Where Do *You* Think You're Going?: Inferring Beliefs about Dynamics from Behavior

Siddharth Reddy, Anca D. Dragan, Sergey Levine
Department of Electrical Engineering and Computer Science
University of California, Berkeley
{sgr,anca,svlevine}@berkeley.edu

Abstract

Inferring intent from observed behavior has been studied extensively within the frameworks of Bayesian inverse planning and inverse reinforcement learning. These methods infer a goal or reward function that best explains the actions of the observed agent, typically a human demonstrator. Another agent can use this inferred intent to predict, imitate, or assist the human user. However, a central assumption in inverse reinforcement learning is that the demonstrator is close to optimal. While models of suboptimal behavior exist, they typically assume that suboptimal actions are the result of some type of random noise or a known cognitive bias, like temporal inconsistency. In this paper, we take an alternative approach, and model suboptimal behavior as the result of internal model misspecification: the reason that user actions might deviate from near-optimal actions is that the user has an incorrect set of beliefs about the rules – the dynamics – governing how actions affect the environment. Our insight is that while demonstrated actions may be suboptimal in the real world, they may actually be near-optimal with respect to the user's *internal* model of the dynamics. By estimating these internal beliefs from observed behavior, we arrive at a new method for inferring intent. We demonstrate in simulation and in a user study with 12 participants that this approach enables us to more accurately model human intent, and can be used in a variety of applications, including offering assistance in a shared autonomy framework and inferring human preferences.

1 Introduction

Characterizing the drive behind human actions in the form of a goal or reward function is broadly useful for predicting future behavior, imitating human actions in new situations, and augmenting human control with automated assistance – critical functions in a wide variety of applications, including pedestrian motion prediction [57], virtual character animation [38], and robotic teleoperation [35]. For example, remotely operating a robotic arm to grasp objects can be challenging for a human user due to unfamiliar or unintuitive dynamics of the physical system and control interface. Existing frameworks for assistive teleoperation and shared autonomy aim to help users perform such tasks [35, 29, 46, 8, 45]. These frameworks typically rely on existing methods for intent inference in the sequential decision-making context, which use Bayesian inverse planning or inverse reinforcement learning to learn the user's goal or reward function from observed control demonstrations. These methods typically assume that user actions are near-optimal, and deviate from optimality due to random noise [56], specific cognitive biases in planning [16, 15, 4], or risk sensitivity [33].

See `https://sites.google.com/view/inferring-internal-dynamics` for supplementary materials, including videos and code.

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

The key insight in this paper is that suboptimal behavior can also arise from a mismatch between the dynamics of the real world and the user's internal beliefs of the dynamics, and that a user policy that appears suboptimal in the real world may actually be near-optimal with respect to the user's internal dynamics model. As resource-bounded agents living in an environment of dazzling complexity, humans rely on intuitive theories of the world to guide reasoning and planning [21, 26]. Humans leverage internal models of the world for motor control [53, 30, 14, 34, 49], goal-directed decision making [7], and representing the mental states of other agents [39]. Simplified internal models can systematically deviate from the real world, leading to suboptimal behaviors that have unintended consequences, like hitting a tennis ball into the net or skidding on an icy road. For example, a classic study in cognitive science shows that human judgments about the physics of projectile motion are closer to Aristotelian impetus theory than to true Newtonian dynamics – in other words, people tend to ignore or underestimate the effects of inertia [11]. Characterizing the gap between internal models and reality by modeling a user's internal predictions of the effects of their actions allows us to better explain observed user actions and infer their intent.

The main contribution of this paper is a new algorithm for intent inference that first estimates a user's internal beliefs of the dynamics of the world using observations of how they act to perform known tasks, then leverages the learned internal dynamics model to infer intent on unknown tasks. In contrast to the closest prior work [28, 22], our method scales to problems with high-dimensional, continuous state spaces and nonlinear dynamics. Our internal dynamics model estimation algorithm assumes the user takes actions with probability proportional to their exponentiated soft Q-values. We fit the parameters of the internal dynamics model to maximize the likelihood of observed user actions on a set of tasks with known reward functions, by tying the internal dynamics to the soft Q function via the soft Bellman equation. At test time, we use the learned internal dynamics model to predict the user's desired next state given their current state and action input.

We run experiments first with simulated users, testing that we can recover the internal dynamics, even in MDPs with a continuous state space that would otherwise be intractable for prior methods. We then run a user study with 12 participants in which humans play the Lunar Lander game (screenshot in Figure 1). We recover a dynamics model that explains user actions better than the real dynamics, which in turn enables us to assist users in playing the game by transferring their control policy from the recovered internal dynamics to the real dynamics.

2 Background

Inferring intent in sequential decision-making problems has been heavily studied under the framework of inverse reinforcement learning (IRL), which we build on in this work. The aim of IRL is to learn a user's reward function from observed control demonstrations. IRL algorithms are not directly applicable to our problem of learning a user's beliefs about the dynamics of the environment, but they provide a helpful starting point for thinking about how to extract hidden properties of a user from observations of how they behave.

In our work, we build on the maximum causal entropy (MaxCausalEnt) IRL framework [55, 6, 44, 36, 28]. In an MDP with a discrete action space $\mathcal{A}$, the human demonstrator is assumed to follow a policy π that maximizes an entropy-regularized reward $R(s, a, s')$ under dynamics $T(s'|s, a)$. Equivalently,

$$\pi(a|s) \triangleq \frac{\exp\left(Q(s,a)\right)}{\sum_{a' \in \mathcal{A}} \exp\left(Q(s,a')\right)}, \tag{1}$$

where Q is the soft Q function, which satisfies the soft Bellman equation [55],

$$Q(s,a) = \mathbb{E}_{s' \sim T(\cdot|s,a)}\left[R(s,a,s') + \gamma V(s')\right], \tag{2}$$

with V the soft value function,

$$V(s) \triangleq \log\left(\sum_{a \in \mathcal{A}} \exp\left(Q(s,a)\right)\right). \tag{3}$$

Prior work assumes T is the true dynamics of the real world, and fits a model of the reward R that maximizes the likelihood (given by Equation 1) of some observed demonstrations of the user acting in the real world. In our work, we assume access to a set of training tasks for which the rewards R are known, fit a model of the internal dynamics T that is allowed to deviate from the real dynamics, then use the recovered dynamics to infer intent (e.g., rewards) in new tasks.

3 Internal Dynamics Model Estimation

We split up the problem of intent inference into two parts: learning the internal dynamics model from user demonstrations on known tasks (the topic of this section), and using the learned internal model to infer intent on unknown tasks (discussed later in Section 4). We assume that the user's internal dynamics model is stationary, which is reasonable for problems like robotic teleoperation when the user has some experience practicing with the system but still finds it unintuitive or difficult to control. We also assume that the real dynamics are known ex-ante or learned separately.

Our aim is to recover a user's implicit beliefs about the dynamics of the world from observations of how they act to perform a set of tasks. The key idea is that, when their internal dynamics model deviates from the real dynamics, we can no longer simply fit a dynamics model to observed state transitions. Standard dynamics learning algorithms typically assume access to (s, a, s') examples, with (s, a) features and s' labels, that can be used to train a classification or regression model $p(s'|s, a)$ using supervised learning. In our setting, we instead have (s, a) pairs that indirectly encode the state transitions that the user expected to happen, but did not necessarily occur, because the user's internal model predicted different outcomes s' than those that actually occurred in the real world. Our core assumption is that the user's policy is near-optimal with respect to the unknown internal dynamics model. To this end, we propose a new algorithm for learning the internal dynamics from action demonstrations: inverse soft Q-learning.

3.1 Inverse Soft Q-Learning

The key idea behind our algorithm is that we can fit a parametric model of the internal dynamics model T that maximizes the likelihood of observed action demonstrations on a set of training tasks with known rewards by using the soft Q function as an intermediary.[1] We tie the internal dynamics T to the soft Q function via the soft Bellman equation (Equation 2), which ensures that the soft Q function is induced by the internal dynamics T. We tie the soft Q function to action likelihoods using Equation 1, which encourages the soft Q function to explain observed actions. We accomplish this by solving a constrained optimization problem in which the demonstration likelihoods appear in the objective and the soft Bellman equation appears in the constraints.

