Ranking Games Meet Generative World Models for Offline Inverse Reinforcement Learning

Peter Phan
pkphan@umass.edu

Giang Nguyen
gnnguyen@umass.edu

Abstract

The motivation for Offline inverse reinforcement (Offline IRL) learning is discern the underlying reward structure and environmental dynamics from a fixed dataset of previously collected experiences. For safety-sensitive applications this paradigm becomes a topic of interest because interacting with the environment may not be possible. Therefore accurate models of the world becomes crucial to avoid compounding errors in estimated rewards. With limited demonstration data of varying expertise, it also becomes important to be able to extrapolate from beyond these demonstrations in order to infer high-quality reward functions. We introduce a bi-level optimization approach for offline IRL which accounts for uncertainty in an estimated world model and uses a ranking loss to encourage learning from intent. We demonstrate the algorithm can match state-of-the-art offline IRL frameworks over the continuous control tasks in MuJoCo and different datasets in the D4RL benchmark.

1 Introduction

Reinforcement learning (RL) is a are of artificial intelligence concerned with enabling agents to learn optimal behavior by maximizing rewards through interaction with their environment. In many applications such as clinical decision making and autonomous driving, online trials for training are not possible due to ethical and safety concerns. Because of this, imitation learning from Learning from Demonstrations (LfD) becomes an important paradigm. Furthermore, often times ground-truth rewards functions are not available and are difficult to specify which motivates the technique of learning rewards from expert data via Inverse Reinforcement Learning (IRL). For safety-sensitive applications, we turn to offline IRL which prohibits access to interactions with the real-world environment. In this setting where data may be scarse and of varying levels of quality it becomes crucial to be able to understand and learn from the available demonstrations to their fullest capacity.

IRL also has the challenge of distribution shift which refers to the compounding error in training caused by reward functions and policies not generalizing well to the real-world environment. This is especially an issue in offline IRL since we only have access to a limited and fixed dataset. Thus there will only be limited coverage of the dynamics model of the real environment. We note that distribution shift remains an area of research that is not fully well understood. Similar works in offline reinforcement learning use conservative policy training to steer away from overestimating values in unseen states in hopes of addressing this issue. In this work we will be following the approach by [] to alleviate distribution shift in order to recover high-quality rewards. The main idea is to incorporate a estimated world model and perform policy updates that maximizes the likelihood of expert demonstrations while being subjected to an uncertainty penalty given by the world model.

A general weakness of IRL is its inability to learn beyond and outperform its demonstrator. It is desirable to seek a policy that can extrapolate beyond its expert because in many contexts we do not have access to – or in the case of offline IRL limited access to – optimal expert demonstrations. We were motivated by [] to show that learning from rankings can effectively tackle this problem. Rather

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

than trying to solely imitate the demonstrator's behavior such as in Behavioral Cloning (BC) and other IRL methods, rankings will allow our reward function to encode the information that explains the intentions behind a demonstrator's actions. In our approach we will use the ranking loss defined by [] to combine the modalities of learning from expert demonstrations and preferences. We will show that this ranking loss can give us performance that is closer to near-expert level performance than other imitation learning methods.

The other problem with IRL is having to solve a RL problem subroutine which will incur costly computation time. Our approach will utilize a more computationally efficient algorithm to solve the problem formulation. As a brief overview, we will avoid having to fully solve a policy optimization problem for each iterative reward estimate. Instead, our algorithm will cast the optimization procedure as a game between two players in which the policy and reward will alternate between their respective update steps. We will show through empirical results that this approach is capable of learning and converging in an efficient manner.

In summary, our project will combine the maximum likelihood (ML) formulation of offline IRL from [] and the ranking loss given by []. The ML formulation will allow us to take conservative policy improvement steps with respect to an estimated world model to alleviate distribution shift. We will follow a computationally and sample efficient game-like procedure to recover a high-quality reward function and its corresponding optimal policy under a conservative Markov decision process (MDP). Finally we will show that a ranking loss will lead to learned policy that outperforms many other imitation methods and match the state-of-the-art performance methods in the offline setting. Our experiments will be conducted on robotic control tasks in MuJoCo [] and collected datasets in D4RL [] benchmark.

2 Related Work

2.1 Inverse Reinforcement Learning

Unlike reinforcement learning (RL) - which relies on a readily available reward function associated with each action-state to model the MDP which restricted RL's performance to fully observable MDP, [5] Inverse Reinforcement Learning (IRL) infers the reward function the agent's optimizing through measurements of agent's behavior's overtime, under a variety of circumstances, and if possible, measurement of sensory input to the agents and/or model of the environment [4].

