REVEALE: Reward Verification and Learning Using Explanations

Saaduddin Mahmud[1], Sandhya Saisubramanian[2] and Shlomo Zilberstein[1]

[1]University of Massachusetts Amherst, Massachusetts, USA
[2]Oregon State University, Oregon, USA

Abstract
When a human expert demonstrates the desired behavior, there often exist multiple reward functions consistent with the observed demonstrations. As a result, agents often learn a proxy reward function to encode their observations. Operating based on proxy rewards may be unsafe. Furthermore, black-box representations make it difficult for the demonstrator to verify the learned reward function and prevent harmful behavior. We investigate the efficiency of using explanations to update and verify a learned reward function, to ensure that it aligns with the demonstrator's intent. The problem is formulated as an inverse reinforcement learning from ranked expert demonstrations, with verification tests to validate the alignment of the learned reward. The agent explains its reward function and the human signals whether the explanation passes the verification test. When the explanation is rejected, the agent presents additional alternative explanations to acquire feedback, such as a preference ordering over explanations, which helps it learn the intended reward. We analyze the efficiency of our approach in learning reward functions from different types of explanations and present empirical results on five domains. Our results demonstrate the effectiveness of our approach in learning and generalizing human-aligned rewards.

Keywords
Reward Alignment, Explanation Generation, Inverse Reinforcement Learning

1. Introduction

With dramatic recent advances in artificial intelligence, autonomous agents are being increasingly deployed in the real world to complete complex and nuanced tasks [1]. A predominant way to train such agents in the absence of a reward function is *learning from demonstration* (LfD) [2]. *Inverse reinforcement learning* (IRL) is a form of LfD designed to retrieve a reward function that captures the demonstrator's behavior [3], allowing agents to learn and generalize the observed behavior to unseen situations.

Despite the success of IRL in many research settings, two key limitations may lead to unsafe behavior of the deployed system: (1) the demonstrations may cover only a subset of states, providing no direct information about acceptable behavior in other states; and (2) a large space of candidate reward functions may be consistent with the demonstrations, each producing slightly different behavior in states that were not visited in the demonstrations. Consequently, an agent may learn a proxy reward that leads to unpredictable, unsafe behavior when encountering novel situations.

Figure 1 illustrates this challenge with an autonomous vehicle (AV) approaching a pedestrian walking two dogs. In the demonstration data, the driver always stops when a human is crossing the street with dogs. Since dogs are often accompanied by humans, the rare case of encountering dogs alone might be missing from the dataset. Consider four different reward functions consistent with the demonstration, $(R_i, 1 \leq i \leq 4)$, each with the same negative reward for not stopping in this case. R_1 does not account for dogs, R_2 rewards stopping for pedestrians with dogs, R_3 rewards stopping for pedestrians or dogs, and R_4 rewards stopping for all objects, including leaves or a plastic bag on the road. In the absence of additional information, the AV may randomly learn one of these reward functions (say R_2), however, R_3 represents the true intent of the demonstrator. When operating based on R_2, the AV may not stop for dogs unaccompanied by humans. This example illustrates the inherent reward ambiguity in IRL and the consequences of learning a proxy reward.

Existing IRL methods aim to resolve reward ambiguity by either introducing heuristics such as Max Margin [4] or Max Entropy [5], or by combining additional information such as a trajectory ranking [6] or a preference differentiator [7]. However, these approaches are not guaranteed to avoid reward ambiguity and they do not verify the learned reward. Recently, Brown et al. [8] introduced an approach to verify the agent's value or policy, but it does not amend the reward if it is misaligned. Further, these methods are not interpretable and may require additional knowledge, such as the value function.