Formulating the optimization problem. Assume the action space $\mathcal{A}$ is discrete.[2] Let $i \in \{1, 2, ..., n\}$ denote the training task, $R_i(s, a, s')$ denote the known reward function for task i, T denote the unknown internal dynamics, and Q_i denote the unknown soft Q function for task i. We represent Q_i using a function approximator $Q_{\boldsymbol{\theta_i}}$ with parameters $\boldsymbol{\theta_i}$, and the internal dynamics using a function approximator $T_{\boldsymbol{\phi}}$ parameterized by $\boldsymbol{\phi}$. Note that, while each task merits a separate soft Q function since each task has different rewards, all tasks share the same internal dynamics.

Recall the soft Bellman equation (Equation 2), which constrains Q_i to be the soft Q function for rewards R_i and internal dynamics T. An equivalent way to express this condition is that Q_i satisfies $\delta_i(s, a) = 0 \; \forall s, a$, where δ_i is the soft Bellman error:

$$\delta_i(s, a) \triangleq Q_i(s, a) - \int_{s' \in \mathcal{S}} T(s'|s, a) \left(R_i(s, a, s') + \gamma V_i(s')\right) \mathrm{d}s'. \tag{4}$$

We impose the same condition on $Q_{\boldsymbol{\theta_i}}$ and $T_{\boldsymbol{\phi}}$, i.e., $\delta_{\boldsymbol{\theta_i}, \boldsymbol{\phi}}(s, a) = 0 \; \forall s, a$. We assume our representations are expressive enough that there exist values of $\boldsymbol{\theta_i}$ and $\boldsymbol{\phi}$ that satisfy the condition. We fit parameters $\boldsymbol{\theta_i}$ and $\boldsymbol{\phi}$ to maximize the likelihood of the observed demonstrations while respecting the soft Bellman equation by solving the constrained optimization problem

$$\begin{aligned} \underset{\{\boldsymbol{\theta_i}\}_{i=1}^n, \boldsymbol{\phi}}{\text{minimize}} \; & \sum_{i=1}^{n} \sum_{(s,a) \in \mathcal{D}_i^{\text{demo}}} - \log \pi_{\boldsymbol{\theta_i}}(a|s) \\ \text{subject to} \; & \delta_{\boldsymbol{\theta_i}, \boldsymbol{\phi}}(s, a) = 0 \; \forall i \in \{1, 2, ..., n\}, s \in \mathcal{S}, a \in \mathcal{A}, \end{aligned} \tag{5}$$

where $\mathcal{D}_i^{\text{demo}}$ are the demonstrations for task i, and $\pi_{\boldsymbol{\theta_i}}$ denotes the action likelihood given by $Q_{\boldsymbol{\theta_i}}$ and Equation 1.

[1] Our algorithm can in principle learn from demonstrations even when the rewards are unknown, but in practice we find that this relaxation usually makes learning the correct internal dynamics too difficult.

[2] We assume a discrete action space to simplify our exposition and experiments. Our algorithm can be extended to handle MDPs with a continuous action space using existing sampling methods [25].

Solving the optimization problem. We use the penalty method [5] to approximately solve the constrained optimization problem described in Equation 5, which recasts the problem as unconstrained optimization of the cost function

$$c(\boldsymbol{\theta}, \boldsymbol{\phi}) \triangleq \sum_{i=1}^{n} \sum_{(s,a) \in \mathcal{D}_i^{\text{demo}}} -\log \pi_{\boldsymbol{\theta_i}}(a|s) + \frac{\rho}{2} \sum_{i=1}^{n} \int_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} (\delta_{\boldsymbol{\theta_i}, \boldsymbol{\phi}}(s,a))^2 \mathrm{d}s, \tag{6}$$

where ρ is a constant hyperparameter, $\pi_{\boldsymbol{\theta_i}}$ denotes the action likelihood given by $Q_{\boldsymbol{\theta_i}}$ and Equation 1, and $\delta_{\boldsymbol{\theta_i}, \boldsymbol{\phi}}$ denotes the soft Bellman error, which relates $Q_{\boldsymbol{\theta_i}}$ to $T_{\boldsymbol{\phi}}$ through Equation 4.

For MDPs with a discrete state space $\mathcal{S}$, we minimize the cost as is. MDPs with a continuous state space present two challenges: (1) an intractable integral over states in the sum over penalty terms, and (2) integrals over states in the expectation terms of the soft Bellman errors δ (recall Equation 4). To tackle (1), we resort to constraint sampling [10]; specifically, randomly sampling a subset of state-action pairs $\mathcal{D}_i^{\text{samp}}$ from rollouts of a random policy in the real world. To tackle (2), we choose a deterministic model of the internal dynamics $T_{\boldsymbol{\phi}}$, which simplifies the integral over next states in Equation 4 to a single term[3].

In our experiments, we minimize the objective in Equation 6 using Adam [31]. We use a mix of tabular representations, structured linear models, and relatively shallow multi-layer perceptrons to model $Q_{\boldsymbol{\theta_i}}$ and $T_{\boldsymbol{\phi}}$. In the tabular setting, $\boldsymbol{\theta_i}$ is a table of numbers with a separate entry for each state-action pair, and $\boldsymbol{\phi}$ can be a table with an entry between 0 and 1 for each state-action-state triple. For linear and neural network representations, $\boldsymbol{\theta_i}$ and $\boldsymbol{\phi}$ are sets of weights.

3.2 Regularizing the Internal Dynamics Model

One issue with our approach to estimating the internal dynamics is that there tend to be multiple feasible internal dynamics models that explain the demonstration data equally well, which makes the correct internal dynamics model difficult to identify. We propose two different solutions to this problem: collecting demonstrations on multiple training tasks, and imposing a prior on the learned internal dynamics that encourages it to be similar to the real dynamics.

Multiple training tasks. If we only collect demonstrations on $n = 1$ training tasks, then at any given state s and action a, the recovered internal dynamics may simply assign a likelihood of one to the next state s' that maximizes the reward function $R_1(s, a, s')$ of the single training task. Intuitively, if our algorithm is given user demonstrations on only one task, then the user's actions can be explained by an internal dynamics model that always predicts the best possible next state for that one task (e.g., the target in a navigation task), no matter the current state or user action. We can mitigate this problem by collecting demonstrations on $n > 1$ training tasks, which prevents degenerate solutions by forcing the internal dynamics to be consistent with a diverse set of user policies.

Action intent prior. In our experiments, we also explore another way to regularize the learned internal dynamics: imposing the prior that the learned internal dynamics $T_{\boldsymbol{\phi}}$ should be similar to the known real dynamics T^{real} by restricting the support of $T_{\boldsymbol{\phi}}(\cdot|s,a)$ to states s' that are reachable in the real dynamics. Formally,

$$T_{\boldsymbol{\phi}}(s'|s,a) \triangleq \sum_{a^{\text{int}} \in \mathcal{A}} T^{\text{real}}(s'|s,a^{\text{int}}) f_{\boldsymbol{\phi}}(a^{\text{int}}|s,a) \tag{7}$$

where a is the user's action, a^{int} is the user's intended action, and $f_{\boldsymbol{\phi}} : \mathcal{S} \times \mathcal{A}^2 \rightarrow [0,1]$ captures the user's 'action intent' – the action they would have taken if they had perfect knowledge of the real dynamics. This prior changes the structure of our internal dynamics model to predict the user's intended action with respect to the real dynamics, rather than directly predicting their intended next state. Note that, when we use this action intent prior, $T_{\boldsymbol{\phi}}$ is no longer directly modeled. Instead, we model $f_{\boldsymbol{\phi}}$ and use Equation 7 to compute $T_{\boldsymbol{\phi}}$.

In our experiments, we examine the effects of employing multiple training tasks and imposing the action intent prior, together and in isolation.

