exp\left(\sum_{t=0}^{T-1} \gamma^t r_\theta(s_t, a_t)\right) . \tag{55}$$

If we perform IRL under the assumption that our trajectory distribution follows the form of Eq. (55), then we recover the *Maximum Entropy* IRL algorithm (abbreviated *MaxEnt* or *ME*) that was first proposed by Ziebart [26]. As we will see in Section 4, this simplified algorithm lends itself well to approximation in environments with unknown (but deterministic) dynamics [5, 4, 6]. Given the relative simplicity of the ME IRL density in Eq. (55), one might be tempted to use ME IRL in environment with stochastic dynamics by simply re-introducing the dropped dynamics factors into the ME IRL density in Eq. (55). Specifically, one could imagine working with a distribution of the form

$$\hat{p}_\theta(\tau) = D(s_0) \prod_{t=0}^{T-1} \mathcal{T}(s_{t+1} \mid s_t, a_t) \exp\left(\gamma^t r_\theta(s_t, a_t)\right) , \tag{56}$$

where the $D(s_0)$ and $\mathcal{T}(s_{t+1}|s_t, a_t)$ factors do not drop out for feasible trajectories because some may be nonzero but less than one.

Unfortunately, "generalising" ME IRL in this way may not produce appropriate behaviour in environments with truly non-deterministic dynamics. The intuitive reason for this is that the ME IRL density in Eq. (55) can assign arbitrarily high likelihood to any feasible trajectory (subject to normalisation constraints), so long as the return associated with that trajectory is high enough. Indeed, by moving the dynamics factors into the exp, we can see that the density takes the form

$$\hat{p}_\theta(\tau) = \exp\left(R_\theta(\tau) + \log D(s_0) + \sum_{t=0}^{T-1} \log \mathcal{T}(s_{t+1}|s_t, a_t)\right) , \tag{57}$$

where $R_\theta(\tau) = \sum_{t=0}^{T-1} \gamma^t r_\theta(s_t, a_t)$. This form suggests that the agent can simply pay a "log cost" in order to choose outcomes that would give it better return—a trajectory τ with extremely high return $R_\theta(\tau)$ relative to other trajectories may have high likelihood even the associated transition probabilities are low. In reality, if a non-deterministic environment has a state transition (s, a, s') with very low (but non-zero) probability ϵ, then no trajectory containing that transition can have likelihood more than ϵ under any physically realisable policy, even if the trajectory has extremely high return.

This mismatch between real MDP dynamics and the ME IRL model can manifest as risk-taking behaviour in the presence of stochastic transitions. For example, in Figure 1, an agent can obtain $s_0 \to s_1 \to s_2$ with 100% probability, or $s_0 \to s_2$ and $s_0 \to s_3$ with 50% probability each. The ME IRL transition model would wrongly believe the agent could obtain $s_0 \to s_2$ with arbitrarily high probability by paying a small cost that depends only on dynamics.

4 Dynamics-free approximations to ME IRL

Real-world environments are often too complex to be described in the tabular form required by MCE IRL. To perform IRL in these settings, we can use "dynamics-free" approximations to MCE IRL which do not require a known world model. In this section we describe three such

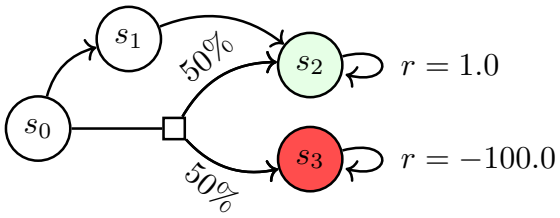

Figure 1: `RiskyPath`: The agent can either take a long but sure path to the goal ($s_0 \to s_1 \to s_2$), or attempt to take a shortcut ($s_0 \to s_2$), with the risk of receiving a low reward ($s_0 \to s_3$). This example is simplified from Ziebart's thesis [25, Figure 6.4].

approximations, all of which assume the environment is deterministic (simplifying MCE to ME IRL) and infinite-horizon (simplifying to a stationary policy).

We will start by describing the assumptions made by these algorithms in more detail, and adapting our previous definitions to the non-tabular setting. Next we consider Guided Cost Learning (GCL), which directly approximates the gradient of the MaxEnt objective using importance-weighted samples [5]. Later we discuss GAN-GCL, which performs GCL-like reward function updates by training a specially structured Generative Adversarial Network (GAN) [9] to distinguish between demonstrated trajectories and imitation trajectories [4]. Finally, we will describe Adversarial IRL (AIRL), another GAN-based approximation to MaxEnt IRL [6], that distinguishes between state–action pairs instead of distinguishing between trajectories. Unlike GCL and GAN-GCL, AIRL has been shown to be successful in standard RL benchmark environments like MuJoCo-based control tasks [6] and some Atari games [22].

4.1 Notation and Assumptions

All the dynamics-free algorithms in this section assume the environment is deterministic. GCL and GAN-GCL are based on ME IRL, which section 3.3 showed is only well-founded for deterministic dynamics. AIRL can be derived using MCE IRL, but many of the key results only hold for deterministic environments.

Additionally, the algorithms are designed for infinite-horizon, discounted MDPs. Counterintuitively, an infinite horizon can simplify the problem. Rather than having a non-stationary policy π_t derived from time-dependent $V^{\text{soft}}_{\theta,t}$ and $Q^{\text{soft}}_{\theta,t}$, we can instead search for a single fixed point to the soft VI equations of Eq. (32) yielding a stationary policy π.

Of course, in practice demonstration trajectories $\tau \sim \mathcal{D}$ must be finite. However, conceptually we can view the trajectories as being infinite, padded with some absorbing terminal state. This model is reasonable if, for example, the demonstrator always completes the task in finite time, after which the demonstration ends.

Additionally, we generalise our definitions Q^{soft}, V^{soft}, A^{soft} (in Eq. (32)) from a linear reward

function $\theta^T \phi(s,a)$ to an arbitrary reward function $r(s,a,s')$:

$$
\begin{aligned}
Q_r^{\text{soft}}(s,a) &\triangleq \sum_{s' \in S} \mathcal{T}(s,a,s') \left(r(s,a,s') + \gamma V_r^{\text{soft}}(s') \right) , \\
V_r^{\text{soft}}(s) &\triangleq \log \left(\sum_{a \in \mathcal{A}} \exp \left(Q_r^{\text{soft}}(s,a) \right) \right) , \\
A_r^{\text{soft}}(s,a) &\triangleq Q_r^{\text{soft}}(s,a) - V_r^{\text{soft}}(s) .
\end{aligned}
$$

4.2 Guided cost learning: Approximation via importance sampling

Consider the gradient of the log discounted likelihood of a demonstration distribution $\mathcal{D}$ under the ME IRL model, which was previously shown to be

$$
\nabla_\theta \mathcal{L}(\mathcal{D}; \theta) = \mathop{\mathbb{E}}_{\tau \sim \mathcal{D}} \left[\sum_{t=0}^{T-1} \gamma^t \nabla_\theta r_\theta(S_t, A_t) \right] - \mathop{\mathbb{E}}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T-1} \gamma^t \nabla_\theta r_\theta(S_t, A_t) \right] . \tag{48}
$$

