On the Feasibility of Learning, Rather than Assuming, Human Biases for Reward Inference

Rohin Shah [1]   **Noah Gundotra** [1]   **Pieter Abbeel** [1]   **Anca D. Dragan** [1]

Abstract

Our goal is for agents to optimize the *right* reward function, despite how difficult it is for us to specify what that is. Inverse Reinforcement Learning (IRL) enables us to infer reward functions from demonstrations, but it usually assumes that the expert is noisily optimal. Real people, on the other hand, often have *systematic* biases: risk-aversion, myopia, etc. One option is to try to characterize these biases and account for them explicitly during learning. But in the era of deep learning, a natural suggestion researchers make is to avoid mathematical models of human behavior that are fraught with specific assumptions, and instead use a purely data-driven approach. We decided to put this to the test – rather than relying on assumptions about which specific bias the demonstrator has when planning, we instead learn the demonstrator's *planning algorithm* that they use to generate demonstrations, as a differentiable planner. Our exploration yielded mixed findings: on the one hand, learning the planner can lead to better reward inference than relying on the wrong assumption; on the other hand, this benefit is dwarfed by the loss we incur by going from an exact to a differentiable planner. This suggest that at least for the foreseeable future, agents need a middle ground between the flexibility of data-driven methods and the useful bias of known human biases. Code is available at `https://tinyurl.com/learningbiases`.

1. Introduction

Our ultimate goal is to enable the design of agents that optimize for the *right* reward function. Unfortunately, designing

(a) Assuming a specific suboptimality | (b) Learning the suboptimality
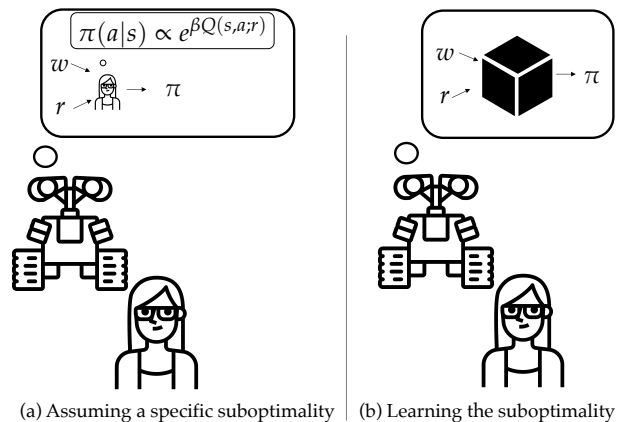
Figure 1. While we could correct for systematic biases by having our AI system reason about explicit models of human reasoning, using the wrong assumption can lead to agents that do not correctly understand what people want. A natural alternative is to learn human biases from data. Our goal in this work is to investigate this alternative and gain insight into what additional assumptions might make it feasible and what algorithmic improvements it demands.

reward functions is challenging (Amodei et al., 2017) and can have unintended side-effects (Hadfield-Menell et al., 2017; Krakovna, 2018). Inverse Reinforcement Learning (IRL) (Russell, 1998; Ng et al., 2000; Abbeel & Ng, 2004) aims to bypass the need for reward design by learning the reward from observed demonstrations of good behavior.

Existing IRL algorithms typically make the assumption that the demonstrator is either optimal, or Boltzmann rational, i.e. taking better actions with higher probability (Ziebart et al., 2008; Finn et al., 2016). However, there is a rich literature showing that humans are *not* optimal, and are biased in *systematic* ways. Consider a grad student who starts writing a paper a month in advance, expecting it to take two weeks, but then misses the deadline. Should we infer that they prefer to lose sleep to pursue a deadline that they then miss? Of course not. This is a classic case of the *planning fallacy* (Buehler et al., 1994): the grad student was wrong in their prediction of how long it would take to complete the paper. An assumption of noisy rationality cannot allow us to correct for this bias: how do we tell whether the grad student underestimated the time required

---

and actually wanted to finish the paper earlier, rather than overestimating the time required and actually wanting to finish the paper even later?

Of course, if we know that humans tend to underestimate how long any given task will take, then we can correct for this bias by having our AI system reason about how it affects human reasoning, as illustrated in figure 1a. IRL algorithms have been developed that can account for particular systematic biases, such as myopia and hyperbolic time discounting (Evans et al., 2016; Evans & Goodman, 2015), sparse noise (Zheng et al., 2014), risk sensitivity (Majumdar et al., 2017), or a bad dynamics model (Reddy et al., 2018). Even suboptimal trajectories or failures (Shiarlis et al., 2016) can be thought of as a biased demonstrator, where the bias is the specific model of failure. However, choosing a particular model of suboptimality is a big assumption, and can lead to arbitrarily bad performance if the assumption is incorrect (Steinhardt, 2017; Steinhardt & Evans, 2017). For example, if we try to explain the grad student's behavior as hyperbolic time discounting (that is, valuing short-term rewards disproportionately more than long-term ones), we might infer that the grad student enjoys long nights of writing over the short term, rather than viewing it as an instrumental goal necessary for submitting the paper. We wouldn't want our AI system deleting our in-progress paper so that we can have the "joy" of rewriting it from scratch!

Given how complex real humans are, it seems hopeless to know exactly which bias a person is displaying, and inevitable that any such assumption will lead to a misspecified bias model. In the age of data-driven methods, it seems almost natural to think that we could learn the bias model as well, rather than relying on a hardcoded assumption. One can view the demonstrator's behavior as a composition of a reward function and a *planning algorithm* that computes what actions to take given a reward function – instead of assuming a planning algorithm (e.g. noisy rational, myopic, etc.), why not learn it? Our work is about exploring the feasibility of this alternative (depicted in figure 1b).

