Feature Expansive Reward Learning: Rethinking Human Input

Andreea Bobu [*1] **Marius Wiggert** [*1] **Claire Tomlin** [1] **Anca Dragan** [1]

Abstract

In collaborative human-robot scenarios, when a person is not satisfied with how a robot performs a task, they can intervene to correct it. Reward learning methods enable the robot to adapt its reward function online based on such human input, but they rely on simple functions of handcrafted features. When the human correction cannot be explained by these features, recent progress in deep Inverse Reinforcement Learning (IRL) suggests that the robot could fall back on demonstrations: ask the human for demonstrations of the task, and recover a reward defined over not just the known features, but also the raw state space. Our insight is that rather than *implicitly* learning about the missing feature(s) from task demonstrations, the robot should instead ask for data that *explicitly* teaches it about what it is missing. We introduce a new type of human input, in which the person guides the robot from areas of the state space where the feature she is teaching is highly expressed to states where it is not. We propose an algorithm for learning the feature from the raw state space and integrating it into the reward function. By focusing the human input on the missing feature, our method decreases sample complexity and improves generalization of the learned reward over the above deep IRL baseline. We show this in experiments with a 7DoF robot manipulator.

1. Introduction

When we deploy robots in human environments, they have to be able to adapt their reward functions to human preferences. For instance in the scenario in Fig. 1, the robot was carrying the cup over the laptop, risking a spill, and the person intervened to correct it. Recent methods interpret such corrections as evidence about the human's desired preference for how to complete the task, enabling the robot to update its reward function online (Bajcsy et al., 2017; Jain et al., 2015; Bajcsy et al., 2018).

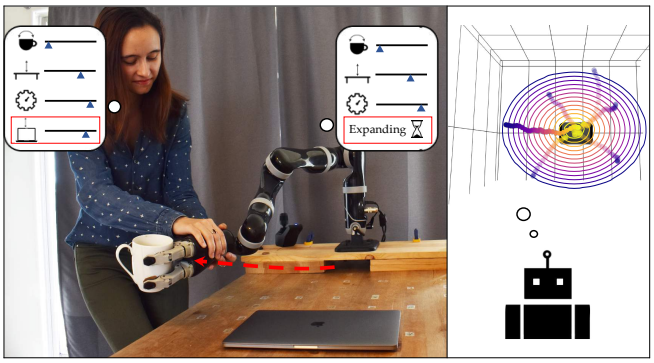

Figure 1. (Left) After the robot detects that its feature space cannot explain the human's input, the person attempts to teach it the concept of distance from the laptop. (Right) The robot queries the human for a few feature traces that teach it the missing feature and adapts the reward to account for it.

Because they have to perform updates online from very little input, these methods resort to representing the reward as a linear function of a small set of hand-engineered *features*. Unfortunately, this puts too much burden on system designers: specifying *a priori* an exhaustive set of *all* the features that end-users might care about is impossible for real-world tasks. While prior work has enabled robots to at least *detect* that the features it has access to cannot explain the human's input (Bobu et al., 2020), it is still unclear how the robot might then construct a feature that can explain it.

A natural answer is in deep IRL methods (Wulfmeier et al., 2016; Finn et al., 2016; Levine et al., 2011), which learn rewards defined directly on the high-dimensional raw state (or observation) space, thereby constructing features automatically. When the robot can't make sense of the human input, it can ask for demonstrations of the task, and learn a reward over not just the known features, but also the raw state space. On the bright side, the learned reward would now be able to explain the demonstrations. On the downside, this may come at the cost of losing generalization when venturing sufficiently far from the demonstrations (Fu et al., 2018; Reddy et al., 2020).

[*]Equal contribution [1]University of California, Berkeley, USA. Correspondence to: Andreea Bobu <abobu@berkeley.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020. Copyright 2020 by the author(s).

In this work, we propose an alternative to relying on demonstrations: we can co-design the learner together with the *type of human feedback* we ask for. Our insight is that instead of learning about the missing feature(s) *implicitly* through the optimal actions, the robot should ask for data that *explicitly* teaches it what is missing. We introduce a new type of human input, which we call *feature traces* – partial trajectories that describe the monotonic evolution of the value of the feature to be learned. To provide a feature trace, the person guides the robot from an area of the state space where the missing feature is highly expressed to states where it is not, in a monotonic fashion.

Looking back at Fig. 1, the person teaches the robot to avoid the laptop by giving a few feature traces: she starts with the arm above the laptop and moves it away until comfortable with the distance from the object. We introduce a reward learning algorithm that harvests the structure inherent to feature traces and uses it to efficiently train a generalizable aspect of the reward: in our example, the distance from the laptop. In experiments on a 7DoF robot arm, we find that by taking control of not only the algorithm but also the kind of data it can receive, our method is able to recover more generalizable rewards with much less human input compared to a learning from demonstrations baseline.

Finally, we discuss our method's potential implications for the general deep reward learning community. Feature traces enable humans to teach robots about salient aspects of the reward in an intuitive way, making it easier to learn overall rewards. This suggests that taking a divide-and-conquer approach focusing on learning important features separately before learning the reward could benefit IRL generalization and sample complexity. Although more work is needed to teach difficult features with even less supervision, we are excited to have taken a step towards better disambiguating complex reward functions that explain human inputs such as demonstrations in the context of reward learning.

