Some Key Concepts Used in Pragmatic Reinforcement Learning and Inverse Reinforcement Learning

Prasanth Sengadu Suresh
THINC Lab, School of Computing,
University of Georgia, Athens, GA 30602
prasuchit@gmail.com

Abstract

While writing my doctoral thesis, it occurred to me that others may benefit from a simple, reliable resource that effectively summarizes key concepts used in reinforcement learning (RL) and inverse reinforcement learning (IRL). Consequently, this tutorial was extracted from the background section of my thesis and hence assumes that the reader has a basic knowledge of probability theory, Markov decision processes (MDPs), and reinforcement learning (RL). Note that notations $variable$ and $variable^t$ denote the variable at current timestep t and $variable^{t+1}$ and $variable'$ denote the variable at the next timestep $t+1$. $variable^*$ denotes the variable at optimality.

1 Modelling the environment

An accurate model of the environment is crucial to make decisions under uncertainty. This model must be sophisticated enough to capture the task, the agent(s) attributes, and environmental uncertainties accurately. Using such a model, an intelligent algorithm can be designed to learn a behavioral policy that guides the agent appropriately.

1.1 Markov decision processes

A common modeling framework adopted throughout reinforcement learning literature is the Markov decision processes (MDP) (Puterman, 1994). Formally, the MDP of an expert is defined as a quadruple $\langle S, A, T, R\rangle$, where S is the set of states defining the environment, A is the expert's set of possible actions, $T : S \times A \times S \to [0, 1]$ gives the transition probabilities from any given state to a next state for each action, and $R : S \times A \to \mathbb{R}$ is the reward function modeling the expert's preferences, rewards, or costs of performing an action from a state. Typically, the learner is aware of the expert's S, A, and T, but not R. Model-free techniques don't assume access to T either.

An important assumption needed to correctly apply a MDP to an agent is that the current state subsumes all relevant information needed for the agent's decision, i.e. the agent's history has no bearing on the policy or transition function. This is known as the Markov assumption and is formally described as: $T(S_{1:t}, A_{1:t}, S_{t+1}) = T(S_t, A_t, S_{t+1})$. Note that we drop subscripts in common usage [1]. A (stationary) policy for a MDP is a mapping from states to actions $\pi : S \to A$ and the discounted, infinite-horizon value of a policy π for a given reward function R at some state $s \in S$, with t denoting time steps is given by:

$$E_s\left[V^\pi(s)\right] = E\left[\sum_{t=0}^{\infty} \gamma^t R(s^t, \pi(s^t))|s, \pi\right].$$

[1] It is important to note that some domains (especially partially observable ones) necessitate the need to condition the policy on the action-observation history and hence require a more general variant of the MDP - POMDP (Kaelbling et al., 1998). We do not consider those cases here.

In this work, we assume that the expert is a rational agent that follows the optimal policy while executing actions at every state. A rational agent is one that monotonically prefers policies that produce higher expected rewards. The forward reinforcement learning problem entails solving the aforementioned MDP by interacting with the environment to obtain feedback in the form of rewards. These rewards are used within the Bellman value function (Eqn (1)) to arrive at the optimal policy.

$$V^0(s) \;=\; \underset{a}{max}\, R_{\boldsymbol{\theta}}(s,a); \quad V^t(s) \;=\; \underset{a}{max}\left(R_{\boldsymbol{\theta}}(s,a) + \gamma \sum_{s'} T(s,a,s')V^{t-1}(s')\right). \tag{1}$$

Once the policy is learned, the agent may now refer to the policy to obtain the optimal action to perform at any given state. This produces a trajectory: $X \;=\; (< s,a,r >, < s,a,r >, ...)$ describing the sequence of states arrived at, actions chosen, and rewards received by the agent over time. Note that expert trajectories used in the context of IRL only contain the state-action pairs and not the step-wise reward.

Note that if you are not interested in multiagent modeling techniques, you can safely skip over Section 1.2.

1.2 Decentralized Markov decision processes

A single-agent MDP as defined in Section 1.1, is insufficient to represent scenarios where multiple agents are involved. Such a system is formally known as a *multiagent system* (Shoham and Leyton-Brown, 2008). The goal(s) of the agents in a multiagent system could be cooperative, competitive, of self-interest, or a mixture of the three. Each agent typically uses all available information about the other agents to make a decision at every step.

Depending on the task, the nature of the agents, and the amount of information they have access to (Xuan et al., 2000), a multiagent system can be modeled in a number of different ways:

1. **Markov game** - where each agent works to maximize its own reward until an equilibrium condition is met. Here each agent gets an individual reward for their actions and their policy maps the global state to their local actions.
2. **Multi-agent Markov decision process (MMDP)** - where every agent is aware of the global state, including attributes specific to themselves, the other agents, and the task. All agents receive a common reward as a result of their collective actions. Agents follow a joint policy that maps global states to global actions.
3. **Decentralized Markov decision process (Dec-MDP)** - specifically a locally fully-observable variant (Goldman and Zilberstein, 2003) is a model where each agent is only aware of their own local state - containing attributes specific to themselves and any directly observable task-related attributes. Like a MMDP, agents receive a common system reward as a result of their collective actions, however, each agent learns an independent policy, mapping their local state to a local action.

Note that all of the above models have a partially observable variant that we do not discuss in this thesis. Since our focus is on HRC, which is by definition decentralized and collaborative, we choose the Dec-MDP to model our collaborative scenarios. A two-agent Dec-MDP is defined using the following tuple:

$$\mathcal{DM} \triangleq \langle S, A, T, R \rangle$$

where the global state, $S = S_i \times S_j$. Here, S_i and S_j are the locally observed states of the two agents i and j, which when combined yield the complete global state of the system; $A = A_i \times A_j$ is the set of joint actions of the two agents; $T : S \times A \times S \to [0,1]$ is the transition function of the multi-agent system; and $R : S \times A \to \mathbb{R}$ is the common reward function. Note that the latter is unknown, whereas the rest of the elements of the model are usually known. As such, the agents know their own local state only, any observable parameters relevant to the interaction, and can act independently while optimizing a task-centric common reward (Melo and Veloso, 2011). Thus, our Dec-MDP is a locally fully observable model whose local states when combined yield the fully observable global state, per Goldman and Zilberstein (2003)'s categorization of such decentralized models. The solution to a Dec-MDP is a vector of policies, $\boldsymbol{\pi}^* = \langle \pi_i^*, \pi_j^* \rangle$, where $\pi_i^* : S_i \to A_i$ and analogously for π_j^*. If the interactions between the two agents are sparse and can occur at some joint states only, we may leverage this domain structure to simplify the model. Let

$S_I \in S$ be the set of states where an interaction may occur, and $S_{NI} = S/S_I$ be the remaining states. Then, we may specify the transition and reward functions as:

$$T(s, \mathbf{a}, s') = \begin{cases} Pr(s'|s, \mathbf{a}) & \text{if } s \in S_{NI} \\ Pr(s'_i|s_i, a_i) \cdot Pr(s'_j|s_j, a_j) & \text{if } s \in S_I \end{cases};$$

$$R(s, \mathbf{a}, s') = \begin{cases} R(s, \mathbf{a}, s') & \text{if } s \in S_{NI} \\ R_i(s_i, a_i, s'_i) + R_j(s_j, a_j, s'_j) & \text{if } s \in S_I \end{cases}$$

where $s \in S$, $s' \in S$, and $\boldsymbol{a} \in A$. Thus, we may exploit the transition and reward independence in the non-interaction states.