Sample-based approximation of the first term is straightforward: we can just sample a subset of trajectories $\{\tau_1, \tau_2, \ldots, \tau_N\}$ from a dataset of observations, then approximate the term with a sample mean

$$
\mathop{\mathbb{E}}_{\tau \sim \mathcal{D}} \left[\sum_{t=0}^{T-1} \gamma^t \nabla_\theta r_\theta(S_t, A_t) \right] \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T-1} \gamma^t \nabla_\theta r_\theta(s_{t,i}, a_{t,i}) . \tag{58}
$$

Approximating the second term without access to dynamics is more complex. One obvious option would be to obtain the true maximum entropy policy $\pi_\theta^*(a|s)$ via (model-free) maximum entropy RL [10], then approximate the term with samples from the maximum entropy policy. However, this would require us to run RL to convergence each time we wanted to update the reward function, which is bound to be both sample- and compute-intensive; we would like a method of evaluating this term without need for π_θ^*.

Guided Cost Learning (GCL) [5] evaluates the second term in the ME IRL gradient using importance sampling. The proposal distribution $\mu(\tau)$ has two components: a learnt sampling distribution $q(\tau)$ that is optimised to roughly cover the highest-density regions of p_θ from Eq. (55), and an approximation $\tilde{p}(\tau)$ to the true demonstrator distribution $p(\tau)$.

The learnt sampling distribution $q(\tau)$ is induced by rollouts of a policy $\pi_q(a|s)$ in the environment. Initially, $\pi_q(a|s)$ will be random. Periodically, $\pi_q(a|s)$ is improved using an RL algorithm to encourage it to maximise the entropy-augmented return

$$
R_q(\tau) = \sum_{t=0}^{T-1} \gamma^t \left[r_\theta(s_t, a_t) + H(\pi_q(\cdot|s_t)) \right] . \tag{59}
$$

If run to optimality, this maximum-entropy RL objective would make $\pi_q(a|s)$ equal to $\pi_\theta^*(a|s)$. However, since GCL is only using π_q to propose samples for importance-sampling, it is acceptable to perform only a few steps of reinforcement learning between updates of the estimated reward parameters θ, so that $\pi_q(a|s)$ loosely approximates the maximum entropy trajectory distribution for $r_\theta(s,a)$.

Now consider the approximation $\tilde{p}(\tau)$ to the true demonstrator density $p(\tau)$. $p(\tau)$ is a product of state transition probabilities and action selection probabilities. However, since we assume

the environment is deterministic, we only have to estimate the action selection probabilities. This can easily be achieved using, e.g., behavioural cloning to obtain a state-conditioned action distribution $\tilde{\pi}_p(a|s)$.

If we assume that $\tilde{p}(\tau) \approx p(\tau)$ (i.e. that our approximation is accurate), then we can include demonstration samples in the set of samples that we use for importance sampling; this is an approximation to proper importance sampling with the true distribution $p(\tau)$. The original GCL paper reported that $p_\theta(\tau)$ tends to reach the point of roughly approximating the demonstrator density $p(\tau)$ much faster than $q(\tau)$ reaches the point of roughly approximating the maxent distribution $p_\theta(\tau)$. Thus it may be that after a few iterations of training, $p_\theta(\tau)$ is still close to zero for most trajectories produced by $q(\tau)$ (i.e. RL has not converged for the reward r_θ), and $\tilde{p}(\tau)$ is likewise close to zero for most trajectories produced by $q(\tau)$ (i.e. $q(\tau)$ is also an ineffective imitator). This would drive the importance weights **TODO** to infinity.

Once we put our two distributions together, we obtain a final sampling distribution

$$\mu(\tau) = \frac{1}{2}q(\tau) + \frac{1}{2}\tilde{p}(\tau) = \frac{1}{2}\prod_{t=0}^{T-1}\pi_q(a_t|s_t) + \frac{1}{2}\prod_{t=0}^{T-1}\tilde{\pi}_p(a_t|s_t)\,. \tag{60}$$

To approximate the second term of the original log likelihood gradient, we can draw a set of samples from μ—by repeatedly taking a rollout sample from $\pi_q(a|s)$, or sampling a demonstration from $\mathcal{D}$, each with 50% probability—and then use the importance-sampled estimate

$$\mathop{\mathbb{E}}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T-1}\gamma^t\nabla_\theta r_\theta(s_t,a_t)\right] \approx \mathop{\mathbb{E}}_{\tau\sim\mu}\left[\frac{1}{\mu(\tau)}\sum_{t=0}^{T-1}\gamma^t\nabla_\theta r_\theta(s_t,a_t)\right]\,. \tag{61}$$

In addition to importance-sampling, the original GCL paper proposes a number of heuristics to stabilise learning in the robotics tasks GCL was evaluated on. For example, it was helpful to include regularisers that encouraged temporally smooth rewards: i.e. the reward is similar for adjacent states in a trajectory. Moreover, GCL assumed rewards increase monotonically over the course of a demonstration trajectory.

Furthermore, GCL was evaluated with reward functions of the form $r_\theta(s,a) = -\|f_\theta(s)\|^2 - w_u\|a\|^2$, where $f_\theta(s,a)$ is a learnt neural network with no final-layer activation function, and w_u is a fixed control weight. This closely matches the form of reward function that humans typically invent for robotic control tasks. We conjecture this form may have provided a very useful inductive bias for the model although, since GCL has not been evaluated with other reward architectures, this is only speculation on our part.

For one of the evaluation tasks, it was also found that the sampling policy $\pi_q(a|s)$ was substantially better at imitating the demonstrations than a policy optimised from scratch on the learnt reward. This suggests that GCL does not always infer reward functions that robustly explain the behaviour of the demonstrator in some environments. It should perhaps be viewed as performing imitation learning rather than inverse reinforcement learning in such settings.

4.3 An interlude on generative adversarial networks

In the next two sub-sections, we will see algorithms that re-cast MaxEnt IRL as a way of learning a specially-structured kind of Generative Adversarial Network (GAN). This sub-section will briefly cover the basic principles behind GANs, which will be useful for understanding the IRL algorithms that follow. Readers unfamiliar with GANs may wish to consult the original GAN paper [9].

GANs include a generator function $x = G(z)$ that maps from random noise z to generated samples x of data. Given some fixed noise distribution $p_n(z)$ (e.g. $\mathcal{N}(0, I)$), we define $p_g(x)$ to represent the distribution over x induced by the generator:

$$p_g(x) = \mathop{\mathbb{E}}_{z \sim p_n} \left[\mathbb{I}[x = G(z)] \right] . \tag{62}$$

The overarching objective of GAN training is for $p_g = p_{\text{data}}$, the true data distribution.