2.2 Offline IRL

IRL involves estimating a reward function and learning its corresponding optimal policy that best fits expert demonstrations. Maximum entropy IRL is a formulation of the IRL problem that gives theoretical performance guarantees on the recovered reward function []. Many sample efficient algorithm have been shown to solve the Maxent IRL formulation. The limitation with online IRL and many online RL methods is the necessity of performing online trials which may not always be possible.

Offline IRL attempts to recover a reward function from a fixed dataset of expert demonstrations. CLARE is a model-based offline IRL approach which performs IRL entirely in an estimated dynamics model. CLARE uses conservatism in its estimated rewards to ensure the state-action pair visitation that of the corresponding policy matches that of the distribution present in collected transition samples. The weakness of this approach is that CLARE's performance becomes reliant and sensitive to the quality of the transition samples and estimated dynamics model. As explored in [trex], matching visitation counts can not recover high-quality reward functions that extrapolates beyond the given demonstrations. [] proposes a model-based offline policy optimization algorithm (MOPO) which incorporates uncertainty to regularize the reward function. MOPO therefore learns a policy that is considered conservative because it stays away from exploring high uncertainty regions. In doing so MOPO can mitigate the distribution shift issue. Other works in offline IRL also incorporates conservatism into the estimated dynamics model to minimize the value function the training policy incurs [45, 47 from ml].

2.3 Learning from Preferences

Learning from preferences is important when high quality expert data is difficult to obtain. Preferences have also been shown to be able to provide better generalization to expert demonstrations. [trex] introduced a reward learning technique that is capable of significantly outperforming its suboptimal demonstrators. (Brown et al., 2019; 2020b;a; Chen et al., 2020) has shown that offline rankings over suboptimal data can be effective for learning reward functions. [rank game] presented a framework for imitation learning in the form of a game that utilizes rankings over suboptimal behaviors to aid policy learning. The paper used a novel ranking loss which can generate new rankings from a set of ranked behaviors or annotated rankings. They showed that their approach guarantees near-expert level performance for imitation learning.

3 Problem Formulation

We will first formalize the offline IRL problem from a maximum likelihood perspective. From this we will cast the bi-level optimization problem as a ranking game for imitation learning. Then we will propose using a ranking loss function as a surrogate objective function for the ML-IRL objective formulation. An overview of our approach can be described as follows:

Algorithm 1 Offline ML-IRL with Ranking loss

1: Initialize policy π_θ^0, reward function r_ϕ, expert data ρ^E
2: Train the world model $\widehat{P}$ on the transition dataset $\mathcal{D}$
3: Specify the penalty function $U(\cdot,\cdot)$ based on $\widehat{P}$
4: **for** $k = 0, 1, 2, \ldots, K-1$ **do**
5: Sample an expert trajectory $\tau_k^E := \{s_t, a_t\}_{t\geq 0}$
6: Sample an agent trajectory $\tau_k^A := \{s_t, a_t\}_{t\geq 0}$ from π_k on $\widehat{P}$
7: Generate rankings based on $\{(\tau_k^A \preceq \tau_k^E)\}$ and r_{ϕ_k}
8: Approximate the soft Q-function $Q_k(\cdot,\cdot)$ by $\widehat{Q}_k(\cdot,\cdot)$
9: Conservatively update policy $\pi_{k+1}(\cdot|s) \propto \exp(\widehat{Q}_k(\cdot,\cdot)), \forall \mathcal{S}$ according to (2b)
10: Do a regression step on the reward r_{ϕ_k} to satisfy the rankings using ranking loss L_k in (6)
11: **end for**

3.1 Offline Maximum Likelihood IRL

An MDP is defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \eta, r, \gamma)$, where $\mathcal{S}$ is the state space, A is the action space, P is the transition dynamic $P : (\mathcal{S} \times \mathcal{A} \times \mathcal{S})_t \to [0, 1]$, $\eta(\cdot)$ is the initial state distribution, r is the reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, and $\gamma \in (0, 1)$ is the discount factor. For any state-action pair under a policy π and a transition dynamics model P we can also define the visitation measure for a pair as $d_P^\pi(s, a) := (1-\gamma)\pi(a|s) \sum_{t=0}^{\infty} \gamma^t P^\pi(s_t = s|s_0 \sim \eta)$.