2 Inverse Reinforcement Learning

While reinforcement learning (RL) is a valid method to tackle several complex scenarios in robotics, designing an appropriate reward function that adequately captures the intricacies of real-world collaborative behaviors is non-trivial (Arora and Doshi, 2021). Therefore, we move to the paradigm of learning from observations (LfO), which entails observing an expert performing a task and learning to perform it similarly. LfO has several branches like inverse RL (IRL), imitation learning (IL), kinesthetic teaching, etc. Branches like IL often use techniques like behavior cloning (BC) (Torabi et al., 2018) which is essentially a supervised learning approach to learn a direct mapping from the states to the actions allowing the learner to mimic the expert. However, these techniques perform poorly on tasks with a lot of uncertainty (Ghasemipour et al., 2020).

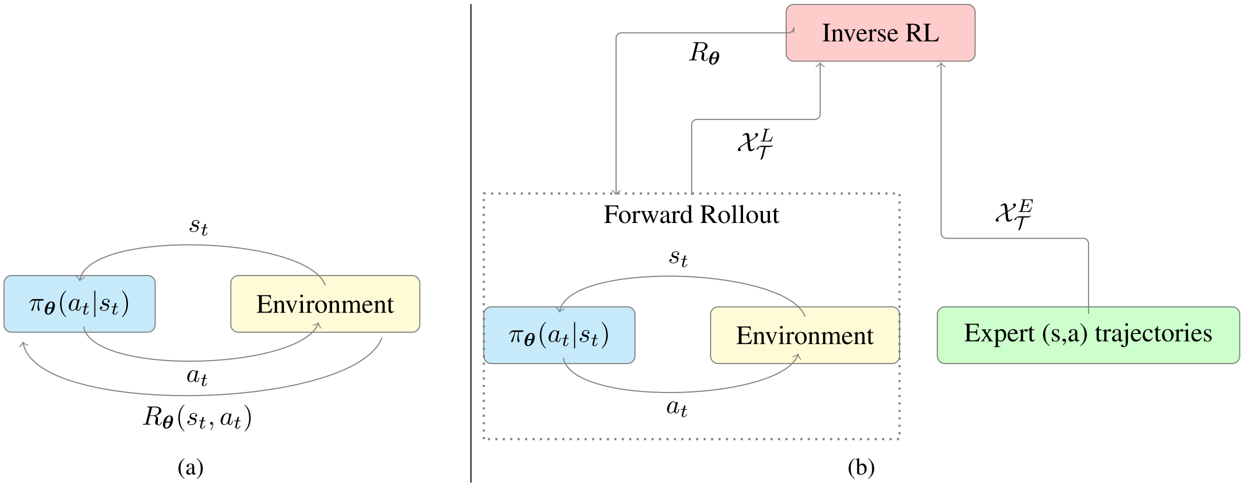
s_t
$\pi_{\boldsymbol{\theta}}(a_t|s_t)$
Environment
a_t
$R_{\boldsymbol{\theta}}(s_t, a_t)$
(a)
Inverse RL
$R_{\boldsymbol{\theta}}$
$\mathcal{X}^L_T$
Forward Rollout
$\mathcal{X}^E_T$
s_t
$\pi_{\boldsymbol{\theta}}(a_t|s_t)$
Environment
Expert (s,a) trajectories
a_t
(b)

Figure 1: Figure shows a comparison between the forward and inverse reinforcement learning paradigms. (a) Shows the forward reinforcement learning architecture where the agent interacts with the environment by performing an action (a_t) in the current state (s_t), receives a reward ($R_{\boldsymbol{\theta}}(s_t, a_t)$) and reaches the next state. Using this experience, it learns a policy ($\pi_{\boldsymbol{\theta}}(a_t|s_t)$). (b) Shows the inverse reinforcement learning paradigm where the observer uses expert demonstrations ($\mathcal{X}^E_T$) and sampled state-action pairs from the currently learned policy ($\mathcal{X}^L_T$) to learn a reward function ($R_{\boldsymbol{\theta}}$) that can be used in forward-rollout to update the learned policy.

Conversely, inverse reinforcement learning (Russell, 1998) refers to the problem of inferring the expert's preferences through a learned reward function using observations of the expert performing the task. This learned reward function can then be used in forward RL to solve the task optimally. *A key difference between imitation learning and inverse RL is that the former focuses on minimizing the classification error between the expert's behavior and the learned behavior, while the latter focuses on learning a behavior with equivalent value as the expert, which is a more relevant objective* (Abbeel and Ng, 2004). IRL literature calls the former objective - learned behavior accuracy (LBA) and the latter objective - inverse learning error (ILE). One of the problems with IRL is that it is an ill-posed problem. Meaning that the agent could learn an uncountable set of different reward functions that accurately explain the observed behavior. This is known as the *degeneracy problem* in IRL literature (Arora and Doshi, 2021). Several approaches have been proposed to solve this problem adequately, including max-margin IRL, apprenticeship learning, maximum entropy IRL, and Bayesian IRL.

Ng et al. (2000) proposed one of the earliest solutions to the IRL problem that constructs a linear program to maximize the margin of value between the action chosen by the expert in each state and all other actions. They further use a penalty term that encourages the use of smaller magnitude rewards. This can then be solved as a linear program to obtain the reward value for each state. They formulate the reward function as a linearly-weighted sum of K basis functions:

$$R_{\boldsymbol{\theta}}(s,a) \triangleq \sum_{k=1}^{K} \boldsymbol{\theta}_k \phi_k(s,a)$$

where K is finite and non-zero, $\boldsymbol{\theta_k}$ are the weights, and $\phi : (S, A) \rightarrow (0, 1)$ is a *feature function*. A binary feature function maps a state from the set of states S and an action from the set of actions A to 0 (false) or 1 (true).