Right off the bat, this enticing and seemingly natural idea hits a wall: unfortunately, when the planning algorithm can be any function mapping reward functions to policies, it is impossible to learn the true reward function even with infinite data, because there are always alternative explanations for the observed policy (Armstrong & Mindermann, 2018; Christiano, 2015). Any particular behavior could be explained either by positing a term in the reward function, or a bias in the planning algorithm. Rather than using data to avoid all assumptions, our work actually investigates whether it is at least feasible to learn the planning algorithm when either a) we get to first observe demonstrations in tasks where we know the reward, and can thus focus only on learning the planner for those demonstrations; or b) we assume

that the demonstrator is good at the tasks, and regularize the planning algorithm towards optimality.

We find that even in the "easy" setting where we are given access to some rewards, this problem is very difficult: there is some benefit from learning systematic biases in the planner, but this is dwarfed by the disadvantage of using a neural network as a planner, relative to using an exact, perhaps slightly misspecified model of rationality. In the case where we instead regularize towards optimality, the resulting algorithm is more robust to different human models, but again the benefit is dwarfed by the inaccuracies introduced by differentiable planning.

2. Examples of Biases

To put this work in context, we start with some examples of the kind of biases a general algorithm should be able to capture and account for. While we use these for illustrative purposes, the whole point of our exploration is that humans might have systematic suboptimalities completely different from these examples. We don't know all the possible biases a priori – if we did, using that information would certainly be the superior choice.

Running Example. We illustrate the effects of these biases on a simple 2D navigation task in figure 2. There are multiple salient locations, each of which has a desirability score (which can be negative, in which case the agent wants to avoid those locations). The agent can move in any of the four cardinal directions, or stay in its current position. Every movement action has a chance of failing and causing the agent to move in a direction orthogonal to the one it chose. Despite their simplicity, there are several ways in which human-like suboptimal behavior can manifest in these environments.

Time inconsistency. Would you prefer to get $100 in 30 days, or $110 in 31 days? Faced with this question, people typically choose the latter. However, thirty days later, when faced with the choice of getting $100 now, or $110 tomorrow, they sometimes choose to take the $100. This reversal of preferences over time would never happen with an optimal agent that maximizes expected sum of discounted rewards. Researchers model this phenomenon using *hyperbolic time discounting*, in which future rewards are discounted more aggressively than exponentially. This leads to a followup question – how do humans make long-term plans, given that their future self will have different preferences? Prior work has considered a spectrum from *naive* agents that assume their future self will have the same preferences as they do, to *sophisticated* agents that perfectly understand how their preferences will change over time and make plans that take such change into account (Frederick et al., 2002).

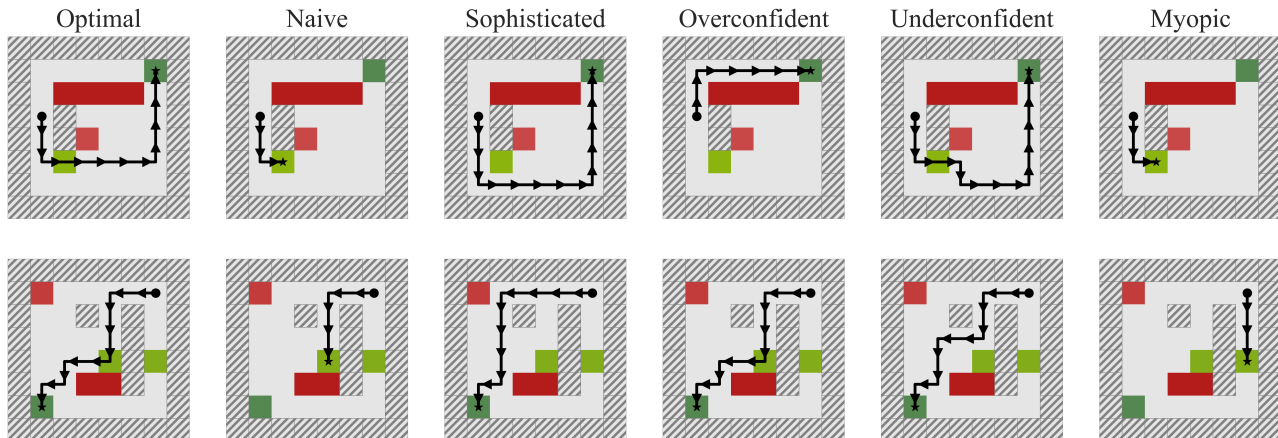In figure 2, when going to a high reward, both the *naive* and

Figure 2. The plans of our synthetic agents on two navigation environments. Actual trajectories could differ due to randomness in the transitions. Green squares indicate positive reward while red squares indicate negative reward, with darker colors indicating higher magnitude of reward.

sophisticated hyperbolic time discounters can be "tempted" by a proximate smaller reward. The naive agent fails to anticipate the temptation, and so once it gets near the smaller positive reward, it caves in to the temptation and stays there. The sophisticated agent explicitly plans to avoid the temptation – it does not collect the smaller reward and instead takes a longer, more dangerous path around the smaller reward to get to the large reward.