To address this issue, we introduce a general framework that utilizes explanations to learn a reward function that is aligned with the demonstrator's intent. Our framework for **re**ward **ve**rification **a**nd **l**earning using

The AAAI-23 Workshop on Artificial Intelligence Safety (SafeAI 2023), Washington D.C., USA.
smahmud@umass.edu (S. Mahmud);
sandhya.sai@oregonstate.edu (S. Saisubramanian);
shlomo@umass.edu (S. Zilberstein)

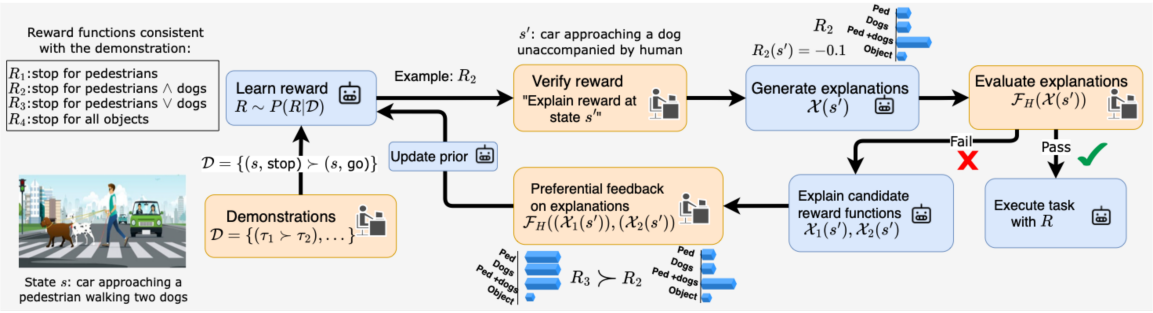

Figure 1: An example of reward learning and verification with REVEALE in autonomous navigation. Existing IRL approaches learn a reward function that aligns with the demonstration (R_2), which may not be the intended reward function. Our approach uses feature attribution based explanations to learn and verify if the system has learned the intended reward function (R_3).

explanations (REVEALE) consists of a *reward learning phase* and a *verification phase*. In the reward learning phase, the agent learns a reward function based on the demonstrations. In the verification phase, the demonstrator verifies the reward alignment through verification tests in the form of queries to the agent. The agent responds by explaining its reward, and the demonstrator signals whether the model passes the verification test. When it fails, the agent queries the demonstrator by presenting additional explanations from alternative candidate reward models. The demonstrator provides feedback by selecting the explanation that matches most closely their intended reward. This is followed by the reward learning phase, in which the agent updates its prior over candidate reward functions, based on the feedback. Thus, REVEALE can *identify and fix inconsistencies* in the reward function by alternating between the learning and verification until the verification test is passed.

We use verification tests in the form of a query: "explain the reward at state s", and explanations in the form of feature attribution. We use feature attribution-based explanations, due to their simplicity, but the framework is general and can work with any form of explanation that can help the demonstrator interpret a reward function. An example of a verification test for the scenario described in Figure 1 is "explain the reward when the AV encounters a dog accompanied by humans," to which the agent would respond with its reward value and feature attributions indicating a low weight for the 'dogs' feature. This reveals a potential weakness of the model in the counterfactual scenario in which the dog is not accompanied by a human (missing from the dataset). When this fails the verification test, the agent explains another candidate reward function (for example, R_2 and R_3). The demonstrator then selects an explanation that is closer to their intended reward (R_3 in this case), indicating that R_3 is preferred over R_2 and the desired behavior is to stop for pedestrians or dogs.

Our key contributions are: (1) introducing a general framework for verifying and learning human-aligned reward from demonstrations and using explanations; (2) presenting an algorithm to generate explanations as feature attributions for reward functions and verify the learned reward through human feedback; (3) analyzing the reduction in reward ambiguity for linear rewards; and (4) demonstrating empirically the effectiveness of our approach on five domains.

2. Related Work

Reward learning Most IRL algorithms learn a reward function using expert trajectories [4, 9, 10]. Recent algorithms utilize additional information to improve reward learning, such as preferences over trajectories [11, 12], prior over reward functions [13], or feature queries [7]. A key obstacle to the safe deployment of an autonomous agent is the long tail of novel situations [14] that cannot be predicted by the demonstrations a priori. In fact, our experiments show that adding additional demonstrations or preferences over trajectories does not guarantee improvement in the learned reward. Further, unlike Basu et al. [7] that uses human feedback to identify a feature that affects trajectory preferences, we use feedback to identify which automatically generated explanation aligns best with the intended reward, thereby reducing reward ambiguity. While the former approach is limited to linear rewards, our approach generalizes to nonlinear cases. Finally, none of the existing IRL approaches perform reward verification. Our approach is complementary to many of the existing reward learning methods, as the reward verification and explanation phases can be used in tandem with any type of reward learning.