Notice that this approach assumes access to a single optimum policy from the expert, and that this representation requires pre-defining these features. An alternative is to learn these feature functions (Levine et al., 2010) or use a neural network representation (Wulfmeier and Posner, 2015), which automatically identifies the features but typically requires far more demonstrations to converge. The learner's task now becomes finding a vector of weights $\boldsymbol{\theta}$ that complete the reward function. One way to narrow down the search space is by calculating an expectation of features that any given policy would obtain during execution:

$$\phi_\pi = \sum_s \mu_\pi(s)\phi(s, \pi(s))$$

where $\mu_\pi(s)$ is the state visitation frequency of state s, as explained in Eq. (2) (μ_π^0 defines the initial state distribution). Let $\mathcal{X}^E$ be the set of expert demonstrations and let X denote a complete trajectory, where $X \in \mathcal{X}^E$ is given by, $X = (s^1, a^1, s^2, a^2, s^3, ..., s^{\mathcal{T}}, a^{\mathcal{T}})$; [2]. Now we can calculate the empirical feature expectations of the expert from an observed trajectory as:

$$\hat{\phi}_E = \frac{1}{|X|} \sum_{(s,a) \in X} \phi_k(s,a).$$

Once we reach a set of weights where the corresponding reward function generates a policy satisfying the constraint: ($\phi_\pi = \hat{\phi}_E$) the IRL problem is considered to be solved. At this point, we can calculate the gradient of the weight vector as

$$\nabla\boldsymbol{\theta} = \phi_\pi - \hat{\phi}_E.$$

wherein if the learned policy results in a feature expectation that is higher than the expert's, the corresponding weight must be reduced, and vice versa.

$$\mu_\pi(s) = \mu_\pi^0(s) + \gamma \sum_{s'} T(s, \pi(s), s')\mu_\pi(s') \tag{2}$$

Since the set of weight vectors that produce a policy matching this criterion is infinite, Abbeel and Ng (2004) formulate a max-margin approach to maximize the value difference between the expert policy and all previously found policies to arrive at the following quadratic program:

$$\max_{m,\theta} \; m \quad \text{subject to} \quad \begin{array}{l} \boldsymbol{\theta}^T \hat{\Phi}_E \geq \boldsymbol{\theta}^T \Phi^{(i)} + m, \;\; i = 0, \ldots, N-1; \\ \| \, \boldsymbol{\theta} \, \|_2 \leq \; 1. \end{array} \tag{3}$$

Where $\boldsymbol{\theta}$ is a vector of weights, $\hat{\Phi}_E$ is the expert's empirical feature expectations, and $\Phi^{(i)}$ is the feature expectations from the policy found at iteration i. Equation 3 tries to solve for a reward function on which the expert does better, by a "margin" of m, than any of the N policies previously obtained. This program can be solved with any quadratic solver, such as SVM, to obtain the candidate feature weights. A convex combination of all produced policies is then generated, weighted such that the feature expectation of the distribution over these policies matches exactly to the expert's empirical feature expectations.

[2] Note the important distinction that expert trajectories in IRL do not include the immediate reward obtained vis-a-vis the definition of a trajectory defined in Section 1.1.

3 Bayesian IRL and MAP inference

One of the most popular methods in IRL is Bayesian IRL (BIRL) (Ramachandran and Amir, 2007a). Before we dive into BIRL, let's take a quick look at Bayes' theorem. The Bayesian probability theory, named after Thomas Bayes, describes the probability of a hypothesis (H) being true, given some evidence (E). In order to do so it uses, the probability of seeing the evidence E, given that the hypothesis H is true - this is called the *likelihood*, and the prior knowledge about the probability of the hypothesis H being true - called the *prior*. This is then divided by the total probability of seeing the evidence E, to arrive at the *posterior* inference. Formally, this can be represented as:

$$\textit{Posterior}\ \Pr(H|E) \ = \ \frac{\overbrace{\Pr(E|H)}^{Likelihood} \times \overbrace{\Pr(H)}^{Prior}}{\underbrace{\Pr(E)}_{\text{Partition function}}}. \tag{4}$$

To present IRL in this formulation, BIRL postulates the reward function as the hypothesis explaining the observed behavior of the expert (which is the evidence). Therefore, the posterior estimates the probability that the learned reward function is true, given the expert's trajectories. BIRL treats the reward function as a random variable and utilizes a prior distribution over the reward function, given as

$$\Pr(R_{\boldsymbol{\theta}}) \ = \ \prod\nolimits_{s\in S, a\epsilon A} \Pr(R_{\boldsymbol{\theta}}(s,a)). \tag{5}$$

Notice that the reward values for the state-action pairs are i.i.d. Ramchandran and Amir [2007b] discuss some example prior distributions including the Gaussian [3]. We may derive the likelihood function for the demonstrated set of trajectories $\mathcal{X}$ as:

$$\Pr(\mathcal{X}|R_{\boldsymbol{\theta}}) = \prod_{X=1}^{|\mathcal{X}|}\prod_{t=1}^{\mathcal{T}} \Pr(s_X^t, a_X^t; R_{\boldsymbol{\theta}}) = \prod_{X=1}^{|\mathcal{X}|} \Pr(s_X^1)\Pr(a_X^1|s_X^1; R_{\boldsymbol{\theta}}) \prod_{t=1}^{\mathcal{T}-1} \Pr(s_X^{t+1}|s_X^t, a_X^t)\ \Pr(a_X^{t+1}|s_X^{t+1}; R_{\boldsymbol{\theta}}).$$

We may rewrite this as,

$$\Pr(\mathcal{X}|R_{\boldsymbol{\theta}}) \ = \ \prod\nolimits_{X=1}^{|\mathcal{X}|} \Pr(s_X^1)\ \pi(a_X^1|s_X^1; R_{\boldsymbol{\theta}}) \times \prod\nolimits_{t=1}^{\mathcal{T}-1}\ T(s_X^t, a_X^t, s_X^{t+1})\ \pi(a_X^{t+1}|s_X^{t+1}; R_{\boldsymbol{\theta}}). \tag{6}$$

The policy is commonly modeled in BIRL as a Boltzmann energy function (Ramachandran and Amir, 2007b; Vroman, 2014) of the form:

$$\pi_{\boldsymbol{\theta}}(a|s) \ = \ \frac{e^{\beta Q_{\boldsymbol{\theta}}(s,a)}}{\sum_{a'\in A} e^{\beta Q_{\boldsymbol{\theta}}(s,a')}} = \frac{e^{\beta Q_{\boldsymbol{\theta}}(s,a)}}{\Xi(s)}. \tag{7}$$

where $\Xi(s)$ is the partition function. As the Boltzmann temperature parameter β becomes large, exploration assigns increasing probability to the action(s) with the largest Q-value(s). One possible assignment to β could be between $0 - 1$ with 0 being fully exploratory and 1 being fully greedy.