Incorrect estimates of probabilities. Humans are notoriously bad at judging probabilities. The *availability heuristic* (Tversky & Kahneman, 1973) refers to the human tendency to rate events as more likely if they are easier to recall. The *recency effect* is a similar effect where recent events are judged to be more probable. These biases are depend heavily enough on context that they don't transfer to our task in any obvious way. So, we use two simplified models – an *overconfident* agent, which expects that the most likely next state is more likely than it actually is, leading it to take what we would call risky behavior, and an *underconfident* agent, which analogously behaves in an overly cautious manner. In figure 2, the overconfident agent takes the shortest path to the reward, underestimating the risk of slipping into the large region of negative reward, while the underconfident agent plans a circuitous route around negative reward that it is unlikely to have actually encountered.

Bounded computation. Researchers have studied models of *bounded rationality*, where humans are assumed to be rational subject to the constraint that they have a bounded amount of computation. This can be thought of as an explanation that many other heuristics and biases are actually computational shortcuts that allow us to reach reasonably good decisions without too much cost (Kahneman, 2003). In our task, we model computation bounds as a small time

horizon for planning, leading to *myopic* behavior. In figure 2, the myopic agent can only see close rewards, and goes directly to them, never even realizing the possibility of going to the highest reward.

3. Problem: Learning Rewards of Demonstrators with Unknown Biases

Notation. A (finite-horizon) Markov Decision Process (MDP) (Puterman, 2014) is a tuple $\langle S, A, T, r, H \rangle$. S is a set of states. A is a set of actions. T is a probability distribution over the next state, given the previous state and action. We write this as $T(s_{t+1}|s_t, a)$. r is a reward function that maps states and actions to rewards $r : S \times A \to \mathbb{R}$. $H \in \mathbb{Z}_+$ is the finite planning horizon for the agent. Since we are interested in the setting where the reward function r is unknown, we will factor MDPs into *world models* $w = \langle S, A, T, H \rangle$ and reward functions r.

Instead of having access to a reward function, we observe the behavior of a demonstrator, who performs the task well but could be suboptimal in systematic ways. We assume that the demonstrator produces (possibly stochastic) policies using a *planning algorithm* $D : (\mathbb{W} \times R) \to (S \to A \to [0, 1])$, or planner for short. Here $\mathbb{W}$ is a space of world models with the same set of states S and actions A, and $R \subseteq S \times A \to \mathbb{R}$ is a space of reward functions that the demonstrator can plan for. Later in this section we illustrate additional assumptions about D. We observe the demonstrator's *policy* $\pi : S \to A$ for a particular world model w, with $\pi_D = D(w, r^*)$ for some unknown reward r^*.

Estimating Biases and Rewards. Given a ***world model*** w and the ***demonstrator's policy*** π_D which may exhibit an unknown bias, determine the ***reward*** r^* that the demonstrator

is optimizing.

We might hope that enough data can solve this problem without any additional assumptions. However, this problem is unsolvable – Armstrong & Mindermann (2018) prove an impossibility result showing that for any potential reward function r', there is some planner D' such that $D'(w, r') = D(w, r^*)$. The proof is simple – simply set $D'(w, r') = \pi_D$ for any r', that is D' always returns π_D regardless of the reward function. What we really explore in this work is thus data-driven approaches that make minimal additional assumptions, rather than none at all.

Inverse reinforcement learning assumes that the demonstrator is (approximately) optimal to get around this issue. It is common to assume Boltzmann rationality, where the probability of an action is proportional to the exponent of its expected value (Baker et al., 2006), i.e. $P(a|s) \propto e^{Q(s,a)}$, where Q is the optimal Q function that satisfies the Bellman equation:

$$Q(s, a) = r(s, a) + \gamma \sum_{s'} \left[T(s'|s, a) \max_{a'} Q(s', a') \right] \quad (1)$$

However, we know that humans are systematically suboptimal, and so we would like to relax this assumption and try other, more realistic assumptions. The pathological solutions in the impossibility result occur partly because the demonstrator can have arbitrary behavior on different environments. While we certainly want the demonstrator to adapt to different environments, the *algorithm* that the demonstrator uses to determine their policy should stay fixed across similar environments. This imposes structure on the demonstrator's planner that can eliminate some possibilities.

Assumption 1: The demonstrator plans in the same way for sufficiently similar environments.

Intuitively, the demonstrator's planning algorithm $D : (\mathbb{W} \times R) \to (S \to A \to [0, 1])$ is "the same" for similar environments. Of course, if D can be any function with this type signature, it can still map any arbitrary (w, r) pair to any arbitrary policy $D(w, r)$, but we will further ensure that D is simple (through regularization). Given a list of world models $W = [w_1 \dots w_n]$ and reward functions $R = [r_1 \dots r_n]$, we define $D(W, R)$ to be the list of the demonstrator's policies $[D(w_1, r_1) \dots D(w_n, r_n)]$.

Note that this is a strong assumption: while it is reasonable to believe that people plan in the same way for variations of the same task, they likely have different biases for different tasks, because they may have domain-specific heuristics. The setting of multiple tasks has been studied before (Gleave & Habryka, 2018; Dimitrakakis & Rothkopf, 2011; Choi & Kim, 2012; Xu et al., 2018), though not for the purpose of inferring systematic biases.

This assumption leads to a slightly easier problem, of recov-

ering rewards from multiple tasks:

Estimating Biases and Rewards for Multiple Tasks. Given a list of world models W and the demonstrator's policies $\Pi_D = D(W, R)$ which may exhibit an unknown bias, determine the list of reward functions R (one for each $w \in W$) that the demonstrator was optimizing.