[3] Another potential solution is sampling states to compute a Monte Carlo estimate of the integral.

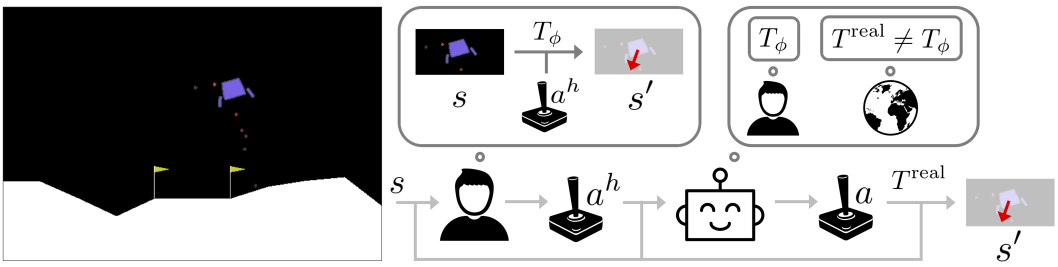

Figure 1: A high-level schematic of our internal-to-real dynamics transfer algorithm for shared autonomy, which uses the internal dynamics model learned by our method to assist the user with an unknown control task; in this case, landing the lunar lander between the flags. The user's actions are assumed to be consistent with their internal beliefs about the dynamics T_ϕ, which differ from the real dynamics T^{real}. Our system models the internal dynamics to determine where the user is trying to go next, then acts to get there.

4 Using Learned Internal Dynamics Models

The ability to learn internal dynamics models from demonstrations is broadly useful for intent inference. In our experiments, we explore two applications: (1) shared autonomy, in which a human and robot collaborate to solve a challenging real-time control task, and (2) learning the reward function of a user who generates suboptimal demonstrations due to internal model misspecification. In (1), intent is formalized as the user's desired next state, while in (2), the user's intent is represented by their reward function.

4.1 Shared Autonomy via Internal-to-Real Dynamics Transfer

Many control problems involving human users are challenging for autonomous agents due to partial observability and imprecise task specifications, and are also challenging for humans due to constraints such as bounded rationality [48] and physical reaction time. Shared autonomy combines human and machine intelligence to perform control tasks that neither can on their own, but existing methods have the basic requirement that the machine either needs a description of the task or feedback from the user, e.g., in the form of rewards [29, 8, 45]. We propose an alternative algorithm that assists the user without knowing their reward function by leveraging the internal dynamics model learned by our method. The key idea is formalizing the user's intent as their desired next state. We use the learned internal dynamics model to infer the user's desired next state given their current state and control input, then execute an action that will take the user to the desired state under the real dynamics; essentially, transferring the user's policy from the internal dynamics to the real dynamics, akin to simulation-to-real transfer for robotic control [13]. See Figure 1 for a high-level schematic of this process.

Equipped with the learned internal dynamics model $T_{\boldsymbol{\phi}}$, we perform internal-to-real dynamics transfer by observing the user's action input, computing the induced distribution over next states using the internal dynamics, and executing an action that induces a similar distribution over next states in the real dynamics. Formally, for user control input a_t^h and state s_t, we execute action a_t, where

$$a_t \triangleq \arg\min_{a \in \mathcal{A}} D_{\text{KL}}(T_{\boldsymbol{\phi}}(s_{t+1}|s_t, a_t^h) \parallel T^{\text{real}}(s_{t+1}|s_t, a)) \tag{8}$$

4.2 Learning Rewards from Misguided User Demonstrations

Most existing inverse reinforcement learning algorithms assume that the user's internal dynamics are equivalent to the real dynamics, and learn their reward function from near-optimal demonstrations. We explore a more realistic setting in which the user's demonstrations are suboptimal due to a mismatch between their internal dynamics and the real dynamics. Users are 'misguided' in that their behavior is suboptimal in the real world, but near-optimal with respect to their internal dynamics. In this setting, standard IRL algorithms that do not distinguish between the internal and the real dynamics learn incorrect reward functions. Our method can be used to learn the internal dynamics, then explicitly incorporate the learned internal dynamics into an IRL algorithm's behavioral model of the user.

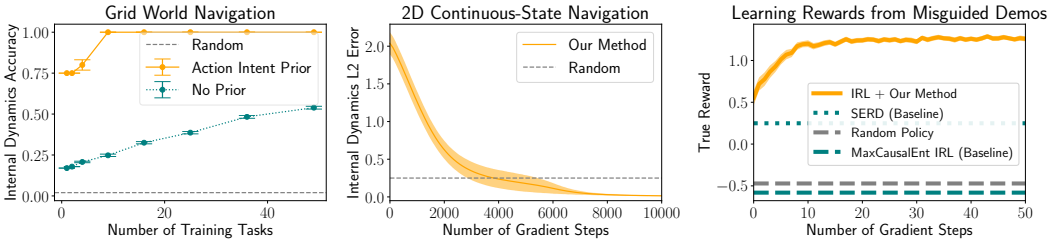

Figure 2: Left, Center: Error bars show standard error on ten random seeds. Our method learns accurate internal dynamics models, the regularization methods in Section 3.2 increase accuracy, and the approximations for continuous-state MDPs in Section 3.1 do not compromise accuracy. **Right:** Error regions show standard error on ten random tasks and ten random seeds each. Our method learns an internal dynamics model that enables MaxCausalEnt IRL to learn rewards from misguided user demonstrations.

In our experiments, we instantiate prior work with MaxCausalEnt IRL [55], which inverts the behavioral model from Equation 1 to infer rewards from demonstrations. We adapt it to our setting, in which the real dynamics are known and the internal dynamics are either learned (separately by our algorithm) or assumed to be the same as the known real dynamics. MaxCausalEnt IRL cannot learn the user's reward function from misguided demonstrations when it makes the standard assumption that the internal dynamics are equal to the real dynamics, but can learn accurate rewards when it instead uses the learned internal dynamics model produced by our algorithm.

5 User Study and Simulation Experiments

The purpose of our experiments is two-fold: (1) to test the correctness of our algorithm, and (2) to test our core assumption that a human user's internal dynamics can be different from the real dynamics, and that our algorithm can learn an internal dynamics model that is useful for assisting the user through internal-to-real dynamics transfer. To accomplish (1), we perform three simulated experiments that apply our method to shared autonomy (see Section 4.1) and to learning rewards from misguided user demonstrations (see Section 4.2). In the shared autonomy experiments, we first use a tabular grid world navigation task to sanity-check our algorithm and analyze the effects of different regularization choices from Section 3.2. We then use a continuous-state 2D navigation task to test our method's ability to handle continuous observations using the approximations described in Section 3.1. In the reward learning experiment, we use the grid world environment to compare the performance of MaxCausalEnt IRL [55] when it assumes the internal dynamics are the same as the real dynamics to when it uses the internal dynamics learned by our algorithm. To accomplish (2), we conduct a user study in which 12 participants play the Lunar Lander game (see Figure 1) with and without internal-to-real dynamics transfer assistance. We summarize these experiments in Sections 5.1 and 5.2. Further details are provided in Section 9.1 of the appendix.

