Adaptive and Efficient Models for Intent Recognition

Tejus Gupta

CMU-RI-TR-21-51

July 12, 2021

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:
Katia Sycara, *chair*
Changliu Liu
Wenhao Luo

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics.

Copyright © 2021 Tejus Gupta. All rights reserved.

To Mom, Dad and Yash.

Abstract

Assistive robots should have the ability to understand the intent of humans, predict their behavior, and plan to provide anticipatory assistance in complex real-life environments. In this thesis, we present adaptive and efficient algorithms for recognizing human intent.

We develop adaptive models for human intent recognition in a simulated search and rescue scenario. Humans vary widely in their behavior style due to different preferences, initial beliefs, internal world models, and planning mechanisms. A generic (non-adaptive) prediction model, therefore, has limited utility in this setting. Our adaptive model can recognize a rescuer's behavior patterns online and make better predictions. We show that adaptive models trained on a wide variety of simulated planning-based agents can transfer to humans and outperform generic models trained on limited human data.

We also present an efficient inverse reinforcement learning algorithm, called f-IRL, which directly optimizes a parameterized reward function to match the demonstrator's state distribution. We show that f-IRL can efficiently learn the demonstrator's intent - it can learn to imitate control policies from just a single demonstration. In addition, we show that the learned reward can be used to transfer policies to different dynamics.

Acknowledgments

I would like to express my gratitude to my family and many friends who have helped me write this thesis. I would like to thank my advisor Katia, for her invaluable guidance especially. I am also thankful to my committee members - Changliu and Wenhao, and all my collaborators - Vidhi, Rohit, Huao, Ini, Dana, Michael, Sophie, Harshit, Yufei, Tianwei, Lisa, and Ben.

Funding

This work was supported by DARPA Award HR001120C0036. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Defense Advanced Research Projects Agency (DARPA).

Contents

When this dissertation is viewed as a PDF, the page header is a link to this Table of Contents.

List of Figures

List of Tables

Chapter 1

Introduction

Intent recognition is a crucial component in shared autonomy and assistance systems. In this thesis, we focus on the adaptivity and efficiency of intent recognition algorithms. Adaptivity allows our algorithms to adapt to specific humans and make better predictions as the system observes the humans' decisions online. Efficiency allows our algorithms to identify the demonstrator's intent using a few demonstrations.

We consider a simulated search and rescue scenario for studying adaptive intent prediction algorithms. Human participants are tasked with searching for and rescuing victims in a simulated setting. We consider the task of predicting the rescuer's triage and navigation decisions in this setting. The rescuers make these decisions to optimize their task performance, and so the predictive model needs to reason about the human's theory of mind and utility of different decisions over long time horizons to make these predictions. These participants vary widely in their preferences, behavior style, initial beliefs, internal world model, etc. A generic prediction model, therefore, has limited utility in this setting. Our adaptive algorithms can recognize a specific rescuer's traits and use them to make better predictions in future missions. For example, it can recognize that a specific human has poor memory and so often forgets information from earlier parts of the mission. The prediction model can then condition future predictions on these learned characteristics of the human.

We train the adaptive models completely in simulation by creating a wide variety of planning-based agents. These planning-based agents are designed to mimic human behavior, and we, therefore, called them faux-human agents. We show that adaptive

models trained on these faux-humans generalize to human trajectories and can do better than models trained on limited human data. These predictions help us understand the rescuer's navigation and triage strategy and would help provide anticipatory assistance to the human.

In the second part of this thesis, we consider the inverse reinforcement learning (IRL) problem and propose an efficient method based on state-marginal matching. The main theoretical result of our work was the derivative of any general f-divergence between the expert and policy state distributions w.r.t. the reward parameters. We demonstrated that the resulting algorithm, called f-IRL, improves upon sample efficiency of previous reward-inference and imitation learning methods. This work enables robots to learn humans' intent and imitate them using just a single demonstration. This also opens the door to provide new types of expert supervision, like which states are unsafe and should be avoided. These capabilities are an essential step towards enabling socially intelligent robots.

Chapter 2

Background

Intent recognition and behavior prediction are essential components for any socially-intelligent robot. Social robots need to be able to understand the intent of humans, predict their behavior, and plan to collaborate with them in complex real-life environments. In this chapter, we discuss various tools used for these tasks. In particular, we review algorithms for theory of mind, trajectory prediction, inverse reinforcement learning, and learning adaptive models (meta-learning). We also describe some applications of these methods for robotic assistance and shared autonomy systems.

2.1 Theory of Mind

Methods based on Theory of Mind (ToM) framework reason in joint belief-intent space to understand the demonstrator's behavior. These methods can attribute a person's suboptimal behavior to false beliefs. The Sally-Anne Test [67] is a popular psychological test used to measure humans' ability to use a theory of mind.

Baker et al. [8] model the agent's decision-making as optimal control over a POMDP, and design inference algorithms that share a person's ability to identify an agent's beliefs and desires from its behavior. Their experiments show that this algorithm better matches the human's predicted belief and desires compared to algorithms that assume that the agent has full observability of the environment or that the agent does not update its beliefs.

Several works have built upon Baker et al. [8] by proposing better models for human

behavior. Evans and Goodman [13], Evans et al. [14] model bounded rationality and can reason about the (possibly faulty) planning mechanism of the agent. Recently, Reddy et al. [53] proposed TOM models that also reason about human's internal model of the world. However, most of these results were demonstrated in smaller settings, and it is challenging to scale Bayesian inference using ToM models to large environments such as ours.

The Machine Theory of Mind [49] model learns to infer agent's intents and latent characteristics directly from data via meta-learning. It first observes the trajectories of a large number of agents to compute its prior on an agent's behavior. It then predicts a specific agent's intentions by computing the posterior conditioned on a (small number of) agent's trajectories. This inference is made directly using the TOM-Net. TOM-Net comprises three modules: a character net, a mental state net, and a prediction net. The character net summarizes the agent's prior trajectories into an embedding (encodes the prior). The mental state net summarizes the agent's current trajectory. The prediction net mimics Bayesian inference to predict subsequent agent behavior. Rabinowitz et al. [49] choose to make models for direct prediction instead of first assuming a generative model of an agent's behavior and inverting it. This trades off data efficiency for expressibility - the TOM-Net was trained on thousands of agent trajectories. Moreover, this approach requires supervision to predict the beliefs of the agents and to recognize false beliefs.

2.2 Inverse Reinforcement Learning

Inverse Reinforcement Learning (IRL) methods aim to learn the reward function from expert demonstrations. The reward can be used to train a policy (apprenticeship learning), or to understand the demonstrator's intent and asist them (shared autonomy).

Batch IRL methods [1, 43, 55] obtain a policy by learning a reward function from sampled trajectories of an expert policy. MaxEntIRL [72] learns a stationary reward by maximizing the likelihood of expert trajectories, i.e., it minimizes forward KL divergence in trajectory space under the maximum entropy RL framework. Similar to MaxEntIRL, Deep MaxEntIRL [68] and GCL [16] optimize the forward KL divergence in trajectory space. A recent work, EBIL [37], optimizes the reverse KL divergence in

the trajectory space by treating the expert state-action marginal as an energy-based model. Another recent method, RED [66], uses support estimation on the expert data to extract a fixed reward, instead of trying to minimize a f-divergence between the agent and expert distribution.

Finn et al. [15] showed that GANs [22] with a special structure in the discriminator can be used to learn the reward. Adversarial IRL [18, 48] follows this direction, and proposes a practical method based on the adversarial reward learning formulation. This method can be shown to minimize the reverse KL divergence between the expert and policy state distribution.

Another direction of IRL research has formulated IRL as a bilevel optimization problem where we aim to find a policy whose performance is close to the expert's on the unknown reward function. This problem can be written as $IRL(\tau_E) = \arg\min_{r \in R} \min_{\pi \in \Pi} E_{\tau_E}[r(s,a)] - E_{\pi}[r(s,a)]$. Abbeel and Ng [1] first introduced this formulation and proposed an efficient solution in their seminal paper. Maximum margin planning [51] recognizes this problem as being equivalent to a structured maximum margin problem, and propose an algorithm based on subgradient descent. This method is applicable to non-linear reward parameterizations as well.

Online IRL methods can recognize the demonstrator's intent from partial trajectories. In a Bayesian framework [50], the actions of the expert acts as a evidence for updating a prior on reward functions. This inference can be performed using Markov chain Monte Carlo (MCMC) methods. In fact, the original IRL objective by [43] can be shown as MAP estimate with a Laplacian prior. This framework can deal with suboptimal experts and limited data, and can incorporate prior information as well. Their key theoretical insight is that the Markov Chain will mix rapidly for a uniform prior since the likelihood function is pseudo-log-concave. Their experiments show that domain knowledge about a problem can be incorporated into the IRL formulation as an informative prior, and that their method is more sample-efficient than Ng et al. [43]'s algorithm.

It is often difficult for humans to demonstrate behaviors for high-dimensional dynamical systems, and a preference-based approach may be practical in such scenarios. In Sadigh et al. [56]'s work, the system actively generates two candidate feasible trajectories for the user to rank, and updates its posterior distribution on rewards. The candidate trajectories are optimized so as to maximize the expected volume

removed from the hypothesis space.

Sadigh et al. [56] use the maximum entropy RL framework. So the probability of the user preferring τ_A over τ_B is $\frac{exp(r(\tau_A))}{exp(r(\tau_A))+exp(r(\tau_B))}$. We can update r based on user's preference via Bayes rule. The trajectories τ_A and τ_B are chosen so that volume reduced under either of user's preference is maximized. In practise, this optimization is done by sampling reward functions from current $p(w)$ and optimizing query so that a large number of these reward functions can be discarded. Note that unnormalized $p(w)$ is maintained implicitly as a multiplication of updates based on previous user preferences (prior is uniform), and reward functions can be efficiently sampled via MCMC since $p(w)$ is log-concave. Their experiments confirm their hypothesis that active query synthesis leads to efficient and accurate reward inference.

Demonstrated data can often be explained more efficiently locally than by a global reward function. The encoding of such behavior using a global intention model requires a reward structure with large number of redundant state-action based rewards. Instead, we can assume that the global task can be decomposed into smaller subtasks that require considerably less modeling effort. These subgoals can be then be used as building blocks to explain complex behavior.

Bayesian Nonparametric Inverse Reinforcement Learning (BNIRL) [39] models the generative process of decisions as subgoal selection (using Chinese Restaurant Process prior), and optimizing for selected subgoal in each segment of trajectory. The inverse model is able to segment trajectories and identify subgoals for each trajectory. The hidden variables in this generative process are the partition assigment z_i for each state s_i in demonstration, and subgoal for each partition g_{z_i}. They use Gibbs sampling to sample from $p(z, g|s_{1:N})$.

Bayesian Multitask Inverse Reinforcement Learning [12] proposed a hierarchical prior over reward functions to account for the fact that different trajectories in a data set could reflect different behavioral intentions, e.g., because they were generated by different domain experts.

Inverse Reinforcement Learning via Nonparametric Subgoal Modeling [61] improves upon limitations of BNIRL, e.g., it is restricted to pure subgoal extraction and does not inherently provide a reasonable mechanism to generalize the expert behavior based on the inferred subgoals.

Apprenticeship Learning About Multiple Intentions [6] assumes that expert tra-

jectories are generated from a set of K reward functions, and uses the EM algorithm to cluster the trajectories and identify the reward functions.

Exploring Hierarchy-Aware Inverse Reinforcement Learning [10] models decision-making as choosing from a (pre-defined) set of options, and does inference over this model using MCMC. This method assumes that we have a known small set of options, since considering the set of all possible options is exponentially large.

2.3 Trajectory Prediction

Direct trajectory prediction with neural sequence models [21] [40] have shown promising results. In particular, the transformer [63] is a popular attention mechanism for sequence modeling tasks and achieves state-of-the-art results on various benchmarks. These models are most commonly used for tasks with a large amount of available data, such as pedestrian and vehicle trajectory forecasting. It is challenging to train these models with a few trajectories for a new task.

Recently Shah et al. [58] investigated the utility of learning the human's model of decision-making compared to assuming a model such as Boltzmann optimal. Their results corroborate the results by Armstrong et al. [4] which states a No Free Lunch theorem for intent recognition for an agent with an unknown decision-making model. This implies that we need to bound the hypothesis space of human models to enable sample-efficient learning. Our work can be seen as a way of constructing this hypothesis space.

2.4 Adaptive Models

Meta-learning, "learning to learn" is a common framework for training models that can adapt quickly to a new input distribution. This could be used, for example, to adapt the model to a new task or to a new human. Meta-learning involves learning a policy modification/learning strategy.