MaxEnt-IRL is a related IRL formulation that recovers a reward function by assigning high rewards to an expert policy and low rewards to every other policy. Let $\tau^E := \{(s_t, a_t)\}_{t=0}^{\infty}$ be an expert trajectory sampled from an expert policy π^E and let similarly τ^A be the trajectory from a RL agent with policy π. Then the Max-Ent-IRL formulation can be written as:

$$\max_r \min_\pi \left\{ \mathbb{E}_{\tau^E \sim \pi^E} \left[\sum_{t=0}^{\infty} \gamma^t \cdot r(s_t, a_t) \right] - \mathbb{E}_{\tau^A \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t \cdot r(s_t, a_t) \right] - H(\pi) \right\}, \tag{1}$$

where $H(\pi) := \mathbb{E}[\sum_{t=0}^{\infty} -\gamma^t \log \pi(a_t|s_t)]$ is the causal entropy term of the policy π. Note that MaxEnt-IRL has to repeatedly solve a policy optimization problem in the inner minimization loop for each reward function in the outer reward maximization.

The ML formulation of offline IRL assumes a trained estimated world model $\widehat{P}(s'|s, a)$ for any $s', s \in \mathcal{S}$ and $a \in \mathcal{A}$. This estimated world model is trained from a transition dataset $\mathcal{D} := \{(s, a, s')\}$. Since we will not have access to interactions to the ground-truth dynamics model P, the estimated model $\widehat{P}$ will be used instead. Given the proposed estimated world model $\widehat{P}$, [ml] proposed the

following model-based ML perspective to offline IRL:

$$\max_{\theta} L(\theta) := \mathbb{E}_{\tau^E \sim (\eta, \pi^E, P)} \left[\sum_{t=0}^{\infty} \gamma^t \log \pi_\theta(a_t|s_t) \right] \tag{2a}$$

$$\text{s.t.} \quad \pi_\theta := \arg\max_{\pi} \mathbb{E}_{\tau^A (\eta, \pi, \widehat{P})} \left[\sum_{t=0}^{\infty} \gamma^t \Big(r(s_t, a_t; \theta) + U(s_t, a_t) + \mathcal{H}(\pi(\cdot|s_t)) \Big) \right], \tag{2b}$$

where $\mathcal{H}(\pi(\cdot|s)) := \sum_{a \in \mathcal{A}} -\pi(a|s) \log \pi(a|s)$ is the entropy of the distribution $\pi(s|s)$; $U(\cdot, \cdot)$ is the uncertainty function of the estimated world model $\widehat{P}(\cdot|s, a)$ for any given state-action pair (s, a).

The formulation is a bi-level optimization problem. The upper-level problem optimizes the reward function $r(\cdot, \cdot; \theta)$ such that the corresponding optimal policy maximizes the log-likelihood of observed expert trajectories taken from the ground-truth dynamics model P. The lower-level problem describes the unique solution to a conservative MDP. It assumes a fixed reward function $r(\cdot, \cdot; \theta)$ and defines the optimal policy to be a policy that maximizes the estimated reward over the estimated dynamics model $\widehat{P}$. Note that we also include the uncertainty penalty to the standard MDP which is what makes this MDP conservative. As a result, the optimal policy under this MDP will not explore uncertain regions of the state-action space that the transition dataset used to train $\widehat{P}$ did not cover. The motivation behind this formulation relies on trying to explain the observed expert behavior within the limited knowledge of the estimated model of the world $\widehat{P}$.

[] provides a statistical guarantee for a surrogate objective function $\widehat{L}(\theta)$ to the objective function $L(\theta)$ in 2a. Essentially, the performance gap between the surrogate and objective function is dependent on the coverage of the transition dataset $\mathcal{D}$ used to construct the estimated world model $\widehat{P}$. The importance piece from this analysis that we would like to highlight for our paper is that under the performance guarantees, the reward parameters can be updated as $\theta_{k+1} = \theta_k + \alpha g_k$. Here α is some hyperparameter coefficient and g_k is a stochastic gradient estimator of $\nabla \widehat{L}(\theta_k)$ defined as:

$$g_k := h(\theta_k; \tau_k^E) - h(\theta_k; \tau_k^A), \tag{3}$$

where $h(\theta, \tau) := \sum_{t=0}^{\infty} \gamma^t \nabla_\theta r(s_t, a_t; \theta)$ is the cumulative reward given by a reward function with parameters θ over a trajectory τ. Since this reward update from offline ML-IRL boils down to the reward difference between expert and agent trajectories, it will act as our motivation for using a ranking loss as a surrogate objective function.