The Q function is given as:

$$Q_{\boldsymbol{\theta}}(s,a) = R_{\boldsymbol{\theta}}(s,a) + \gamma \sum_{s'\in S} T(s,a,s') \times \sum_{a'\in A} Q_{\boldsymbol{\theta}}(s',a')\ \pi_{\boldsymbol{\theta}}(s',a'). \tag{8}$$

Methods for both maximum likelihood (Vroman, 2014; Jain et al., 19) and maximum-a-posteriori (Choi and Kim, 2011) inferences of the reward function exist, which use the likelihood function of Eq. 6 and, in case of MAP inference, the prior as well [4]. MAP inference for IRL has been shown to be more accurate, benefiting from its use of the prior (Choi and Kim, 2011). Formally, we may write MAP inference in *unnormalized* log form [5] as:

$$R^*_{\boldsymbol{\theta}} \ = \ \arg\max_{\mathcal{R}} \ \underbrace{L_{\boldsymbol{\theta}}}_{\text{log-posterior}} \ = \ \arg\max_{\mathcal{R}} \ \underbrace{L^{lh}_{\boldsymbol{\theta}}}_{\text{log-likelihood}} \ + \ \underbrace{L^{pr}_{\boldsymbol{\theta}}}_{\text{log-prior}}\ . \tag{9}$$

[3] Note that Gaussian distribution is one of the most commonly used priors in Bayesian probability theory because its conjugate prior is also a Gaussian distribution, which makes it easy to estimate.

[4] We consider the prior to be a Gaussian distribution in our case.

[5] Log form is preferred because it preserves function monotonicity and simplifies product terms into sum terms. This is an especially useful feature in optimization since computing differentials of product terms can get hairy.

where $\mathcal{R}$ is the continuous space of reward functions, and

$$L_{\boldsymbol{\theta}} = \log \Pr(R_{\boldsymbol{\theta}}|\mathcal{X}); \quad L^{lh}_{\boldsymbol{\theta}} = \log \Pr(\mathcal{X}|R_{\boldsymbol{\theta}}); \quad L^{pr}_{\boldsymbol{\theta}} = \log \Pr(R_{\boldsymbol{\theta}}). \tag{10}$$

Consequently, the partial differential of Eqn 9 becomes:

$$\frac{\partial L_{\boldsymbol{\theta}}}{\partial \boldsymbol{\theta}} = \frac{\partial L^{lh}_{\boldsymbol{\theta}}}{\partial \boldsymbol{\theta}} + \frac{\partial L^{pr}_{\boldsymbol{\theta}}}{\partial \boldsymbol{\theta}}.$$

The partial differential of the log-likelihood and log-prior are given as:

$$\frac{\partial L^{lh}_{\boldsymbol{\theta}}}{\partial \boldsymbol{\theta}} = \sum_{x\in\mathcal{X}} \sum_{t=1}^{\mathcal{T}} \frac{1}{\pi_{\boldsymbol{\theta}}(s^t, a^t)} \frac{\partial \pi_{\boldsymbol{\theta}}(s^t, a^t)}{\partial \boldsymbol{\theta}}; \quad \frac{\partial L^{pr}_{\boldsymbol{\theta}}}{\partial \boldsymbol{\theta}} = \left(\frac{-(\boldsymbol{\theta} - \mu_{\boldsymbol{\theta}})}{\sigma^2_{\boldsymbol{\theta}}} \right).$$

Choi and Kim (2011) presents a gradient-based approach to obtain R^*, which searches the reward optimality region H^π only. Given the expert's policy, Ng and Russell [2000] show that this region can be obtained as:

$$H^\pi \triangleq I - (I^A - \gamma T)(I - \gamma T^\pi)^{-1} E^\pi. \tag{11}$$

where I is the identity matrix, T is the transition matrix, E^π is an $|S| \times |S||A|$ matrix with the $(s, (s', a'))$ element being 1 if s = s' and $\pi(s')$ = a'. I^A is an $|S||A| \times |S|$ matrix constructed by stacking the $|S| \times |S|$ identity matrices $|A|$ times. The reward update rule in the gradient ascent is given as,

$$R^{new}_{\boldsymbol{\theta}} \leftarrow R_{\boldsymbol{\theta}} + \delta_t \nabla_{\boldsymbol{\theta}} \Pr(R_{\boldsymbol{\theta}}|\mathcal{X}). \tag{12}$$

where δ_t is an appropriate step size (or the learning rate). As computing $\nabla_{\boldsymbol{\theta}} \Pr(R_{\boldsymbol{\theta}}|\mathcal{X})$ involves calculating an optimal policy, this may slow down the computations. By checking if the gradient lies within the new reward optimality region using the condition: $H^\pi \cdot R^{new}_{\boldsymbol{\theta}} \leq 0$, we can reuse the same gradient and reduce the computational time.

4 Maximum Entropy IRL

The principle of maximum entropy originates from information theory and was proposed by E.T.Jaynes in 1957 when he put forth the idea that entropy in information theory and entropy in statistical mechanics is based on the same principle. The theory of maximum entropy argues that while there may be an infinite number of probability distributions that may satisfy a set of constraints, the one with the maximum entropy is the one that makes the least assumptions about the data, apart from what is required to satisfy the constraints. This makes the learned distribution maximally uncertain about the parts of data that it hasn't encountered before and hence is an encoding of all the provided constraint information and nothing else. As described by Ziebart et al. (2008), MaxEntIRL builds a probability distribution over all possible expert trajectories as:

$$\max_{\Delta} \left(\underbrace{-\sum_{X\in\mathcal{X}} \Pr(X)\ log\ \Pr(X)}_{\textit{H(A|S) - Shannon entropy}} \right) \quad \text{subject to} \quad \begin{aligned} &\sum_{X\in\mathcal{X}} \Pr(X) = 1, \\ &\sum_{X\in\mathcal{X}} \Pr(X) \sum_{\langle s,a\rangle \in X} \phi_k(s,a) = \hat{\phi}_k \quad \forall k. \end{aligned} \tag{13}$$