One research direction is metric-based meta-learning which learns a similarity metric between tasks and uses the network most suited to the test task. Prototypical Networks [60] is a representative method in this class of algorithms.

Another research direction is based on learning parameters of an optimization method. For example, MAML [17] trains the parameters of the model such that the model can be easily fine-tuned using a small amount of data from a new task. Recent work such as Reptile [44] has focused on improving the stability of optimization-based meta-learning algorithms.

Learning a Prior over Intent via Meta-Inverse Reinforcement Learning [69] uses MAML to learn a reward prior. This reward prior can be quickly updated using MaxEnt IRL to learn the user's intent from just a few demonstrations.

Scalable Meta Inverse Reinforcement Learning through Context-Conditional Policies (SMILe) [19] assumes access to a context variable describing the environment and demonstrators, along with expert demonstrations. During meta-training, they train context-conditioned discriminators and policies using off-policy learning (similar to AIRL). In their experiments, they assume that the expert demonstrations themselves form the context, and train an encoder to generate the context from trajectories.

Meta-Inverse Reinforcement Learning with Probabilistic Context Variables [70] is a similar work that learns a trajectory to context encoder and context-conditioned reward functions.

Chapter 3

Adaptive Models for Human Intent Prediction

3.1 Introduction

People exhibit a wide variety of preferences, beliefs, and styles of behavior. Therefore, a generic model of human behavior has limited utility for intent recognition. This motivates our goal of learning intent recognition models that can adapt online upon observing a specific person. We model the individual differences in people completing the same task so that our model can learn to recognize them online and improve its predictions.

Intent prediction is a crucial component for any human-robot interaction system. The robot can use the predicted human intent to provide anticipatory assistance or collaborate with humans. Liu et al. [35] showed that effective intent prediction could reduce joint task completion rate, are preferable to humans, and are perceived to be collaborative. Even though Bayesian goal inference has a significant error rate of 25-40%, humans found it almost as helpful as robots with knowledge of ground-truth intent. This suggests that an attempt to predict and adapt to human goals—even under limited accuracy— can significantly affect the team's objective performance and the human's perception of the robot.

Previous work has shown the utility of intent prediction in several collaborative

and assistance tasks. For example, Nikolaidis et al. [45] proposed a framework for learning human intents from demonstrations that enables the robot to compute a robust policy for a collaborative task with a human. Similar work has shown the utility of intent prediction for shared autonomy systems [28, 38, 52].

However, a significant limitation of previous intent recognition methods is learning or assuming a generic model of human behavior. For example, Inverse Reinforcement Learning (IRL) methods assume a human-decision model and invert this model to infer the demonstrator's intent from its actions. The most popular model is the Boltzmann optimal model [71] which assumes that the human's policy is the maximum entropy policy for its internal reward. Recent work has proposed more sophisticated modes of decision-making that account for limited planning ability, false beliefs, and incorrect internal models of the world. However, these models remain uniform across people. They do not model the individual differences between people.

In this work, we demonstrate the utility of adaptive models for human intent recognition. As the model observes a person's behavior, it recognizes their personal traits in an online manner and specializes in its predictions. For example, an adaptive model can recognize an aggressive driver and predict their behavior accordingly in future timesteps. Such models are necessary and beneficial for accurate intent recognition.

Our approach for training adaptive models uses prediction models with context variables and is motivated by similar work for transferring control policies from simulation to the real world. This context variable encodes the individual characteristics of a person and allows the predictor to adapt to the given human appropriately. Previous work has used meta-learning to learn a context predictor using simulated data. For example, Rapid Motor Adaptation [31] uses an adaptation network to predict the physical characteristics of the terrain, such as friction. Our critical insight is to create a variety of simulated planning-based agents and use them to simulate the diversity of real humans. These planning-based agents have a variety of planning horizons, tolerance to risk, high-level goals, belief update mechanisms, etc. We use these simulated agents to train our predictions model and context predictors.

We show that adaptation methods can be prone to issues due to distribution shift between the planning-based agents and real humans. Since it is challenging (impossible?) to model the variety of real humans accurately, this distribution shift will

always exist. We show that certain classes of optimization-based context-predictors are more robust to this distribution shift.

We demonstrate our results on human subjects in a simulated search and rescue task. The participants are tasked with searching for and rescuing victims in a 3D simulation of a realistic office building with several rooms and corridors where the disaster has taken place. Before starting the mission, we provide the rescuer with the building's original floor plan to plan their route. During the mission, they may encounter several changes due to the damage, such as wall collapse. Some critically injured victims take more time to triage and may expire sooner than other victims. This task requires a good navigation strategy to explore the map efficiently and make triage decisions about whether to come back to rescue low-priority victims later on in the mission. We consider the task of predicting the human rescuer's navigation and victim triage decisions.

We trained our adaptive models to utilize information from one mission of a person to make predictions at later missions of the same person. We empirically demonstrate that our models adapt their predictions online to make more accurate predictions. The second mission has different map perturbations, victim locations, and initial player beliefs, and therefore the models can not memorize the rescuer's actions and need to generalize.

In summary, we contribute an empirical analysis of different approaches for training adaptive intent prediction models.

3.2 Method

Our adaptation models are trained using meta-learning on trajectories from planning-based agents. We now describe both the design of planning-based agents and the meta-learning algorithms we employ.

3.2.1 Faux-human Agents

We create a wide variety of planning-based agents that are designed to quickly search the map and triage as many victims as possible. We formulate the search and rescue task as a partially observable Markov decision process, and create online planners

for completing this task. These planning-based agents are designed to simulate the variations in human behavior, and we, therefore, refer to them as 'faux-human' agents.

However, humans do not always act rationally, and therefore, these naive faux-human agents' behavior may be quite different from human behavior. We ameliorate this issue by collecting a small set of pilot human data and incorporating the rescuers' observed biases into the faux-human agents. Some of the observed biases in the rescuer's decision-making were: (1) choosing subgoals in a soft-optimal manner (modeled using Boltzmann distribution), (2) planning over room sequences instead of low-level actions, and (3) using greedy frontier-based search for short-horizon combined with long term planning aligned with the cliques in the graph representation of areas of a map.

Moreover, we model the individual differences between rescuers to capture the heterogeneity of human behavior. We create a variety of simulated agents based on online planning with different planning horizons, tolerance to risk, triage strategy preferences, and belief state accuracy. We thus obtain a rich and diverse set of faux-human agents that incorporate human biases while optimizing for the task objective.

3.2.2 Adaptive Models

We train an adaptive intent prediction model $p(x_{t+1}|x_t, c)$, where x_t is the current mission state and c is the context variable. The context variable captures the individual traits of each rescuer, such as their planning horizon, high-level strategy, tolerance to risk, etc. This model is trained using supervised learning using privileged information about the current agent's context variable. Knowledge of the context variable allows the predictor to adapt to the given agent appropriately. We randomly sample simulated faux-human trajectories to train this model.

We empirically compare different meta-learning algorithms for training adaptive intent prediction models. These methods primarily differ in how they obtain the context variable based on a person's past decisions. We broadly classify these methods into three classes and discuss them below.

A1. Context Predictor: We can estimate the context variable c using an a context predictor model $\phi(c|\tau)$. The context predictor module uses the history of the

human's decisions (states $\{x_0, ..., x_{t-1}\}$ and actions $\{a_0, ..., a_{t-1}\}$) to output $\hat{c}$ which is an estimate of the true context variable c.

We train the context predictor on randomly sampled faux-human agents. The context-predictor's parameters are updated to minimize a regression loss with respect to the ground-truth c. Since we trained exclusively on simulation, we have access to the ground-truth context variable.

A2. End-to-End Sequence Models: We can train recurrent intent predictors on a diverse set of faux-human trajectories to automatically learn adaptive behavior. These models can learn to adjust their predictions during testing to recognize the specific person's behavior style and adapt their intents accordingly, i.e., by implementing a learning algorithm internally.

A3. Online Context Optimization: We can directly search for the best context variable using the human's partial trajectory τ instead of directly predicting it. We use online Bayesian optimization to update our posterior distribution over the human context variable as we observe more data. In this case, our prediction for the person's intent is given by a weighted average of predictions using context variables. The weights are the current distribution over the context variables.

3.3 Study Design

Participants completed an approximately three-hour experiment in which they search for victims and rescue them in a Minecraft task environment. This simulated environment consisted of a structurally damaged office building with victims trapped inside. The building's map and the participant's screen are shown in Fig 3.1.

The building contains 26 area segments consisting of corridors, rooms, and elevators. The initial building layout and segment connectivity were changed by perturbations such as collapses, wall openings, and sporadic fires. The participants were given a blueprint of the building depicting its layout prior to the collapse and needed to work around blockages and obstacles in their search. There are 20 injured victims inside the building who need to be rescued. Out of these, five victims are severely injured (yellow) and might die if not treated in time. Other victims are denoted in green. Both victims depend on the first responders' help to stabilize and evacuate out of the building.

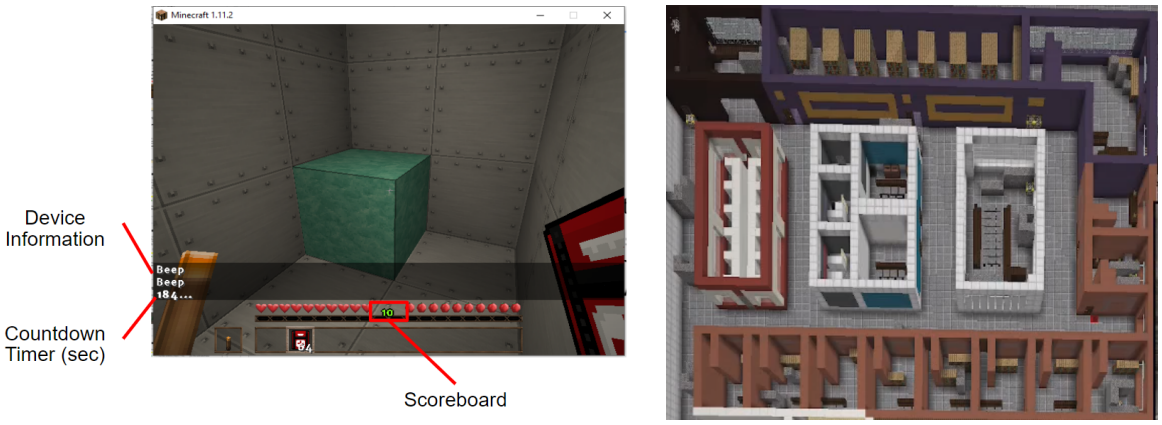


Figure 3.1: Human participants as rescuer see the view in (a). The bird's eye view of the environment as in (b).

Participants with Minecraft experience took part in an approximately three-hour experiment, in which they completed three 15 minute missions each. The participants received a training program that introduces the rules of the mission and provides some hands-on experience with the environment. The missions for each player had different maps, victim locations, and initial player beliefs. The missions had different complexity levels, randomly ordered. The complexity was manipulated by varying victims' location, the number/layout of blockages, occlusions, and the number/layout of openings.

The initial information given to the rescuer was also manipulated across both missions of the same subject. The rescuers were equipped with a device (intended to model a rescue dog) that signaled when a regular or critical victim was nearby. The device beeped at the door if live victims are present in the room and at the opening of a collapsed wall if victims are present in the connected room. It beeped twice if there was a yellow victim or a mix of yellow and green victims, once if there are only green victims. The signaling device did not beep for any other reasons. All the rescuers were equipped with this device, but only some of the participants were instructed about how to use it. The other participants could either learn to use it from experience, or could ignore the device.

We model the rescuer's intent across the victim saving strategies, and navigation behavior in terms of the next area to visit. First, we predict whether the rescuer's

trial decision whenever they find a regular victim. With time and proximity to the victims, these preferences tend to change in humans, making it a challenging sequential binary prediction task. Second, given the area segments of the rooms and corridors from the original floorplan, we formulate the next location prediction as a multi-class classification problem.