In GAN training, we alternate between training a *discriminator* $D(x) \in (0, 1)$ to distinguish between p_g and p_{data}, and training $G(z)$ to produce samples that appear "real" to $D(x)$. Specifically, $D(x)$ is treated as a classifier of whether x is a sample from p_{data} (high) or p_g (low), and must minimise the cross-entropy loss

$$L_D = - \mathop{\mathbb{E}}_{z \sim p_z} \left[\log(1 - D(G(z))) \right] - \mathop{\mathbb{E}}_{x \sim p_{\text{data}}} \left[\log D(x) \right] . \tag{63}$$

In tandem, the generator $G(z)$ is trained to minimise

$$L_G = - \mathop{\mathbb{E}}_{z \sim p_n} \left[\log D(G(z)) \right] , \tag{64}$$

which encourages its outputs to appear more realistic to the discriminator. A generator cost based on $-\log D(G(z))$ rather than $\log(1 - D(G(z)))$ might seem like an odd choice, given that the latter appears in the discriminator objective Eq. (63). However, Goodfellow et al. observe that $D(G(z))$ is very low at the beginning of training, when the generator is poor, and so $-\log D(G(z))$ provides larger gradients than $\log(1 - D(G(z)))$. Importantly, optimising either objective over an unrestricted class of generators will lead to the same fixed points for the "game" played between G and D. Specifically, for any given generator distribution p_g, the Bayes-optimal discriminator will be [9, proposition 1]:

$$D^*_G(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)} . \tag{65}$$

Hence, the best worst-case loss attainable by the generator occurs when $p_g = p_{\text{data}}$, at which point we must have $D^*_G(x) = 0.5$ everywhere.

4.4 GAN-GCL: A trajectory-centric, GAN-based approximation

Superficially, the process of learning a reward function and imitation policy in MCE IRL resembles the learning of a discriminator and generator in a GAN. In this analogy, the discriminator ($\approx$ reward function) is trained to distinguish between real and generated samples ($\approx$ put high reward on only the demonstrations), while the generator is periodically updated to produce samples that are more likely to "fool" the discriminator ($\approx$ obtain high reward). Although this analogy appears simplistic, Finn et al. [4] have shown that a form of GAN training with a specially-structured discriminator is in fact mathematically equivalent to reward inference.

Specifically, we train a GAN in which the generator distribution is given by the importance sampling policy π_q from GCL; that is, $p_g(\tau) = q(\tau)$. Unlike the original GAN, we can directly evaluate the density of the generator distribution p_g: it is simply

$$p_g(\tau) = q(\tau) = \prod_{t=0}^{T} \pi_q(a_t | s_t) . \tag{66}$$

where we assume deterministic initial state and transition dynamics, and that τ is feasible.

The objective of the discriminator is still to distinguish between (trajectory) samples from the generator, and samples from the real (demonstration) dataset. If we substitute our generator distribution into Eq. (65), we find that the optimal discriminator for any given π_q must be

$$D_\theta(\tau) = \frac{p_{\text{data}}(\tau)}{p_{\text{data}}(\tau) + q(\tau)} \,. \tag{67}$$

Thus, an appropriate family of function approximators is given by

$$D_\theta(\tau) = \frac{\frac{1}{Z}\exp R_\theta(\tau)}{\frac{1}{Z}\exp R_\theta(\tau) + q(\tau)} = \frac{\frac{1}{Z}\exp R_\theta(\tau)}{2\tilde{\mu}(\tau)} \,, \tag{68}$$

where $\tilde{\mu}(\tau) = \frac{1}{2Z}\exp R_\theta(\tau) + \frac{1}{2}q(\tau)$. If we recover an R_θ equivalent to the demonstrator's true reward, then this will match the optimal discriminator in Eq. (67). Otherwise, it can be viewed as an approximation of the optimal discriminator whenever $p_\theta \approx p$. Further, if we assume that $p_\theta \approx p$, then we are justified in using the density of $\tilde{\mu}$ as an approximation for the density of the original mixture distribution $\mu(\tau) = \frac{1}{2}p(\tau) + \frac{1}{2}q(\tau)$ from GCL. We will now show that applying a slightly modified variant of GAN training to this combination of generator and discriminator yields an algorithm equivalent to GCL.

Generator objective First, we'll show that the generator (i.e. imitation policy) optimises the correct objective to recover the soft-optimal policy with respect to r_θ at convergence. Instead of using the original GAN loss of $\mathbb{E}_{\tau\sim p_g}[-\log D(\tau)]$, we'll use a sum of GAN losses of the form

$$L_G = \mathop{\mathbb{E}}_{\tau\sim p_g}\left[\log(1 - D(\tau)) - \log D(\tau)\right] \,. \tag{69}$$

As noted in the previous section, this combined objective is no less reasonable than the original GAN objective, since it will still lead to the same fixed points of the min–max game that gives rise the generator and discriminator. We can also show that it expands out to the entropy-regularised RL objective:

$$\begin{aligned} L_G &= \mathop{\mathbb{E}}_{\tau\sim q}\left[\log \frac{q(\tau)}{2\tilde{\mu}(\tau)} - \log \frac{\frac{1}{Z}\exp R_\theta(\tau)}{2\tilde{\mu}(\tau)}\right] && (70)\\ &= \mathop{\mathbb{E}}_{\tau\sim q}\left[-R_\theta(\tau) + \log q(\tau)\right] + \log Z && (71)\\ &= -\mathop{\mathbb{E}}_{\tau\sim q}\left[R_\theta(\tau)\right] + H(q) + \log Z \,. && (72) \end{aligned}$$

Note that $\log Z$ does not depend on q, and thus does not affect optimisation of π_q. This is thus equivalent to the objective in Eq. (59) that was used to train the sampling policy π_q in the original GCL algorithm.

Discriminator objective Now we'll turn to the discriminator objective, which is given by

$$\begin{aligned} L_D &= -\mathop{\mathbb{E}}_{\tau\sim q}\left[\log(1 - D(\tau))\right] - \mathop{\mathbb{E}}_{\tau\sim p_{\text{data}}}\left[\log D(\tau)\right] && (73)\\ &= -\mathop{\mathbb{E}}_{\tau\sim q}\left[\log \frac{q(\tau)}{2\tilde{\mu}(\tau)}\right] - \mathop{\mathbb{E}}_{\tau\sim p_{\text{data}}}\left[\log \frac{\frac{1}{Z}\exp R_\theta(\tau)}{2\tilde{\mu}(\tau)}\right] && (74)\\ &= -\mathop{\mathbb{E}}_{\tau\sim q}\left[\log q(\tau)\right] - \mathop{\mathbb{E}}_{\tau\sim p_{\text{data}}}\left[R_\theta(\tau)\right] + \log Z + 2\mathop{\mathbb{E}}_{\tau\sim \mu}\left[\log \tilde{\mu}(\tau)\right] + c \,. && (75) \end{aligned}$$