2. Method

We consider a robot operating in the presence of a human. The robot has access to a feature set defined over states $\vec{\phi}(s)$, and is optimizing its current estimate of the reward function, $r_\theta(\vec{\phi}(s))$. Here, θ is a vector of parameters specifying how the features combine. If the robot is not executing the task according to the person's preferences, the human can intervene with input a_H. For instance, a_H might be an external torque that the person applies to change the robot's current configuration. Or, they might stop the robot and kinesthetically demonstrate the task, resulting in a trajectory. Building on prior work, we assume the robot can evaluate whether its existing feature space can explain the human input (Sec. 2.4). If it can, the robot directly updates its reward function parameters θ, also in line with prior work (Bajcsy et al., 2017; Ratliff et al., 2006) (Sec. 2.3). But what if it can not? Below, we introduce a method for augmenting the robot's feature space by soliciting further feature-specific human input (Sec. 2.1) and using it to learn a new mapping directly from the raw state space (Sec. 2.2). [1]

2.1. Collecting Feature Traces from Human Input

A state feature is an arbitrary complex mapping $\phi(s) : \mathbb{R}^d \rightarrow [0, 1]$. To learn a non-linear representation of ϕ, we introduce *feature traces* $\xi = s_{0:n}$, a novel type of human input defined as a sequence of n states monotonically decreasing in feature value, i.e. $\phi(s_i) \geq \phi(s_j), \forall i < j$. When learning a feature, the robot first queries the human for a set Ξ of N traces. The person gives a trace by simply moving the system from a state s_0 of their choosing to an end state, nosily ensuring monotonicity.

The power of feature traces lies in their inherent structure. Our algorithm, thus, makes certain assumptions to harvest this structure during the learning procedure. First, we assume that, that the feature values of states along the collected traces $\xi \in \Xi$ are monotonically decreasing. Since humans are imperfect, we allow users to violate the monotonicity assumption by modeling them as noisily rational, following the classic Bradley Terry and Luce-Shepard models of preference (Bradley & Terry, 1952; Luce, 1959):

$$P(s \succ s') = P(\phi(s) > \phi(s')) = \frac{e^{\phi(s)}}{e^{\phi(s)} + e^{\phi(s')}} \quad . \quad (1)$$

Our method also assumes by default that $\phi(s_0) \approx 1$ and $\phi(s_n) \approx 0$, meaning the human starts in a state s_0 where the missing feature is highly expressed, then leads the system to a state s_n along decreasing feature values. Since in some situations providing a 1-0 trace is difficult, our algorithm optionally allows the human to give labels $l_0, l_n \in [0, 1]$[2] for the respective feature values.

To illustrate how a human might offer feature traces, let's turn again to Fig. 1. Here, the person is teaching the robot to keep the mug away from the laptop. The person starts a trace at s_0 by placing the end-effector close to the object center, then leads the robot away from the laptop to s_n. Our method works best when the person tries to be informative, i.e. covers diverse areas of the space: the traces illustrated move radially in all directions and start at different heights. While for some features, like distance from a known object, it is easy to be informative, for others, like slowing down near objects, it might be more difficult. This limitation can

[1] Prior work proposed tackling this issue by iterative boosting (Ratliff et al., 2007) or constructing binary features (Levine et al., 2010; Choi & Kim, 2013). We instead focus on features that can be non-binary, complex non-linear functions of the raw state space.

[2] Since providing decimal labels is challenging, the person gives instead a rating between 0 and 10.

be potentially alleviated using active learning, thereby shifting the burden away from the human to select informative traces, onto the robot to make queries for traces by proposing informative starting states. For instance, the robot could fit an ensemble of functions from traces online, and query for new traces from states where the ensemble disagrees (Reddy et al., 2019).

2.2. Learning a Feature Function

We represent the missing feature by a neural network $\phi_\psi(s) : \mathbb{R}^d \rightarrow [0, 1]$. We use the feature traces ξ, their inherent monotonicity, and the approximate knowledge about $\phi(s_0)$ and $\phi(s_n)$ to train ϕ_ψ.

Using Eq. 1, we frame feature learning as a classification problem with a Maximum Likelihood objective over a dataset of tuples $(s, s', y) \in \mathcal{D}$, where $y \in \{0, 0.5, 1\}$ is a label indicating which state has higher feature value. First, due to monotonicity along a feature trace $\xi = (s_0, s_1, \ldots, s_n)$, we have $\phi_\psi(s_i) \geq \phi_\psi(s_j), \forall j > i$, so $y = 1$ if $j > i$ and 0 otherwise. This results in $\binom{n}{2}$ tuples per trace. Second, we encode our knowledge of $\phi(s_0) \approx 1$ and $\phi(s_n) \approx 1$ for all $\xi \in \Xi$ by encouraging indistinguishable feature values at the starts and ends of traces as $(s_0^i, s_0^j, 0.5), (s_n^i, s_n^j, 0.5) \ \forall \ \xi_i, \xi_j \in \Xi, \ i \neq j, \ i > j$. This results in a total dataset of $|\mathcal{D}| = \sum_{i=1}^{N} \binom{n^i}{2} + 2\binom{N}{2}$ that is already significantly large for a small set of feature traces. The final cross-entropy loss $L(\psi)$ is then given by:

$$- \sum_{(s,s',y)\in\mathcal{D}} y \log(P(s \succ s')) + (1 - y) \log(1 - P(s \succ s')) \tag{2}$$

which expands into the sum of a monotonicity loss for $s \succ s'$ and an indistinguishability loss for $s \sim s'$:

$$- \sum_{s\succ s'} \log \frac{e^{\phi_\psi(s)}}{e^{\phi_\psi(s')} + e^{\phi_\psi(s)}} - \frac{1}{2} \sum_{s\sim s'} \log \frac{e^{\phi_\psi(s)+\phi_\psi(s')}}{(e^{\phi_\psi(s)} + e^{\phi_\psi(s')})^2}. \tag{3}$$

The optionally provided labels l_0, l_n are incorporated as bonus for s_0, s_n as $\phi_\psi(s_0)' = \phi_\psi(s_0) - l_0$ and $\phi_\psi(s_n)' = \phi_\psi(s_n) + (1 - l_n)$ to encourage the labeled feature values to approach the labels.