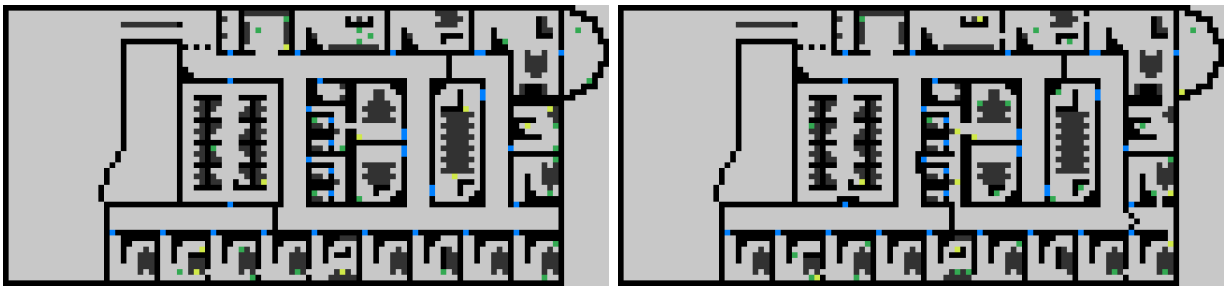


Figure 3.2: Different Map Perturbations for the Search and Rescue Missions

3.4 Experiments

Data: We collected data for 70 human subjects to evaluate our intent prediction models. These 70 subjects completed 3 search and rescue missions each. We used a 70% training, 15% validation and 15% testing split for all the models.

Mission State Representation: The mission is represented by a 146-dimensional feature vector that summarizes the trajectory. The features include 6 features for each of the 24 map regions and the estimated training condition of the player and the mission timer ($146 = 6 \times 24 + 2$).

For each room, the feature vector contains information about:

1. Whether the player has explored this room.
2. The number of yellow victims (which the player has seen but not rescued).
3. The number of green victims (which the player has seen but not rescued).
4. Whether the player is in this room.
5. Beep information for this room.

6. The distance of room from the current location of the player.

Context variable representation: The context variable for each agent is a 12 dimension vector. It summarizes the agent's planning horizon, Boltzmann constant (for soft-optimal action selection), belief update mechanisms, internal model of the environment, high-level triage strategy and usage of the signaling device.

Method: We train two baseline models for comparing our adaptive models with. The first baseline model (B1) is an intent prediction model trained on limited human data (with regularization). This model can learn to capture the patterns of human behavior from the data. Since we have access to limited human data, this model is heavily regularized.

The second baseline model (B2) generates trajectories from near-optimal planning-based agents and uses them to train an intent prediction model. This planning-based agent plans with a long horizon to efficiently search and rescue as many victims as possible, and performs much better than most human rescuers. Since there is a mismatch between the trajectory distribution of the optimal planning-based agents and humans, we expect to observe a generalization gap when we test it on human data.

Next, we train two varieties of robust models. The models are trained on various faux-human agents (with different planning horizons, belief-update mechanisms, etc.). Since this model is trained on a wide distribution of planning-based agents, we expect it to generalize better to the human data. The first robust model (R1) is trained by randomly sampling faux-humans, while the second robust model (R2) searches for faux-humans on which the current model is not doing well and updates its parameters to do better on these.

Our next model aims to adaptively use the robust models and the models trained on the human data. The model trained on the human data can recognize patterns in the human data but might not do well during testing if the mission state was not seen during training. On the other hand, the robust model better handles this distribution shift but is very pessimistic. Our adaptive model (R1+B1) uses the human model when it is certain but uses the robust model as a fallback when the human model is uncertain. We estimate the uncertainty of the model by training an ensemble of

models and computing the disagreement between the ensemble for a mission state.

Our adaptive models are described in Section 3.2. We use a LSTM to model the context predictor $\psi(c|\tau)$. See the Appendix for other details.

3.5 Results

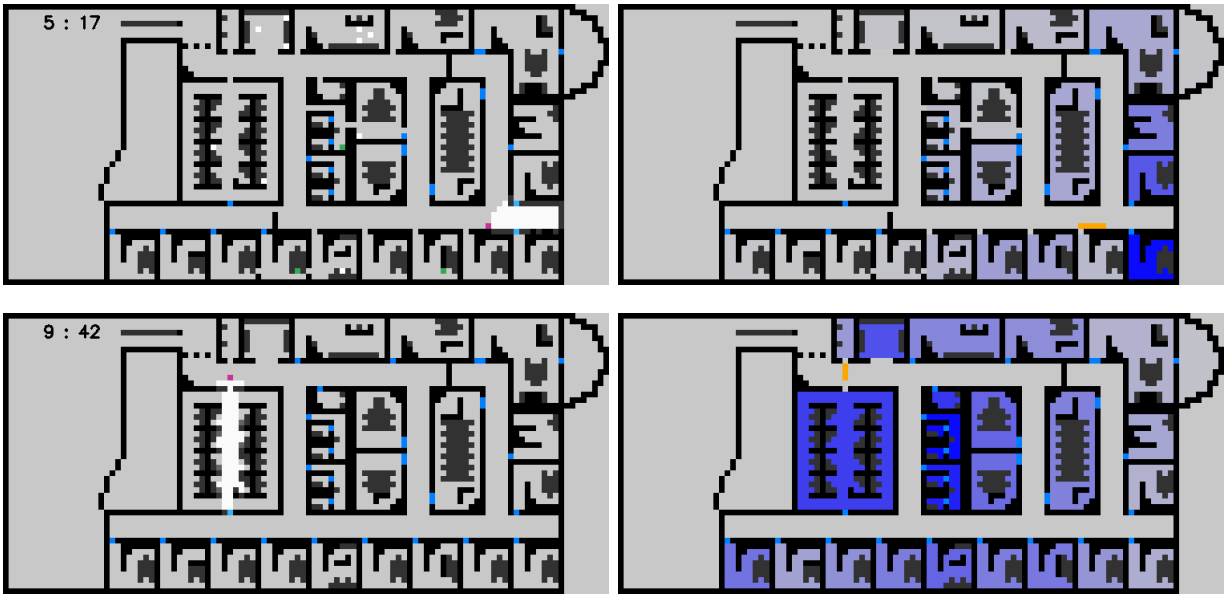


Figure 3.3: Examples of Intent Prediction : Mission State (Left), Predicted Intent (Right)

In our experiments, we seek answers to the following questions:

1. Can we combine use the robust model and human model to get the best of both worlds - good accuracy near human trajectory distribution with robust predictions for out-of-distribution trajectories?
2. Is the robust model more accurate than the model trained on trajectories from the optimal planning-based agent?
3. Can our adaptive intent prediction models outperform a generic model?

We evaluate our models on the test human subjects. We collect 3 mission trajectories for each human with different map perturbations, victim locations, and initial rescuer beliefs. We adapt our models using the first mission trajectory and evaluate

the models on later missions.

Method	Navigation Decision Prediction Accuracy	Triage Strategy Prediction Accuracy
(B1) Model trained on limited human data	71.8%	74.2%
(B2) Model trained on optimal trajectories	56.9%	61.8%
(R) Robust model trained on distribution of faux-humans	66.5%	63.2%
(R+B1) Combine robust model and human model based on uncertainty	73.9%	75.9%
(A1) Predict context variable using context predictor $\psi(c\|\tau)$	72.3%	82.7%
(A2) Train end-to-end sequence model on distribution of faux-humans	69.0%	81.1%
(A3) Optimize distribution over context variables online	78.1%	86.4%

Table 3.1: Accuracy for predicting the navigational intent for different algorithms.

Table 3.1 shows the accuracy for our models on both predictive tasks. We discuss our results on the navigation decision prediction task in detail below.

We observe that the generic intent prediction model trained on limited human data (B1) obtains an accuracy of 71.8%. On the other hand, the model trained on trajectories of the optimal planner only (B2) has an accuracy of 56.9%. This gap can be explained by the mismatch between the trajectory distribution of humans and the optimal planner.

The robust model (R) obtains an accuracy of 66.5%. This result suggests that the diversity of faux-human agents helps combat the distribution shift to human trajectories. This result is especially impressive since the model is trained using only simulated data.

We observe that the robust model and the human model complement each other, and combining them gives us an accuracy boost over just using one of them. We use the human model when it has low epistemic uncertainty and fallback to the robust model for tail events. This combination gives us an accuracy of 73.9%. This is the highest accuracy we obtain using a generic prediction model.

We obtain an accuracy ranging from 69% to 78.1% using the adaptive models. This shows that the adaptive models can use information from the first trajectory to improve model predictions in the second trajectory of the same human. In particular, we find that searching for the context variable online using Bayesian optimization performs best.

3.6 Conclusion

In summary, we show that adaptive models can be used to improve predictions as the model observes humans' behavior. These models outperform generic prediction models in our experiments. Moreover, we show that adaptive models can be trained exclusively on simulated data. Despite only having access to the context variables during training, our model can identify the behavior traits of a rescuer and use them to make better predictions. We did not model the learning effects between missions, which is an important direction for future work.

Chapter 4

Inverse Reinforcement Learning via State Marginal Matching

4.1 Introduction

Imitation learning (IL) is a powerful tool to design autonomous behaviors in robotic systems. Although reinforcement learning methods promise to learn such behaviors automatically, they have been most successful in tasks with a clear definition of the reward function. Reward design remains difficult in many robotic tasks such as driving a car [47], tying a knot [46], and human-robot cooperation [24]. Imitation learning is a popular approach to such tasks, since it is easier for an expert teacher to demonstrate the desired behavior rather than specify the reward [5, 25, 27].

Methods in IL frameworks are generally split into behavior cloning (BC) [7] and inverse reinforcement learning (IRL) [1, 43, 55]. BC is typically based on supervised learning to regress expert actions from expert observations without the need for further interaction with the environment, but suffers from the covariate shift problem [54]. On the other hand, IRL methods aim to learn the reward function from expert demonstrations, and use it to train the agent policy. Within IRL, adversarial imitation learning (AIL) methods (GAIL [26], AIRL [18], f-MAX [20], SMM [32]) train a discriminator to guide the policy to match the expert's state-action distribution.

AIL methods learn a *non-stationary* reward by iteratively training a discriminator

and taking a single policy update step using the reward derived from the discriminator. After convergence, the learned AIL reward cannot be used for training a new policy from scratch, and is thus discarded. In contrast, IRL methods such as ours learn a **stationary reward** such that, if the policy is trained from scratch using the reward function until convergence, then the policy will match the expert behavior. We argue that learning a stationary reward function can be useful for solving downstream tasks and transferring behavior across different dynamics. Traditionally, IL methods assume access to expert demonstrations and minimize some divergence between policy and expert's trajectory distribution. However, in many cases, it may be easier to directly specify the state distribution of the desired behavior rather than to provide fully-specified demonstrations of the desired behavior [32]. For example, in a safety-critical application, it may be easier to specify that the expert never visits some unsafe states, instead of tweaking reward to penalize safety violations [11]. Similarly, we can specify a uniform density over the whole state space for exploration tasks, or a Gaussian centered at the goal for goal-reaching tasks. Reverse KL instantiation for f-divergence in f-IRL allows for unnormalized density specification, which further allows for easier preference encoding.

In this paper, we propose a new method, f-IRL, that learns a stationary reward function from the expert density via gradient descent. To do so, we derive an analytic gradient of any arbitrary f-divergence between the agent and the expert state distribution w.r.t. reward parameters. We demonstrate that f-IRL is especially useful in the limited data regime, exhibiting better sample efficiency than prior work in terms of the number of environment interactions and expert trajectories required to learn the MuJoCo benchmark tasks.

We also demonstrate that the reward functions recovered by f-IRL can accelerate the learning of hard-to-explore tasks with sparse rewards, and these same reward functions can be used to transfer behaviors across changes in dynamics.

4.2 Preliminaries

In this section, we review notation on maximum entropy (MaxEnt) RL [33] and state marginal matching (SMM) [32] that we build upon in this work.

MaxEnt RL. Consider a Markov Decision Process (MDP) represented as a tuple

$(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \rho_0, T)$ with state-space $\mathcal{S}$, action-space $\mathcal{A}$, dynamics $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$, reward function $r(s, a)$, initial state distribution ρ_0, and horizon T. The optimal policy π under the maximum entropy framework [71] maximizes the objective $\sum_{t=1}^{T} \rho_{\pi,t}(s_t, a_t) r(s_t, a_t) + \alpha H(\cdot|s_t)$. Here $\rho_{\pi,t}$ is the state-action marginal distribution of policy π at timestamp t, and $\alpha > 0$ is the entropy temperature. The trajectory distribution induced by the soft-optimal policy is $p(\tau) = p_d(\tau) e^{r(\tau)/\alpha}/Z$, where $p_d(\tau) = \rho_0(s_0) \prod_{i=1}^{T-1} \pi(a_t|s_t) p(s_{t+1}|s_t, a_t)$ is the probability of trajectory τ under the dynamics, and $r(\tau)$ is the cumulative rewards in the trajectory τ, and Z is the partition function.