Consider the gradient of this loss with respect to Z, where we simply treat Z as a learnt parameter of our model:

$$\frac{\partial L_D}{\partial Z} = \frac{\partial}{\partial Z}\left(\log Z + 2 \mathop{\mathbb{E}}_{\tau\sim\mu}\left[\log \tilde{\mu}(\tau)\right]\right) \tag{76}$$

$$= \frac{1}{Z} + 2 \mathop{\mathbb{E}}_{\tau\sim\mu}\left[\frac{\partial}{\partial Z}\log\left(\frac{1}{2Z}\exp R_\theta(\tau) + \frac{1}{2}q(\tau)\right)\right] \tag{77}$$

$$= \frac{1}{Z} - \frac{1}{Z^2}\mathop{\mathbb{E}}_{\tau\sim\mu}\left[\frac{1}{\tilde{\mu}(\tau)}\exp R_\theta(\tau)\right]. \tag{78}$$

If we equate this derivative to zero and solve for Z, we find that at optimality we must have

$$Z^\star = \mathop{\mathbb{E}}_{\tau\sim\mu}\left[\frac{1}{\tilde{\mu}(\tau)}\exp R_\theta(\tau)\right]. \tag{79}$$

If $\tilde{\mu} \approx \mu$ (which holds when $p_\theta \approx p$), then this is an importance-sampled estimate of the partition function $Z = \sum_\tau \exp R_\theta(\tau)$. In other words, when p_θ is close to convergence, optimising Z to a global minimum is roughly equivalent to taking an importance-sampled estimate of the partition function.

Now consider the gradient of L_D with respect to the reward parameters θ. If we have already optimised Z to convergence after the last update to θ (e.g. by simply setting it to the above importance-sampled estimate), then the gradient of L_D with respect to θ is

$$\nabla_\theta L_D = \nabla_\theta\left(-\mathop{\mathbb{E}}_{\tau\sim p_{\text{data}}}\left[R_\theta(\tau)\right] + 2\mathop{\mathbb{E}}_{\tau\sim\mu}\left[\log\tilde{\mu}(\tau)\right]\right) \tag{80}$$

$$= -\mathop{\mathbb{E}}_{\tau\sim p_{\text{data}}}\left[\nabla_\theta R_\theta(\tau)\right] + 2\mathop{\mathbb{E}}_{\tau\sim\mu}\left[\frac{1}{\tilde{\mu}(\tau)}\nabla_\theta\tilde{\mu}(\tau)\right] \tag{81}$$

$$= -\mathop{\mathbb{E}}_{\tau\sim p_{\text{data}}}\left[\nabla_\theta R_\theta(\tau)\right] + 2\mathop{\mathbb{E}}_{\tau\sim\mu}\left[\frac{1}{\tilde{\mu}(\tau)}\nabla_\theta\left(\frac{1}{2Z}\exp R_\theta(\tau) + \frac{1}{2}q(\tau)\right)\right] \tag{82}$$

$$= \mathop{\mathbb{E}}_{\tau\sim\mu}\left[\frac{p_\theta(\tau)}{\tilde{\mu}(\tau)}\nabla_\theta R_\theta(\tau)\right] - \mathop{\mathbb{E}}_{\tau\sim p_{\text{data}}}\left[\nabla_\theta R_\theta(\tau)\right]. \tag{83}$$

Again, when $\tilde{\mu} \approx \mu$, this is approximately equal to the gradient of the MCE IRL log likelihood, using a similar importance-sampling strategy to the GCL gradient in Eq. (61).

4.5 Adversarial IRL: A state-centric, GAN-based approximation

In the previous section, we saw how GANs can be used to produce an IRL method similar to GCL. However, GAN-GCL obtains poor performance in practice [6, Table 2]. In particular, performing discrimination over entire *trajectories* τ can complicate training. Adversarial IRL (AIRL) [6] instead discriminates between states or state-action pairs. Concretely, it learns a (stationary) stochastic policy $\pi(a \mid s)$ and reward function $f_\theta(s, a)$.

The AIRL discriminator is defined by:

$$D_\theta(s, a) = \frac{\exp\left(f_\theta(s, a)\right)}{\exp\left(f_\theta(s, a)\right) + \pi(a \mid s)}. \tag{84}$$

The generator π is trained with the discriminator confusion:

$$\hat{r}(s, a) \triangleq \log\left(D_\theta(s, a)\right) - \log\left(1 - D_\theta(s, a)\right). \tag{85}$$

The discriminator is trained with cross-entropy loss, updating only the parameters θ corresponding to the reward function f_θ:

$$L(\theta) \triangleq - \mathop{\mathbb{E}}_{\tau\sim\pi}\left[\sum_{t=1}^{T} \log\left(1 - D_\theta(s_t, a_t)\right)\right] - \mathop{\mathbb{E}}_{\tau\sim\mathcal{D}}\left[\sum_{t=1}^{T} \log D_\theta(s_t, a_t)\right]. \tag{86}$$

4.5.1 Policy objective is entropy-regularised f_θ

We train the generator $\pi(a \mid s)$ using the reward:

$$\hat{r}(s, a) \triangleq \log\left(D_\theta(s, a)\right) - \log\left(1 - D_\theta(s, a)\right). \tag{85}$$

Similar to GAN-GCL, this is a sum of two widely used GAN objectives: the minimax objective $-\log\left(1 - D_\theta(s, a)\right)$ and the more widely used $\log\left(D_\theta(s, a)\right)$ objective. Combining the two objectives is reasonable since both objectives have the same fixed point. Moreover, the combined objective has a pleasing interpretation in terms of f_θ:

$$\hat{r}(s, a) = f_\theta(s, a) - \log \pi(a \mid s). \tag{87}$$

Summing over entire trajectories, we obtain the entropy-regularised policy objective:

$$\mathop{\mathbb{E}}_{\pi}\left[\sum_{t=0}^{T} \gamma^t \hat{r}(s_t, a_t)\right] = \mathop{\mathbb{E}}_{\pi}\left[\sum_{t=0}^{T} \gamma^t \left(f_\theta(s_t, a_t) - \log \pi(a_t \mid s_t)\right)\right]. \tag{88}$$

It is known that the policy π that maximises this objective is [25, 10]:

$$\pi(a \mid s) = \exp\left(Q^{\text{soft}}_{f_\theta}(s, a) - V^{\text{soft}}_{f_\theta}(s)\right) = \exp\left(A^{\text{soft}}_{f_\theta}(s, a)\right).$$

4.5.2 $f_\theta(s, a)$ recovers the optimal advantage

Recall that for a fixed generator G, the optimal discriminator D is

$$D^*_G(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)}, \tag{65}$$

where $p_{\text{data}}(x)$ is the true data distribution (in our case, expert trajectories), and $p_g(x)$ is the distribution induced by the generator.