2.3. Online Reward Update

Once we have a new feature, the robot updates its feature vector $\vec{\phi} \leftarrow (\vec{\phi}, \phi_\psi)$. At this point, the robot goes back to the original human input a_H that previously could not be explained by the old features and uses it to update its estimate of the reward parameters θ. Here, any prior work on online reward learning from user input is applicable, but we highlight one example to make the exposition complete.

Algorithm 1 Feature Adaptive Reward Learning (FERL)

Input: Features $\vec{\phi} = [\phi_1, \ldots, \phi_f]$, weight θ, confidence threshold ϵ, robot trajectory τ, N number of human queries.
while executing τ **do**
  if a_H **then**
    $\hat{\beta} \leftarrow$ estimate_confidence(a_H) as in Sec. 2.4.
    if $\beta < \epsilon$ **then**
      $\Xi = \{\}$
      for $i \leftarrow 1$ **to** N **do**
        $\xi \leftarrow$ query_feature_trace() as in Sec. 2.1.
        $\Xi \leftarrow \Xi \cup \xi$.
      end
      $\phi_{new} \leftarrow$ learn_feature(Ξ) as in Sec. 2.2.
      $\vec{\phi} \leftarrow (\vec{\phi}, \phi_{new}), \theta \leftarrow (\theta, 0.0)$.
    end
    $\theta \leftarrow$ update_reward(a_H) as in Sec. 2.3.
    $\tau \leftarrow$ replan_trajectory(θ).
  end
end

For instance, take the setting where the human's original input a_H was an external torque, applied as the robot was tracking a trajectory τ that was optimal with respect to its current reward. Prior work (Bajcsy et al., 2017) has modeled this as inducing a deformed trajectory τ_H, by propagating the change in configuration to the rest of the trajectory. Further, let θ define linear weights on the reward features. Then, the robot updates its estimate $\hat{\theta}$ in the direction of the feature change from τ to τ_H

$$\hat{\theta}' = \hat{\theta} - \alpha\Big(\sum_{s\in\tau_H} \vec{\phi}(s) - \sum_{s\in\tau} \vec{\phi}(s)\Big) = \hat{\theta} - \alpha(\Phi(\tau_H) - \Phi(\tau)) \ , \tag{4}$$

where Φ is the cumulative feature sum along a trajectory. If instead, the human intervened with a full demonstration, work on online learning from demonstrations (Sec. 3.2 in (Ratliff et al., 2006)) has derived the same update with τ_H now the human demonstration. In our implementation, we use corrections and follow (Bajcsy et al., 2018), which shows that people more easily correct on feature at a time, and only update the θ index corresponding to the feature that changes the most (if our feature learning worked correctly, this is the new feature we just learned). After the update, the robot replans its trajectory using the new reward.

2.4. Confidence Estimation

We lastly have to detect that a feature is missing in the first place. Prior work does so by looking at how people's choices are modeled via the Boltzmann noisily-rational decision model:

$$P(\tau_H \mid \theta, \beta) = \frac{e^{\beta R_\theta(\tau_H)}}{\int_{\bar{\tau}_H} e^{\beta R_\theta(\bar{\tau}_H)} d\bar{\tau}_H} \ , \tag{5}$$

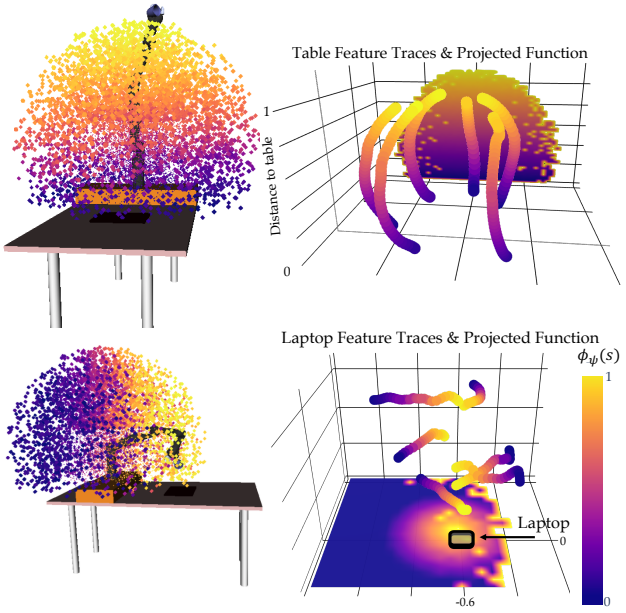

Figure 2. Visualization of the experimental setup, learned feature values $\phi_\psi(s)$, and training feature traces ξ for *table* (up) and *laptop* (down). We display the feature values $\phi_\psi(s)$ for states s sampled from the reachable set of the 7DoF arm, as well as their projections onto the yz and xy planes.

where the human chooses trajectories proportional to their exponentiated reward (Baker et al., 2007; Von Neumann & Morgenstern, 1945). The coefficient $\beta \in [0, \infty)$ controls the degree to which the robot expects to observe human interventions consistent with its feature space. Typically, β is fixed, recovering the Maximum Entropy IRL (Ziebart et al., 2008) observation model. Inspired by work in (Fridovich-Keil et al., 2019; Fisac et al., 2018; Bobu et al., 2020), we instead reinterpret it as a confidence in the robot's features' ability to explain human data. To detect missing features, we estimate $\hat{\beta}$ via a Bayesian belief update $b'(\theta, \beta) \propto P(\tau_H \mid \theta, \beta) b(\theta, \beta)$. If $\hat{\beta}$ is above a threshold ϵ, our method updates the reward as usual with its current features; if $\hat{\beta} < \epsilon$, the current features are insufficient and the robot stops and enters the feature learning mode. Algorithm 1 summarizes the full procedure.