We assume access to an expert state marginal $\rho_E(s)$ which is feasible to specify in many tasks. If expert observations are available, we can fit a density model to the samples. Let $r_\theta(s)$ be a parameterized differentiable reward function only dependent on state. Let trajectory τ be a time series of visited states $\tau = (s_0, s_1, \ldots, s_T)$. The optimal MaxEnt trajectory distribution $\rho_\theta(\tau)$ under reward r_θ can be computed as $\rho_\theta(\tau) = \frac{1}{Z} p(\tau) e^{r_\theta(\tau)/\alpha}$, where

$$p(\tau) = \rho_0(s_0) \prod_{t=0}^{T-1} p(s_{t+1}|s_t, a_t) \ , \quad r_\theta(\tau) = \sum_{t=1}^{T} r_\theta(s_t), \quad Z = \int p(\tau) e^{r_\theta(\tau)/\alpha} d\tau.$$

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

$$\rho_\theta(s) \propto \int p(\tau) e^{r_\theta(\tau)/\alpha} \eta_\tau(s) d\tau \tag{4.1}$$

where $\eta_\tau(s) \sum_{t=1}^{T} 1(s_t = s)$ is the visitation count of a state s in a particular trajectory τ.

State Marginal Matching. Given the expert state density $p_E(s)$, one can train a policy to match the expert behavior by minimizing the following f-divergence objective:

$$L_f(\theta) = D_f(\rho_E(s) \ || \ \rho_\theta(s)) \tag{4.2}$$

where common choices for the f-divergence D_f [2, 20] include forward KL divergence, reverse KL divergence, and Jensen-Shannon divergence. Our proposed f-IRL algorithm will compute the analytical gradient of Eq. 4.2 w.r.t. θ and use it to optimize

the reward function via gradient descent.

4.3 Learning Stationary Rewards via State-Marginal Matching

In this section, we describe our algorithm f-IRL, which takes the expert state density as input, and optimizes the f-divergence objective (Eq. 4.2) via gradient descent. Our algorithm trains a policy whose state marginal is close to that of the expert, and a corresponding stationary reward function that would produce the same policy if the policy were trained with MaxEnt RL from scratch.

4.3.1 Analytic Gradient for State Marginal Matching in f-divergence

One of our main contributions is the exact gradient of the f-divergence objective (Eq. 4.2) w.r.t. the reward parameters θ. This gradient will be used by f-IRL to optimize Eq. 4.2 via gradient descent. The proof is provided in Appendix A.1.

[**f-divergence analytic gradient**] The analytic gradient of the f-divergence $L_f(\theta)$ between state marginals of the expert (ρ_E) and the soft-optimal agent w.r.t. the reward parameters θ is given by:

$$\nabla_\theta L_f(\theta) = \frac{1}{\alpha T} \text{cov}_{\tau \sim \rho_\theta(\tau)} \left(\sum_{t=1}^{T} h_f \left(\frac{\rho_E(s_t)}{\rho_\theta(s_t)} \right), \sum_{t=1}^{T} \nabla_\theta r_\theta(s_t) \right) \tag{4.3}$$

where $h_f(u) f(u) - f'(u)u$, $\rho_E(s)$ is the expert state marginal and $\rho_\theta(s)$ is the state marginal of the soft-optimal agent under the reward function r_θ, and the covariance is taken under the agent's trajectory distribution $\rho_\theta(\tau)$.[1]

Choosing the f-divergence to be Forward Kullback-Leibler (FKL), Reverse Kullback-Leibler (RKL), or Jensen-Shannon (JS) instantiates h_f (see Table 4.1). Note that the gradient of the RKL objective has a special property in that we can specify the expert as an *unnormalized* log-density (i.e. energy), since in $h_{\text{RKL}}(\frac{\rho_E(s)}{\rho_\theta(s)}) =$

[1]Here we assume f is differentiable, which is often the case for common f-divergence (e.g. KL divergence).

Name	f-divergence $D_f(P \mid\mid Q)$	Generator $f(u)$	$h_f(u)$
FKL	$\int p(x) \log \frac{p(x)}{q(x)} dx$	$u \log u$	$-u$
RKL	$\int q(x) \log \frac{q(x)}{p(x)} dx$	$-\log u$	$1 - \log u$
JS	$\frac{1}{2} \int p(x) \log \frac{2p(x)}{p(x)+q(x)} + q(x) \log \frac{2q(x)}{p(x)+q(x)} dx$	$u \log u - (1+u) \log \frac{1+u}{2}$	$-\log(1+u)$

Table 4.1: Selected list of f-divergences $D_f(P \mid\mid Q)$ with generator functions f and h_f defined in Theorem 4.3.1, where f is convex, lower-semicontinuous and $f(1) = 0$.

$1 - \log \rho_E(s) + \log \rho_\theta(s)$, the normalizing factor of $\rho_E(s)$ does not change the gradient (by linearity of covariance). This makes density specification much easier in a number of scenarios. Intuitively, since h_f is a monotonically decreasing function ($h'_f(u) = -f''(u)u < 0$) over $\mathbb{R}^+$, the gradient descent tells the reward function to increase the rewards of those state trajectories that have higher sum of density ratios $\sum_{t=1}^{T} \frac{\rho_E(s_t)}{\rho_\theta(s_t)}$ so as to minimize the objective.

4.3.2 Learning a Stationary Reward by Gradient Descent

We now build upon Theorem 4.3.1 to design a practical algorithm for learning the reward function r_θ (Algorithm. 1). Given expert information (state density or observation samples) and an arbitrary f-divergence, the algorithm alternates between using MaxEnt RL with the current reward, and updating the reward parameter using gradient descent based on the analytic gradient.

If the provided expert data is in the form of expert state density $\rho_E(s)$, we can fit a density model $\hat{\rho}_\theta(s)$ to estimate agent state density $\rho_\theta(s)$ and thus estimate the density ratio required in gradient. If we are given samples from expert observations s_E, we can fit a discriminator $D_\omega(s)$ in each iteration to estimate the density ratio by optimizing the binary cross-entropy loss:

$$\max_{\omega} s \sim s_E \log D_\omega(s) + s \sim \rho_\theta(s) \log(1 - D_\omega(s)) \tag{4.4}$$

where the optimal discriminator satisfies $D^*_\omega(s) = \frac{\rho_E(s)}{\rho_E(s)+\rho_\theta(s)}$ [22], thus the density ratio can be estimated by $\frac{\rho_E(s)}{\rho_\theta(s)} \approx \frac{D_\omega(s)}{1-D_\omega(s)}$, which is the input to h_f.

Algorithm 1 Inverse RL via State Marginal Matching (f-IRL)

InputInput OutputOutput Expert state density $\rho_E(s)$ or expert observations s_E , f-divergence
Learned reward r_θ, Policy π_θ Initialize r_θ, and density estimation model (provided $\rho_E(s)$) or disciminator D_ω (provided s_E)
for $i \leftarrow 1$ *Iter* **do**
$\pi_\theta \leftarrow$ MaxEntRL(r_θ) and collect agent trajectories τ_θ
    if provided $\rho_E(s)$ **then** Fit the density model $\hat{\rho}_\theta(s)$ to the state samples from τ_θ
    elseprovided s_E Fit the discriminator D_ω by Eq. 4.4 using expert and agent state samples from s_E and τ_θ
Compute sample gradient $\hat{\nabla}_\theta L_f(\theta)$ for Eq. 4.3 over τ_θ
$\theta \leftarrow \theta - \lambda \hat{\nabla}_\theta L_f(\theta)$

4.3.3 Robust Reward Recovery under State-only Ground-truth Reward

IRL methods are different from IL methods in that they recover a reward function in addition to the policy. A hurdle in this process is often the reward ambiguity problem, explored in [18, 42]. This ambiguity arises due to the fact that the optimal policy remains unchanged under the following reward transformation [42]:

$$\hat{r}(s, a, s') = r_{\text{gt}}(s, a, s') + \gamma\Phi(s') - \Phi(s) \tag{4.5}$$

for any function Φ. In the case where the ground-truth reward is a function over states only (i.e., $r_{\text{gt}}(s)$), f-IRL is able to recover the *disentangled* reward function (r_{IRL}) that matches the ground truth reward r_{gt} up to a constant. The obtained reward function is robust to different dynamics – for any underlying dynamics, r_{IRL} will produce the same optimal policy as r_{gt}. We formalize this claim in Appendix A.1.4 (based on Theorem 5.1 of AIRL [18]).

AIRL uses a special parameterization of the discriminator to learn state-only rewards. A disadvantage of their approach is that AIRL needs to approximate a separate reward-shaping network apart from the reward network. In contrast, our method naturally recovers a state-only reward function.

4.3.4 Practical Modification in the Exact Gradient

In practice with high-dimensional observations, when the agent's current trajectory distribution is far off from the expert trajectory distribution, we find that there is little supervision available through our derived gradient, leading to slow learning. Therefore, when expert trajectories are provided, we bias the gradient (Eq. 4.3) using a mixture of agent and expert trajectories inspired by GCL [16], which allows for richer supervision and faster convergence. Note that at convergence, the gradient becomes unbiased as the agent's and expert's trajectory distribution matches.

$$\tilde{\nabla}_\theta L_f(\theta) := \frac{1}{\alpha T} \mathrm{cov}_{\tau \sim \frac{1}{2}(\rho_\theta(\tau)+\rho_E(\tau))} \left(\sum_{t=1}^{T} h_f \left(\frac{\rho_E(s_t)}{\rho_\theta(s_t)} \right), \sum_{t=1}^{T} \nabla_\theta r_\theta(s_t) \right) \tag{4.6}$$

where the expert trajectory distribution $\rho_E(\tau)$ is uniform over samples τ_E.

4.4 Experiments

In our experiments, we seek answers to the following questions:

1. Can f-IRL learn a policy that matches the given expert state density?
2. Can f-IRL learn good policies on high-dimensional continuous control tasks in a sample-efficient manner?
3. Can f-IRL learn a reward function that induces the expert policy?
4. How can learning a stationary reward function help solve downstream tasks?

Comparisons. To answer these questions, we compare f-IRL against two classes of existing imitation learning algorithms: (1) those that learn only the policy, including Behavior Cloning (BC), GAIL [26], and f-MAX-RKL[2] [20]; and (2) IRL methods that learn both a reward and a policy simultaneously, including MaxEnt IRL [72] and AIRL [18]. The rewards/discriminators of the baselines are parameterized to be state-only. We use SAC [23] as the base MaxEnt RL algorithm. Since the original AIRL uses TRPO [57], we re-implement a version of AIRL that uses SAC as the

[2]A variant of AIRL [18] proposed in [20] only learns a policy and does not learn a reward.

underlying RL algorithm for fair comparison. For our method (f-IRL), MaxEnt IRL, and AIRL, we use a MLP for reward parameterization.

Tasks. We evaluate the algorithms on several tasks:

- **Matching Expert State Density**: In Section 4.4.1, the task is to learn a policy that matches the given expert state density.
- **Inverse Reinforcement Learning Benchmarks**: In Section 4.4.2, the task is to learn a reward function and a policy from expert trajectory samples. We collected expert trajectories by training SAC [23] to convergence on each environment. We trained all the methods using varying numbers of expert trajectories $\{1, 4, 16\}$ to test the robustness of each method to the amount of available expert data.
- **Using the Learned Reward for Downstream Tasks**: In Section 4.4.3, we first train each algorithm to convergence, then use the learned reward function to train a new policy on a related downstream task. We measure the performance on downstream tasks for evaluation.

We use five MuJoCo continuous control locomotion environments [9, 62] with joint torque actions, illustrated in Figure 4.1. Further details about the environment, expert information (samples or density specification), and hyperparameter choices can be found in Appendix A.2.