5.1 Simulation Experiments

Shared autonomy. The grid world provides us with a domain where exact solutions are tractable, which enables us to verify the correctness of our method and compare the quality of the approximation in Section 3.1 with an exact solution to the learning problem. The continuous task provides a more challenging domain where exact solutions via dynamic programming are intractable. In each setting, we simulate a user with an internal dynamics model that is severely biased away from the real dynamics of the simulated environment. The simulated user's policy is near-optimal with respect to their internal dynamics, but suboptimal with respect to the real dynamics. Figure 2 (left and center plots) provides overall support for the hypothesis that our method can effectively learn tabular and continuous representations of the internal dynamics for MDPs with discrete and continuous state spaces. The learned internal dynamics models are accurate with respect to the ground truth internal dynamics, and internal-to-real dynamics transfer successfully assists the simulated users. The learned internal dynamics model becomes more accurate as we increase the number of training tasks, and the action intent prior (see Section 3.2) increases accuracy when the internal dynamics are similar to the real dynamics. These results confirm that our approximate algorithm is correct and yields solutions

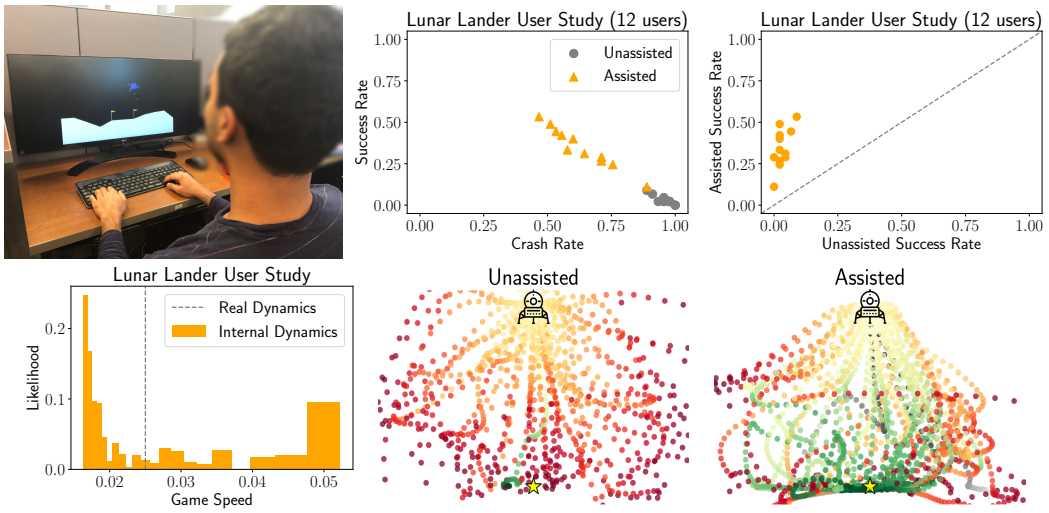

Figure 3: Human users find the default game environment – the real dynamics – to be difficult and unintuitive, as indicated by their poor performance in the unassisted condition (top center and right plots) and their subjective evaluations (in Table 1). Our method observes suboptimal human play in the default environment, learns a setting of the game physics under which the observed human play would have been closer to optimal, then performs internal-to-real dynamics transfer to assist human users in achieving higher success rates and lower crash rates (top center and right plots). The learned internal dynamics has a slower game speed than the real dynamics (bottom left plot). The bottom center and right plots show successful (green) and failed (red) trajectories in the unassisted and assisted conditions.

that do not significantly deviate from those of an exact algorithm. Further results and experimental details are discussed in Sections 9.1.1 and 9.1.2 of the appendix.

Learning rewards from misguided user demonstrations. Standard IRL algorithms, such as MaxCausalEnt IRL [55], can fail to learn rewards from user demonstrations that are 'misguided', i.e., systematically suboptimal in the real world but near-optimal with respect to the user's internal dynamics. Our algorithm can learn the internal dynamics model, and we can then explicitly incorporate the learned internal dynamics into the MaxCausalEnt IRL algorithm to learn accurate rewards from misguided demonstrations. We assess this method on a simulated grid world navigation task. Figure 2 (right plot) supports our claim that standard IRL is ineffective at learning rewards from misguided user demonstrations. After using our algorithm to learn the internal dynamics and explicitly incorporating the learned internal dynamics into an IRL algorithm's model of the user, we see that it's possible to recover accurate rewards from these misguided demonstrations. Additional information on our experimental setup is available in Section 9.1.3 of the appendix.

In addition to comparing to the standard MaxCausalEnt IRL baseline, we also conducted a comparison (shown in Figure 2) with a variant of the Simultaneous Estimation of Rewards and Dynamics (SERD) algorithm [28] that simultaneously learns rewards and the internal dynamics instead of assuming that the internal dynamics are equivalent to the real dynamics. This baseline performs better than random, but still much worse than our method. This result is supported by the theoretical analysis in Armstrong et al. [2], which characterizes the difficulty of simultaneously deducing a human's rationality – in our case, their internal dynamics model – and their rewards from demonstrations.

5.2 User Study on the Lunar Lander Game

Our previous experiments were conducted with simulated expert behavior, which allowed us to control the corruption of the internal dynamics. However, it remains to be seen whether this model of suboptimality effectively reflects real human behavior. We test this hypothesis in the next experiment, which evaluates whether our method can learn the internal dynamics accurately enough to assist real users through internal-to-real dynamics transfer.

Task description. We use the Lunar Lander game from OpenAI Gym [9] (screenshot in Figure 1) to evaluate our algorithm with human users. The objective of the game is to land on the ground,

Table 1: Subjective evaluations of the Lunar Lander user study from 12 participants. Means reported below for responses on a 7-point Likert scale, where 1 = Strongly Disagree, 4 = Neither Disagree nor Agree, and 7 = Strongly Agree. p-values from a one-way repeated measures ANOVA with the presence of assistance as a factor influencing responses.

	p-value	Unassisted	Assisted
I enjoyed playing the game	$< .001$	3.92	**5.92**
I improved over time	$< .0001$	3.08	**5.83**
I didn't crash	$< .001$	1.17	**3.00**
I didn't fly out of bounds	$< .05$	1.67	**3.08**
I didn't run out of time	$> .05$	5.17	6.17
I landed between the flags	$< .001$	1.92	**4.00**
I understood how to complete the task	$< .05$	6.42	**6.75**
I intuitively understood the physics of the game	$< .01$	4.58	**6.00**
My actions were carried out	$> .05$	4.83	5.50
My intended actions were carried out	$< .01$	2.75	**5.25**

without crashing or flying out of bounds, using two lateral thrusters and a main engine. The action space $\mathcal{A}$ consists of six discrete actions. The state $s \in \mathbb{R}^9$ encodes position, velocity, orientation, and the location of the landing site, which is one of nine values corresponding to $n = 9$ distinct tasks. The physics of the game are forward-simulated by a black-box function that takes as input seven hyperparameters, which include engine power and game speed. We manipulate whether or not the user receives internal-to-real dynamics transfer assistance using an internal dynamics model trained on their unassisted demonstrations. The dependent measures are the success and crash rates in each condition. The task and evaluation protocol are discussed further in Section 9.2 of the appendix.

Analysis. In the default environment, users appear to play as though they underestimate the strength of gravity, which causes them to crash into the ground frequently (see the supplementary videos). Figure 3 (bottom left plot) shows that our algorithm learns an internal dynamics model characterized by a slower game speed than the real dynamics, which makes sense since a slower game speed induces smaller forces and slower motion – conditions under which the users' action demonstrations would have been closer to optimal. These results support our claim that our algorithm can learn an internal dynamics model that explains user actions better than the real dynamics.

When unassisted, users often crash or fly out of bounds due to the unintuitive nature of the thruster controls and the relatively fast pace of the game. Figure 3 (top center and right plots) shows that users succeed significantly more often and crash significantly less often when assisted by internal-to-real dynamics transfer (see Section 9.2 of the appendix for hypothesis tests). The assistance makes the system feel easier to control (see the subjective evaluations in Table 1 of the appendix), less likely to tip over (see the supplementary videos), and move more slowly in response to user actions (assistance led to a 30% decrease in average speed). One of the key advantages of assistance was its positive effect on the rate at which users were able to switch between different actions: on average, unassisted users performed 18 actions per minute (APM), while assisted users performed 84 APM. Quickly switching between firing various thrusters enabled assisted users to better stabilize flight. These results demonstrate that the learned internal dynamics can be used to effectively assist the user through internal-to-real dynamics transfer, which in turn gives us confidence in the accuracy of the learned internal dynamics. After all, we cannot measure the accuracy of the learned internal dynamics by comparing it to the ground truth internal dynamics, which is unknown for human users.

6 Related Work

The closest prior work in intent inference and action understanding comes from inverse planning [3] and inverse reinforcement learning [37], which use observations of a user's actions to estimate the user's goal or reward function. We take a fundamentally different approach to intent inference: using action observations to estimate the user's beliefs about the world dynamics.