3. FERL for Feature Learning

Before investigating the benefits of our method (FERL) for reward learning, we analyze the quality of the features it can learn. We present results for six different features of varying complexity.

Experimental Design. We conduct our experiments on a 7-DoF JACO robotic arm. We investigate six features that arise in the context of personal robotics: a) *table*: distance of the End-Effector (EE) to the table; b) *coffee*: keeping the coffee cup upright; c) *laptop*: 0.3m xy-plane distance of the EE to a laptop position, to avoid passing over the laptop at any height; d) *test laptop location*: same as *laptop* but the test position differs from the training positions; e) *proxemics*: keeping the EE away from the human, more so when moving in front of them, and less so when moving on their side; f) *between objects*: 0.2m xy-plane distance of the EE to two objects – the feature penalizes collision with objects, and, to a lesser extent, passing in between the two objects. This feature requires some traces with explicit labels l_0, l_n. We approximate all features ϕ_ψ by small neural networks (2 layers, 64 units each), and train them on a set Ξ of feature traces.

For each feature, we collected a set $\mathcal{F}$ of 20 feature traces (40 for the complex *test laptop location* and *between objects*) from which we sample subsets $\Xi \in \mathcal{F}$ for training. As described in Sec. 2.1, the human teaching the feature starts at a state where the feature is highly expressed e.g. for *laptop* that is the EE positioned vertically above the laptop. She then moves the EE away until the distance is equal to the desired bump radius. She does this for a few different directions and heights to provide a diverse dataset. For each feature, we determine what an informative and intuitive set of traces would be, i.e. how to choose the starting states to cover enough of the space, as exemplified in Fig. 2.

Our raw state space consists of the 27D xyz positions of all robot joints and objects in the scene, as well as the rotation matrix of the EE with respect to the robot base. We assume known object positions but they could be obtained from a vision system. It was surprisingly difficult to train on both positions and orientations due to spurious correlations in the raw state space, hence we show results for training only on positions or only on orientations. This speaks to the need for methods that can handle correlated input spaces, which we defer to future work.

We test feature learning by manipulating the number of traces the learner gets access to, and measuring error compared to the ground truth feature ϕ_{True} on a test set of states $\mathcal{S}_{\text{Test}}$. To form $\mathcal{S}_{\text{Test}}$, we uniformly sample 10,000 states from the robot's reachable set. Importantly, many of these test points are far from the training traces, probing the generalization of the learned features ϕ_ψ. We measure error via the Mean-Squared-Error (MSE), $\text{MSE} = \frac{1}{|\mathcal{S}_{\text{Test}}|} \sum_{\mathcal{S}_{\text{Test}}} ||\phi_\psi(s) - \phi_{\text{True}}(s)||^2$. To ground the MSE values, we normalize them with the mean MSE of a randomly initialized untrained feature function, $\text{MSE}_{\text{norm}} = \frac{\text{MSE}}{\text{MSE}_{\text{random}}}$, hence a value of 1.0 equals random performance. For each number of feature traces N, we run 10 experiments sampling different traces from $\mathcal{F}$, and calculate MSE_{norm}.

We test these hypotheses: *H1)* With enough data, FERL learns good features; *H2)* FERL learns increasingly better features with more data; *H3)* FERL becomes less input-

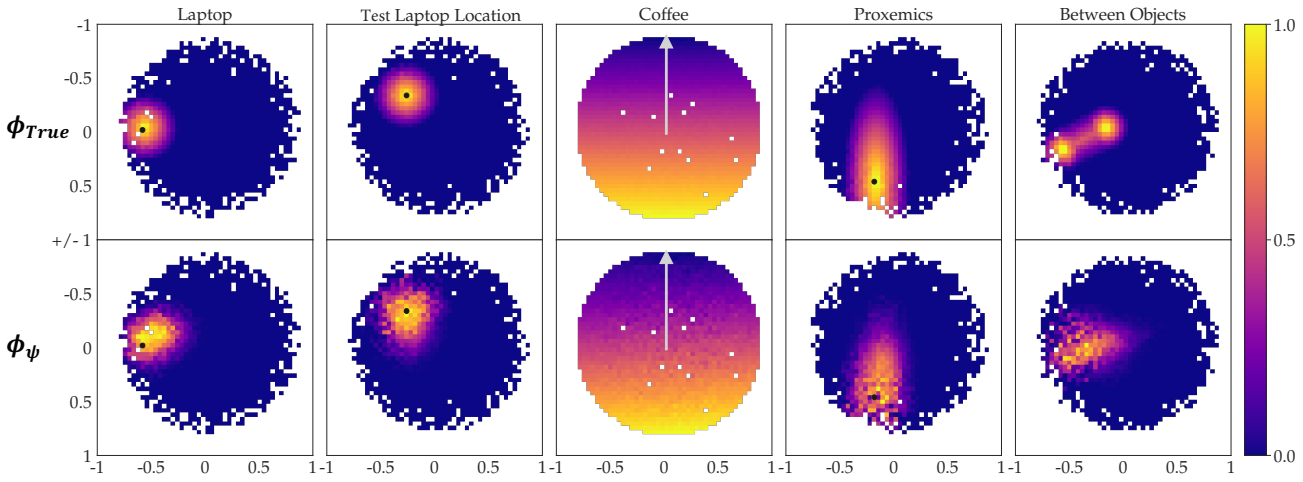

Figure 3. The plots display the ground truth ϕ_{True} and learned feature values ϕ_{ψ} over $\mathcal{S}_{\text{Test}}$, averaged and projected onto a representative 2D subspace: the xy-plane for all Euclidean features, and the xz orientation plane for *coffee* (the arrow represents the cup upright).