Normally in GAN training, it is computationally efficient to *sample* from the generator distribution p_g but expensive to evaluate the *density* $p_g(x)$ of a given sample. Fortunately, in AIRL (like GAN-GCL), density evaluation is cheap since the generator $p_g(x)$ is defined by a stochastic policy $\pi(a \mid s)$, which explicitly defines a distribution.

Suppose the generator is always pitted against the optimal discriminator in Eq. (65). It is known that the generator achieves a global maximum of the minimum discriminator cross-entropy loss in eq. 86 if and only if $p_g(x) = p_{\text{data}}(x)$, which is attained in our case when the generator $\pi(a \mid s)$ is equal to the expert policy $\pi_E(a \mid s)$.

The optimal discriminator for an optimal generator will always output 0.5: i.e. $f^*_\theta(s, a) = \log \pi_E(a \mid s)$. Moreover, the optimal Maximum Causal Entropy policy π_E for a reward function r is $\pi_E(a \mid s) = \exp\left(A^{\text{soft}}_r(s, a)\right)$. So, if the GAN converges, then at optimality $f^*_\theta(s, a) = A^{\text{soft}}_r(s, a)$ and $\pi(a \mid s) = \pi_E(a \mid s)$.

4.5.3 Reward shaping in MCE

In this section, we will introduce a classical result of (hard) optimal policy equivalence under potential shaping due to Ng et al. [15], and then generalise it to the case of maximum entropy (soft) optimal policies.

Definition 4.1. *Let r and r' be two reward functions. We say r and r' induce the* same hard optimal policy *under transition dynamics $\mathcal{T}$ if, for all states s:*

$$\arg\max_a Q_{r,\mathcal{T}}(s,a) = \arg\max_a Q_{r',\mathcal{T}}(s,a)\,. \tag{89}$$

Theorem 4.1. *Let r and r' be two reward functions. r and r' induce the same hard optimal policy under all transition dynamics $\mathcal{T}$ if:*

$$r'(s,a,s') = \lambda\left(r(s,a,s') + \gamma\phi(s') - \phi(s)\right)\,, \tag{90}$$

for some $\lambda > 0$ and potential-shaping function $\phi : \mathcal{S} \to \mathbb{R}$.

Proof. See [15, 7]. □

Definition 4.2. *Let r and r' be two reward functions. We say they induce the* same soft optimal policy *under transition dynamics $\mathcal{T}$ if, for all states s and actions a:*

$$A^{\text{soft}}_{r,\mathcal{T}}(s,a) = A^{\text{soft}}_{r',\mathcal{T}}(s,a)\,. \tag{91}$$

Theorem 4.2. *Let r and r' be two reward functions. r and r' induce the same soft optimal policy under all transition dynamics $\mathcal{T}$ if $r'(s,a,s) = r(s,a,s') + \gamma\phi(s') - \phi(s)$ for some potential-shaping function $\phi : \mathcal{S} \to \mathbb{R}$.*

Proof. We have:

$$Q^{\text{soft}}_{r',\mathcal{T}}(s,a) = \mathop{\mathbb{E}}_{s'\sim\mathcal{T}}\left[r(s,a,s') + \gamma\phi(s') - \phi(s) + \gamma V^{\text{soft}}_{r',\mathcal{T}}(s')\middle|s,a\right]\,. \tag{92}$$

So:

$$Q^{\text{soft}}_{r',\mathcal{T}}(s,a) + \phi(s) = \mathop{\mathbb{E}}_{s'\sim\mathcal{T}}\Bigg[r(s,a,s') + \gamma\log\sum_{a'\in\mathcal{A}}\exp\left(Q^{\text{soft}}_{r',\mathcal{T}}(s',a') + \phi(s')\right)\Bigg|s,a\Bigg]\,. \tag{93}$$

Thus $Q^{\text{soft}}_{r',T}(s,a) + \phi(s)$ satisfies the soft Bellman backup for r, so:

$$Q^{\text{soft}}_{r,\mathcal{T}}(s,a) = Q^{\text{soft}}_{r',\mathcal{T}}(s,a) + \phi(s)\,. \tag{94}$$

It follows that the optimal advantage is invariant to shaping:

$$A^{\text{soft}}_{r',\mathcal{T}}(s,a) = Q^{\text{soft}}_{r',\mathcal{T}}(s,a) - V^{\text{soft}}_{r',\mathcal{T}}(s) \tag{95}$$

$$= Q^{\text{soft}}_{r',\mathcal{T}}(s,a) - \log \sum_{a\in\mathcal{A}} \exp\left(Q^{\text{soft}}_{r',\mathcal{T}}(s,a)\right) \tag{96}$$

$$= Q^{\text{soft}}_{r,\mathcal{T}}(s,a) + \phi(s) - \log \sum_{a\in\mathcal{A}} \exp\left(Q^{\text{soft}}_{r,\mathcal{T}}(s,a) + \phi(s)\right) \tag{97}$$

$$= Q^{\text{soft}}_{r,\mathcal{T}}(s,a) - \log \sum_{a\in\mathcal{A}} \exp\left(Q^{\text{soft}}_{r,\mathcal{T}}(s,a)\right) \tag{98}$$

$$= A^{\text{soft}}_{r,\mathcal{T}}(s,a)\,. \tag{99}$$

□

Remark 4.1. *Note that rescaling* r does *change the soft optimal advantage function (and the consequent soft-optimal policy). Let* $r'(s,a,s') = \lambda(s,a,s')r(s,a,s')$*. Consider a simple MDP with a single state* s *and two actions,* a *and* a'*, with* $r(s,a,s) = 0$ *and* $r(s,a',s) = 1$*. Since there is only a single state, the optimal advantage function reduces to the instantaneous reward:*

$$A^{\text{soft}}_{r,\mathcal{T}}(s,x) = r(s,x,s)\,. \tag{100}$$

Rescaling by $\lambda \neq 1$ *will therefore change the soft optimal advantage function (and the consequent soft-optimal policy).*

4.5.4 Discriminator objective

In this section, we show that minimising the loss of the discriminator corresponds to ME IRL in deterministic dynamics when f_θ is already an advantage for some reward function.