Value alignment Value alignment focuses on ensuring that an agent behavior is aligned with its user's intentions. Unlike the inverse reward design approach [15] that aims to retrieve the intended reward by treating the specified reward function as a proxy, we learn the true reward func-

tion using human feedback on automatically generated explanations of potential reward models. While some recent work focused on value alignment verification (VAV) with a minimum number of queries [8], our work differs in that: (1) we use human feedback in the form of preferences over reward explanations to verify the reward, while VAV uses reward weights, value weights or trajectory preferences for verification; and (2) VAV can detect but cannot amend misaligned rewards, while our approach verifies and rectifies incorrect rewards. Further, VAV can check the consistency of the value function only in situations that occur during training, and cannot verify the performance in novel situations that the agents may encounter after deployment.

Explainable AI For autonomous systems to be widely adopted, user trust in the systems' capability must be built [16], and it is widely accepted that explanations can induce trust [17]. Much of the existing work on explainable AI uses feature attributions as explanations to help understand the relationship between input features and the output of a learned model. Some of the widely used techniques are LIME [18], meanRESP[19], SHAP [20], gradient as explanation (GaE) [21] and saliency maps [22]. Besides feature attribution, there are other broad classes of automated explanation generation methods such as model reconciliation [23] and policy summarization [24]. While these existing approaches typically use explanations to improve interpretability, we use them to verify and improve the reward model. Relevant to our work is [25] which uses a policy summarization technique to explain reward function to humans in order to induce trust. Another related line of work uses a model reconciliation method to improve humans' understanding of reward function for better collaboration [26]. Unlike these approaches where the focus is to induce trust or improve collaboration, our framework uses explanations to simultaneously learn and verify human-aligned reward.

3. Background

Markov decision process A Markov Decision Process (MDP) M is represented by the tuple $M = (S, A, T, R, S_0, \gamma)$ where S is a finite set of states, A is a finite set of actions, $T : S \times A \times S \to [0, 1]$ is the transition function, $R : S \times A \to \mathbb{R}$ is the reward function, S_0 is the initial state distribution, and $\gamma \in [0, 1)$ is the discount factor. A policy $\pi : S \times A \to [0, 1]$ is a mapping from states to a distribution over actions. The state and state-action values of a policy π are $V^\pi(s) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R(s_t) \mid s_0, \pi]$ and $Q^\pi(s, a) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R(s_t) \mid s_0 = s, a_0 = a, \pi], \forall s \in S$ and $a \in A$. The optimal values are denoted by $V^*(s) = \max_\pi V^\pi(s)$ and $Q^*(s, a) = \max_\pi Q^\pi(s, a)$.

Bayesian IRL A Bayesian framework for IRL defines a probability distribution of reward functions given a demonstration dataset $\mathscr{D}$ using Bayes rule, $\mathscr{P}(R|\mathscr{D}) \propto (\mathscr{D}|R)P(R)$. Various algorithms define $P(\mathscr{D}|R)$ differently. We use the definition from B-REX [11], as it is scalable. Let τ^i and τ^j denote two different trajectories. Expert demonstrations are given in the form of preferences, $\tau^i \succ \tau^j$, indicating that τ^i is preferred over τ^j. The demonstration data is denoted by $\mathscr{D} = \{(\tau_1^1 \succ \tau_1^2), (\tau_2^1 \succ \tau_2^2), ..., (\tau_n^1 \succ \tau_n^2), \}$, where the trajectory τ_i^1 is preferred over trajectory τ_i^2. Hence, such data are called *preferential dataset*. B-REX defines $\mathscr{P}(\mathscr{D}|R)$ as:

$$\mathscr{P}(\mathscr{D}|R) = \prod_{(\tau_i \succ \tau_j) \in \mathscr{D}} \frac{e^{\beta R(\tau_i)}}{e^{\beta R(\tau_i)} + e^{\beta R(\tau_j)}},$$

where, $R(\tau) = \sum_{s \in \tau} R(s)$ and $\beta \in [0, \infty)$.

4. The REVEALE Framework

Consider an agent operating in an environment modeled as a Markov decision process (MDP), $M = (S, A, T, \mathbf{R}, S_0, \gamma)$ where the reward function $\mathbf{R}$ is initially *unknown* to the agent. The agent aims to learn $\mathbf{R}$ using expert demonstration data $\mathscr{D}$. We consider a factored state representation.