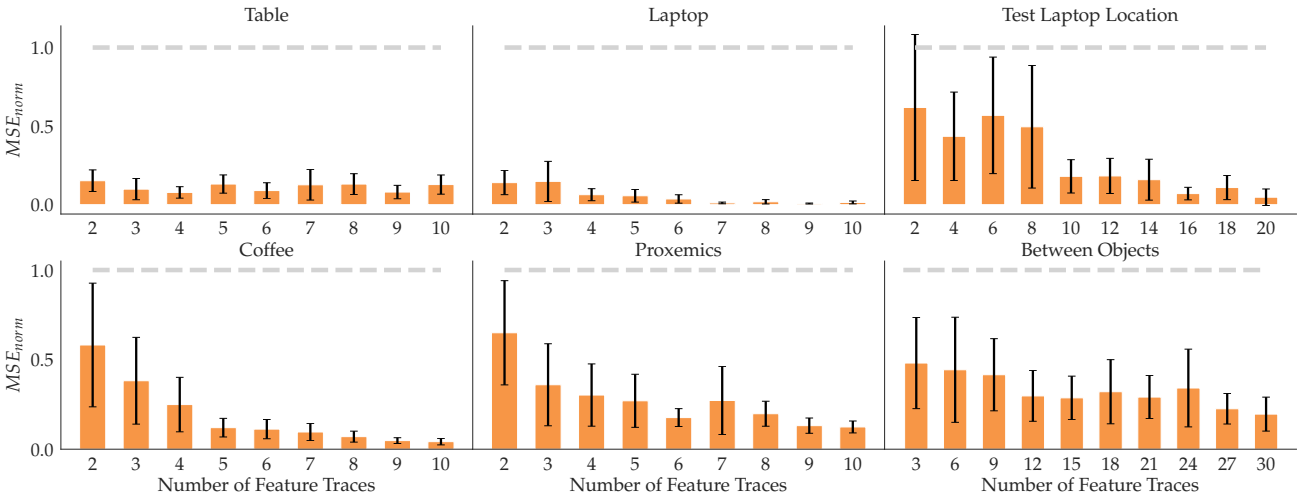

Figure 4. For each feature, we show the MSE_{norm} mean and standard error across 10 random seeds with an increasing number of traces (orange) compared to random performance (gray).

sensitive with more data.

Qualitative results. We first inspect qualitative results with FERL for $N = 10$. In Fig. 2 we show the learned *table* and *laptop* features ϕ_{ψ} by visualizing the position of the EE for all 10,000 points in our test set. The color of the points encodes the learned feature values $\phi_{\psi}(s)$ from low (blue) to high (yellow): *table* is highest when the EE is farthest, while *laptop* peaks when the EE is above the laptop. In Fig. 3, we visualize the remaining features by projecting the test points on 2D sub-spaces and plotting the average feature value per 2D grid point. For Euclidean features we used the EE's *xy*-plane and for *coffee* we project the x-axis basis vector of the EE after forward kinematic rotations onto the *xz*-plane (arrow up represents the cup upright). White pixels are an artifact of sampling.

The top row of Fig. 3 illustrates the Ground Truth (GT) feature values ϕ_{True} and the bottom row the trained feature ϕ_{ψ}. We observe that ϕ_{ψ} resembles ϕ_{True} very well for most features. Our most complex feature, *between objects*, does not recreate the GT as well, although it does learn the general shape. In a post-hoc analysis, we observed that, in fact, our method can learn the fine-grained GT structure for this feature when the raw input space is smaller. This implies that spurious correlation in the input space is a problem, hence for complex features more data or active learning methods to collect informative traces are required.

Quantitative analysis. We now discuss the quantitative results of our experiments for all 6 features. Fig. 4 displays the means and standard errors across 10 seeds for each feature with data increase. The figure drives four core

observations: 1) ϕ_ψ perform significantly better than random, supporting *H1*; 2) the mean MSE_{norm} decreases with data, supporting *H2* and demonstrating that FERL can learn high quality features with little human input; 3) the standard error of MSE_{norm} decreases with data, meaning with more traces FERL is insensitive to the exact input received, supporting *H3*; 4) the more complex a feature, the more traces are needed for good performance: while *table* and *laptop* perform well with just $N=4$, some other features require more traces. In summary, the qualitative and quantitative results support our hypotheses and suggest that our training methodology requires few traces to reliably learn feature functions ϕ_ψ that generalize well to states not seen during training.

4. FERL for Reward Learning

To evaluate FERL for reward learning we compare it to an IRL baseline on 3 representative settings.

Experimental Setup. We compare FERL reward learning to an adapted Maximum Entropy Inverse Reinforcement Learning (ME-IRL) baseline (Ziebart et al., 2008; Finn et al., 2016; Wulfmeier et al., 2016) learning a deep reward function from demonstrations. We model the ME-IRL reward function r_ω as a neural network with 2 layers, 128 units each. For a fair comparison, we gave r_ω access to the known features: once the 27D Euclidean input is mapped to a final neuron, a last layer combines it with the known feature vector.