Theorem 4.3. *Suppose* f_θ *is an optimal advantage function for the given MDP for some reward function* r_θ*. Then minimising the cross-entropy loss of the discriminator is equivalent to maximising the log-likelihood of observations under the Maximum Entropy IRL model.*

Specifically, recall that the gradient of the log likelihood is

$$\nabla_\theta \mathcal{L}(\mathcal{D};\theta) = \mathop{\mathbb{E}}_{\tau\sim\mathcal{D}}\left[\sum_{t=0}^{T-1} \gamma^t \nabla_\theta r_\theta(S_t,A_t)\right] - \mathop{\mathbb{E}}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T-1} \gamma^t \nabla_\theta r_\theta(S_t,A_t)\right]. \tag{48}$$

We will show that the gradient of the discriminator objective is:

$$-\nabla_\theta L(\theta) = \frac{1}{2}\mathop{\mathbb{E}}_{\tau\sim\mathcal{D}}\left[\sum_{t=0}^{T} \nabla_\theta f_\theta(s_t,a_t)\right] - \frac{1}{2}\mathop{\mathbb{E}}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta f_\theta(s_t,a_t)\right]. \tag{101}$$

Proof. Note that $L(\theta)$ is the discriminator loss, and so we wish to maximise the discriminator objective $-L(\theta)$. This has the gradient:

$$-\nabla_\theta L(\theta) = \mathop{\mathbb{E}}_{\tau\sim\mathcal{D}}\left[\sum_{t=0}^{T} \nabla_\theta \log D_\theta(s_t,a_t,s_{t+1})\right] + \mathop{\mathbb{E}}_{\tau\sim\pi}\left[\sum_{t=0}^{T} \nabla_\theta \log\left(1 - D_\theta(s_t,a_t,s_{t+1})\right)\right]. \tag{102}$$

Observe that:

$$\log D_\theta(s_t,a_t,s_{t+1}) = f_\theta(s_t,a_t) - \log\left(\exp\left(f_\theta(s_t,a_t)\right) + \pi(a_t \mid s_t)\right). \tag{103}$$

Thus:

$$\nabla_\theta \log D_\theta(s_t, a_t) = \nabla_\theta f_\theta(s_t, a_t) - \frac{\exp\left(f_\theta(s_t, a_t)\right) \nabla_\theta f_\theta(s_t, a_t)}{\exp\left(f_\theta(s_t, a_t)\right) + \pi(a_t \mid s_t)} \,. \tag{104}$$

Similarly:

$$\log\left(1 - D_\theta(s_t, a_t)\right) = \pi(a_t \mid s_t) - \log\left(\exp\left(f_\theta(s_t, a_t)\right) + \pi(a_t \mid s_t)\right) \,. \tag{105}$$

So:

$$\nabla_\theta \log\left(1 - D_\theta(s_t, a_t)\right)) = -\frac{\exp\left(f_\theta(s_t, a_t)\right) \nabla_\theta f_\theta(s_t, a_t)}{\exp\left(f_\theta(s_t, a_t)\right) + \pi(a_t \mid s_t)} \,. \tag{106}$$

Recall we train the policy $\pi(a_t \mid s_t)$ to maximize eq. 88. The optimal maximum entropy policy for a given f_θ is $\pi^*_{f_\theta}(a_t \mid s_t) = \exp\left(A^{\text{soft}}_{f_\theta}(s_t, a_t)\right)$.

By assumption, f_θ is the advantage for some reward function r_θ, so:

$$\begin{aligned}
f_\theta(s, a) &= A^{\text{soft}}_{r_\theta}(s, a) && (107)\\
&= Q^{\text{soft}}_{r_\theta}(s, a) - V^{\text{soft}}_{r_\theta}(s) && (108)\\
&= \sum_{s' \in \mathcal{S}} \mathcal{T}(s, a, s') \left(r_\theta(s, a, s') + \gamma V^{\text{soft}}_r(s')\right) - V^{\text{soft}}_r(s) && (109)\\
&= r_\theta\left(s, a, \mathcal{T}(s, a)\right) + \gamma V^{\text{soft}}_r\left(\mathcal{T}(s, a)\right) - V^{\text{soft}}_r(s) \,, && (110)
\end{aligned}$$

where we write $s' = \mathcal{T}(s, a)$ for the deterministic next-state. Restricting ourselves only to feasible transitions (s, a, s'), we can alternatively write:

$$f_\theta(s, a, s') = r(s, a, s') + \gamma V^{\text{soft}}_r(s') - V^{\text{soft}}_r(s) \,. \tag{111}$$

That is, f_θ is r_θ shaped by potential function $V^{\text{soft}}_r(s)$. Applying theorem 4.2, it follows that:

$$A^{\text{soft}}_{f_\theta}(s, a) = A^{\text{soft}}_{r_\theta}(s, a) \tag{112}$$

but by assumption $f_\theta(s, a) = A^{\text{soft}}_{r_\theta}(s, a)$, so we have that $f_\theta(s, a)$ is idempotent under the advantage operator:

$$A^{\text{soft}}_{f_\theta}(s, a) = f_\theta(s, a) \,. \tag{113}$$

Thus the optimal policy is $\pi^*_{f_\theta}(a_t \mid s_t) = \exp\left(f_\theta(s_t, a_t)\right)$. Substituting this expression gives

$$\nabla_\theta \log D_\theta(s_t, a_t) = \frac{1}{2} \nabla_\theta f_\theta(s_t, a_t) \,, \tag{114}$$

and

$$\nabla_\theta \log\left(1 - D_\theta(s_t, a_t)\right)) = -\frac{1}{2} \nabla_\theta f_\theta(s_t, a_t) \,. \tag{115}$$

So:

$$\begin{aligned}
-\nabla_\theta L(\theta) &= \frac{1}{2} \sum_{t=0}^{T} \left(\mathop{\mathbb{E}}_{(s_t, a_t) \sim \mathcal{D}} \left[\nabla_\theta f_\theta(s_t, a_t)\right] - \mathop{\mathbb{E}}_{(s_t, a_t) \sim \pi_t} \left[\nabla_\theta f_\theta(s_t, a_t)\right] \right) && (116)\\
&= \frac{1}{2} \mathop{\mathbb{E}}_{\tau \sim \mathcal{D}} \left[\sum_{t=0}^{T} \nabla_\theta f_\theta(s_t, a_t) \right] - \frac{1}{2} \mathop{\mathbb{E}}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} \nabla_\theta f_\theta(s_t, a_t) \right] \,. && (117)
\end{aligned}$$

□

Remark 4.2. *Section 4.5.2 showed the globally optimal f_θ is the optimal soft advantage function. However, there is no guarantee that the f_θ is ever a soft advantage function during training. So this theorem does not demonstrate convergence, but does provide intuition for why AIRL often works well in practice.*

4.5.5 Disentangled rewards

In section 4.5.3, we saw that if a reward function r' is a potential shaped version of r then r' induces the same soft Q-values as r up to a state-dependent baseline. In the case that both reward functions are state-only, i.e. $r(s)$ and $r'(s)$, then potential shaping (when $\gamma < 1$) reduces to the special case of $r'(s) = r(s) + k$ for some constant k. Perhaps surprisingly, AIRL can determine state-only rewards up to a constant provided the transition dynamics $\mathcal{T}$ satisfies a strong requirement known as the *decomposability condition*.