Here, $\mathcal{X}$ is the set of expert trajectories, $H(A|S)$ denotes the conditional entropy distribution of the policy, Δ is the space of all distributions $\Pr(X)$. This is also known as maximizing Shannon entropy (Zellner and Highfield, 1988). In order to simplify this, we apply Lagrangian relaxation to Eq 13 which is the *primal-problem*, to form an objective function that subsumes both the constraints, and then we solve the Lagrangian dual. The new objective function becomes:

$$\begin{aligned} \mathcal{L}(Pr, \boldsymbol{\theta}, \eta) \quad &= -\sum_{X\in\mathcal{X}} \Pr(X)\ log \Pr(X) + \sum_k \theta_k \left(\sum_{X\in\mathcal{X}} \Pr(X) \sum_{\langle s,a\rangle\in X} \phi_k(s,a) - \hat{\phi}_k \right) \\ &\qquad + \eta \left(\sum_{X\in\mathcal{X}} \Pr(X) - 1 \right). \end{aligned} \tag{14}$$

Where $\boldsymbol{\theta}$ and η are the Lagrange multipliers and since Eq. 14 is convex, taking the partial derivative of $\mathcal{L}$ with respect to $\Pr(X)$ and setting it to zero gives us the optimum:

$$\frac{\partial \mathcal{L}}{\partial \Pr(X)} = -log\ \Pr(X) - 1 + \sum_k \theta_k \sum_{\langle s,a \rangle \in X} \phi_k(s,a) + \eta = 0;$$

$$\text{where,} \quad \Pr(X) = \frac{e^{\sum_k \theta_k \sum_{\langle s,a \rangle \in X} \phi_k(s,a)}}{\Xi(\boldsymbol{\theta})}. \tag{15}$$

$\Xi(\boldsymbol{\theta})$ is the normalization constant $e^{\eta} \cdot e^{-1}$; where η may be obtained using the Karush-Kuhn-Tucker (KKT) conditions.Substituting $\Pr(X)$ from Eq. 15 into the Lagrangian Eq. 14), we arrive at the dual $\mathcal{L}^{\text{dual}}(\boldsymbol{\theta})$ which is concave. We now have:

$$\mathcal{L}^{\text{dual}}(\boldsymbol{\theta}) = log\ \Xi(\boldsymbol{\theta}) - \sum\nolimits_k \theta_k \hat{\phi}_k.$$

With its gradient being,

$$\nabla \mathcal{L}^{\text{dual}}(\boldsymbol{\theta}) \ = \ \sum\nolimits_{X \in \mathcal{X}} \Pr(X) \sum_{\langle s,a \rangle \in X} \phi_k(s,a) - \hat{\phi}_k.$$

This method is built on the hypothesis that the expert follows a particular trajectory with a probability proportional to the reward accrued along it. While this is valid in a deterministic setting, in a stochastic MDP, an approximation is considered (Bogert, 2016).

$$\Pr(Y) \approx \frac{\prod_{<s,a,s'> \in Y} T(s,a,s') e^{\sum_k \theta_k \sum_{\langle s,a \rangle \in Y} \phi_k(s,a)}}{\Xi(\boldsymbol{\theta})}.$$

4.1 Maximum Causal Entropy

In order to avoid the bias that maximizing Shannon entropy introduces towards actions with uncertain (and possibly) risky outcomes, maximum causal entropy must be adopted in stochastic MDPs. Causal entropy is the sum of the entropies of the policy action selected conditioned on the state at that timestep, $H(A_{0:T-1} || S_{0:T-1}) = \sum_{t=0}^{T-1} \gamma^t H(A_t | S_t)$ (Gleave and Toyer, 2022). This has the useful property that it conditions only on the information available to the agent until the current timestep, namely, the current state, as well as the prior states and actions. Conversely, conventional Shannon entropy computes the entropy over the entire trajectory distribution, introducing an unwanted dependency on the transition dynamics (Ziebart et al., 2010). Formally, we can derive this as:

$$\begin{aligned}
H(S_{0:T-1}, A_{0:T-1}) &= \sum_{t=0}^{T-1} \gamma^t H(S_t, A_t | S_{0:t-1}, A_{0:t-1}) && \textit{chain rule} \\
&= \sum_{t=0}^{T-1} \gamma^t H(S_t | S_{0:t-1}, A_{0:t-1}) + H(A_t | S_{0:t-1}, A_{0:t-1}) && \textit{chain rule} \\
&= \sum_{t=0}^{T-1} \gamma^t H(S_t | S_{t-1}, A_{t-1}) + H(A_t | S_t) && \textit{independence} \\
&= \sum_{t=0}^{T-1} \underbrace{\gamma^t H(S_t | S_{t-1}, A_{t-1})}_{\textit{state transition entropy}} + \underbrace{H(A_{0:T-1} || S_{0:T-1})}_{\textit{causal entropy}} && \textit{causal ent.}
\end{aligned}$$

5 Adversarial Inverse Reinforcement Learning

The idea of adversarial training using two competing neural networks was first popularized by Goodfellow et.al, in 2014. This was given the moniker generative adversarial networks (GANs) and describes two neural networks (often deep networks) called the generator and the discriminator that compete against each other during training; upon convergence, the optimal generator accurately depicts the probability distribution of the training data that it received. The generator network learns to generate samples by attempting to fool the discriminator network into believing its samples are real data. Both estimation procedures use the same function to drive learning, and sans a delicate balance between them, the learning is thrown off course. Drawing inspiration from GANs, generative adversarial imitation learning (GAIL) architecture was proposed by Ho and Ermon (2016).

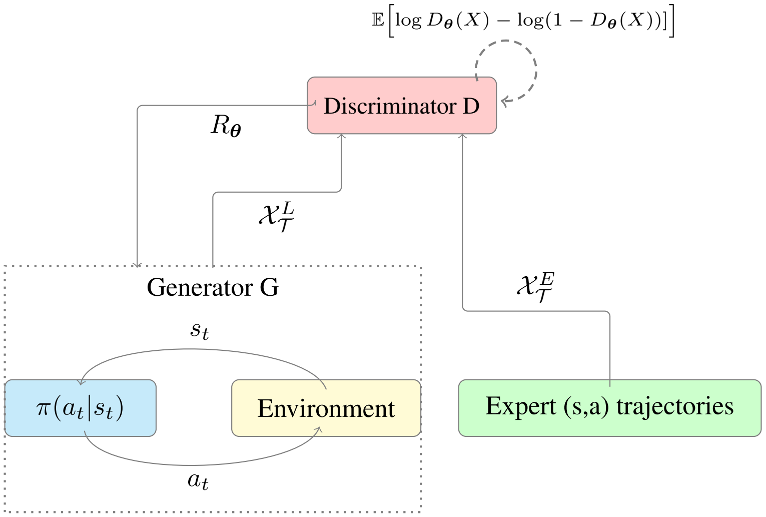
$\mathbb{E}\Big[\log D_{\boldsymbol{\theta}}(X) - \log(1 - D_{\boldsymbol{\theta}}(X))\Big]$
Discriminator D
$R_{\boldsymbol{\theta}}$
$\mathcal{X}_T^L$
$\mathcal{X}_T^E$
Generator G
s_t
$\pi(a_t|s_t)$
Environment
a_t
Expert (s,a) trajectories