3.2 Two-Player Ranking Game Formulation

[rank] casts the bi-level optimization problem between a policy and reward learner as a ranking game between two players. The formulation is based on the Stackelberg game (Başar & Olsder, 1998) between two players, where one player is set as the leader and the other as the follower. The leader will optimize its objective with the assumption that the follower will respond with the optimization to their own objective. Explicitly, given a leader player A and a follower player B with parameters θ_A, θ_B with objectives $\mathcal{L}_A(\theta_A, \theta_B)$ and $\mathcal{L}_B(\theta_A, \theta_B)$, the Stackelberg game solves the following bi-level optimization:

$$\min_{\theta_A} \mathcal{L}_A(\theta_A, \theta_B) \tag{4a}$$

$$\text{s.t.} \quad \theta_B^*(\theta_A) = \arg\min_{\theta} \mathcal{L}_B(\theta_A, \theta). \tag{4b}$$

The authors of [rank] propose two class of algorithms that follows this game formulation, Policy as leader (PAL) and Reward as leader (RAL). For our implementation we will be sticking to the PAL algorithm. However, we note that [rank] provides a thorough comparison between the two algorithm classes. Particularly, they showed that PAL adapts faster to changes in demonstrations' intent and RAL adapts faster to changes in environment dynamics.

In the case for offline ML-IRL we will adopt the class of ranking loss function L_k from [rank] as a surrogate to 2a. The loss function L_k tries to induce a performance gap of $k \in [0, R_{max}]$ for all behavior preferences in a dataset of behavioral pairs $\mathcal{D} = \{(\rho^{\pi^i}, \rho^{\pi^j})\}$. In the offline ML-IRL setting, we will assume to have no access to a dataset of behavior rankings. Instead, we will use the behavior denoted by the set of trajectories produced by a learning agent policy ρ^{π^A} and the behavior of expert

demonstrations ρ^{π^E}. For legibility we will denote these behaviors as ρ^A and ρ^E respectively. ρ^A will be described as the set of trajectories that is generated from rolling out the learning agent policy π on the estimated world model $\hat{P}$. ρ^E will be our fixed demonstrations dataset for ML-IRL. We will prespecify the rankings to be $\rho^A \preceq \rho^E$. Then the ranking loss can be defined as the following regression loss:

$$L_k(\mathcal{D}, r) = \mathbb{E}_{(\rho^A, \rho^E)\sim\mathcal{D}}\left[\mathbb{E}_{s,a\sim\rho^A}[(r(s,a)-0)^2] + \mathbb{E}_{s,a\sim\rho^E}[(r(s,a)-k)^2]\right] \tag{5}$$

3.2.1 Automatically Generating Rankings

We will use a form of data augmentation to generate further rankings between our agent and expert trajectories. Specifically, we will use mixup (Zhang et al,. 2017) regularization in the trajectory space. Mixup has been shown in to improve generalization and adversarial robustness in reward learning []. Furthermore, these rankings represent the true underlying rankings of behaviors in the case where the expert reward function is linear in observations []. To generate these rankings, we will interpolate trajectories between trajectories of two differently ranked behaviors. In our case, we have the ranking $\rho^A \preceq \rho^E$ and will generate the additional rankings: $\rho^A = \rho_{\lambda_0,AE} \preceq \rho_{\lambda_1,AE} \preceq \rho_{\lambda_2,AE} \dots \preceq \rho_{\lambda_P,AE} \preceq \rho_{\lambda_{P+1},AE} = \rho^E$ where $0 = \lambda_0 < \lambda_1 < \dots \lambda_P < 1 = \lambda_{P+1}$. Trajectories from a behavior $\rho_{\lambda_p,AE}$ is given as the following interpolation: $\tau_{\lambda_P,AE} = \lambda_p\tau^A + (1-\lambda_p)\tau^E$. Then the corresponding target reward for this trajectory is given as $k_p = \lambda_p\mathbf{0} + (1-\lambda_p)k\mathbf{1}$. For instance, if our agent trajectory is $\tau^A = [0,1]^\top$, our expert trajectory is $\tau^E = [2,1]^\top$, and $\lambda_p = 0.2$, then we will have $\tau_{\lambda_p,AE} = [1.6,1]^\top$ with a target reward of $k_p = 0.8k$. Then for inducing a performance gap between $p+2$ consecutive rankings, the final ranking loss function for our approach is:

$$SL_k(\mathcal{D}, r) = \frac{1}{p+2}\sum_{i=0}^{p+1}\mathbb{E}_{s\sim\rho^{AE}_{\lambda_i}(s,a)}[(r(s,a)-k_i)^2] \tag{6}$$

In our implementation we set the regression targets k_i apriori according to the exp-β family of parameterized mappings. This mapping entails a rate of increase in return exponential in $\alpha(\frac{dk_\alpha}{d\alpha} \propto e^{\beta\alpha})$, where β is a temperature parameter.