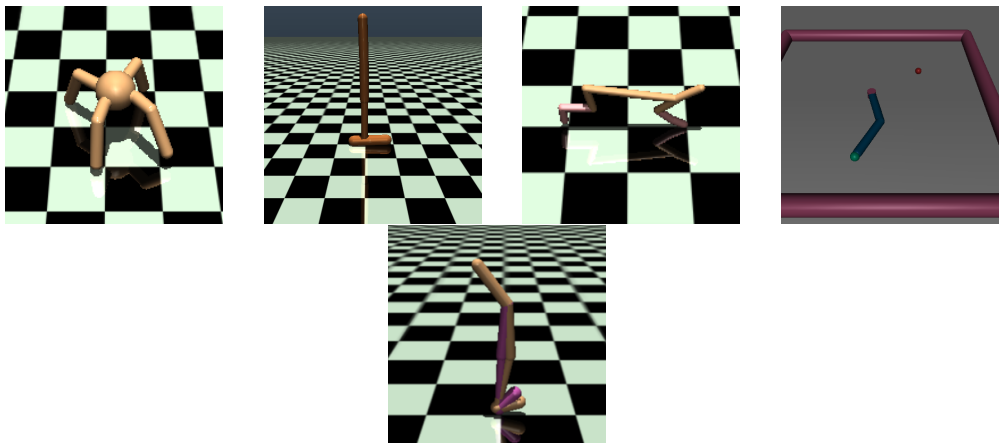
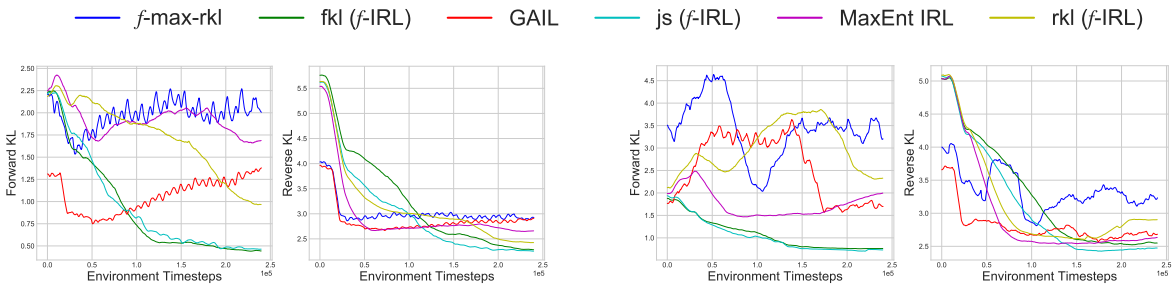


Figure 4.1: **Environments**: (left to right) Ant-v2, Hopper-v2, HalfCheetah-v2, Reacher-v2, and Walker2d-v2.



(a) Expert Density: Gaussian (b) Expert Density: Mixture of two Gaussians

Figure 4.2: Forward (left) and Reverse (right) KL curves in the Reacher environment for different expert densities of all methods. Curves are smoothed in a window of 120 evaluations.

4.4.1 Matching the Specified Expert State Density

First, we check whether f-IRL can learn a policy that matches the given expert state density of the fingertip of the robotic arm in the 2-DOF Reacher environment. We evaluate the algorithms using two different expert state marginals: (1) a Gaussian distribution centered at the goal for single goal-reaching, and (2) a mixture of two Gaussians, each centered at one goal. Since this problem setting assumes access to the expert density only, we use importance sampling to generate expert samples required by the baselines.

In Figure 4.2, we report the estimated forward and reverse KL divergences in state marginals between the expert and the learned policy. For f-IRL and MaxEnt IRL, we use Kernel Density Estimation (KDE) to estimate the agent's state marginal. We observe that the baselines demonstrate unstable convergence, which might be because those methods optimize the f-divergence approximately. Our method {FKL, JS} f-IRL outperforms the baselines in the forward KL and the reverse KL metric, respectively.

4.4.2 Inverse Reinforcement Learning Benchmarks

Next, we compare f-IRL and the baselines on IRL benchmarks, where the task is to learn a reward function and a policy from expert trajectory samples. We use the modification proposed in Section 4.3.4 to alleviate the difficulty in optimizing the

f-IRL objective with high-dimensional states.

Policy Performance. We check whether f-IRL can learn good policies on high-dimensional continuous control tasks in a sample-efficient manner from expert trajectories. Figure 4.3 shows the learning curves of each method in the four environments with *one* expert trajectory provided. f-IRL and MaxEnt IRL demonstrate much faster convergence in most of the tasks than f-MAX-RKL. Table 4.2 shows the final performance of each method in the four tasks, measured by the ratio of agent returns (evaluated using the ground-truth reward) to expert returns. While MaxEnt IRL provides a strong baseline, f-IRL outperforms all baselines on most tasks especially in Ant, where the FKL (f-IRL) has much higher final performance and is less sensitive to the number of expert trajectories compared to the baselines. In contrast, we found the original implementation of f-MAX-RKL to be extremely sensitive to hyperparameter settings. We also found that AIRL performs poorly even after tremendous tuning, similar to the findings in [36, 37].

Recovering the Stationary Reward Function. We also evaluate whether f-IRL can recover a stationary reward function that induces the expert policy. To do so, we train a SAC agent from scratch to convergence using the reward model obtained from each IRL method. We then evaluate the trained agents using the ground-truth reward to test whether the learned reward functions are good at inducing the expert policies.

Table 4.3 shows the ratio of the final returns of policy trained from scratch using the rewards learned from different IRL methods with one expert trajectory provided, to expert returns. Our results show that MaxEnt IRL and f-IRL are able to learn *stationary* rewards that can induce a policy close to the optimal expert policy.

4.4.3 Using the Learned Stationary Reward for Downstream Tasks

Finally, we investigate how the learned stationary reward can be used to learn related, downstream tasks.

Reward prior for downstream hard-exploration tasks. We first demonstrate the utility of the learned stationary reward by using it as a prior reward for the downstream task. Specifically, we construct a didactic point mass environment

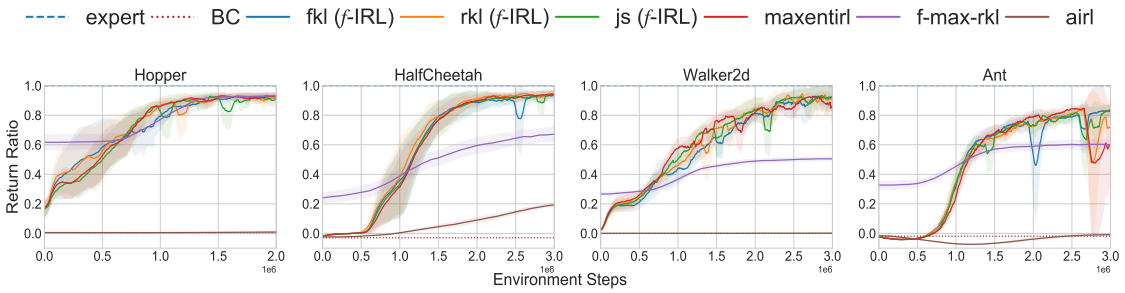


Figure 4.3: Training curves for f-IRL and 4 other baselines - BC, MaxEnt IRL, f-MAX-RKL and AIRL with one expert demonstration. Solid curves depict the mean of 3 trials and the shaded area shows the standard deviation. The dashed blue line represents the expert performance and the dashed red line shows the performance of a BC agent at convergence.

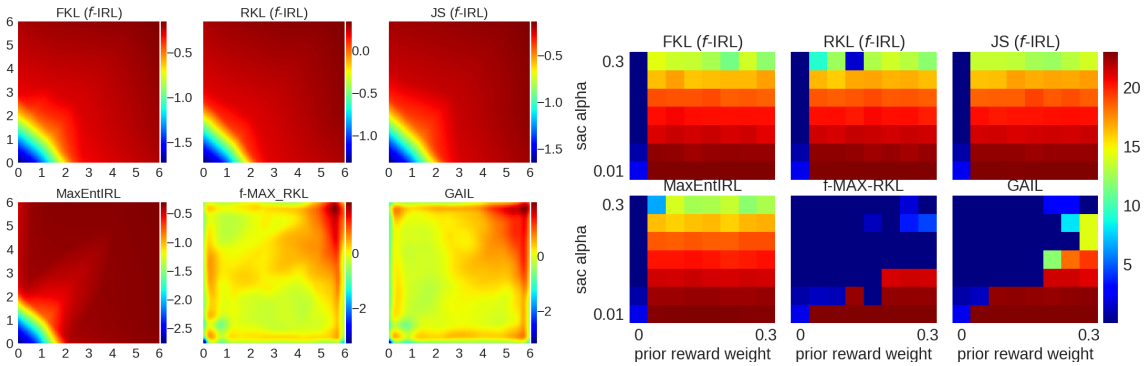


Figure 4.4: Left: Extracted final reward of all compared methods for the uniform expert density in the point environment. Right: The task return (in terms of r_{task}) with different α and prior reward weight λ. The performance of vanilla SAC is shown in the leftmost column with $\lambda = 0$ in each subplot.

that operates under linear dynamics in a 2D 6×6 room, and actions are restricted to $[-1, 1]$. The prior reward is obtained from a *uniform* expert density over the whole state space, and is used to ease the learning in the hard-exploration task, where we design a difficult goal to reach with distraction rewards (full details in appendix A.2).

We use the learned prior reward r_{prior} to augment the task reward r_{task} as follows:

$$r(s, a, s') = r_{\text{task}}(s, a, s') + \lambda(\gamma r_{\text{prior}}(s') - r_{\text{prior}}(s)) \tag{4.7}$$

where $\lambda \geq 0$ is the weight of prior reward and γ is the discount factor. The main

Method	Hopper			Walker2d			HalfCheetah			Ant		
Expert return	3592.63 ± 19.21			5344.21 ± 84.45			12427.49 ± 486.38			5926.18 ± 124.56		
# Expert traj	1	4	16	1	4	16	1	4	16	1	4	16
BC	0.00	0.13	0.16	0.00	0.05	0.08	0.00	0.01	0.02	0.00	0.22	0.47
MaxEnt IRL	0.93	0.92	**0.94**	0.88	0.88	**0.91**	**0.95**	**0.98**	0.91	0.54	0.71	0.81
f-MAX-RKL	**0.94**	**0.93**	0.91	0.49	0.49	0.47	0.71	0.41	0.65	0.60	0.65	0.62
AIRL	0.01	0.01	0.01	0.00	0.00	0.00	0.19	0.19	0.19	0.00	0.00	0.00
FKL (f-IRL)	0.93	0.90	0.93	**0.90**	0.90	0.90	0.94	0.97	0.94	**0.82**	**0.83**	**0.84**
RKL (f-IRL)	0.93	0.92	0.93	0.89	0.90	0.85	**0.95**	0.97	**0.96**	0.63	0.82	0.81
JS (f-IRL)	0.92	**0.93**	**0.94**	0.89	**0.92**	0.88	0.93	**0.98**	0.94	0.77	0.81	0.73

Table 4.2: We report the ratio between the average return of the trained (stochastic) policy vs. that of the expert policy for different IRL algorithms using 1, 4 and 16 expert trajectories. All results are averaged across 3 seeds. Negative ratios are clipped to zero.

Method	Hopper	Walker2d	HalfCheetah	Ant
AIRL	-	-	-0.03	-
MaxEntIRL	**0.93**	**0.92**	0.96	0.79
f-IRL	**0.93**	0.88	**1.02**	**0.82**

Table 4.3: The ratios of final return of the obtained policy against expert return across IRL methods. We average f-IRL over FKL, RKL, and JS. '-' indicates that we do not test learned rewards since AIRL does poorly at these tasks in Table 4.2.

Policy Transfer using GAIL	AIRL	MaxEntIRL	f-IRL	Ground-truth Reward
-29.9	130.3	**145.5**	141.1	315.5

Table 4.4: Returns obtained after transferring the policy/reward on modified Ant environment using different IL methods.

theoretical result of [42] dictates that adding a potential-based reward in this form will not change the optimal policy. GAIL and f-MAX-RKL do not extract a reward function but rather a discriminator, so we derive a prior reward from the discriminator in the same way as [20, 26].

Figure 4.4 illustrates that the reward recovered by {FKL, RKL, JS} f-IRL and the

baseline MaxEnt IRL are similar: the reward increases as the distance to the agent's start position, the bottom left corner, increases. This is intuitive for achieving the target uniform density: states farther away should have higher rewards. f-MAX-RKL and GAIL's discriminator demonstrate a different pattern which does not induce a uniform state distribution. The leftmost column in the Figure 4.4 (Right) shows the poor performance of SAC training without reward augmentation ($\lambda = 0$). This verifies the difficulty in exploration for solving the task. We vary λ in the x-axis, and α in SAC in the y-axis, and plot the final task return (in terms of r_{task}) as a heatmap in the figure. The presence of larger red region in the heatmap shows that our method can extract a prior reward that is more robust and effective in helping the downstream task attain better final performance with its original reward.

Reward transfer across changing dynamics. Lastly, we evaluate the algorithms on transfer learning across different environment dynamics, following the setup from [18]. In this setup, IL algorithms are provided expert trajectories from a quadrupedal ant agent which runs forward. The algorithms are tested on an ant with two of its legs being disabled and shrunk. This requires the ant to significantly change its gait to adapt to the disabled legs for running forward.

We found that a forward-running policy obtained by GAIL fails to transfer to the disabled ant. In contrast, IRL algorithms such as f-IRL are successfully able to learn the expert's reward function using expert demonstrations from the quadrupedal ant, and use the reward to train a policy on the disabled ant. The results in Table 4.4 show that the reward learned by f-IRL is robust and enables the agent to learn to move forward with just the remaining two legs.