Since the person uses the same planner across all tasks, an agent can have an easier time recovering rewards for each task by leveraging the common structure across the tasks. This is especially appealing for agents that would get to observe people for some period of time before trying to assist them. However, Assumption 1 still falls prey to the impossibility result. Consider the case where the demonstrator is optimal. Given the assumptions so far, we could infer that the demonstrator is *minimizing* expected reward for the reward function $-r^*$, since that perfectly predicts π_D. This is very bad, as we could infer a reward that incentivizes the *worst* possible behavior!

When humans take action, we typically assume that they are doing something that is reasonable for achieving their goals, even if it is not optimal:

Assumption 2a: The demonstrator is "close" to optimal.

This is a weaker version of the standard IRL assumption of Boltzmann rationality. In section 4.3, we derive a natural algorithm that takes advantage of this assumption, by regularizing the planner towards optimality. This gives us an algorithm for the problem of estimating biases and rewards for multiple tasks that does not obviously fail due to an impossibility result.

We also explore an alternative approach, based on the fact that we have strong priors about what humans are trying to optimize for. Intuitively, these priors allow us to infer how good they are at achieving their goals, and in what ways they are systematically biased, which can be used to better infer goals in new settings. We formalize this by assuming that we observe some tasks where we know the demonstrator's reward function and policy.

This again gives us a natural algorithm, detailed in section 4.2 that does not obviously fail due to the impossibility result, since we can easily infer that the demonstrator is "close" to optimal from the tasks for which we do observe the demonstrator's reward function.

Assumption 2b: We know what reward function the demonstrator is optimizing for *some* tasks.

Estimating Biases and Rewards with Access to Tasks with Known Rewards. Given a list of world models W, a list of the demonstrator's policies $\Pi_D = D(W, R)$, a list of world models W_{known} with known rewards R_{known} and a list of the demonstrator's policies $\Pi_{\text{known}} = D(W_{\text{known}}, R_{\text{known}})$,

determine the reward functions R that D was optimizing.

While our problem formulations above assume that we have access to full policies Π_D, none of the algorithms rely on this assumption – it is easy to modify them to work with trajectories instead.

4. Algorithms to Estimate Biases and Rewards

The idea that we investigate in this work is whether it is beneficial to learn a model of how the demonstrator plans. Once we have learned the planning algorithm D, we are faced with an inverse problem: we want to find the r such that $\pi_D = D(w, r)$. This resembles the problem of feature visualization for image classifiers (Olah et al., 2017), and suggests a natural approach: as long as the planner D is differentiable, we can invert its "understanding" using backpropagation to infer the reward from the policy.

4.1. Architecture

We model the demonstrator planning algorithm D using a *differentiable planner* f_θ, which is a neural net that can express planning algorithms whose parameters θ can be updated using gradient descent. f has the same type as the demonstrator's planner D, namely $(\mathbb{W} \times R) \to (S \to A \to [0, 1])$. Thus, the inputs to the differentiable planner f are a world model $w \in \mathbb{W}$ and a reward function $r \in \mathbb{R}$; the output is a stochastic policy $\pi \in (S \to A \to [0, 1])$. We determine how well $\langle f, R \rangle$ matches the demonstrator's policy π_D with the cross entropy loss $\mathcal{L}(f_\theta(W, R), \Pi_D) = \sum_i \mathcal{L}(f_\theta(w_i, r_i), \pi_{D,i})$.

While the algorithms can work with any differentiable planner, in this work we use a *value iteration network* (VIN) (Tamar et al., 2016). A VIN is a fully differentiable neural network that embeds an approximate value iteration algorithm inside a feed-forward classification network. For environments where transitions only depend on "nearby" states (as in navigation tasks), the Bellman update can be performed using an appropriate convolution, and the computation of values from Q-values can be done with a max-pooling layer. By leaving the filters for the convolutions unspecified, the VIN can automatically learn the transition probabilities. Of course, the VIN is merely one architecture for a differentiable planner; we could equally well use other planners (Srinivas et al., 2018; Pascanu et al., 2017; Guez et al., 2018). The algorithms we study will become stronger as research in this area advances.

The components of the algorithms. This architecture enables two main operations that are important for inferring rewards and biases, which we illustrate in figure 3. First, given world models W, reward functions R (either known or hypothesized), and the demonstrator's policies Π_D, we can train a corresponding planner using gradient descent (figure 3a):

$$\theta = \min_{\theta'} \mathcal{L}(f_{\theta'}(W, R), \Pi_D) \qquad \text{(TRAIN-PLANNER)}$$

Second, given world models W, demonstrator's policies Π_D, and some planner parameters θ, we can infer the corresponding reward functions using gradient descent (figure 3b):

$$R = \min_{R'} \mathcal{L}(f_\theta(W, R'), \Pi_D) \qquad \text{(TRAIN-REWARD)}$$

It is also possible to perform both of these at the same time by training the planner parameters and rewards jointly given world models W and the demonstrator's policies Π_D:

$$R, \theta = \min_{R', \theta'} \mathcal{L}(f_{\theta'}(W, R'), \Pi_D) \qquad \text{(TRAIN-JOINTLY)}$$

4.2. Learning the planner from known rewards first (Assumption 2b)

Consider the simpler setting when we have access to a set of tasks with *known* rewards. The known rewards can be used to infer the planning algorithm used by the demonstrator, which can then be used to infer rewards in the remaining cases. So, the planner is first trained on the world models for which we have rewards. Then, learned planner weights allow us to infer the reward on the world models for which we don't know the reward. This algorithm is illustrated in Algorithm 1.