4 Methodology

We compare our approach to other imitation learning approaches on MuJoCo benchmarks having continuous state and action spaces. At first we considered using the ranking loss from 6 as a regularizer term to the offline ML objective function in 2a. Then we used the ranking loss as a standalone objective function, replacing the original objective formulation described in [ml]. We performed a comparison between these two approaches with the baseline offline ML approach. Then we performed experiments to show that our approach also improves with estimated world models that have bigger coverage as claimed in [].

We test the performance of our approach on two robotics training tasks from the MuJoCo simulator, as well as datasets in the D4RL benchmarks. The environments we tested were halfcheetah and hopper. The datasets we used were medium-replay and medium-expert. For each task we had separate estimated world model trained on medium-replay and medium-expert dataset, with the medium-expert dataset having more converge of the world dynamics. In all our experiments, access to the ground-truth reward or interactions with the environment were not available besides for testing. We ran each algorithm for 800 episodes and averaged the runs over at least 4 different seeds. Due to computational limitations, all ranking related results were generated by us while other numbers were taken from other works to compare. In addition, we reproduced the results from [ml] for their halfcheetah and hopper experiments using offline ML-IRL. Each training run for one agent took between 7-9 hours to complete.

In our implementation we used the provided trained estimated world model by [ml]. The estimated dynamics models $\widehat{P}$ were constructed as an ensemble of 5 Gaussian distributions, each parameterized by neural networks as $\widehat{P}^i_{\phi,\varphi}(s_{t+1}|s_t,a_t) = \mathcal{N}(\mu^i_\phi(s_t,a_t), \Sigma^i_\varphi(s_t,a_t))$. Each Gaussian was was trained via likelihood maximization over the transition dataset. Then the uncertainty function is

constructed as the aleatoric uncertainty as $U(s,a) = -\max_{i=1,\ldots,N} ||\Sigma^i_\varphi(s,a)||_{\mathrm{F}}$. For the offline RL optimization, the implementation of MOPO from [ml 25] was followed. The reward function was parameterized as a three-layer neural network which took in a state-action pair as input, has hidden layers of size 256, a batch normalization layer, ReLU activation, and reward output clamped between -10 and 10. The reward model sampled the agent trajectories on the world model and expert demonstration from the dataset to compute the ranking loss according to (6).

5 Experiments and Result

First we showed that our approach still coheres to the correlation between performance and coverage extent of the estimated that was shown in [ml].

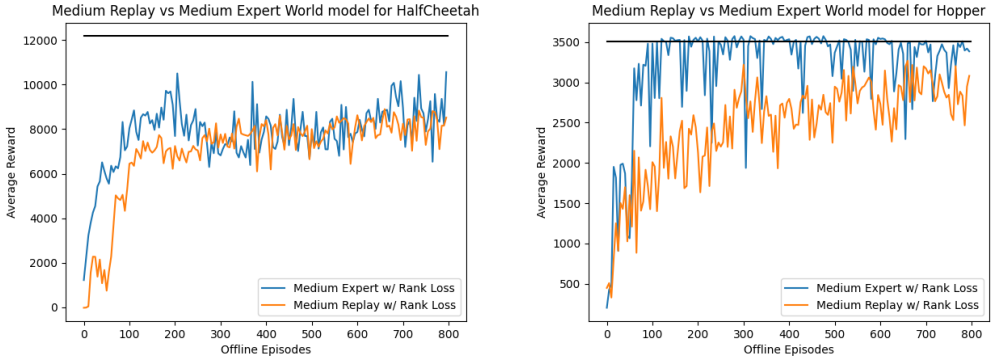

Figure 1: Comparison between using a world dynamics model trained using transitions from the medium-replay dataset versus medium-expert dataset. In both cases, using the medium-expert was more successful than using medium-replay which correlates the the theoretical analysis that says our surrogate objective function should be nearer to the original objective with the more coverage of the environment the estimated dynamics model has.