The simultaneous estimation of rewards and dynamics (SERD) instantiation of MaxCausalEnt IRL [28] aims to improve the sample efficiency of IRL by forcing the learned real dynamics model to explain observed state transitions as well as actions. The framework includes terms for the

demonstrator's beliefs of the dynamics, but the overall algorithm and experiments of Herman et al. [28] constrain those beliefs to be the same as the real dynamics. Our goal is to learn an internal dynamics model that may deviate from the real dynamics. To this end, we propose two new internal dynamics regularization techniques, multi-task training and the action intent prior (see Section 3.2), and demonstrate their utility for learning an internal dynamics model that differs from the real dynamics (see Section 5.1). We also conduct a user experiment that shows human actions in a game environment can be better explained by a learned internal dynamics model than by the real dynamics, and that augmenting user control with internal-to-real dynamics transfer results in improved game play. Furthermore, the SERD algorithm is well-suited to MDPs with a discrete state space, but intractable for continuous state spaces. Our method can be applied to MDPs with a continuous state space, as shown in Sections 5.1 and 5.2.

Golub et al. [22] propose an internal model estimation (IME) framework for brain-machine interface (BMI) control that learns an internal dynamics model from control demonstrations on tasks with linear-Gaussian dynamics and quadratic reward functions. Our work is (1) more general in that it places no restrictions on the functional form of the dynamics or the reward function, and (2) does not assume sensory feedback delay, which is the fundamental premise of using IME for BMI control.

Rafferty et al. [43, 41, 42] use an internal dynamics learning algorithm to infer a student's incorrect beliefs in online learning settings like educational games, and leverage the inferred beliefs to generate personalized hints and feedback. Our algorithm is more general in that it is capable of learning continuous parameters of the internal dynamics, whereas the cited work is only capable of identifying the internal dynamics given a discrete set of candidate models.

Modeling human error has a rich history in the behavioral sciences. Procrastination and other time-inconsistent human behaviors have been characterized as rational with respect to a cost model that discounts the cost of future action relative to that of immediate action [1, 32]. Systematic errors in human predictions about the future have been partially explained by cognitive biases like the availability heuristic and regression to the mean [50]. Imperfect intuitive physics judgments have been characterized as approximate probabilistic inferences made by a resource-bounded observer [26]. We take an orthogonal approach in which we assume that suboptimal behavior is primarily caused by incorrect beliefs of the dynamics, rather than uncertainty or biases in planning and judgment.

Humans are resource-bounded agents that must take into account the computational cost of their planning algorithm when selecting actions [24]. One way to trade-off the ability to find high-value actions for lower computational cost is to plan using a simplified, low-dimensional model of the dynamics [27, 19]. Evidence from the cognitive science literature suggests humans find it difficult to predict the motion of objects when multiple information dimensions are involved [40]. Thus, we arrive at an alternative explanation for why humans may behave near-optimally with respect to a dynamics model that differs from the real dynamics: even if users have perfect knowledge of the real dynamics, they may not have the computational resources to plan under the real dynamics, and instead choose to plan using a simplified model.

7 Discussion

Limitations. Although our algorithm models the soft Q function with arbitrary neural network parameterizations, the internal dynamics parameterizations we use are smaller, with at most seven parameters for continuous tasks. Increasing the number of dynamics parameters would require a better approach to regularization than those proposed in Section 3.2.

Summary. We contribute an algorithm that learns a user's implicit beliefs about the dynamics of the environment from demonstrations of their suboptimal behavior in the real environment. Simulation experiments and a small-scale user study demonstrate the effectiveness of our method at recovering a dynamics model that explains human actions, as well as its utility for applications in shared autonomy and inverse reinforcement learning.

Future work. The ability to learn internal dynamics models from demonstrations opens the door to new directions of scientific inquiry, like estimating young children's intuitive theories of physics and psychology without eliciting verbal judgments [52, 18, 23]. It also enables applications that involve intent inference, including adaptive brain-computer interfaces for prosthetic limbs [12, 47] that help users perform control tasks that are difficult to fully specify.

8 Acknowledgements

We would like to thank Oleg Klimov for open-sourcing his implementation of the Lunar Lander game, which was originally developed by Atari in 1979, and inspired by the lunar modules built in the 1960s and 70s for the Apollo space program. We would also like to thank Eliezer Yudkowsky for the fanfiction novel, Harry Potter and the Methods of Rationality – Harry's misadventure with the rocket-assisted broomstick in chapter 59 inspired us to try to close the gap between intuitive physics and the real world. This work was supported in part by a Berkeley EECS Department Fellowship for first-year Ph.D. students, Berkeley DeepDrive, computational resource donations from Amazon, NSF IIS-1700696, and AFOSR FA9550-17-1-0308.

References

[1] George A Akerlof. Procrastination and obedience. *The American Economic Review*, 81(2):1–19, 1991.

[2] Stuart Armstrong and Sören Mindermann. Impossibility of deducing preferences and rationality from human policy. *arXiv preprint arXiv:1712.05812*, 2017.

[3] Chris L Baker, Rebecca Saxe, and Joshua B Tenenbaum. Action understanding as inverse planning. *Cognition*, 113(3):329–349, 2009.

[4] Leon Bergen, Owain Evans, and Joshua Tenenbaum. Learning structured preferences. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 32, 2010.

[5] Dimitri P Bertsekas. *Constrained optimization and Lagrange multiplier methods*. Academic press, 2014.

[6] Michael Bloem and Nicholas Bambos. Infinite time horizon maximum causal entropy inverse reinforcement learning. In *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, pages 4911–4916. IEEE, 2014.

[7] Matthew Botvinick and James An. Goal-directed decision making in prefrontal cortex: a computational framework. In *Advances in neural information processing systems*, pages 169–176, 2009.

[8] Alexander Broad, TD Murphey, and Brenna Argall. Learning models for shared control of human-machine systems with unknown dynamics. *Robotics: Science and Systems Proceedings*, 2017.

[9] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[10] Giuseppe Calafiore and Fabrizio Dabbene. *Probabilistic and randomized methods for design under uncertainty*. Springer, 2006.

[11] Alfonso Caramazza, Michael McCloskey, and Bert Green. Naive beliefs in "sophisticated" subjects: Misconceptions about trajectories of objects. *Cognition*, 9(2):117–123, 1981.

[12] Jose M Carmena. Advances in neuroprosthetic learning and control. *PLoS biology*, 11(5):e1001561, 2013.

[13] Mark Cutler and Jonathan P How. Efficient reinforcement learning for robots using informative simulated priors. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 2605–2612. IEEE, 2015.

[14] Michel Desmurget and Scott Grafton. Forward modeling allows feedback control for fast reaching movements. *Trends in cognitive sciences*, 4(11):423–431, 2000.

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

[16] Owain Evans, Andreas Stuhlmüller, and Noah D Goodman. Learning the preferences of ignorant, inconsistent agents. In *AAAI*, pages 323–329, 2016.

[17] 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.

[18] Jerry A Fodor. A theory of the child's theory of mind. *Cognition*, 1992.

[19] David Fridovich-Keil, Sylvia L Herbert, Jaime F Fisac, Sampada Deglurkar, and Claire J Tomlin. Planning, fast and slow: A framework for adaptive real-time safe trajectory planning. *arXiv preprint arXiv:1710.04731*, 2017.

[20] Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*, 2017.

[21] Tobias Gerstenberg and Joshua B Tenenbaum. Intuitive theories. *Oxford handbook of causal reasoning*, pages 515–548, 2017.

[22] Matthew Golub, Steven Chase, and M Yu Byron. Learning an internal dynamics model from control demonstration. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 606–614, 2013.

[23] Alison Gopnik and Henry M Wellman. The theory theory. *Mapping the mind: Domain specificity in cognition and culture*, page 257, 1994.

[24] Thomas L Griffiths, Falk Lieder, and Noah D Goodman. Rational use of cognitive resources: Levels of analysis between the computational and the algorithmic. *Topics in cognitive science*, 7(2):217–229, 2015.