Algorithm 1 IRL-WITH-REWARDS: Estimating biases and rewards with access to tasks with known rewards. Requires that Assumptions 1 and 2b hold.

1: $\theta \Leftarrow$ TRAIN-PLANNER$(W_{\text{known}}, R_{\text{known}}, \Pi_{\text{known}})$
2: **return** TRAIN-REWARD(W, θ, Π_D)

4.3. Learning the planner and rewards simultaneously (Assumption 2a)

Perhaps the most natural algorithm to solve our problem would be to jointly train the planner and rewards on the given set of world models and policies. However, this falls prey to the impossibility result: there is no way to distinguish between reward maximization with an optimal reward r^* and reward minimization with the reward $-r^*$. So, it would be useful to regularize the planner so that it is "close" to optimal, in accordance with Assumption 2a.

A natural way to do this regularization is by *initializing* the planner to be optimal, and then finetuning the result by
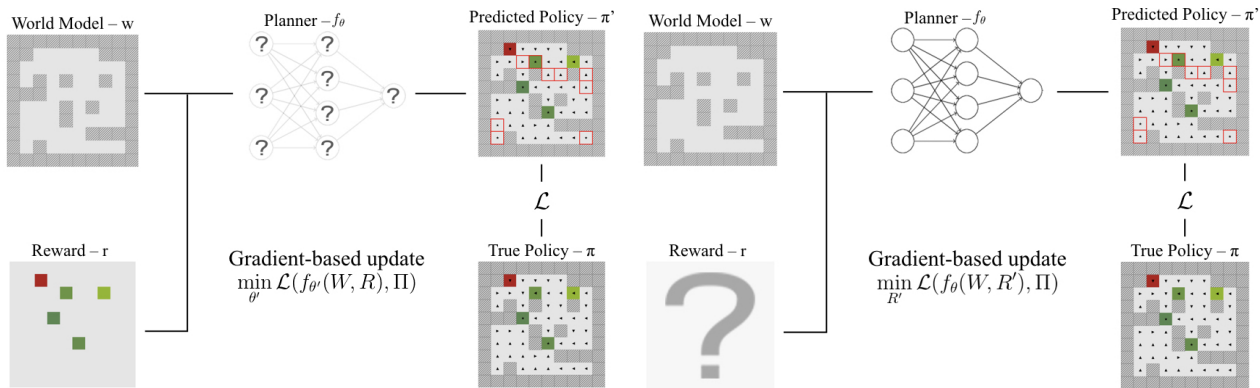
(a) Training the planner f_θ. We hold the world model w, reward r, and policy π fixed, and update f_θ with gradient descent.

(b) Training the reward R. We hold the world model w, planner f_θ, and policy π fixed, and update the reward r with gradient descent.

Figure 3. The architecture and operations on it used in the algorithms.

training jointly as in Equation TRAIN-JOINTLY. Since the reward inference requires a differentiable planner, we need a method that sets a differentiable planner to be optimal (that is, the planner that maximizes expected reward). This can be done by simulating data from an optimal agent with randomly generated world models and rewards, and use this to train the planner to mimic an optimal agent. The resulting algorithm is illustrated in Algorithm 2. We show in section 5.4 that the initialization is crucial for good performance as we would expect.

Algorithm 2 IRL-WITHOUT-REWARDS: Estimating biases and rewards for multiple tasks with no known rewards. Requires that Assumptions 1 and 2a hold.

1: $W_\text{sim}, R_\text{sim} \Leftarrow$ Generate world models and rewards
2: $\Pi_\text{sim} \Leftarrow$ Run optimal agent on $\langle W_\text{sim}, R_\text{sim} \rangle$
3: $\theta_\text{init} \Leftarrow$ TRAIN-PLANNER$(W_\text{sim}, R_\text{sim}, \Pi_\text{sim})$
4: $R_\text{init} \Leftarrow$ TRAIN-REWARD$(W, \theta_\text{init}, \Pi_D)$
5: $\theta, R \Leftarrow$ TRAIN-JOINTLY(W, Π_D) {using θ_init and R_init as initializations}
6: **return** R

---

5. Evaluation

We evaluate the algorithms by simulating demonstrators with different biases, and testing whether the same method can correctly infer reward for all these demonstrators.

5.1. Experiment details

In all experiments below, results are averaged over 10 runs with different seeds, on randomly generated 14x14 gridworlds that have 7 squares with non-zero rewards. We ensure that all such squares can be reached from the start state, and that at least half of the positions in grid are not walls.

We use a Value Iteration Network with 10 iterations as the

differentiable planner, and set the space of rewards to be $S \to \mathbb{R}$; that is, any state can be mapped to any reward, but the reward is assumed not to depend on the action. We added an extra convolutional layer to the initial part of the VIN (which learns a proxy reward) as initial experiments showed that this could better learn an optimal planner for our gridworlds; other than that the architecture remains as described in Tamar et al. (2016). We apply L2 regularization to the VIN with scale 0.0001, and do not regularize the reward.

For all experiments, we kept the number of demonstrations fixed to 8000. For Algorithm 1, this was split into 7000 policies with rewards that were used to train the planner, and 1000 on which rewards had to be inferred. Note that this does *not* include any simulated data – for example, Algorithm 2 would get 8000 biased policies, and would *also* simulate a further 7000 policies from an optimal agent in order to initialize the planner and reward.