Also for a fair comparison, we took great care to collect a set of demonstrations for ME-IRL designed to be as informative as possible. Moreover, FERL and ME-IRL rely on different types of human input: FERL on feature traces ξ and pushes a_H and ME-IRL on a set of near-optimal demonstrations $\mathcal{D}^*$. To level the amount of data each method has access to, we collected the sets of traces Ξ and demonstrations $\mathcal{D}^*$ such that ME-IRL has more data points: the average number of states per demonstration/trace in our experiments were 61 and 39, respectively.

We run experiments in three settings in which two features are known and one feature is unknown. In case 1, the laptop feature is unknown and the true reward is $r_{\text{true}} = (0,10,10)(\phi_{\text{coffee}},\phi_{\text{table}},\phi_{\text{laptop}})^T$, in case 2, the table feature is unknown and $r_{\text{true}} = (0,10,10)(\phi_{\text{coffee}},\phi_{\text{laptop}},\phi_{\text{table}})^T$, and in case 3 the proxemics feature is unknown and $r_{\text{true}} = (0,10,10)(\phi_{\text{coffee}},\phi_{\text{table}},\phi_{\text{proxemics}})^T$. We include ϕ_{coffee} with zero weight to evaluate if the methods can learn to ignore irrelevant features.

The gradient of the Maximum Entropy objective with respect to the reward parameters ω can be estimated by: $\nabla_\omega \mathcal{L} \approx \frac{1}{|\mathcal{D}^*|}\sum_{\tau\in\mathcal{D}^*}\nabla_\omega R_\omega(\tau) - \frac{1}{|\mathcal{D}^\omega|}\sum_{\tau\in\mathcal{D}^\omega}\nabla_\omega R_\omega(\tau)$

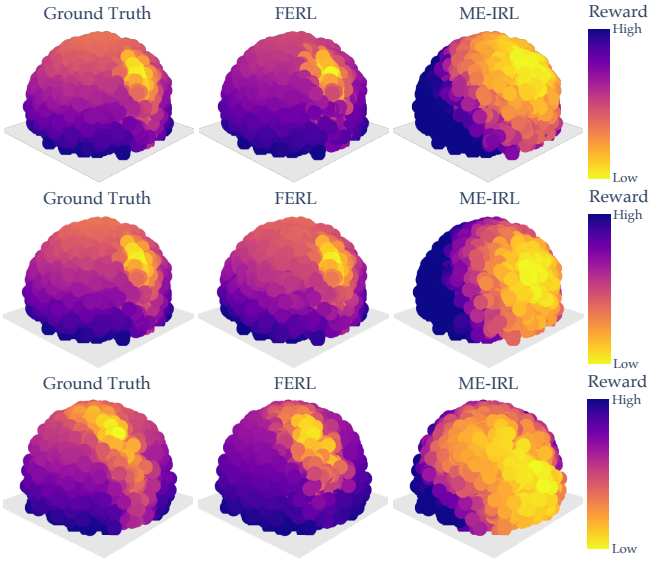

Figure 5. Visual comparison of the ground truth, FERL, and ME-IRL rewards, for case 1 (up), case 2 (middle), and case 3 (down).

(Wulfmeier et al., 2016; Finn et al., 2016). $R_\omega(\tau) = \sum_{s\in\tau} r_\omega(s)$ is the parametrized reward, $\mathcal{D}^*$ the set of expert demonstrations, and $\mathcal{D}^\omega$ are trajectory samples from the r_ω induced near optimal policy. We use TrajOpt (Schulman et al., 2013) to obtain the current set of samples $\mathcal{D}^\omega$, and optimize the loss with Adam (Kingma & Ba, 2015). We validated our ME-IRL implementation in experiments and observe that it quickly learns a reward that induces a state expectation very close to the expert demonstrations $\mathcal{D}^*$.

We compare the two reward learning methods across two metrics commonly used in the IRL literature (Choi & Kim, 2011): 1) *Reward Accuracy*: how close to GT the learned reward is by some distance metric, and 2) *Behavior Accuracy*: how well do the behaviors induced by the learned rewards compare to the GT optimal behavior, measured by evaluating the induced trajectories on GT reward.

For *Reward Accuracy*, we manipulate the number of traces/demonstrations each learner gets access to, and measure the MSE compared to the GT reward on $\mathcal{S}_{\text{Test}}$, similar to Sec. 3. For *Behavior Accuracy*, we train FERL and ME-IRL with a set of 10 traces/demonstrations that we deemed to be most informative. We then use TrajOpt (Schulman et al.) to produce optimal trajectories for 100 randomly selected start-goal pairs under the learned rewards. We evaluate the trajectories with the GT reward r_{true} and divide by the reward of the GT induced trajectory for easy relative comparison.

We use these metrics to test the hypotheses: *H1)* FERL learns rewards that better generalize to the state space than ME-IRL; *H2)* FERL performance is less sensitive to the specific input than ME-IRL.