[25] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. *arXiv preprint arXiv:1702.08165*, 2017.

[26] Jessica Hamrick, Peter Battaglia, and Joshua B Tenenbaum. Internal physics models guide probabilistic judgments about object dynamics. In *Proceedings of the 33rd annual conference of the cognitive science society*, pages 1545–1550. Cognitive Science Society Austin, TX, 2011.

[27] Sylvia L Herbert, Mo Chen, SooJean Han, Somil Bansal, Jaime F Fisac, and Claire J Tomlin. Fastrack: a modular framework for fast and guaranteed safe motion planning. *arXiv preprint arXiv:1703.07373*, 2017.

[28] Michael Herman, Tobias Gindele, Jörg Wagner, Felix Schmitt, and Wolfram Burgard. Inverse reinforcement learning with simultaneous estimation of rewards and dynamics. In *Artificial Intelligence and Statistics*, pages 102–110, 2016.

[29] Shervin Javdani, Siddhartha S Srinivasa, and J Andrew Bagnell. Shared autonomy via hindsight optimization. *arXiv preprint arXiv:1503.07619*, 2015.

[30] Mitsuo Kawato. Internal models for motor control and trajectory planning. *Current opinion in neurobiology*, 9(6):718–727, 1999.

[31] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[32] Jon Kleinberg and Sigal Oren. Time-inconsistent planning: a computational problem in behavioral economics. In *Proceedings of the fifteenth ACM conference on Economics and computation*, pages 547–564. ACM, 2014.

[33] Anirudha Majumdar, Sumeet Singh, Ajay Mandlekar, and Marco Pavone. Risk-sensitive inverse reinforcement learning via coherent risk models. In *Robotics: Science and Systems*, 2017.

[34] Biren Mehta and Stefan Schaal. Forward models in visuomotor control. *Journal of Neurophysiology*, 88(2):942–953, 2002.

[35] Katharina Muelling, Arun Venkatraman, Jean-Sebastien Valois, John E Downey, Jeffrey Weiss, Shervin Javdani, Martial Hebert, Andrew B Schwartz, Jennifer L Collinger, and J Andrew Bagnell. Autonomy infused teleoperation with application to brain computer interface controlled manipulation. *Autonomous Robots*, pages 1–22, 2017.

[36] Gergely Neu and Csaba Szepesvári. Apprenticeship learning using inverse reinforcement learning and gradient methods. *arXiv preprint arXiv:1206.5264*, 2012.

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

[38] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *arXiv preprint arXiv:1804.02717*, 2018.

[39] David Premack and Guy Woodruff. Does the chimpanzee have a theory of mind? *Behavioral and brain sciences*, 1(4):515–526, 1978.

[40] Dennis R Proffitt and David L Gilden. Understanding natural dynamics. *Journal of Experimental Psychology: Human Perception and Performance*, 15(2):384, 1989.

[41] Anna N Rafferty and Thomas L Griffiths. Diagnosing algebra understanding via inverse planning.

[42] Anna N Rafferty, Rachel Jansen, and Thomas L Griffiths. Using inverse planning for personalized feedback. In *EDM*, pages 472–477, 2016.

[43] Anna N Rafferty, Michelle M LaMar, and Thomas L Griffiths. Inferring learners' knowledge from their actions. *Cognitive Science*, 39(3):584–618, 2015.

[44] Deepak Ramachandran and Eyal Amir. Bayesian inverse reinforcement learning. *Urbana*, 51(61801):1–4, 2007.

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

[46] Wilko Schwarting, Javier Alonso-Mora, Liam Pauli, Sertac Karaman, and Daniela Rus. Parallel autonomy in automated vehicles: Safe motion generation with minimal intervention. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 1928–1935. IEEE, 2017.

[47] Krishna V Shenoy and Jose M Carmena. Combining decoder design and neural adaptation in brain-machine interfaces. *Neuron*, 84(4):665–680, 2014.

[48] Herbert A Simon. Bounded rationality and organizational learning. *Organization science*, 2(1):125–134, 1991.

[49] Emanuel Todorov. Optimality principles in sensorimotor control. *Nature neuroscience*, 7(9):907, 2004.

[50] Amos Tversky and Daniel Kahneman. Judgment under uncertainty: Heuristics and biases. *science*, 185(4157):1124–1131, 1974.

[51] Eiji Uchibe. Model-free deep inverse reinforcement learning by logistic regression. *Neural Processing Letters*, pages 1–15, 2017.

[52] Friedrich Wilkening and Trix Cacchione. Children's intuitive physics. *The Wiley-Blackwell Handbook of Childhood Cognitive Development, Second edition*, pages 473–496, 2010.

[53] Daniel M Wolpert, Zoubin Ghahramani, and Michael I Jordan. An internal model for sensorimotor integration. *Science*, 269(5232):1880–1882, 1995.

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

[55] Brian D Ziebart, J Andrew Bagnell, and Anind K Dey. Modeling interaction via the principle of maximum causal entropy. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pages 1255–1262. Omnipress, 2010.

[56] 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.

[57] Brian D Ziebart, Nathan Ratliff, Garratt Gallagher, Christoph Mertz, Kevin Peterson, J Andrew Bagnell, Martial Hebert, Anind K Dey, and Siddhartha Srinivasa. Planning-based prediction for pedestrians. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 3931–3936. IEEE, 2009.

9 Appendix

This appendix contains additional discussion of experiments.

9.1 Experiments

9.1.1 Grid World Navigation

In Section 3.1 (in the main paper), we described two ways to adapt our algorithm to MDPs with a continuous state space: constraint sampling, and choosing a deterministic model of the internal dynamics. In this section, we evaluate our method on an MDP with a discrete state space in order to avoid the need for these two tricks. Our goal is to learn the internal dynamics and use it to assist the user through internal-to-real dynamics transfer. To sanity-check our algorithm and analyze its behavior under various hyperparameter settings and regularization choices, we implement a simple, deterministic grid world environment in which the simulated user attempts to navigate to a target position.

Hypothesis. Our algorithm is capable of learning accurate tabular representations of the internal dynamics for MDPs with a discrete state space. The two regularization schemes proposed in Section 3.2 (in the main paper) improve the quality of the learned internal dynamics model.

Task description. The state space consists of 49 states arranged in a 7x7 grid. The action space consists of four discrete actions that deterministically move the agent one step in each of the cardinal directions. The reward function emits a large bonus when the agent hits the target, a large penalty when the agent goes out of bounds, and includes a shaping term that rewards the agent for moving closer to the target. An episode lasts at most 100 timesteps. Each of the 49 states is a potential target, so the environment naturally yields 49 distinct tasks.

Corrupting the internal dynamics. To simulate suboptimal behavior, we create two users: one user whose action labels have been randomly scrambled in the same way at all states (e.g., the user's 'left' button actually moves them down instead, and this confusion is the same throughout the state space), and a different user whose action labels have been randomly scrambled in potentially different ways depending on which state they're in (e.g., 'left' takes them down in the top half of the grid, but takes them right in the bottom half). We refer to these two corruption models as 'globally scrambled actions' and 'locally scrambled actions' respectively. The users behave near-optimally with respect to their internal beliefs of the action labels, i.e., their internal dynamics, but because their beliefs about the action labels are incorrect, they act suboptimally in the actual environment.

Evaluation. We evaluate our method on its ability to learn the internal dynamics models of the simulated suboptimal users, i.e., on its ability to unscramble their actions, given demonstrations of their failed attempts to solve the task. The dependent measures are the next-state prediction accuracy of the learned internal dynamics compared to the ground truth internal dynamics, as well as the user's success rate when they are assisted with internal-to-real dynamics transfer (see Section 4.1 in the main paper) using the learned internal dynamics.

Implementation details. We use tabular representations of the $Q_{\boldsymbol{\theta}_i}$ functions and the internal dynamics $T_{\boldsymbol{\phi}}$. We collect 1000 demonstrations per training task, set $\rho = 2 \cdot 10^{-3}$ in Equation 6 (in the main paper), and enumerate the constraints in Equation 6 (in the main paper) instead of sampling.