5.2. Evaluating reward inference

Hypothesis. The hypothesis we put to the test in this work is that accounting for unknown systematic bias should outperform the assumption of a particular inaccurate bias, e.g. noisy rationality or the lack thereof.

Manipulated variables. In order to test this, we manipulate whether we *learn* the demonstrator model or *assume* it. To avoid confounds introduced by changing the inference algorithm, we use the same algorithm for both. In the learning case, we train the planner on the ground truth demonstrator data; in the assume case, we train it on data generated from a) a Boltzmann-rational demonstrator; and b) an optimal demonstrator – these are the two models commonly assumed by IRL algorithms. Keeping the algorithm the same enables us to isolate the effect of adapting to an unknown model from the effect of having to use an approximate differentiable planner rather than a perfect one. We
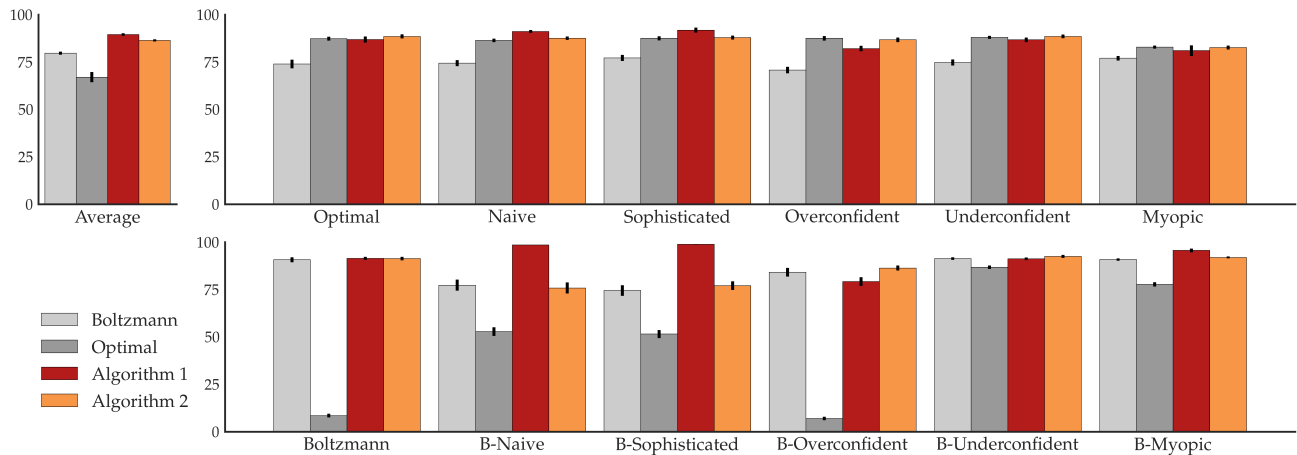
Figure 4. Reward obtained when planning with the inferred reward, as a percentage of the maximum possible reward, for different bias models and algorithms. We implement five types of biases by modifying the value iteration algorithm to produce a different set of Q-values. The top row shows results for agents that choose between the best actions from these Q-values, while in the bottom row the agent chooses actions with probability proportional to the exponent of the Q-value (the Boltzmann assumption). The average over both rows is displayed in the top left.

will quantify the second effect, i.e. the approximation error introduced by the VIN, in section 5.3.

In the setting where we learn the bias, we further manipulate whether we have access to known rewards for some tasks or not – i.e. whether we use Algorithm 1 or Algorithm 2.

Finally, we manipulate the *actual bias* of the demonstrator. Following Evans et al. (2016), we implement the myopic, naive and sophisticated synthetic demonstrators as modifications of the value iteration algorithm. Similarly, we implement the overconfident and underconfident demonstrators by modifying the transition probability distributions used to plan in value iteration. We also include an optimal demonstrator, and stochastic (Boltzmann) versions of all demonstrators.

Dependent measures. We measure the reward obtained by planning optimally with the inferred reward function, as a percentage of the maximum possible reward that could be obtained.

Comparisons among differentiable planners. Figure 4 shows the results for learning a demonstrator model vs. assuming an optimal or a Boltzmann demonstrator. The top left subfigure plots what happens on average, across all synthetic demonstrators we tested. The results do provide support to the hypothesis: both learning methods (orange) outperform assuming a model (gray). Looking at the breakdown per demonstrator, we see that assuming optimal does not do well when the demonstrator has any noise (bottom graph). Similarly, assuming Boltzmann does not do well when the demonstrator is not noisy (top graph).

The learning methods tend to perform on par with the best

of two choices. In some cases, like the naive and sophisticated hyperbolic discounters, especially the noisy ones, the learning methods outperform both optimal and Boltzmann assumptions. The optimal assumption outperforms the learning methods in some of the non-noisy cases. We hypothesize that this is because as long as the demonstrator eventually reaches the best reward location, assuming optimality allows us to figure out this location. It is then possible to perform near-optimally on our task with knowledge of the best reward location, by navigating to that location and staying there.

Interestingly, Algorithm 1 does not always outperform Algorithm 2, despite it having access to known rewards. We believe this has to do with the fact that Algorithm 2 exploits Assumption 2a (demonstrator close to optimal) and initializes from training on simulated optimal demonstrator data. Algorithm 1 does not rely on this assumption and therefore does not benefit from this initialization, even though the assumption is correct for most of the models we test.