Definition 4.3 (Decomposability Condition)**.** *We define two states u, v as being* 1-step linked *under a transition distribution $\mathcal{T}(s' \mid s, a)$ if there exists a state s and actions a, b such that $\mathcal{T}(u \mid s, a) > 0$ and $\mathcal{T}(v \mid s, b) > 0$.*

We define two states u, v as being $n+1$-step linked *if they are in the transitive closure of the n-step linked relation. That is, they are n-step linked or there is a chain of intermediate states $s_1, s_2, \ldots, s_n$ such that u is 1-step linked to s_1, s_i is 1-step linked to s_{i+1} and s_n is 1-step linked to v.*

We define two states u, v as being linked *if there is some $n \in \mathbb{N}$ for which they are n-step linked.*

A transition distribution $\mathcal{T}$ is decomposable *if all pairs of states in the MDP are linked.*

The decomposability condition can be counterintuitive, so we consider some examples before using the definition further.

A simple MDP that does **not** satisfy the condition is a two-state cyclic MDP, where it is only possible to transition from state A to B and vice-versa. There is no state that can reach both A and B, so they are not 1-step linked. They are therefore also not n-step linked for any n, since there are no possible intermediary states. However, the MDP would be decomposable if the dynamics were extended to allow self-transitions (from $A \to A$ and $B \to B$).

A similar pattern holds in gridworlds. Imagine a checkerboard pattern on the grid. If all actions move to an adjacent cell (left, right, up or down), then all the successors of white cells are black, and vice-versa. Consequently, cells are only ever 1-step linked to cells of the same colour. Taking the transitive closure, all cells of the same colour are linked together, but never to cells of a different colour. However, if you add a 'stay' action to the gridworld then all cells are linked.

We have not been able to determine either way whether the decomposability condition is satisfied in standard RL benchmarks, such as MuJoCo tasks or Atari games.

Theorem 4.4. *Let $\mathcal{T}$ be a* ***deterministic*** *dynamics model satisfying the decomposability condition, and let $\gamma > 0$. Let $r(s)$ and $r'(s)$ be two reward models producing the same MCE policy in $\mathcal{T}$. That is, for all states s and actions a:*

$$A^{\text{soft}}_{r',\mathcal{T}}(s, a) = A^{\text{soft}}_{r,\mathcal{T}}(s, a)\,. \tag{118}$$

Then $r'(s) = r(s) + k$, for some constant k.

Proof. We start by considering the general case of a stochastic dynamics model $\mathcal{T}$ and reward functions over (s, a, s') triples. We introduce the simplifying assumptions only when necessary, to highlight why we make these assumptions.

Substituting the definition for A^{soft} in Eq. (118):

$$Q^{\text{soft}}_{r',\mathcal{T}}(s,a) - V^{\text{soft}}_{r',\mathcal{T}}(s) = Q^{\text{soft}}_{r,\mathcal{T}}(s,a) - V^{\text{soft}}_{r,\mathcal{T}}(s)\,. \tag{119}$$

So:

$$Q^{\text{soft}}_{r',\mathcal{T}}(s,a) = Q^{\text{soft}}_{r,\mathcal{T}}(s,a) - f(s)\,, \tag{120}$$

where $f(s) = V^{\text{soft}}_{r',\mathcal{T}}(s) - V^{\text{soft}}_{r,\mathcal{T}}(s)$.

Now:

$$\begin{aligned} Q^{\text{soft}}_{r',\mathcal{T}}(s,a) &= Q^{\text{soft}}_{r,T}(s,a) - f(s) \\ &= \mathop{\mathbb{E}}_{s'\sim\mathcal{T}}\left[r(s,a,s') - f(s) + \gamma \log \sum_{a'} \exp\left(Q^{\text{soft}}_{r,T}(s',a')\right) \middle| s, a \right] \\ &= \mathop{\mathbb{E}}_{s'\sim\mathcal{T}}\Bigg[r(s,a,s') - f(s) + \gamma f(s') \\ &\qquad\qquad + \gamma \log \sum_{a'} \exp\left(Q^{\text{soft}}_{r,T}(s',a') - f(s')\right) \Bigg| s, a \Bigg] \\ &= \mathop{\mathbb{E}}_{s'\sim\mathcal{T}}\left[r(s,a,s') - f(s) + \gamma f(s') + \gamma \log \sum_{a'} \exp\left(Q^{\text{soft}}_{r',T}(s',a')\right) \middle| s, a \right]\,. \end{aligned} \tag{121}$$

Contrast this with the Bellman backup on r':

$$Q^{\text{soft}}_{r',\mathcal{T}}(s,a) = \mathop{\mathbb{E}}_{s'\sim\mathcal{T}}\left[r'(s,a,s') + \gamma \log \sum_{a'} \exp\left(Q^{\text{soft}}_{r',T}(s',a')\right) \middle| s, a \right]\,. \tag{122}$$

So, equating these two expressions for $Q^{\text{soft}}_{r',\mathcal{T}}(s,a)$:

$$\mathop{\mathbb{E}}_{s'\sim\mathcal{T}}[r'(s,a,s')|s,a] = \mathop{\mathbb{E}}_{s'\sim\mathcal{T}}[r(s,a,s') - f(s) + \gamma f(s')|s,a]\,.$$

In the special case of deterministic dynamics, then if $s' = \mathcal{T}(s,a)$, we have:

$$r'(s,a,s') = r(s,a,s') - f(s) + \gamma f(s')\,. \tag{123}$$

This looks like r' being a potential-shaped version of r, but note this equality may not hold for transitions that are not feasible under this dynamics model $\mathcal{T}$.

If we now constrain r' and r to be state-only, we get:

$$r'(s) - r(s) + f(s) = \gamma f(s')\,. \tag{124}$$

In particular, since $\gamma \neq 0$ this implies that for a given state s, all possible successor states s' (reached via different actions) must have the same value $f(s')$. In other words, all 1-step linked states have the same f value. Moreover, 2-step linked states are simply the transitive closure

of the 1-step linked states, so since equality is transitive must also have the same f values. By induction, all linked states must have the same f value. Since by assumption $\mathcal{T}$ is decomposable, then $f(s) = c$ for some constant, and so:

$$r'(s) = r(s) + (\gamma - 1)c = r(s) + k\,. \tag{125}$$

□

4.5.6 Recovering rewards

Lemma 4.1. *Suppose the transition distribution $\mathcal{T}$ is decomposable. Let $a(s), b(s), c(s), d(s)$ be functions of the state. Suppose that for all states s, actions a and successor states s' for which $\mathcal{T}(s' \mid s, a) > 0$, then*

$$a(s) - c(s) = b(s') - d(s')\,. \tag{126}$$

Then for all s,

$$a(s) = c(s) + k_1 \tag{127}$$

$$b(s) = d(s) + k_2\,, \tag{128}$$

where $k_1, k_2 \in \mathbb{R}$ are constants.

Proof. Let $f(s) = a(s) - c(s)$ and $g(s') = b(s') - d(s')$.

Base case: Since $f(s) = g(s')$ for any successor state s' of s, it must be that $g(s')$ takes on the same value for all successor states s' for s. This shows that all 1-step linked states have the same $g(s')$.

Inductive case: Moreover, this extends by transitivity. Suppose that all n-step linked states s' have the same $g(s')$. Let u and v be $n+1$-step linked via intermediate state s. So u and v are both n-step linked to s. But then $g(u) = g(s) = g(v)$. So in fact all $n+1$ step linked states s' have the same $g(s')$.