Manipulated factors. We manipulate (1) whether or not the user receives assistance in the form of internal-to-real dynamics transfer using the learned internal dynamics model – a binary variable; (2) the number of training tasks on which we collect demonstrations – an integer-valued variable between 1 and 49; (3) the structure of the internal dynamics model – a categorical variable that can take on two values: state intent, which structures the internal dynamics in the usual way, or action intent, which uses Equation 7 (in the main paper) instead; and (4) the user's internal dynamics corruption scheme – a categorical variable that can take on two values: globally scrambled actions, or locally scrambled actions.

Analysis. Figure 4 provides overall support for the hypothesis that our method can effectively learn a tabular representation of the internal dynamics for an MDP with a discrete state space. The learned internal dynamics models are accurate with respect to the ground truth internal dynamics, especially when the user's internal dynamics corruption is systematic throughout the state space (top and bottom left plots).

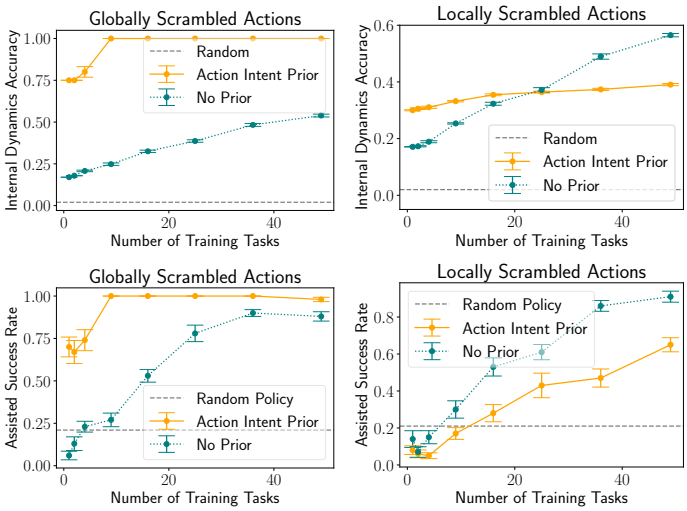

Figure 4: Error bars show standard error on ten random seeds. Corrupting the internal dynamics of the simulated user by scrambling actions the same way at all states (top and bottom left plots) induces a much easier internal dynamics learning problem than scrambling actions differently at each state (top and bottom right plots).

We also compare the two regularization schemes discussed in Section 3.2 (in the main paper): training on multiple tasks, and imposing an action intent prior. Internal models are easier to learn when the user demonstrates their behavior on multiple training tasks, as shown by the increase in accuracy as the number of tasks (on the horizontal axis) increases. Regularizing the internal dynamics using action intent can be useful in some cases when the internal dynamics systematically deviate from the real dynamics, like when the user's actions are scrambled in the same way throughout the state space (top and bottom left plots, compare orange vs. teal curve), but can have a varying effect in other cases where the internal dynamics are severely biased away from reality, like when the action scrambling varies between states (top and bottom right plots, compare orange vs. teal curve).

9.1.2 2D Continuous Navigation

In the previous section, we adopted a tabular grid world environment in order to avoid constraint sampling in Equation 6 (in the main paper). Now, we would like to show that our method still works even when we sample constraints to be able to handle a continuous state space.

Hypothesis. Our algorithm can learn accurate continuous representations of the internal dynamics for MDPs with a continuous state space.

Task description. As mentioned in Section 1 (in the main paper), a classic study in cognitive science shows that people's intuitive judgments about the physics of projectile motion are closer to Aristotelian impetus theory than to true Newtonian dynamics [11]. In other words, people tend to ignore or underestimate the effects of inertia. Inspired by this study, we create a simple 2D environment in a which a simulated user must move a point mass from its initial position to a target position as quickly as possible using a discrete action space of four arrow keys and continuous, low-dimensional observations of position and velocity. The system follows deterministic, linear dynamics. Formally,

$$\boldsymbol{x_{t+1}} = A\boldsymbol{x_t} + B\boldsymbol{u_t} \tag{9}$$

where $\boldsymbol{x} = (x, y, v_x, v_y)^\intercal$ denotes the state, $\boldsymbol{u} \in \{(\pm 0.01, 0)^\intercal, (0, \pm 0.01)^\intercal\}$ denotes the control,

$$A = \begin{pmatrix} 1 & 0 & a_{13} & 0 \\ 0 & 1 & 0 & a_{24} \\ 0 & 0 & a_{33} & 0 \\ 0 & 0 & 0 & a_{44} \end{pmatrix}, B = \begin{pmatrix} b_{11} & 0 \\ 0 & b_{22} \\ b_{31} & 0 \\ 0 & b_{42} \end{pmatrix}.$$

At the beginning of each episode, the state is reset to $\boldsymbol{x_0} = (x_0 \sim \text{Unif}(0,1), y_0 \sim \text{Unif}(0,1), 0, 0)$. The episode ends if the agent reaches the target (gets within a 0.02 radius around the target), goes out of bounds (outside the unit square), or runs out of time (takes longer than 200 timesteps).

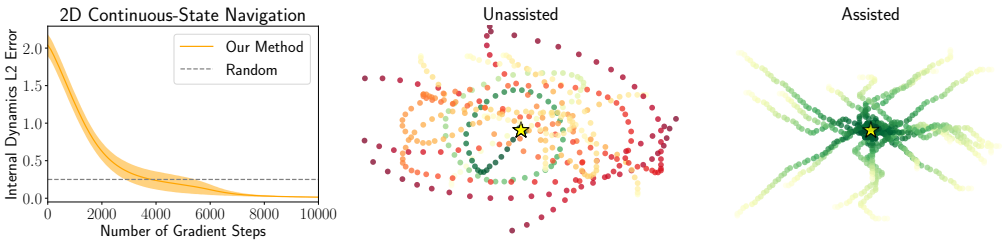

Figure 5: Our method is able to assist the simulated suboptimal user through internal-to-real dynamics transfer. Sample paths followed by the unassisted and assisted user on a single task are shown above. Red paths end out of bounds; green, at the target marked by a yellow star.

Corrupting the internal dynamics. In the simulation, actions control acceleration and inertia exists; in other words, $b_{11} = b_{22} = 0$ and the rest of the parameters are set to 1. We create a simulated suboptimal user that behaves as if their actions control velocity and inertia does not exist, which causes them to follow trajectories that oscillate around the target or go out of bounds. The user behaves near-optimally with respect to their internal beliefs about the dynamics, but because their beliefs are incorrect, they act suboptimally in the real environment.

Evaluation. As before, we evaluate our method on its ability to learn the internal dynamics models of the simulated suboptimal user given demonstrations of their failed attempts to solve the task. We manipulate whether or not the user receives assistance in the form of internal-to-real dynamics transfer using the learned internal dynamics model. The dependent measures are the L2 error of the learned internal dynamics model parameters with respect to the ground truth internal dynamics parameters, and the success and crash rates of the user in each condition.

Implementation details. We fix the number of training tasks at $n = 49$, and use a multi-layer perceptron with one hidden layer of 32 units to represent the $Q_{\boldsymbol{\theta_i}}$ functions. We use a linear model based on Equation 9 to represent the internal dynamics $T_{\boldsymbol{\phi}}$, in which $\hat{a}_{13}, \hat{a}_{24}, \hat{a}_{33}, \hat{a}_{44}, \hat{b}_{11}, \hat{b}_{22}, \hat{b}_{31}, \hat{b}_{42} \in [0, 1]$. We collect 1000 demonstrations per training task, set $\rho = 2$ in Equation 6 (in the main paper), and sample constraints in Equation 6 (in the main paper) by collecting 500 rollouts of a random policy in the real world (see Section 3.1 in the main paper for details).