Definition 1. *Given an MDP M with an unknown reward function $\mathbf{R}$, a REVEALE instance is defined as $\langle M, \mathscr{D}, \mathscr{R}, \mathscr{P}, S_V, \mathscr{E}, \mathscr{X}, V_T, \mathscr{F} \rangle$ where*

- *M denotes the underlying MDP with $\mathbf{R}$ initially unknown to the agent;*
- *$\mathscr{D}$ denotes the demonstration data;*
- *$\mathscr{R}$ denotes the space of possible reward functions for M that are consistent with $\mathscr{D}$;*
- *$\mathscr{P}$ is the probability distribution over the reward functions in $\mathscr{R}$, with $\mathscr{P}(R|D) \in [0, 1]$ denoting the probability of a reward function $R \in \mathscr{R}$ being the true reward function[1] $\mathbf{R}$, given the demonstration data $\mathscr{D}$;*
- *$S_V \subseteq S$ denotes the set of states used by the human to verify the agent's learned reward;*
- *$\mathscr{E}$ is the space of explanations, corresponding to $\mathscr{R}$;*
- *$\mathscr{X} : S_V \times V_T \to \mathscr{E}$ is the agent's explanation generation function that generates an explanation $e \in \mathscr{E}$, given a verification test $v_T \in V_T$ and a state $s_V \in S_V$; and*
- *$\mathscr{F}$ is the feedback provided by the human, in response to the agent's explanation $e \in \mathscr{E}$.*

[1] A reward function that captures the true intention of the demonstrator

We assume that the agent has access to a limited number of expert demonstrations to learn $\mathbf{R}$. While the general framework can leverage different types of demonstrations, explanation generation, and verification techniques, we target settings where the demonstrations are in the form of trajectory preferences, explanations are generated as reward value and feature attribution, and verification tests describe the reward function at certain critical states identified by the human. In Section 5, we briefly discuss how to handle other forms of explanations, demonstrations, and IRL algorithms.

Demonstration data Similar to [11], we use expert demonstrations in the form of pairwise preference over trajectories, $\mathscr{D}=\{(\tau_1^1 \succ \tau_1^2), \dots, (\tau_n^1 \succ \tau_n^2)\}$, called a *preferential dataset*, due to its computational efficiency.

Based on [9, 27], the space of reward functions consistent with a policy and preferential demonstrations are defined as follows.

Definition 2. *Given an MDP M, a* ***policy-consistent reward set****, denoted $\Delta(\pi)$, is a set of reward functions under which π is optimal, $\Delta(\pi) = \{R \in \mathscr{R} \mid \pi, s_0, V_R^{\pi}(s_0) = V^*(s_0)\}$ where s_0 is the initial state.*

Definition 3. *Given an MDP M and preferential dataset $\mathscr{D}$, a* ***demonstration-consistent reward set****, denoted $\Delta(\mathscr{D})$, is a set of reward functions under which the preferred trajectories have a higher reward than the less preferred trajectories, $\Delta(\mathscr{D}) = \{R \in \mathscr{R} \mid R(\tau_i^1) > R(\tau_i^2), \forall (\tau_i^1 \succ \tau_i^2) \in \mathscr{D}\}$.*

In practice, $\mathscr{D}$ may not cover all states and there may be mismatches in the training and testing environments [15, 28]. Such situations lead to reward ambiguity and the agent may end up learning a proxy reward. REVEALE overcomes these drawbacks by generating explanations of the reward functions that are consistent with the expert's policy/demonstrations, which are verified by the demonstrator.

Explanation generation function ($\mathscr{X}$) The agent's explanations involve two components: (1) the *reward* at the verification test state s_V, denoted by $R(s_V)$, following the most likely reward function; and (2) *feature attribution* indicating the influence of state features on the reward function. Attribution-based explanations are commonly used in interpretable machine learning [29].

A feature attribution is a scoring function denoting the contribution of each state feature to the output, $FA : S \times \mathscr{R} \rightarrow \mathbb{R}^{|S|}$. This is a form of *local* explanation as it only explains the reward function at each state in isolation. We use established local explanation techniques mentioned earlier: gradient as explanation (GaE), LIME, and saliency maps.