By induction, it follows that all linked states s' have the same $g(s')$. In a decomposable MDP, this implies $g(s')$ is constant. A similar argument holds for $f(s)$. □

Theorem 4.5. *Suppose the reward network is parameterized by*

$$f(s, a, s') = g_\theta(s) + \gamma h_\phi(s') - h_\phi(s)\,. \tag{129}$$

Suppose the ground-truth reward is state-only, $r(s)$. Suppose moreover that the MDP has deterministic dynamics $\mathcal{T}$ satisfying the decomposability condition, and that $\gamma > 0$. Then if f attains the global minimum, $f(s, s') = f^*(s, s')$, we have:

$$g_{\theta^*}(s') = r(s') + k_1, \quad h_{\phi^*}(s') = V_r^{\text{soft}}(s') + k_2\,, \tag{130}$$

where $k_1, k_2 \in \mathbb{R}$ are constants and s' is the successor of some state s.

Proof. We know the global minimum $f^*(s, a, s') = A^{\text{soft}}(s, a)$, from section 4.5.2. Now:

$$A^{\text{soft}}(s, a) = Q_r^{\text{soft}}(s, a) - V_r^{\text{soft}}(s) \tag{131}$$

$$= r(s) + \gamma V_r^{\text{soft}}(s') - V_r^{\text{soft}}(s)\,. \tag{132}$$

So for all states s, actions a and resulting deterministic successor states $s' = \mathcal{T}(s, a)$ we have:

$$g_{\theta^*}(s) + \gamma h_{\phi^*}(s') - h_{\phi^*}(s) = r(s) + \gamma V_r^{\text{soft}}(s') - V_r^{\text{soft}}(s)\,. \tag{133}$$

Now applying lemma 4.1 with $a(s) = g_{\theta^*}(s) - h_{\phi^*}(s)$, $c(s) = r(s) - V_r^{\text{soft}}(s)$, $b(s') = \gamma h_{\phi^*}(s')$ and $d(s') = \gamma V_r^{\text{soft}}(s')$ gives:

$$g_{\theta^*}(s) - h_{\phi^*}(s) = a(s) = c(s) + \alpha = r(s) - V_r^{\text{soft}}(s) + \alpha \tag{134}$$

$$\gamma h_{\phi^*}(s') = b(s') = d(s') + \beta = \gamma V_r^{\text{soft}}(s') + \beta\,. \tag{135}$$

where $\alpha, \beta \in \mathbb{R}$ are constants. Rearranging and using $\gamma \neq 0$ we have:

$$g_{\theta^*}(s) = r(s) + h_{\phi^*}(s) - V_r^{\text{soft}}(s) + \alpha \tag{136}$$

$$h_{\phi^*}(s') = V_r^{\text{soft}}(s') + \frac{\beta}{\gamma} = V_r^{\text{soft}}(s') + k_2\,. \tag{137}$$

Then, provided s is the successor of some state, we can apply Eq. (137) obtaining:

$$g_{\theta^*}(s) = r(s) + (\beta + k_2) = r(s) + k_1\,, \tag{138}$$

as required. □

Remark 4.3. *Note this theorem makes several strong assumptions. In particular, it requires that f attains the global minimum, but – at the time of writing – there is no proof that AIRL converges. Additionally, many environments have stochastic dynamics or are not 1-step linked.*

Note that in stochastic dynamics there may not be any function $f(s, s')$ that is always equal to $A^{\text{soft}}(s, a)$. This is because there may exist a, a' such that $A^{\text{soft}}(s, a) \neq A^{\text{soft}}(s, a')$ due to differing successor state distributions $\mathcal{T}(\cdot \mid s, a)$ and $\mathcal{T}(\cdot \mid s, a')$, but both successor state distributions may have non-zero probability of successor state s'.

References

[1] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *ICML*, 2004.

[2] Kavosh Asadi and Michael L. Littman. An alternative softmax operator for reinforcement learning. In *ICML*, 2017.

[3] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[4] Chelsea Finn, Paul Christiano, Pieter Abbeel, and Sergey Levine. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. arXiv: 1611.03852v3 [cs.LG], 2016.

[5] Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *ICML*, 2016.

[6] Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. In *ICLR*, 2018.

[7] Adam Gleave, Michael Dennis, Shane Legg, Stuart Russell, and Jan Leike. Quantifying differences in reward functions. arXiv: 2006.13900v1 [cs.LG], 2020.

[8] Adam Gleave, Pedro Freire, Steven Wang, and Sam Toyer. seals: Suite of environments for algorithms that learn specifications. `https://github.com/HumanCompatibleAI/seals`, 2020.

[9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014.

[10] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *ICML*, 2017.

[11] Hong Jun Jeon, Smitha Milli, and Anca D. Dragan. Reward-rational (implicit) choice: A unifying formalism for reward learning, 2020.

[12] Andrej Karpathy. Keynote talk. CVPR Workshop on Scalability in Autonomous Driving, 2020.

[13] Ilya Kostrikov, Kumar Krishna Agrawal, Debidatta Dwibedi, Sergey Levine, and Jonathan Tompson. Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning. In *ICLR*, 2019.

[14] Andrew Y Ng and Stuart J Russell. Algorithms for inverse reinforcement learning. In *ICML*, 2000.

[15] Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, 1999.

[16] Deepak Ramachandran and Eyal Amir. Bayesian inverse reinforcement learning. In *IJCAI*, 2007.

[17] Dorsa Sadigh, Anca D. Dragan, S. Shankar Sastry, and Sanjit A. Seshia. Active preference-based learning of reward functions. In *RSS*, July 2017.

[18] Rohin Shah, Dmitrii Krasheninnikov, Jordan Alexander, Pieter Abbeel, and Anca Dragan. Preferences implicit in the state of the world. In *ICLR*, 2019.

[19] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2018.

[20] Tesla. Upgrading autopilot: Seeing the world in radar. `https://www.tesla.com/blog/upgrading-autopilot-seeing-world-radar`, 2016. Accessed: 2020-07-27.

[21] Flemming Topsøe. Information-theoretical optimization techniques. *Kybernetika*, 1979.

[22] Aaron Tucker, Adam Gleave, and Stuart Russell. Inverse reinforcement learning for video games. *arXiv:1810.10593*, 2018.

[23] Mel Vecerik, Oleg Sushkov, David Barker, Thomas Rothörl, Todd Hester, and Jon Scholz. A practical approach to insertion with variable socket position using deep reinforcement learning. In *ICRA*, 2019.

[24] Markus Wulfmeier, Peter Ondruska, and Ingmar Posner. Maximum entropy deep inverse reinforcement learning. arXiv: 1507.04888v3 [cs.LG], 2015.

[25] Brian D Ziebart. *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. PhD thesis, CMU, 2010.

[26] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, 2008.