Analysis. Our algorithm correctly learns the following internal dynamics parameters: (1) $\hat{a}_{33} = \hat{a}_{44} = 0$ in the learned internal dynamics, which corresponds to the user's belief that inertia does not exist; (2) $\hat{b}_{11} = \hat{b}_{22} = 1$ and $\hat{b}_{31} = \hat{b}_{42} = 0$ in the learned internal dynamics, which matches the user's belief that they have velocity control instead of acceleration control. The learned internal dynamics maintains $\hat{a}_{13} = \hat{a}_{24} = 1$, as in the real dynamics, which makes sense since the user's behavior is consistent with these parameters. Figure 5 (left plot) demonstrates the stability of our algorithm in converging to the correct internal dynamics.

Figure 5 (center and right plots) shows examples of trajectories followed by the simulated suboptimal user on their own and when they are assisted by internal-to-real dynamics transfer. The assisted user tends to move directly to the target instead of oscillating around it or missing it altogether.

9.1.3 Learning Rewards from Misguided User Demonstrations

The previous simulation experiments show that our algorithm can learn internal dynamics models that are useful for shared autonomy. Now, we explore a different application of our algorithm: learning rewards from demonstrations generated by a user with a misspecified internal dynamics model. In order to compare to prior methods that operate on tabular MDPs, we adopt the grid world setup from Section 9.1.1, with globally scrambled actions as the internal dynamics corruption scheme.

Hypothesis. Standard IRL algorithms can fail to learn rewards from user demonstrations that are 'misguided', i.e., suboptimal in the real world but near-optimal with respect to the user's internal dynamics. Our algorithm can learn the internal dynamics model, then we can explicitly incorporate the learned internal dynamics into standard IRL to learn accurate rewards from misguided demonstrations.

Evaluation. We evaluate our method on its ability to learn an internal dynamics model that is useful for 'debiasing' misguided user demonstrations, which serve as input to the MaxCausalEnt

IRL algorithm described in Section 4.2 (in the main paper). We manipulate whether we use the learned internal dynamics, or assume the internal dynamics to be the same as the real dynamics. The dependent measure is the true reward collected by a policy that is optimized for the rewards learned by MaxCausalEnt IRL.

Implementation details. We implement the MaxCausalEnt IRL algorithm [55, 28]. The reward function is represented as a table $R(s)$.

Analysis. Figure 2 (in the main paper, right plot) supports our claim that standard IRL is not capable of learning rewards from misguided user demonstrations, and that after using our algorithm to learn the internal dynamics and explicitly incorporating the learned internal dynamics into an IRL algorithm's behavioral model of the user, we learn accurate rewards.

9.2 User Study on the Lunar Lander Game

Task description. The reward function emits a large bonus at the end of the episode for landing between the flags, a large penalty for crashing or going out of bounds, and is shaped to penalize speed, tilt, and moving away from the landing site. The physics of the game are deterministic.

Evaluation protocol. We evaluate our method on its ability to learn the internal dynamics models of human users given demonstrations of their failed attempts to solve the task in the default environment. We manipulate whether or not the user receives assistance in the form of internal-to-real dynamics transfer using the learned internal dynamics. The dependent measures are the success and crash rates in each condition.

Implementation details. We fix the number of training tasks at $n = 9$ and use a multi-layer perceptron with one hidden layer of 32 units to represent the $Q_{\boldsymbol{\theta_i}}$ functions. We collect 5 demonstrations per training task per user, set $\rho = 2 \cdot 10^{-3}$ in Equation 6 (in the main paper), and sample constraints in Equation 6 (in the main paper) by collecting 100 rollouts of a random policy in the real world (see Section 3.1 in the main paper for details).

The physics of the game are governed in part by a configurable vector $\boldsymbol{\psi} \in \mathbb{R}^7$ that encodes engine power, game speed, and other relevant parameters. Since we cannot readily access an analytical expression of the dynamics, only a black-box function that forward-simulates the dynamics, we cannot simply parameterize our internal dynamics model using $\boldsymbol{\psi}$ (see Section 3.1 in the main paper for details). Instead, we draw 100 random samples of $\boldsymbol{\psi}$ and represent our internal dynamics model as a categorical probability distribution over the samples. In other words, we approximate the continuous space of possible internal dynamics models using a discrete set of samples. To accommodate this representation, we modify Equation 4 (in the main paper):

$$\begin{aligned}\delta_{\boldsymbol{\theta_i},\boldsymbol{\phi}}(s,a) &\triangleq Q_{\boldsymbol{\theta_i}}(s,a) - \mathbb{E}_{j\sim\mathrm{Cat}(100,\boldsymbol{\phi})}\left[\int_{s'\in\mathcal{S}} T_{\boldsymbol{\psi}_j}(s'|s,a)\left(R_i(s,a,s') + \gamma V_{\boldsymbol{\theta_i}}(s')\right)\mathrm{d}s'\right] \\ &= Q_{\boldsymbol{\theta_i}}(s,a) - \sum_{j=1}^{100} \boldsymbol{\phi}_j \cdot \int_{s'\in\mathcal{S}} T_{\boldsymbol{\psi}_j}(s'|s,a)\left(R_i(s,a,s') + \gamma V_{\boldsymbol{\theta_i}}(s')\right)\mathrm{d}s'.\end{aligned}$$

Subject allocation. We recruited 9 male and 3 female participants, with an average age of 24. Each participant was provided with the rules of the game and a short practice period of 9-18 episodes to familiarize themselves with the controls and dynamics. Each user played in both conditions: unassisted, and assisted. To avoid the confounding effect of humans learning to play the game better over time, we counterbalanced the order of the two conditions. Each condition lasted 45 episodes.

Counterbalancing the order of the two conditions sometimes requires testing the user in the assisted condition *before* the unassisted condition, which begs the question: where do the demonstrations used to train the internal dynamics model used in internal-to-real dynamics transfer assistance come from, if not the data from the unassisted condition? We train the internal dynamics model used to assist the k-th participant on the pooled, unassisted demonstrations of all previous participants $\{1, 2, ..., k-1\}$. After the k-th participant completes both conditions, we train an internal dynamics model solely on unassisted demonstrations from the k-th participant and verify that the resulting internal dynamics model is the same as the one used to assist the k-th participant.

Analysis. After inspecting the results of our random search over the internal dynamics space, we found that the game speed parameter in $\boldsymbol{\psi}$ had a much larger influence on the quality of the learned

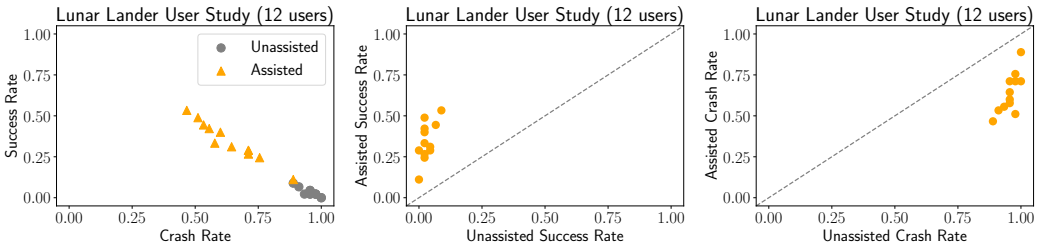

Figure 6: Assistance in the form of internal-to-real dynamics transfer increases success rates and decreases crash rates.

internal dynamics and the resulting internal-to-real dynamics transfer than the other six parameters. Hence, in Figure 3 (in the main paper, bottom left plot), we show the results of a grid search on the game speed parameter, holding the other six parameters constant at their default values. The game speed parameter governs the size of the time delta with which the game engine advances the physics simulation at each discrete step. This parameter indirectly controls the strength of the forces in the game physics: smaller time deltas lead to smaller forces and generally slower motion, and larger deltas to larger forces and consequently faster motion.

We ran a one-way repeated measures ANOVA with the presence of assistance as a factor influencing success and crash rates, and found that $f(1, 11) = 109.58, p < 0.0001$ for the success rate and $f(1, 11) = 126.33, p < 0.0001$ for the crash rate. The assisted user succeeds significantly more often and crashes significantly less often than the unassisted user. Figure 6 shows the raw data.