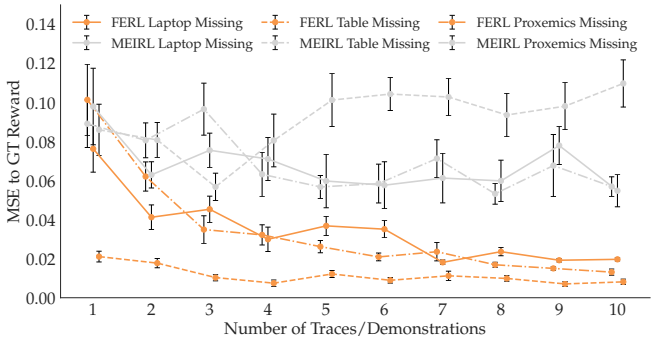

Figure 6. MSE of FERL and ME-IRL to GT reward.

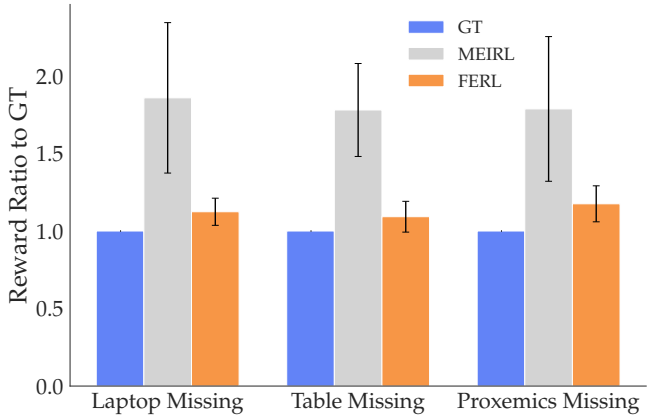

Figure 7. Induced trajectories' reward ratio.

Qualitative Comparison. In Fig. 5, we show the learned FERL and ME-IRL rewards as well as the GT for our 3 cases evaluated at the test points. For case 1 (top), as we can see, by first learning the *laptop* feature and then the reward on the extended feature vector, FERL is able to learn a fine-grained reward structure closely resembling GT. Meanwhile, ME-IRL learns some structure capturing where the laptop is, but not enough to result in a good trade-off between the active features. Similarly, we observe that in case 2 (middle) FERL is able to learn a fine-grained reward structure closely resembling GT. In case 3 (bottom), for the more difficult *proxemics*, FERL with just 10 features traces is not perfect but still recovers most of the reward structure. In both cases, ME-IRL only learns a coarse structure with a broad region of low reward which does not capture the intricate trade-off of the true reward function.

Quantitative Analysis. To compare *Reward Accuracy*, we show in Fig. 6 the MSE mean and standard error across 10 seeds, with increasing training data. We visualize results from all 3 cases, with FERL in orange and ME-IRL in gray. FERL is closer to GT than ME-IRL no matter the amount of data, supporting *H1*. Additionally, the consistently decreasing mean MSE for FERL suggests that our method gets better with more data; in contrast, the same trend is inexistent with ME-IRL. Supporting *H2*, the high standard error that ME-IRL displays implies that ME-IRL is highly sensitive to the demonstrations provided and the learned reward likely overfits to the expert demonstrations. With more data, this shortcoming might disappear; however, this would pose an additional burden on the human, which our method successfully alleviates.

Lastly, we looked at *Behavior Accuracy* for the two learned rewards. Fig. 7 illustrates the reward ratios to GT for all three cases. The GT ratio is 1 by default, and the closer to 1 the ratios are, the better the performance because all rewards are negative. The figure confirms *H1*, showing that FERL rewards produce trajectories that are preferred under the GT reward over ME-IRL reward trajectories.

5. Discussion of Results, Implications, and Future Work

In this work, we proposed FERL, a framework for learning rewards from corrections when the initial feature set cannot capture human preferences. Based on our insight that the robot should ask for data that *explicitly* teaches it what is missing, we introduced feature traces ξ as a novel type of human input that allows for intuitive teaching and learning of non-linear features from high-dimensional state spaces. In experiments, we analyzed the quality of the learned features and showed that FERL outperforms a deep reward learning from demonstrations baseline (ME-IRL) in terms of data-efficiency, generalization, and sensitivity to input data.

Potential Implications for Learning Complex Rewards from Demonstrations. Reward learning from raw state space with expressive function approximators is considered difficult because there exists a large set of functions $r_\theta(s)$ that could explain the human input, e.g. for demonstrations many functions $r_\theta(s)$ induce policies that match the state expectation of the demonstrations. The higher dimensional the state $s \in \mathbb{R}^d$, the more information from human input is needed to disambiguate between those functions sufficiently to find a reward r_θ which accurately captures human preferences and thereby generalizes to states not seen during training and not just replicates the demonstrations' state expectations as in IRL. We are hopeful that our method of collecting feature traces rather than just demonstrations has potential implications broadly for non-linear (deep) reward learning, as a way to better disambiguate the reward and improve generalization.

While in this paper we focused on adapting a reward online, we also envision our method used as part of a "divide-and-conquer" alternative to IRL: first, collect feature traces for the important non-linear criteria of the reward, and then use demonstrations to figure out how to (shallowly) combine

them. The reason this might help relative to relying on demonstrations for everything is that demonstrations aggregate a lot of information. First, by learning features, we can isolate learning what matters, from learning how to trade off or combine what matters into a single value (the features vs. their combination) – in contrast, demonstrations have to teach the robot about both at once. Second, feature traces give information about states that are not going to be on optimal trajectories, be it states with high feature values that are undesirable, or states with low feature values where other, more important features have high values. Third, feature traces are also structured by the monotonicity assumption: they tell us relative feature values of the states along a trace, whereas demonstrations only tell us about the reward value in aggregate across a trajectory. These might be the reasons why we saw the result in Fig. 6, 7, where the FERL reward reliably generalized better to new states than the demonstration-only based IRL.