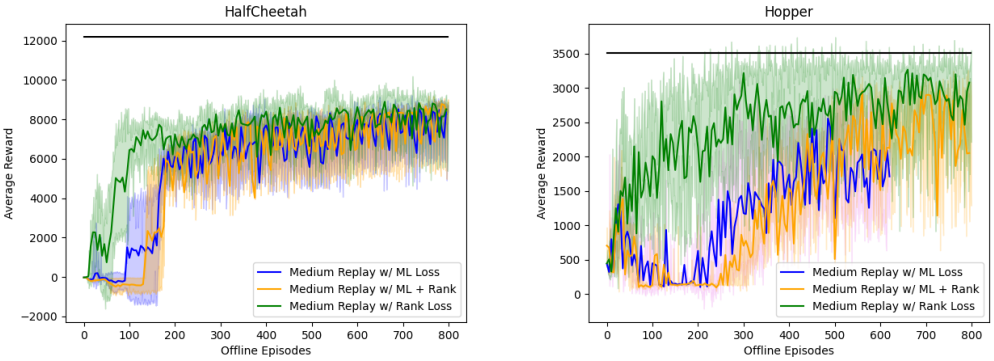

Figure 2: Comparison between using the "ML Loss" formulation as originally described in [ml], using "ML + Rank" which is the previous mentions loss formulation with a ranking loss described in 6 as a regularizer, and "Rank Loss" which is using the ranking loss function from 6 by itself. In both cases of HalfCheetah and Hopper, using rank loss as its standalone loss led to the fastest convergence and highest peak average rewards.

6 Discussion

While we had intended to implement both PAL and RAL separately to evaluate their performance, we were only able to complete PAL due to limitations in time and computing resources. However, we

remain keen to explore the integration of both PAL and RAL in our optimization game in the future. We also would like to try different generative models to see how our model will perform. Currently our Offline IRL implementation still contains an inner RL loop which affects the runtime negatively. However, with the emergence of new techniques aimed at eliminating the inner RL loop inside IRL, such as [7], we are optimistic about the potential performance enhancements in the future.

7 Contribution

This project is a collaborative research effort between Peter Phan and Giang Nguyen. Peter was responsible for the implementation of the algorithm and running the experiments. Both Peter and Giang were responsible for the write-up.

References

[1] Daniel S. Brown, Wonjoon Goo, Prabhat Nagarajan, and Scott Niekum. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In *International Conference on Machine Learning*, 2019. URL https://api.semanticscholar.org/CorpusID:119111734.

[2] Justin Fu, Aviral Kumar, Ofir Nachum, G. Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *ArXiv*, abs/2004.07219, 2020. URL https://api.semanticscholar.org/CorpusID:215827910.

[3] Frédérick Garcia and Emmanuel Rachelson. *Markov Decision Processes*, chapter 1, pages 1–38. John Wiley Sons, Ltd, 2013. ISBN 9781118557426. doi: https://doi.org/10.1002/9781118557426.ch1. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118557426.ch1.

[4] Andrew Y. Ng and Stuart J. Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, page 663–670, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc. ISBN 1558607072.

[5] Stuart Russell. Learning agents for uncertain environments (extended abstract). In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, COLT' 98, page 101–103, New York, NY, USA, 1998. Association for Computing Machinery. ISBN 1581130570. doi: 10.1145/279943.279964. URL https://doi.org/10.1145/279943.279964.

[6] Harshit S. Sikchi, Akanksha Saran, Wonjoon Goo, and Scott Niekum. A ranking game for imitation learning. *ArXiv*, abs/2202.03481, 2022. URL https://api.semanticscholar.org/CorpusID:246652580.

[7] Gokul Swamy, David J. Wu, Sanjiban Choudhury, J. Andrew Bagnell, and Zhiwei Steven Wu. Inverse reinforcement learning without reinforcement learning. *ArXiv*, abs/2303.14623, 2023. URL https://api.semanticscholar.org/CorpusID:257767360.

[8] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y. Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *ArXiv*, abs/2005.13239, 2020. URL https://api.semanticscholar.org/CorpusID:218900501.

[9] Siliang Zeng, Chenliang Li, Alfredo Garcia, and Min-Fong Hong. When demonstrations meet generative world models: A maximum likelihood framework for offline inverse reinforcement learning. In *Neural Information Processing Systems*, 2023. URL https://api.semanticscholar.org/CorpusID:256868343.

A Appendix