Figure 2: Architecture of adversarial inverse reinforcement learning (AIRL). Discriminator D receives expert demos and sampled trajectories, using which it minimizes the reverse KL divergence between expert and learned state-action marginal distributions to learn a function $f_{\boldsymbol{\theta}}$. G learns policy using $f_{\boldsymbol{\theta}} - \log(\pi)$ as the reward. We use the dotted arrow at D to denote the training loop with the objective function denoted over the dotted arrow. The definitions of $\mathcal{X}_T^E$ and $\mathcal{X}_T^L$ are the same as Fig. 1b.

GAIL by definition being an imitation learning technique learns to mimic the expert by arriving at a policy distribution that accurately represents the distribution of the input expert demonstrations. As discussed in Section 2, blindly mimicking the expert's behavior doesn't work well in environments with high uncertainty. Therefore, Fu et.al formulated Adversarial Inverse Reinforcement Learning (AIRL) in 2018 by extending Chelsea Finn's work - Guided Cost Learning (Finn et al., 2016). AIRL builds on the maximum causal entropy IRL framework (Ziebart, 2010), based on an entropy-regularized MDP. While AIRL shares similarities with GAIL (Ho and Ermon, 2016), GAIL does not analytically solve for the reward function and instead uses $-log(1 - D_{\boldsymbol{\theta}})$ (where $D_{\boldsymbol{\theta}}$ is the discriminator at the current epoch), as the reward function.

In most imitation learning and inverse RL methods, the stochastic policy is represented using a Boltzmann energy distribution, where the parameters of the distribution are optimized to maximize the likelihood of the observed data points, thus arriving at a policy distribution identical to the expert's. Alternatively, a simple approach can also be taken to this problem by minimizing the classification error of the output policy from the generator and the observed data, without learning a discriminator or energy function. However, when the generative model does not have the capacity to represent the entire data distribution sufficiently, maximizing likelihood directly without an energy function, will lead to a moment-matching distribution that tries to "cover" all of the modes, leading to a solution that puts much of its mass in parts of the space that have negligible probability under the true distribution.

Consequently, even if the generator trained through maximum likelihood has the same capacity as the one with an energy distribution, the latter exhibits mode-seeking behavior as long as the energy function is more flexible than the generator. Evidently, this phenomenon is often achieved at the cost of tractability, since generating samples from an energy function requires training a generator, which in the case of IRL, is forward policy optimization (Finn et al., 2016). While methods like behavior cloning (BC) that employ such maximum likelihood techniques to learn a behavior matching that of the demonstrating agent, these techniques prove particularly ineffective in complex scenarios due to compounding errors. In other words, when the policy erroneously deviates from the learned behavior, it encounters states other than what it was exposed to during training, making it more likely to continue making mistakes. This phenomenon is referred to as the covariate shift (Ghasemipour et al., 2020).

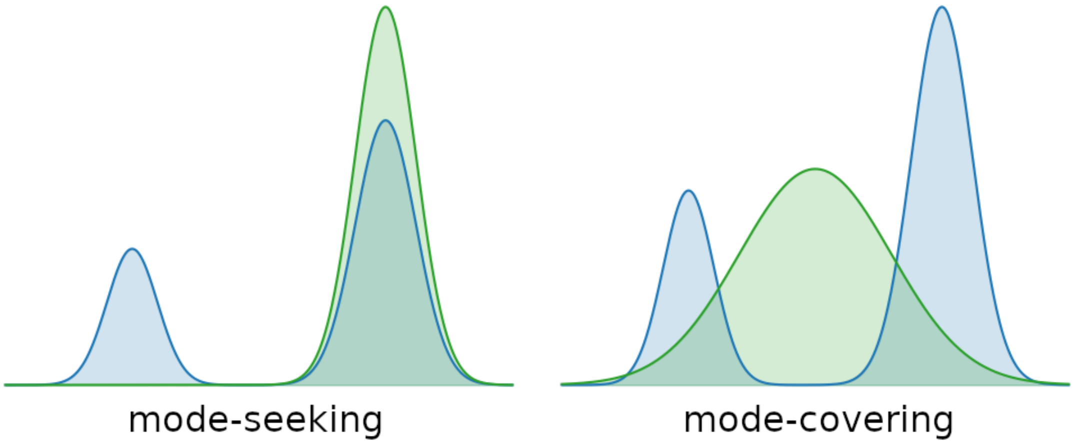

Figure 3: Comparison between the distributions learned by minimizing reverse KL divergence verses Jensen-Shannon divergence (Menéndez et al., 1997) or forward KL divergence. Notice that the latter induces a mean-seeking or mode-covering behavior, whereas the former induces a mode-seeking behavior. In many scenarios, it is preferable to produce only realistic, highly probable samples, by "filling in" as many modes as possible, at the trade-off of reduced variance (Finn et al., 2016). Image credit: Sander AI

Forward reinforcement learning aims to find the optimal policy π^* that maximizes the expected entropy-regularized discounted reward, under π, T, and ρ_0:

$$\pi^* = \arg\max_{\pi} E_{X\sim\pi}\Big[\sum_{t=0}^{\mathcal{T}} \gamma^t \Big(r(s_t, a_t) + H(\pi(.|s_t))\Big)\Big].$$

The distribution of the trajectories derived from the optimal policy $\pi^*(a|s)$ has been shown to take the form $\pi^*(a|s) \propto \exp Q^*_{soft}(s_t, a_t)$ in the MaxEntIRL formulation (Ziebart, 2010; Haarnoja et al., 2017), where:

$$Q^*_{soft}(s_t, a_t) = r_t(s, a) + E_{(s_{t+1},\ldots)\sim\pi}\underbrace{\Big[\sum_{t'=t}^{\mathcal{T}} \gamma^t \Big(r(s'_t, a'_t) + H(\pi(.|s'_t))\Big)\Big]}_{\text{Entropy regularized discounted reward}}.$$