Limitations and Future Work. There are four main limitations of FERL which we seek to address in future work. First, due to the current pandemic, we could not run a user study, so we do not know how non-expert users provide feature traces. Second, with the current feature learning protocol, it is cumbersome to teach discontinuous features, so we would like to extend our approach by other feature learning protocols.Third, while we show that FERL works reliably in 27D, we want to extend it to higher dimensional state spaces. Our initial results in the appendix show that this is difficult if the raw states are highly correlated. We believe techniques from causal learning, such as Invariant Risk Minimization (Arjovsky et al., 2019), can be helpful. Lastly, we want to further ease the human supervision burden by developing an active learning approach where the robot autonomously picks starting states most likely to result in informative feature traces. To prevent learning incorrect features, we want to enable the human to validate a learned feature, e.g. through visualization before it is added to the feature set.

References

Arjovsky, M., Bottou, L., Gulrajani, I., and Lopez-Paz, D. Invariant risk minimization. *ArXiv*, abs/1907.02893, 2019.

Bajcsy, A., Losey, D. P., O'Malley, M. K., and Dragan, A. D. Learning robot objectives from physical human interaction. In *CoRL*, 2017.

Bajcsy, A., Losey, D. P., O'Malley, M. K., and Dragan, A. D. Learning from physical human corrections, one feature at a time. In *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, HRI '18, pp. 141–149, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-4953-6. doi: 10.1145/3171221.3171267. URL http://doi.acm.org/10.1145/3171221.3171267.

Baker, C., B Tenenbaum, J., and R Saxe, R. Goal inference as inverse planning. 01 2007.

Bobu, A., Bajcsy, A., Fisac, J. F., Deglurkar, S., and Dragan, A. D. Quantifying hypothesis space misspecification in learning from human–robot demonstrations and physical corrections. *IEEE Transactions on Robotics*, pp. 1–20, 2020.

Bradley, R. A. and Terry, M. E. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

Choi, J. and Kim, K.-E. Inverse reinforcement learning in partially observable environments. *Journal of Machine Learning Research*, 12(Mar):691–730, 2011.

Choi, J. and Kim, K.-E. Bayesian nonparametric feature construction for inverse reinforcement learning. In *Twenty-Third International Joint Conference on Artificial Intelligence*, 2013.

Finn, C., Levine, S., and Abbeel, P. Guided cost learning: Deep inverse optimal control via policy optimization. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pp. 49–58. JMLR.org, 2016.

Fisac, J. F., Bajcsy, A., Herbert, S. L., Fridovich-Keil, D., Wang, S., Tomlin, C. J., and Dragan, A. D. Probabilistically safe robot planning with confidence-based human predictions. *Robotics: Science and Systems (RSS)*, 2018.

Fridovich-Keil, D., Bajcsy, A., Fisac, J. F., Herbert, S. L., Wang, S., Dragan, A. D., and Tomlin, C. J. Confidence-aware motion prediction for real-time collision avoidance. *International Journal of Robotics Research*, 2019.

Fu, J., Luo, K., and Levine, S. Learning robust rewards with adverserial inverse reinforcement learning. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rkHywl-A-.

Jain, A., Sharma, S., Joachims, T., and Saxena, A. Learning preferences for manipulation tasks from online coactive feedback. *The International Journal of Robotics Research*, 34(10):1296–1313, 2015.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

Levine, S., Popovic, Z., and Koltun, V. Feature construction for inverse reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 1342–1350, 2010.

Levine, S., Popovic, Z., and Koltun, V. Nonlinear inverse reinforcement learning with gaussian processes. In *Advances in Neural Information Processing Systems*, pp. 19–27, 2011.

Luce, R. D. *Individual choice behavior.* John Wiley, Oxford, England, 1959.

Ratliff, N., Bradley, D. M., Chestnutt, J., and Bagnell, J. A. Boosting structured prediction for imitation learning. In *Advances in Neural Information Processing Systems*, pp. 1153–1160, 2007.

Ratliff, N. D., Bagnell, J. A., and Zinkevich, M. A. Maximum margin planning. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pp. 729–736, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595933832. doi: 10.1145/1143844.1143936. URL https://doi.org/10.1145/1143844.1143936.

Reddy, S., Dragan, A. D., Levine, S., Legg, S., and Leike, J. Learning human objectives by evaluating hypothetical behavior, 2019.

Reddy, S., Dragan, A. D., and Levine, S. Sqil: Imitation learning via reinforcement learning with sparse rewards. *arXiv: Learning*, 2020.

Schulman, J., Ho, J., Lee, A., Awwal, I., Bradlow, H., and Abbeel, P. Finding locally optimal, collision-free trajectories with sequential convex optimization.

Schulman, J., Ho, J., Lee, A. X., Awwal, I., Bradlow, H., and Abbeel, P. Finding locally optimal, collision-free trajectories with sequential convex optimization. In *Robotics: science and systems*, volume 9, pp. 1–10. Citeseer, 2013.

Von Neumann, J. and Morgenstern, O. *Theory of games and economic behavior*. Princeton University Press Princeton, NJ, 1945.

Wulfmeier, M., Wang, D. Z., and Posner, I. Watch this: Scalable cost-function learning for path planning in urban environments. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2089–2095, 2016.

Ziebart, B. D., Maas, A., Bagnell, J. A., and Dey, A. K. Maximum entropy inverse reinforcement learning. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*, AAAI'08, pp. 1433–1438. AAAI Press, 2008. ISBN 978-1-57735-368-3. URL http://dl.acm.org/citation.cfm?id=1620270.1620297.