5.3. Tradeoff between being adaptive to bias vs. using exact planning

Our paper is about investigating the viability of the "don't assume specific biases" idea. To be adaptive to different kinds of biases the agent might see, it has to learn a model of the demonstrator's planning algorithm via a differentiable planner. Unfortunately, this causes planning to be approximate – it seems like whatever benefit we get from the adapting to biases, we lose because of the approximation. But these planners will become more practical, they can make this idea practical as well.
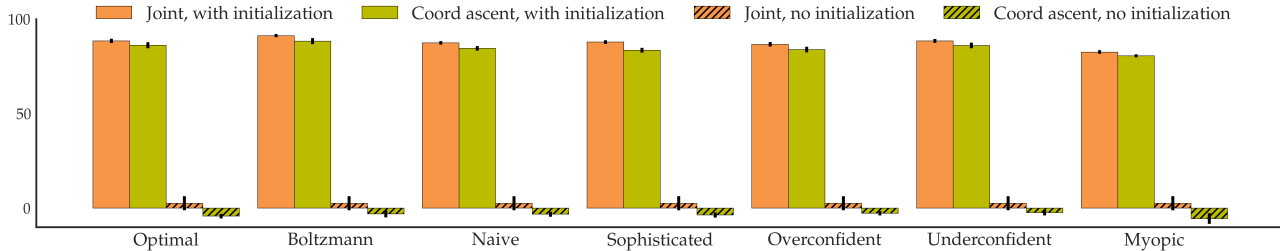
Figure 5. Percent reward obtained for different bias models using variations of Algorithm 2, which does not get access to any known rewards. These algorithms can vary along two dimensions – whether they are initialized with the assumption that the demonstrator is rational, and whether they train the planner and reward jointly or with coordinate ascent. The original version of Algorithm 2 does initialize, and trains jointly.

To quantify this loss, we replace the VIN with a differentiable exact model of the demonstrator, and infer the reward by backpropagating through the exact model. Since value iteration is not differentiable, we implement soft value iteration, where max operations are replaced with logsumexp operations, and measure percent reward obtained when inferring rewards for an optimal demonstrator.

Results. With an exact model of the demonstrator, we get $(98.1 \pm 0.1)\%$ of the maximum reward when performing optimal planning on the inferred rewards, while we get $(86.9 \pm 1.6)\%$ with Algorithm 1 and $(86.2 \pm 1.6)\%$ with Algorithm 2. Again, better planners would improve both algorithms.

5.4. How important are the various parts of the algorithm?

Algorithm 2 was predicated on Assumption 2a, that the demonstrator's planner was "close" to rational, which motivated the initialization step where the planner is trained to mimic an optimal agent. We test how important this is by modifying Algorithm 2 to infer rewards without an initialization (removing lines 1-4). We include versions of the algorithm where we perform coordinate ascent by alternating planner training and reward training instead of training the planner and reward jointly.

Results. Figure 5 shows the results for a subset of demonstrators (full results are in the supplementary material). We can see that the initialization is indeed crucial for good performance, as expected. It also turns out that the joint training outperforms coordinate ascent.

The importance of joint training. We may also ask what value the joint training adds over the initialization. Without the joint training, Algorithm 2 is simply training the planner to be optimal and then inferring rewards, and so is identical to the Optimal case in Figure 4. We can see that the joint training does add value over the initialization.

6. Discussion

Summary. It seems daunting to try to characterize human biases, and yet assuming the wrong bias can lead to agents that do not correctly understand what people want. A natural alternative is to let the data characterize the bias: to learn how people generate their actions given what they want, with all the biases they might have. Our goal in this work was to investigate this approach and gain insight into whether it just works right off the bat, and, if not, what additional structure it benefits from and what improvements in state of the art algorithms it requires. Overall, we have found that it might be possible to maintain flexibility in learning systematic biases while regularizing the learned planner to be close to optimal; but also that for this to be practical within a deep learning architecture will need more progress in differentiable planning.

Limitations and future work. A core limitation of our analysis is that it was conducted on simple problems in simulation, rather than on real problems with real people. This makes sense as a starting point, because it allows us to have ground truth to evaluate against, and allows us to generate large datasets that would be hard to collect with real humans. However, as differentiable planners are able to handle higher complexity, the analysis ought to eventually move to more realistic tasks with human data.

Further, we did not explore all possible assumptions that could simplify the planner learning task. The assumption that the demonstrator has the same bias across many tasks is key to the algorithms, but is very strong. This analysis could be extended by using meta-learning to learn a prior over planners, and by inferring the demonstrator's beliefs as in Baker & Tenenbaum (2014) (e.g. via TOMNets (Rabinowitz et al., 2018)). We are excited to look into this, and into what additional inductive bias we could leverage, in our future work.

References

Abbeel, P. and Ng, A. Y. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 1. ACM, 2004.

Amodei, D., Christiano, P., and Ray, A. Learning from human preferences. https://blog.openai.com/deep-reinforcement-learning-from-human-preferences/, 2017.

Armstrong, S. and Mindermann, S. Occam's razor is insufficient to infer the preferences of irrational agents. In *Advances in Neural Information Processing Systems*, pp. 5603–5614, 2018.

Baker, C., Saxe, R., and Tenenbaum, J. B. Bayesian models of human action understanding. In *Advances in neural information processing systems*, pp. 99–106, 2006.

Baker, C. L. and Tenenbaum, J. B. Modeling human plan recognition using bayesian theory of mind. *Plan, activity, and intent recognition: Theory and practice*, pp. 177–204, 2014.