4.5 Conclusion

In summary, we have proposed f-IRL, a practical IRL algorithm that distills an expert's state distribution into a stationary reward function. Our f-IRL algorithm can learn from either expert samples (as in traditional IRL), or a specified expert density (as in SMM [32]), which opens the door to supervising IRL with different types of data. These types of supervision can assist agents in solving tasks faster, encode preferences for how tasks are performed, and indicate which states are unsafe and should be avoided. Our experiments demonstrate that f-IRL is more sample efficient

in the number of expert trajectories and environment timesteps as demonstrated on MuJoCo benchmarks.

Chapter 5

Conclusions

Our work aimed to develop algorithms for adaptive and efficient intent recognition algorithms. These could be useful, for example, to provide anticipatory assistance to human rescuers in disaster scenarios.

We demonstrated our adaptive algorithms on human subjects in a disaster scenario. Our methods were based on training adaptive models on a wide variety of faux-humans, so that they generalize to humans. We showed that our models could learn from a rescuer's decisions in the first mission to make better predictions in later missions. These algorithms can be extended to personalize the assistance policy for the rescuer as well.

We also showed that our proposed algorithm f-IRL is sample-efficient. It could learn the demonstrator's intent from a single trajectory in simulated control tasks, and use it for imitation. The algorithm could use the learned intent to imitate the demonstrator in a completely different environment.

Appendix A

Inverse Reinforcement Learning using State Marginal Matching

A.1 Derivation and Proof

This section provides the derivation and proof for the main paper. Section A.1.1 and A.1.2 provide the derivation of Theorem 4.3.1, and section A.1.4 provides the details about section 4.3.3.

A.1.1 Analytical Gradient of State Marginal Distribution

In this subsection, we start by deriving a general result - gradient of state marginal distribution w.r.t. parameters of the reward function. We will use this gradient in the next subsection A.1.2 where we derive the gradient of f-divergence objective.

Based on the notation introduced in section **??**, we start by writing the probability of trajectory $\tau = (s_0, s_1, \ldots, s_T)$ of fixed horizon T under the optimal MaxEnt trajectory distribution for $r_\theta(s)$ [71].

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

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

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

$$\rho_\theta(\tau) = \frac{p(\tau)e^{\sum_{t=1}^{T} r_\theta(s_t)/\alpha}}{\int p(\tau)e^{\sum_{t=1}^{T} r_\theta(s_t)/\alpha}d\tau} \tag{A.2}$$

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

$$\rho_\theta(s) \propto \int p(\tau)e^{\sum_{t=1}^{T} r_\theta(s_t)/\alpha}\eta_\tau(s)d\tau \tag{A.3}$$

where

$$\eta_\tau(s) = \sum_{t=1}^{T} 1(s_t = s) \tag{A.4}$$

is the empirical frequency of state s in trajectory τ (omitting the starting state s_0 as the policy cannot control the initial state distribution).

The marginal distribution over states can now be written as:

$$\rho_\theta(s) \propto \int p(\tau)e^{\sum_{t=1}^{T} r_\theta(s_t)/\alpha}\eta_\tau(s)d\tau \tag{A.5}$$

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

$$\begin{aligned}
\rho_\theta(s) &= \frac{\int p(\tau)e^{\sum_{t=1}^{T} r_\theta(s_t)/\alpha}\eta_\tau(s)d\tau}{\int\int p(\tau')e^{\sum_{t=1}^{T} r_\theta(s'_t)/\alpha}\eta_{\tau'}(s')d\tau' ds'} \\
&= \frac{\int p(\tau)e^{\sum_{t=1}^{T} r_\theta(s_t)/\alpha}\eta_\tau(s)d\tau}{\int p(\tau')e^{\sum_{t=1}^{T} r_\theta(s'_t)/\alpha}\int \eta_{\tau'}(s')ds' d\tau'} \\
&= \frac{\int p(\tau)e^{\sum_{t=1}^{T} r_\theta(s_t)/\alpha}\eta_\tau(s)d\tau}{T\int p(\tau')e^{\sum_{t=1}^{T} r_\theta(s'_t)/\alpha}d\tau'}
\end{aligned} \tag{A.6}$$

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

to simplify notation in further calculations.

$$\begin{aligned} f(s) &= \int p(\tau) e^{\sum_{t=1}^{T} r_\theta(s_t)/\alpha} \eta_\tau(s) d\tau \\ Z &= T \int p(\tau) e^{\sum_{t=1}^{T} r_\theta(s_t)/\alpha} d\tau \\ \rho_\theta(s) &= \frac{f(s)}{Z} \end{aligned} \tag{A.7}$$

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

$$\frac{df(s)}{dr_\theta(s^*)} = \frac{1}{\alpha} \int p(\tau) e^{\sum_{t=1}^{T} r_\theta(s_t)/\alpha} \eta_\tau(s) \eta_\tau(s^*) d\tau \tag{A.8}$$

$$\frac{dZ}{dr_\theta(s^*)} = \frac{T}{\alpha} \int p(\tau) e^{\sum_{t=1}^{T} r_\theta(s_t)/\alpha} \eta_\tau(s^*) d\tau = \frac{T}{\alpha} f(s^*) \tag{A.9}$$

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

$$\begin{aligned} \frac{d\rho_\theta(s)}{dr_\theta(s^*)} &= \frac{Z \frac{df(s)}{dr_\theta(s^*)} - f(s) \frac{dZ}{dr_\theta(s^*)}}{Z^2} \\ &= \frac{\int p(\tau) e^{\sum_{t=1}^{T} r_\theta(s_t)/\alpha} \eta_\tau(s) \eta_\tau(s^*) d\tau}{\alpha Z} - \frac{f(s)}{Z} \frac{T f(s^*)}{\alpha Z} \\ &= \frac{\int p(\tau) e^{\sum_{t=1}^{T} r_\theta(s_t)/\alpha} \eta_\tau(s) \eta_\tau(s^*) d\tau}{\alpha Z} - \frac{T}{\alpha} \rho_\theta(s) \rho_\theta(s^*) \end{aligned} \tag{A.10}$$

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

rule.

$$\begin{aligned}
\frac{d\rho_\theta(s)}{d\theta} &= \int \frac{d\rho_\theta(s)}{dr_\theta(s^*)} \frac{dr_\theta(s^*)}{d\theta} ds^* \\
&= \frac{1}{\alpha} \int \left(\frac{\int p(\tau) e^{\sum_{t=1}^T r_\theta(s_t)/\alpha} \eta_\tau(s) \eta_\tau(s^*) d\tau}{Z} - T\rho_\theta(s)\rho_\theta(s^*) \right) \frac{dr_\theta(s^*)}{d\theta} ds^* \\
&= \frac{1}{\alpha Z} \int \int p(\tau) e^{\sum_{t=1}^T r_\theta(s_t)/\alpha} \eta_\tau(s) \eta_\tau(s^*) \frac{dr_\theta(s^*)}{d\theta} ds^* d\tau - \frac{T}{\alpha} \rho_\theta(s) \int \rho_\theta(s^*) \frac{dr_\theta(s^*)}{d\theta} ds^* \\
&= \frac{1}{\alpha Z} \int p(\tau) e^{\sum_{t=1}^T r_\theta(s_t)/\alpha} \eta_\tau(s) \sum_{t=1}^T \frac{dr_\theta(s_t)}{d\theta} d\tau - \frac{T}{\alpha} \rho_\theta(s) \int \rho_\theta(s^*) \frac{dr_\theta(s^*)}{d\theta} ds^*
\end{aligned} \tag{A.11}$$

A.1.2 Analytical Gradient of f-divergence objective

f-divergence [2] is a family of divergence, which generalizes forward/reverse KL divergence. Formally, let P and Q be two probability distributions over a space Ω, then for a convex and lower-semicontinuous function f such that $f(1) = 0$, the f-divergence of P from Q is defined as:

$$D_f(P \mid\mid Q) := \int_\Omega f\left(\frac{dP}{dQ}\right) dQ \tag{A.12}$$

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

$$\min_\theta L_f(\theta) := D_f(\rho_E \mid\mid \rho_\theta) = \int_{\mathcal{S}} f\left(\frac{\rho_E(s)}{\rho_\theta(s)}\right) \rho_\theta(s) ds \tag{A.13}$$

Now we show the proof of **Theorem 4.3.1** on the gradient of f-divergence objective:

The gradient of the f-divergence objective can be derived by chain rule:

$$\begin{aligned}\nabla_\theta L_f(\theta) &= \int \nabla_\theta \left(f\left(\frac{\rho_E(s)}{\rho_\theta(s)}\right)\rho_\theta(s)\right) ds \\ &= \int \left(f\left(\frac{\rho_E(s)}{\rho_\theta(s)}\right) - f'\left(\frac{\rho_E(s)}{\rho_\theta(s)}\right)\frac{\rho_E(s)}{\rho_\theta(s)}\right)\frac{d\rho_\theta(s)}{d\theta} ds \\ &\int h_f\left(\frac{\rho_E(s)}{\rho_\theta(s)}\right)\frac{d\rho_\theta(s)}{d\theta} ds\end{aligned} \tag{A.14}$$

where we denote $h_f(u)f(u) - f'(u)u.$ for convenience.[1]

Substituting the gradient of state marginal distribution w.r.t θ in Eq. A.11, we have:

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

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

$$\begin{aligned}&\nabla_\theta L_f(\theta) \\ &= \frac{1}{\alpha T}\left(\rho_\theta(\tau)\sum_{t=1}^T h_f\left(\frac{\rho_E(s_t)}{\rho_\theta(s_t)}\right)\sum_{t=1}^T \nabla_\theta r_\theta(s_t) - \rho_\theta(\tau)\sum_{t=1}^T h_f\left(\frac{\rho_E(s_t)}{\rho_\theta(s_t)}\right)\rho_\theta(\tau)\sum_{t=1}^T \nabla_\theta r_\theta(s_t)\right) \\ &= \frac{1}{\alpha T}\mathrm{cov}_{\tau\sim\rho_\theta(\tau)}\left(\sum_{t=1}^T h_f\left(\frac{\rho_E(s_t)}{\rho_\theta(s_t)}\right), \sum_{t=1}^T \nabla_\theta r_\theta(s_t)\right)\end{aligned} \tag{A.16}$$

[1]Note that if $f(u)$ is non-differentiable at some points, such as $f(u) = |u-1|/2$ at $u = 1$ for Total Variation distance, we take one of its subderivatives.

Thus we have derived the analytic gradient of f-divergence for state-marginal matching as shown in Theorem 4.3.1.

A.1.3 Extension to Integral Probability Metrics in f-IRL

Integral Probability Metrics (IPM) [41] is another class of divergence based on dual norm, examples of which include Wasserstein distance [3] and MMD [34]. We can use Kantorovich-Rubinstein duality [65] to rewrite the IPM-based state marginal matching as:

$$L_{\mathrm{B}}(\theta) = \|\rho_E(s) - \rho_\theta(s)\|_B := \max_{D_\omega \in B} \rho_E(s) D_\omega(s) - \rho_\theta(s) D_\omega(s) \tag{A.17}$$

where B is a symmetric convex set of functions and D_ω is the critic function in [3].

Then the analytical gradient of the objective $L_{\mathrm{B}}(\theta)$ can be derived to be:

$$\nabla_\theta L_{\mathrm{B}}(\theta) = -\frac{1}{\alpha T} \mathrm{cov}_{\tau \sim \rho_\theta(\tau)} \left(\sum_{t=1}^{T} D_\omega\left(s_t\right), \sum_{t=1}^{T} \nabla_\theta r_\theta(s_t) \right) \tag{A.18}$$

where the derivation directly follows the proof of Theorem 4.3.1.

A.1.4 f-IRL Learns Disentangled Rewards w.r.t. Dynamics

We follow the derivation and definitions as given in Fu et al. [18] to show that f-IRL learns disentangled rewards. We show the definitions and theorem here for completeness. For more information, please refer to Fu et al. [18].

We first redefine the notion of "disentangled rewards".

[Disentangled rewards] A reward function $r'(s, a, s')$ is (perfectly) disentangled with respect to ground truth reward $r_{\mathrm{gt}}(s, a, s')$ and a set of dynamics $\mathcal{T}$ such that under all dynamics in $T \in \mathcal{T}$, the optimal policy is the same: $\pi^*_{r',T}(a|s) = \pi^*_{r_{\mathrm{gt}},T}(a|s)$

Disentangled rewards can be loosely understood as learning a reward function which will produce the same optimal policy as the ground truth reward for the environment, on any underlying dynamics.

To show how f-IRL recovers a disentangled reward function, we need go through the definition of "Decomposability condition"

[Decomposability condition] Two states s_1,s_2 are defined as "1-step linked" under a dyanamics or transition distribution $T(s'|a,s)$, if there exists a state that can reach s_1 and s_2 with positive probability in one timestep. Also, this relationship can transfer through transitivity: if s_1 and s_2 are linked, and s_2 and s_3 are linked then we can consider s_1 and s_3 to be linked.
A transition distribution T satisfies the decomposibility condition if all states in the MDP are linked with all other states.