In the context of AIRL, $\pi(a|s)$ is the probability of the adaptive sampler, and the goal is to minimize the reverse KL divergence between the trajectory distribution generated by the generator and that induced by the reward function (Alsaleh and Sayed, 2021). The structure of the discriminator is adapted from GCL (Finn et al., 2016) where the discriminator $D_{\boldsymbol{\theta}}(X)$ learns a function $f_{\boldsymbol{\theta}}(X)$ (Fu et al., 2018) which at optimality is the expert policy's advantage function. This structure is very similar to a typical model for binary classification, with a sigmoid as the final layer. By subtracting $\log(\pi(X))$ from the input to the sigmoid $f_{\boldsymbol{\theta}}$, the discriminator can be made completely independent of the generator. This change is very simple to implement and is applicable in any setting where the density $\pi(X)$ can be cheaply evaluated (Finn et al., 2016).

$$\begin{aligned}
\mathcal{S}(p) &= \frac{1}{1+\exp\{-p\}} \quad \rightarrow \quad \text{Regular sigmoid function with input } p \\
\mathcal{S}(f_{\boldsymbol{\theta}} - \log(\pi(X))) &= \frac{1}{1+\exp\{-(f_{\boldsymbol{\theta}} - \log(\pi(X)))\}} \\
&= \frac{\exp\{f_{\boldsymbol{\theta}} - \log\pi(X)\}}{\exp\{f_{\boldsymbol{\theta}} - \log(\pi(X))\}+1} = \frac{\frac{\exp\{f_{\boldsymbol{\theta}}\}}{\exp\{\log\pi(X)\}}}{\frac{\exp\{f_{\boldsymbol{\theta}}\}}{\exp\{\log\pi(X)\}}+1} \\
&= \frac{\frac{\exp\{f_{\boldsymbol{\theta}}\}}{\cancel{\exp\{\log\pi(X)\}}}}{\frac{\exp\{f_{\boldsymbol{\theta}}\}+\exp\{\log\pi(X)\}}{\cancel{\exp\{\log\pi(X)\}}}} = \frac{\exp\{f_{\boldsymbol{\theta}}\}}{\exp\{f_{\boldsymbol{\theta}}\}+\pi(X)}.
\end{aligned}$$

The learned softmax policy is given as,

$$\pi(X) = exp\Big(A^{soft}_{f_{\boldsymbol{\theta}}}(s,a)\Big) = exp\Big(Q^{soft}_{f_{\boldsymbol{\theta}}}(s,a) - V^{soft}_{f_{\boldsymbol{\theta}}}(s)\Big) \tag{16}$$

The learned softmax policy is given as,

$$\pi(X) = exp\Big(A^{soft}_{f_{\boldsymbol{\theta}}}(s,a)\Big) = exp\Big(Q^{soft}_{f_{\boldsymbol{\theta}}}(s,a) - V^{soft}_{f_{\boldsymbol{\theta}}}(s)\Big) \tag{17}$$

The soft advantage function in Eqn 17 (Gleave and Toyer, 2022) is computed through the forward rollout before the next discriminator update and used as:

$$D_{\boldsymbol{\theta}}(X) = \frac{\exp\{f_{\boldsymbol{\theta}}(X)\}}{\exp\{f_{\boldsymbol{\theta}}(X)\} + \pi(X)} \tag{18}$$

The goal of the discriminator is to minimize the binary cross-entropy loss (Mao et al., 2023) between the expert and learned distributions as:

$$L(\theta) = \sum_{t=0}^{T}\left(-\mathop{\mathbb{E}}_{(s_t,a_t)\sim X_E^t}[\log D_\theta(s_t,a_t)] - \mathop{\mathbb{E}}_{(s_t,a_t)\sim X_L^t}[\log(1 - D_\theta(s_t,a_t))]\right).$$

Therefore the reward function can be denoted as (we drop the subscripts here to avoid clutter),

$$R_{\boldsymbol{\theta}}(X) \leftarrow \log D_{\boldsymbol{\theta}}(X) - \log(1 - D_{\boldsymbol{\theta}}(X)). \tag{19}$$

Substituting the value of $D_{\boldsymbol{\theta}}$ from Eqn 18 into Eqn 19, we get,

$$\begin{aligned}
R_{\boldsymbol{\theta}}(X) &= \log\Big(\frac{\exp\{f_{\boldsymbol{\theta}}(X)\}}{\exp\{f_{\boldsymbol{\theta}}(X)\} + \pi(X)}\Big) - \log\Big(1 - \frac{\exp\{f_{\boldsymbol{\theta}}(X)\}}{\exp\{f_{\boldsymbol{\theta}}(X)\} + \pi(X)}\Big) \\
&= \Big(\log\exp\{f_{\boldsymbol{\theta}}(X)\} - \log\Big(\exp\{f_{\boldsymbol{\theta}}(X)\} + \pi(X)\Big)\Big) \\
&\qquad\qquad\qquad\qquad - \log\Big(\frac{\cancel{\exp\{f_{\boldsymbol{\theta}}(X)\}} + \pi(X) - \cancel{\exp\{f_{\boldsymbol{\theta}}(X)\}}}{\exp\{f_{\boldsymbol{\theta}}(X)\} + \pi(X)}\Big) \\
&= \Big(f_{\boldsymbol{\theta}}(X) - \log(\exp\{f_{\boldsymbol{\theta}}(X)\} + \pi(X))\Big) - \log\Big(\frac{\pi(X)}{\exp\{f_{\boldsymbol{\theta}}(X)\} + \pi(X)}\Big) \\
&= f_{\boldsymbol{\theta}}(X) - \cancel{\log\Big(\exp\{f_{\boldsymbol{\theta}}(X)\} + \pi(X)\Big)} - \log(\pi(X)) + \cancel{\log\Big(\exp\{f_{\boldsymbol{\theta}}(X)\} - \log(\pi(X))\Big)} \\
&= f_{\boldsymbol{\theta}}(X) - \log(\pi(X)). \qquad \rightarrow \qquad \text{Entropy regularized reward.}
\end{aligned} \tag{20}$$

In contrast to GCL's formulation, AIRL's discriminator does not use a partition function $(1/Z)$ for the $\exp\{f_{\boldsymbol{\theta}}(X)\}$ term because for a given state, $Z(s)$ remains constant and is assumed to be implicitly learned as part of $f_{\boldsymbol{\theta}}(X)$, although it cannot be extracted [6]. AIRL aims to minimize the *reverse KL divergence* between the learner's and expert's marginal state-action distribution $KL(\rho_\pi(s,a)||\rho_{exp}(s,a))$ since this results in mode-seeking behavior in contrast to BC methods that use forward KL divergence with a mode-covering behavior (Ghasemipour et al., 2020).

$$KL(\rho_\pi(s,a)||\rho_{exp}(s,a)) = \sum_{(s,a)\in X}\sum_{t=0}^{T-1}\Pr_\pi(s^t,a^t)\Big[\log(\Pr_\pi(s^t,a^t)) - \log(\Pr_{exp}(s^t,a^t))\Big]. \tag{21}$$

Note that Fu et al. (2018)'s disentangled reward formulation comes with the following strong assumptions:

1. The MDP modelling the environment is ergodic. In other words, every state gets visited eventually.
2. The MDP has deterministic environment transitions (Geng et al., 2020; Venuto, 2020).
3. The deterministic transition dynamics satisfies a strong requirement known as the decomposability condition (Gleave and Toyer, 2022). In other words, all pairs of states in the MDP are linked.