Buehler, R., Griffin, D., and Ross, M. Exploring the "planning fallacy": Why people underestimate their task completion times. *Journal of personality and social psychology*, 67(3):366, 1994.

Choi, J. and Kim, K.-E. Nonparametric bayesian inverse reinforcement learning for multiple reward functions. In *Advances in Neural Information Processing Systems*, pp. 305–313, 2012.

Christiano, P. The easy goal inference problem is still hard. https://ai-alignment.com/the-easy-goal-inference-problem-is-still-hard-fad030e0a876, 2015.

Dimitrakakis, C. and Rothkopf, C. A. Bayesian multitask inverse reinforcement learning. In *European Workshop on Reinforcement Learning*, pp. 273–284. Springer, 2011.

Evans, O. and Goodman, N. D. Learning the preferences of bounded agents. In *NIPS Workshop on Bounded Optimality*, volume 6, 2015.

Evans, O., Stuhlmüller, A., and Goodman, N. D. Learning the preferences of ignorant, inconsistent agents. In *AAAI*, pp. 323–329, 2016.

Finn, C., Levine, S., and Abbeel, P. Guided cost learning: Deep inverse optimal control via policy optimization. In *International Conference on Machine Learning*, pp. 49–58, 2016.

Frederick, S., Loewenstein, G., and O'donoghue, T. Time discounting and time preference: A critical review. *Journal of economic literature*, 40(2):351–401, 2002.

Gleave, A. and Habryka, O. Multi-task maximum entropy inverse reinforcement learning. *arXiv preprint arXiv:1805.08882*, 2018.

Guez, A., Weber, T., Antonoglou, I., Simonyan, K., Vinyals, O., Wierstra, D., Munos, R., and Silver, D. Learning to search with mctsnets. *arXiv preprint arXiv:1802.04697*, 2018.

Hadfield-Menell, D., Milli, S., Abbeel, P., Russell, S. J., and Dragan, A. Inverse reward design. In *Advances in Neural Information Processing Systems*, pp. 6768–6777, 2017.

Kahneman, D. A perspective on judgment and choice: mapping bounded rationality. *American psychologist*, 58(9): 697, 2003.

Krakovna, V. Specification gaming examples in ai. https://vkrakovna.wordpress.com/2018/04/02/specification-gaming-examples-in-ai/, 2018.

Majumdar, A., Singh, S., Mandlekar, A., and Pavone, M. Risk-sensitive inverse reinforcement learning via coherent risk models. In *Robotics: Science and Systems*, 2017.

Ng, A. Y., Russell, S. J., et al. Algorithms for inverse reinforcement learning. In *Icml*, pp. 663–670, 2000.

Olah, C., Mordvintsev, A., and Schubert, L. Feature visualization. *Distill*, 2(11):e7, 2017.

Pascanu, R., Li, Y., Vinyals, O., Heess, N., Buesing, L., Racanière, S., Reichert, D., Weber, T., Wierstra, D., and Battaglia, P. Learning model-based planning from scratch. *arXiv preprint arXiv:1707.06170*, 2017.

Puterman, M. L. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Rabinowitz, N. C., Perbet, F., Song, H. F., Zhang, C., Eslami, S., and Botvinick, M. Machine theory of mind. *arXiv preprint arXiv:1802.07740*, 2018.

Reddy, S., Dragan, A., and Levine, S. Where do you think you're going?: Inferring beliefs about dynamics from behavior. In *Advances in Neural Information Processing Systems*, pp. 1454–1465, 2018.

Russell, S. Learning agents for uncertain environments. In *Proceedings of the eleventh annual conference on Computational learning theory*, pp. 101–103. ACM, 1998.

Shiarlis, K., Messias, J., and Whiteson, S. Inverse reinforcement learning from failure. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, pp. 1060–1068. International Foundation for Autonomous Agents and Multiagent Systems, 2016.

Srinivas, A., Jabri, A., Abbeel, P., Levine, S., and Finn, C. Universal planning networks. *arXiv preprint arXiv:1804.00645*, 2018.

Steinhardt, J. Latent variables and model mis-specification. https://jsteinhardt.wordpress.com/2017/01/10/latent-variables-and-model-mis-specification/, 2017.

Steinhardt, J. and Evans, O. Model mis-specification and inverse reinforcement learning. https://jsteinhardt.wordpress.com/2017/02/07/model-mis-specification-and-inverse-reinforcement-learning/, 2017.

Tamar, A., Wu, Y., Thomas, G., Levine, S., and Abbeel, P. Value iteration networks. In *Advances in Neural Information Processing Systems*, pp. 2154–2162, 2016.

Tversky, A. and Kahneman, D. Availability: A heuristic for judging frequency and probability. *Cognitive psychology*, 5(2):207–232, 1973.

Xu, K., Ratner, E., Dragan, A., Levine, S., and Finn, C. Learning a prior over intent via meta-inverse reinforcement learning. *arXiv preprint arXiv:1805.12573*, 2018.

Zheng, J., Liu, S., and Ni, L. M. Robust bayesian inverse reinforcement learning with sparse behavior noise. In *AAAI*, pp. 2198–2205, 2014.

Ziebart, B. D., Maas, A. L., Bagnell, J. A., and Dey, A. K. Maximum entropy inverse reinforcement learning. In *AAAI*, volume 8, pp. 1433–1438. Chicago, IL, USA, 2008.