This condition is mild and can be satisfied by any of the environments used in our experiments.

Theorem A.1.4 and A.1.4 formalize the claim that f-IRL recovers disentangled reward functions with respect to the dynamics. The notation $Q^*_{r,T}$ denotes the optimal Q function under reward function r and dynamics T, and similarly $\pi^*_{r,T}$ is the optimal policy under reward function r and dynamics T.

Let $r_{gt}(s)$ be the expert reward, and T be a dynamics satisfying the decomposability condition as defined in [18]. Suppose f-IRL learns a reward r_{IRL} such that it produces an optimal policy in T: $Q^*_{r_{\text{IRL}},T}(s,a) = Q^*_{r_{gt},T}(s,a) - f(s)$,where $f(s)$ is an arbitrary function of the state. Then we have:

$r_{\text{IRL}}(s) = r_{gt}(s)+C$ for some constant C, and thus $r_{\text{IRL}}(s)$ is robust to all dynamics. Refer to Theorem 5.1 of AIRL [18].

If a reward function $r'(s,a,s')$ is disentangled with respect to all dynamics functions, then it must be state-only. Refer to Theorem 5.2 of AIRL [18].

A.2 Implementation Details

A.2.1 Matching the Specified Expert State Density on Reacher (Sec 4.4.1)

Environment: The OpenAI gym `Reacher-v2` environment [9] has a robotic arm with 2 DOF on a 2D arena. The state space is 8-dimensional: sine and cosine of both joint angles, and the position and velocity of the arm fingertip in x and y direction. The action controls the torques for both joints. The lengths of two bodies are $r_1 = 0.1, r_2 = 0.11$, thus the trace space of the fingertip is an annulus with $R = r_1 + r_2 = 0.21$ and $r = r_2 - r_1 = 0.01$. Since r is very small, it can be

approximated as a disc with radius $R = 0.21$. The time horizon is $T = 30$. We remove the object in original reacher environment as we only focus on the fingertip trajectories.

Expert State Density: The domain is x-y coordinate of fingertip position. We experiment with the following expert densities:

- *Single Gaussian:* $\mu = (-R, 0) = (-0.21, 0), \sigma = 0.05$.
- *Mixture of two equally-weighted Gaussians:* $\mu_1 = (-R/\sqrt{2}, -R/\sqrt{2}), \mu_2 = (-R/\sqrt{2}, R/\sqrt{2}), \sigma_1 = \sigma_2 = 0.05$

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

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

For other baselines including AIRL, f-MAX-RKL, GAIL, we refer to the f-MAX [20] authors' official implementation[2]. We use the default discriminator architecture as in [20]. In detail, first the input is linearly embedded into a 128-dim vector. This hidden state then passes through 6 Resnet blocks of 128-dimensions; the residual path uses batch normalization and tanh activation. The last hidden state is then linearly embedded into a single-dimensional output, which is the logits of the discriminator. The logit is clipped to be within the range $[-10, 10]$. The discriminator is optimized using Adam with a learning rate of 0.0003 and a batch size of 128.

At each epoch, for all methods, we train SAC for 10 episodes using the current reward/discriminator. We warm-start SAC policy and critic networks from networks trained at previous iteration. We do not empty the replay buffer, and leverage data collected in earlier iterations for training SAC. We found this to be effective empirically, while saving lots of computation time for the bilevel optimization.

For f-IRL and MaxEntIRL, we update the reward for 2 gradient steps in each

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

iteration. For AIRL, f-MAX-RKL and GAIL, the discriminator takes 60 gradient steps per epoch. We train all methods for 800 epochs.

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

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

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

$$\begin{aligned}\nabla_\theta L_{\text{MaxEntIRL}}(\theta) &= \frac{1}{\alpha}\sum_{t=1}^{T}\left(s_t \sim \rho_{E,t}\nabla r_\theta(s_t) - s_t \sim \rho_{\theta,t}\nabla r_\theta(s_t)\right) \\ &= \frac{T}{\alpha}\left(s \sim \rho_E \nabla r_\theta(s) - s \sim \rho_\theta \nabla r_\theta(s)\right) \\ &= \frac{T}{\alpha}\left(s \sim \rho_\theta \frac{\rho_E(s)}{\hat{\rho}_\theta(s)}\nabla r_\theta(s) - s \sim \rho_\theta \nabla r_\theta(s)\right)\end{aligned} \tag{A.19}$$

  where $\rho_t(s)$ is state marginal at timestamp t, and $\rho(s) = \sum_{t=1}^{T}\rho_t(s)/T$ is state marginal averaged over all timestamps, and we fit a density model to the agent distribution as $\hat{\rho}_\theta$.

- *For GAIL, AIRL, f-MAX-RKL*: Original discriminator needs to be trained using expert samples, thus we use the same density model as described above, and then use importance sampling to compute the discriminator objective:

$$\max_D L(D) = s \sim \rho_\theta \frac{\rho_E(s)}{\hat{\rho}_\theta(s)} \log D(s) + s \sim \rho_\theta \log(1 - D(s)) \tag{A.20}$$

Evaluation: For the approximation of both forward and reverse KL divergence, we use non-parametric Kozachenko-Leonenko estimator [29, 30] with lower error [59]

compared to plug-in estimators using density models. Suggested by [64][3], we choose $k = 3$ in k-nearest neighbor for Kozachenko-Leonenko estimator. Thus for each evaluation, we need to collect agent state samples and expert samples for computing the estimators.

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

A.2.2 Inverse Reinforcement Learning Benchmarks (Sec 4.4.2)

Environment: We use the `Hopper-v2`, `Ant-v2`, `HalfCheetah-v2`, `Walker2d-v2` environments from OpenAI Gym.

Expert Samples: We use SAC to train expert policies for each environment. SAC uses the same policy and critic networks, and the learning rate as section A.2.1. We train using a batch size of 100, a replay buffer of size 1 million, and set the temperature parameter α to be 0.2. The policy is trained for 1 million timesteps on Hopper, and for 3 million timesteps on the other environments. All algorithms are tested on 1, 4, and 16 trajectories collected from the expert stochastic policy.

Training Details: We train f-IRL, Behavior Cloning (BC), MaxEntIRL, AIRL, and f-MAX-RKL to imitate the expert using the provided expert trajectories.

We train f-IRL using Algorithm 1. Since we have access to expert samples, we use the practical modification described in section 4.3.4 for training f-IRL, where we feed a mixture of 10 agent and 10 expert trajectories (resampled with replacement from provided expert trajectories) into the reward objective.

SAC uses the same hyperparameters used for training expert policies. Similar to the previous section, we warm-start the SAC policy and critic using trained networks from previous iterations, and train them for 10 episodes. At each iteration, we update the reward parameters once using Adam optimizer. For the reward network of f-IRL and MaxEntIRL, we use the same reward structure as section A.2.1 with the learning rate of 0.0001, and ℓ_2 weight decay of 0.001. We take one gradient step for the reward

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

update.

MaxEntIRL is trained in the standard manner, where the expert samples are used for estimating reward gradient.

For Behavior cloning, we use the expert state-action pairs to learn a stochastic policy that maximizes the likelihood on expert data. The policy network is same as the one used in SAC for training expert policies.

For f-MAX-RKL and AIRL, we tuned the hyperparameters based on the code provided by f-MAX that is used for *state-action* marginal matching in Mujoco benchmarks. For f-MAX-RKL, we fix SAC temperature $\alpha = 0.2$, and tuned reward scale c and gradient penalty coefficient λ suggested by the authors, and found that $c = 0.2, \lambda = 4.0$ worked for {Ant, Hopper, Walker2d} *with* the normalization in each dimension of states and with a replay buffer of size 200000. However, for HalfCheetah, we found it only worked with $c = 2.0, \lambda = 2.0$ *without* normalization in states and with a replay buffer of size 20000. For the other hyperparameters and training schedule, we keep them same as f-MAX original code: e.g. the discriminator is parameterized as a two-layer MLP of hidden size 128 with tanh activation and the output clipped within [-10,10]; the discriminator and policy are alternatively trained once for 100 iterations per 1000 environment timesteps.

For AIRL, we re-implement a version that uses SAC as the underlying RL algorithm for a fair comparison, whereas the original paper uses TRPO. Both the reward and the value model are parameterized as a two-layer MLP of hidden size 256 and use ReLU as the activation function. For SAC training, we tune the learning rates and replay buffer sizes for different environments, but find it cannot work on all environments other than HalfCheetah even after tremendous tuning. For reward and value model training, we tune the learning rate for different environments. These hyper-parameters are summarized in table A.1. We set $\alpha = 1$ in SAC for all environments. For every 1000 environment steps, we alternatively train the policy and the reward/value model once, using a batch size of 100 and 256.

Evaluation: We compare the trained policies by f-IRL, BC, MaxEntIRL, AIRL, and f-MAX-RKL by computing their returns according to the ground truth return on each environment. We report the mean of their performance across 3 seeds.

For the IRL methods, f-IRL, MaxEntIRL, and AIRL, we also evaluate the learned reward functions. We train SAC on the learned rewards, and evaluate the performance

Hyper-parameter	Ant	Hopper	Walker	HalfCheetah
SAC learning rate	$3e-4$	$1e-5$	$1e-5$	$3e-4$
SAC replay buffer size	1000000	1000000	1000000	10000
Reward/Value model learning rate	$1e-4$	$1e-5$	$1e-5$	$1e-4$

Table A.1: AIRL IRL benchmarks task-specific hyper-parameters.

of learned policies according to ground-truth rewards.

A.2.3 Reward Prior for Downstream Hard-exploration Tasks (Sec 4.4.3.1)

Environment: The pointmass environment has 2D square state space with range $[0, 6]^2$, and 2D actions that control the delta movement of the agent in each dimension. The agent starts from the bottom left corner at coordinate $(0, 0)$.

Task Details: We designed a hard-to-explore task for the pointmass. The grid size is 6×6, the agent is always born at $[0, 0]$, and the goal is to reach the region $[5.95, 6] \times [5.95, 6]$. The time horizon is $T = 30$. The agent only receives a reward of 1 if it reaches the goal region. To make the task more difficult, we add two distraction goals: one is at $[5.95, 6] \times [0, 0.05]$, and the other at $[0, 0.05] \times [5.95, 6]$. The agent receives a reward of 0.1 if it reaches one of these distraction goals. Vanilla SAC always converges to reaching one of the distraction goals instead of the real goal.

Training Details: We use SAC as the RL algorithm. We train SAC for 270 episodes, with a batch size of 256, a learning rate of 0.003, and a replay buffer size of 12000. To encourage the exploration of SAC, we use a random policy for the first 100 episodes.

A.2.4 Reward Transfer across Changing Dynamics (Sec 4.4.3.2)

Environment: In this experiment, we use Mujoco to simulate a healthy Ant, and a disabled Ant with two broken legs (Figure A.1). We use the code provided by Fu et al. [18]. Note that this Ant environment is a slightly modified version of the `Ant-v2` available in OpenAI gym.

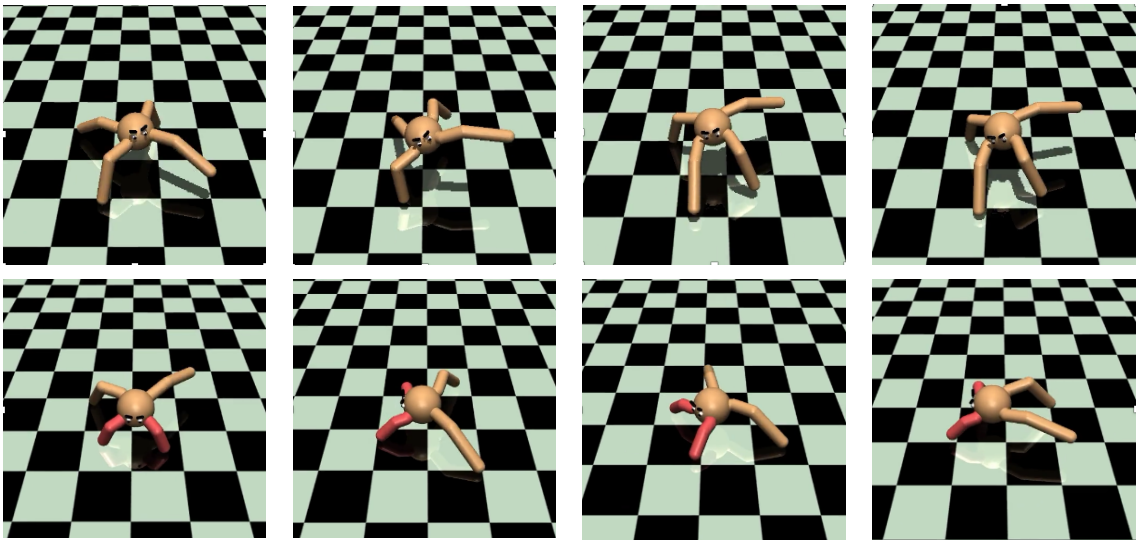


Figure A.1: **Top row**: A healthy Ant executing a forward walk. **Bottom row**: A successful transfer of walking behavior to disabled Ant with 2 legs active. The disabled Ant learns to use the two disabled legs as support and crawl forward, executing a very different gait than previously seen in healthy Ant.