[6] (Fu et al., 2018) AIRL OpenReview comment.

References

M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1994. ISBN 0471619779.

L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

Y. Shoham and K. Leyton-Brown. *Multiagent systems: Algorithmic, game-theoretic, and logical foundations*. Cambridge University Press, 2008.

P. Xuan, V. Lesser, and S. Zilberstein. s. In *Proceedings Fourth International Conference on MultiAgent Systems*, pages 467–468. IEEE, 2000.

C. V. Goldman and S. Zilberstein. Optimizing information exchange in cooperative multi-agent systems. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '03, pages 137–144, New York, NY, USA, 2003. ACM. ISBN 1-58113-683-8.

F. S. Melo and M. Veloso. Decentralized mdps with sparse interactions. *Artificial Intelligence*, 175(11):1757–1789, 2011.

S. Arora and P. Doshi. A survey of inverse reinforcement learning: Challenges, methods and progress. *Artificial Intelligence*, 297:103500, 2021.

F. Torabi, G. Warnell, and P. Stone. Behavioral cloning from observation. *arXiv preprint arXiv:1805.01954*, 2018.

S. K. S. Ghasemipour, R. Zemel, and S. Gu. A divergence minimization perspective on imitation learning methods. In *Conference on Robot Learning*, pages 1259–1277, unknown, 2020. PMLR, unknown.

S. Russell. Learning agents for uncertain environments (extended abstract). In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, COLT' 98, pages 101–103, New York, NY, USA, 1998. ACM. ISBN 1-58113-057-0.

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, 2004.

A. Y. Ng, S. J. Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, page 2, 2000.

S. Levine, Z. Popović, and V. Koltun. Feature construction for inverse reinforcement learning. In *Proceedings of the 23rd International Conference on Neural Information Processing Systems*, NIPS'10, pages 1342–1350, USA, 2010. Curran Associates Inc.

M. Wulfmeier and I. Posner. Maximum Entropy Deep Inverse Reinforcement Learning. *arXiv preprint*, 2015.

P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pages 1–8, New York, NY, USA, 2004. ACM. ISBN 1-58113-838-5.

D. Ramachandran and E. Amir. Bayesian inverse reinforcement learning. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 2586–2591, San Francisco, CA, USA, 2007a. Morgan Kaufmann Publishers Inc.

D. Ramachandran and E. Amir. Bayesian inverse reinforcement learning. In *IJCAI*, volume 7, pages 2586–2591, 2007b.

M. C. Vroman. *Maximum likelihood inverse reinforcement learning*. PhD thesis, Rutgers University-Graduate School-New Brunswick, 2014.

V. Jain, P. Doshi, and B. Banerjee. Model-free irl using maximum likelihood estimation. In *AAAI Conference on Artificial Intelligence*, pages 3951–3958, 19.

J. Choi and K.-E. Kim. Map inference for bayesian inverse reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 1989–1997, 2011.

E. T. Jaynes. Information theory and statistical mechanics. *Phys. Rev.*, 106:620–630, May 1957.

B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey. Maximum entropy inverse reinforcement learning. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*, AAAI'08, pages 1433–1438, None, 2008. AAAI Press. ISBN 978-1-57735-368-3.

A. Zellner and R. A. Highfield. Calculation of maximum entropy distributions and approximation of marginalposterior distributions. *Journal of Econometrics*, 37(2):195–209, 1988.

K. Bogert. *Inverse Reinforcement Learning for Robotic Applications: Hidden Variables, Multiple Experts and Unknown Dynamics*. PhD thesis, University of Georgia, 2016.

A. Gleave and S. Toyer. A primer on maximum causal entropy inverse reinforcement learning. *arXiv preprint arXiv:2203.11409*, 1:arxiv, 2022.

B. D. Ziebart, J. A. Bagnell, and A. K. Dey. Modeling interaction via the principle of maximum causal entropy. In *ICML*, 2010.

I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014.

J. Ho and S. Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29:None, 2016.

J. Fu, K. Luo, and S. Levine. Learning robust rewards with adverserial inverse reinforcement learning. In *International Conference on Learning Representations*, pages 1–10, None, 2018. unknown. URL `https://openreview.net/forum?id=rkHywl-A-`.

C. Finn, S. Levine, and P. Abbeel. Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization. *arXiv preprint*, arXiv:1603.00448:0, 2016.

B. Ziebart. *Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy*. PhD thesis, Carnegie Mellon University, December 2010.

M. Menéndez, J. Pardo, L. Pardo, and M. Pardo. The jensen-shannon divergence. *Journal of the Franklin Institute*, 334(2):307–318, 1997.

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.

S. K. S. Ghasemipour, R. Zemel, and S. Gu. A divergence minimization perspective on imitation learning methods. In *Conference on Robot Learning*, pages 1259–1277, 2020.

T. Haarnoja, H. Tang, P. Abbeel, and S. Levine. Reinforcement learning with deep energy-based policies. In *International Conference on Machine Learning*, pages 1352–1361, unknown, 2017. PMLR, unknown.

R. Alsaleh and T. Sayed. Markov-game modeling of cyclist-pedestrian interactions in shared spaces: A multi-agent adversarial inverse reinforcement learning approach. *Transportation research part C: emerging technologies*, 128: 103191, 2021.

A. Mao, M. Mohri, and Y. Zhong. Cross-entropy loss functions: Theoretical analysis and applications. In *International Conference on Machine Learning*, pages 23803–23828. PMLR, 2023.

S. Geng, H. Nassif, C. Manzanares, M. Reppen, and R. Sircar. Deep pqr: Solving inverse reinforcement learning using anchor actions. In *International Conference on Machine Learning*, pages 3431–3441, unknown, 2020. PMLR, unknown.

D. Venuto. *Robust Adversarial Inverse Reinforcement Learning with Temporally Extended Actions*. McGill University (Canada), None, 2020.