Expert Samples: We use SAC to obtain a forward-running policy for the Ant. We use the same network structure and training parameters as section A.2.2 for training this policy. We use 16 trajectories from this policy as expert demonstrations for the task.

Training Details: We train f-IRL and MaxEntIRL using the same network structure and training parameters as section A.2.2. We also run AIRL, but couldn't match the performance reported in Fu et al. [18].

Evaluation: We evaluate f-IRL and MaxEntIRL by training a policy on their learned rewards using SAC. We report the return of this policy on the disabled Ant environment according to the ground-truth reward for forward-running task. Note that we directly report results for policy transfer using GAIL, and AIRL from Fu et al. [18].

Bibliography

[1] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1, 2004. 2.2, 4.1

[2] Syed Mumtaz Ali and Samuel D Silvey. A general class of coefficients of divergence of one distribution from another. *Journal of the Royal Statistical Society: Series B (Methodological)*, 28(1):131–142, 1966. 4.2, A.1.2

[3] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017. A.1.3, A.1.3

[4] Stuart Armstrong and Sören Mindermann. Occam's razor is insufficient to infer the preferences of irrational agents. *arXiv preprint arXiv:1712.05812*, 2017. 2.3

[5] Christopher G Atkeson and Stefan Schaal. Robot learning from demonstration. In *ICML*, volume 97, pages 12–20. Citeseer, 1997. 4.1

[6] Monica Babes, Vukosi N Marivate, Kaushik Subramanian, and Michael L Littman. Apprenticeship learning about multiple intentions. In *ICML*, 2011. 2.2

[7] Michael Bain and Claude Sammut. A framework for behavioural cloning. In *Machine Intelligence 15*, pages 103–129, 1995. 4.1

[8] Chris Baker, Rebecca Saxe, and Joshua Tenenbaum. Bayesian theory of mind: Modeling joint belief-desire attribution. In *Proceedings of the annual meeting of the cognitive science society*, volume 33, 2011. 2.1

[9] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016. 4.4, A.2.1

[10] Chris Cundy and Daniel Filan. Exploring hierarchy-aware inverse reinforcement learning. *arXiv preprint arXiv:1807.05037*, 2018. 2.2

[11] Gal Dalal, Krishnamurthy Dvijotham, Matej Vecerik, Todd Hester, Cosmin Paduraru, and Yuval Tassa. Safe exploration in continuous action spaces. *arXiv preprint arXiv:1801.08757*, 2018. 4.1

[12] Christos Dimitrakakis and Constantin A Rothkopf. Bayesian multitask inverse

reinforcement learning. In *European workshop on reinforcement learning*, pages 273–284. Springer, 2011. 2.2

[13] Owain Evans and Noah D Goodman. Learning the preferences of bounded agents. In *NIPS Workshop on Bounded Optimality*, volume 6, 2015. 2.1

[14] Owain Evans, Andreas Stuhlmüller, and Noah Goodman. Learning the preferences of ignorant, inconsistent agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016. 2.1

[15] Chelsea Finn, Paul Christiano, Pieter Abbeel, and Sergey Levine. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *arXiv preprint arXiv:1611.03852*, 2016. 2.2

[16] Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *International conference on machine learning*, pages 49–58, 2016. 2.2, 4.3.4

[17] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR, 2017. 2.4

[18] Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*, 2017. 2.2, 4.1, 4.3.3, 4.3.3, 4.4, 2, 4.4.3, A.1.4, A.2.4

[19] Seyed Kamyar Seyed Ghasemipour, Shixiang Gu, and Richard Zemel. Smile: Scalable meta inverse reinforcement learning through context-conditional policies. 2019. 2.4

[20] Seyed Kamyar Seyed Ghasemipour, Richard Zemel, and Shixiang Gu. A divergence minimization perspective on imitation learning methods. *arXiv preprint arXiv:1911.02256*, 2019. 4.1, 4.2, 4.4, 2, 4.4.3, A.2.1

[21] Francesco Giuliari, Irtiza Hasan, Marco Cristani, and Fabio Galasso. Transformer networks for trajectory forecasting, 2020. URL https://arxiv.org/abs/2003.08111. 2.3

[22] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014. 2.2, 4.3.2

[23] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018. 4.4

[24] Dylan Hadfield-Menell, Stuart J Russell, Pieter Abbeel, and Anca Dragan. Cooperative inverse reinforcement learning. In *Advances in neural information*

processing systems, pages 3909–3917, 2016. 4.1

[25] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 4.1

[26] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in neural information processing systems*, pages 4565–4573, 2016. 4.1, 4.4, 4.4.3

[27] Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26):eaau5872, 2019. 4.1

[28] Shervin Javdani, Siddhartha S Srinivasa, and J Andrew Bagnell. Shared autonomy via hindsight optimization. *Robotics science and systems: online proceedings*, 2015, 2015. 3.1

[29] LF Kozachenko and Nikolai N Leonenko. Sample estimate of the entropy of a random vector. *Problemy Peredachi Informatsii*, 23(2):9–16, 1987. A.2.1

[30] Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. Estimating mutual information. *Physical review E*, 69(6):066138, 2004. A.2.1

[31] Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. Rma: Rapid motor adaptation for legged robots. *arXiv preprint arXiv:2107.04034*, 2021. 3.1

[32] Lisa Lee, Benjamin Eysenbach, Emilio Parisotto, Eric Xing, Sergey Levine, and Ruslan Salakhutdinov. Efficient exploration via state marginal matching. *arXiv preprint arXiv:1906.05274*, 2019. 4.1, 4.2, 4.5

[33] Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018. 4.2

[34] Chun-Liang Li, Wei-Cheng Chang, Yu Cheng, Yiming Yang, and Barnabás Póczos. Mmd gan: Towards deeper understanding of moment matching network. In *Advances in Neural Information Processing Systems*, pages 2203–2213, 2017. A.1.3

[35] Chang Liu, Jessica B Hamrick, Jaime F Fisac, Anca D Dragan, J Karl Hedrick, S Shankar Sastry, and Thomas L Griffiths. Goal inference improves objective and perceived performance in human-robot collaboration. *arXiv preprint arXiv:1802.01780*, 2018. 3.1

[36] Fangchen Liu, Zhan Ling, Tongzhou Mu, and Hao Su. State alignment-based imitation learning. *arXiv preprint arXiv:1911.10947*, 2019. 4.4.2

[37] Minghuan Liu, Tairan He, Minkai Xu, and Weinan Zhang. Energy-based imitation learning. *arXiv preprint arXiv:2004.09395*, 2020. 2.2, 4.4.2

[38] Negar Mehr, Roberto Horowitz, and Anca D Dragan. Inferring and assisting

with constraints in shared autonomy. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 6689–6696. IEEE, 2016. 3.1

[39] Bernard Michini and Jonathan P How. Bayesian nonparametric inverse reinforcement learning. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 148–163. Springer, 2012. 2.2

[40] Alessio Monti, Alessia Bertugli, Simone Calderara, and Rita Cucchiara. Dag-net: Double attentive graph neural network for trajectory forecasting, 2020. URL https://arxiv.org/abs/2005.12661. 2.3

[41] Alfred Müller. Integral probability metrics and their generating classes of functions. *Advances in Applied Probability*, pages 429–443, 1997. A.1.3

[42] Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, volume 99, pages 278–287, 1999. 4.3.3, 4.4.3

[43] Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, pages 663–670, 2000. 2.2, 4.1

[44] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018. 2.4

[45] Stefanos Nikolaidis, Keren Gu, Ramya Ramakrishnan, and Julie Shah. Efficient model learning for human-robot collaborative tasks. arxiv, 2014. 3.1

[46] Takayuki Osa, Naohiko Sugita, and Mamoru Mitsuishi. Online trajectory planning and force control for automation of surgical tasks. *IEEE Transactions on Automation Science and Engineering*, 15(2):675–691, 2017. 4.1

[47] Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *Advances in neural information processing systems*, pages 305–313, 1989. 4.1

[48] Ahmed H Qureshi, Byron Boots, and Michael C Yip. Adversarial imitation via variational inverse reinforcement learning. *arXiv preprint arXiv:1809.06404*, 2018. 2.2

[49] Neil Rabinowitz, Frank Perbet, Francis Song, Chiyuan Zhang, SM Ali Eslami, and Matthew Botvinick. Machine theory of mind. In *International conference on machine learning*, pages 4218–4227. PMLR, 2018. 2.1

[50] Deepak Ramachandran and Eyal Amir. Bayesian inverse reinforcement learning. In *IJCAI*, volume 7, pages 2586–2591, 2007. 2.2

[51] Nathan D Ratliff, J Andrew Bagnell, and Martin A Zinkevich. Maximum margin planning. In *Proceedings of the 23rd international conference on Machine learning*, pages 729–736, 2006. 2.2

[52] Siddharth Reddy, Anca D Dragan, and Sergey Levine. Shared autonomy via deep reinforcement learning. *arXiv preprint arXiv:1802.01744*, 2018. 3.1

[53] Siddharth Reddy, Anca D Dragan, and Sergey Levine. Where do you think you're going?: Inferring beliefs about dynamics from behavior. *arXiv preprint arXiv:1805.08010*, 2018. 2.1

[54] Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 661–668, 2010. 4.1

[55] Stuart Russell. Learning agents for uncertain environments. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 101–103, 1998. 2.2, 4.1

[56] Dorsa Sadigh, Anca D Dragan, Shankar Sastry, and Sanjit A Seshia. Active preference-based learning of reward functions. 2017. 2.2

[57] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897, 2015. 4.4

[58] Rohin Shah, Noah Gundotra, Pieter Abbeel, and Anca Dragan. On the feasibility of learning, rather than assuming, human biases for reward inference. In *International Conference on Machine Learning*, pages 5670–5679. PMLR, 2019. 2.3

[59] Shashank Singh and Barnabás Póczos. Analysis of k-nearest neighbor distances with application to entropy estimation. *arXiv preprint arXiv:1603.08578*, 2016. A.2.1

[60] Jake Snell, Kevin Swersky, and Richard S Zemel. Prototypical networks for few-shot learning. *arXiv preprint arXiv:1703.05175*, 2017. 2.4

[61] Adrian Šošić, Abdelhak M Zoubir, and Heinz Koeppl. Inverse reinforcement learning via nonparametric subgoal modeling. In *2018 AAAI Spring Symposium Series*, 2018. 2.2

[62] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012. 4.4

[63] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 2.3

[64] Greg Ver Steeg. Non-parametric entropy estimation toolbox (npeet). 2000. A.2.1

[65] Cédric Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008. A.1.3

[66] Ruohan Wang, Carlo Ciliberto, Pierluigi Amadori, and Yiannis Demiris. Random

expert distillation: Imitation learning via expert policy support estimation. *arXiv preprint arXiv:1905.06750*, 2019. 2.2

[67] Heinz Wimmer and Josef Perner. Beliefs about beliefs: Representation and constraining function of wrong beliefs in young children's understanding of deception. *Cognition*, 13(1):103–128, 1983. 2.1

[68] Markus Wulfmeier, Peter Ondruska, and Ingmar Posner. Maximum entropy deep inverse reinforcement learning. *arXiv preprint arXiv:1507.04888*, 2015. 2.2

[69] Kelvin Xu, Ellis Ratner, Anca Dragan, Sergey Levine, and Chelsea Finn. Learning a prior over intent via meta-inverse reinforcement learning. In *International Conference on Machine Learning*, pages 6952–6962. PMLR, 2019. 2.4

[70] Lantao Yu, Tianhe Yu, Chelsea Finn, and Stefano Ermon. Meta-inverse reinforcement learning with probabilistic context variables. *arXiv preprint arXiv:1909.09314*, 2019. 2.4

[71] Brian D Ziebart. Modeling purposeful adaptive behavior with the principle of maximum causal entropy. 2010. 3.1, 4.2, A.1.1

[72] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008. 2.2, 4.4