Verification and feedback We use verification tests of the form "Explain the reward at state s_V" with $s_V \in S_V$ selected by the demonstrator. Hence, $\mathscr{X}(s_V, v_T) = \mathscr{X}(s_V), \forall s_V, v_T$. However, S_V can also be generated automatically using techniques such as policy summarization [30]. The agent responds by automatically generating explanations, consisting of the reward value at s_V and the feature attribution describing the reward function.

Approval: A binary signal indicating whether the demonstrator approves ($\mathscr{F}_A(\mathscr{X}(s_V)) = 1$) or disapproves ($\mathscr{F}_A(\mathscr{X}(s_V)) = 0$) the explanation, denoting the outcome of the verification test.

Explanation feedback: When the verification test fails, the demonstrator provides accurate feedback on explanations generated by the agent in one of the following two forms, used by the agent to update $\mathscr{P}$.

(1) Oracle explanations, typically provided by the human, in the form of exact feature attribution corresponding to their intended reward, $\mathscr{F}_O(\mathscr{X}(s_V)) = \mathscr{X}_O(s_V), \forall s_V \in S_V$, where $\mathscr{X}_O(s_V)$ denotes the exact feature attribution generated by the Oracle. Though this is an ideal setting as $\mathscr{X}_O(s_V)$ provides features that are critical to learning the intended reward, this type of feedback can be harder to collect in practice, except for simpler domains.

(2) Pairwise preference over feature attributions generated by the agent, $\mathscr{F}_P(\mathscr{X}_1(s_V), \mathscr{X}_2(s_V)) = \mathscr{X}_P(s_V), \forall s_V \in S_V$ where $\mathscr{X}_P(s_V) \in \{\mathscr{X}_1(s_V), \mathscr{X}_2(s_V)\}$ and $\mathscr{X}_1(s_V)$ and $\mathscr{X}_2(s_V)$ denote explanations of two different reward models that are consistent with $\mathscr{D}$. This is a more realistic form of feedback that identifies the explanation that better captures the intended reward.

Definition 4. *Given an MDP M and a set of verification states S_V, an* ***explanation-consistent reward set****, denoted $\Delta(\mathscr{X}(S_V))$, is a set of reward functions whose corresponding explanations are approved by the demonstrator, $\Delta(\mathscr{X}(S_V)) = \{R \in \mathscr{R} \mid \mathscr{F}_A(\mathscr{X}_R(s_V)) = 1, \forall s_V \in S_V\}$.*

Definition 5. ***Reward Ambiguity*** *is a measure proportional to the size of the consistent reward set Δ, such as $|\Delta|$ when Δ is finite and discrete, and volume of Δ, $\mathscr{V}(\Delta)$, when Δ is continuous, as in a simplex in $\mathbb{R}^k$.*

REVEALE aims to eliminate reward ambiguity, by reducing the size of consistent reward set $\Delta(\mathscr{D}) \cap \Delta(\mathscr{X}(S_V))$, using verification tests and feedback on explanations.

Definition 6. *A solution of a REVEALE instance is a reward function, $R \in \Delta(\mathscr{D}) \cap \Delta(\mathscr{X}(S_V))$, that is better aligned with the demonstrator's intent.*

An optimal solution eliminates reward ambiguity and identifies a reward function that is aligned with the demonstrator's intended reward, when feasible. The following section presents an algorithm that produces a solution.

Algorithm 1: ILV

Input: Demonstration data $\mathscr{D}$, verification states S_V
Output: Final Reward R_b and test score

1: $\mathscr{C} = \emptyset$
2: $\mathscr{C}_{s_V} = \emptyset; \forall s_V \in S_V$
3: $R_b \sim \mathscr{R}$
4: $T_b = [-\infty, -\infty]$
5: $\epsilon = |S_V|$
6: **while** $\epsilon > 0$ **do**
7: R_m = MAP estimation of $\mathscr{P}(R|\mathscr{D}, \mathscr{C})$
8: $T_m = [\mathscr{P}(\mathscr{D}|R_m), 0]$
9: **for** $s_V \in S_V$ **do**
10: $\mathscr{F}_A(\mathscr{X}_{R_m}(s_V)) = Get\_approval(\mathscr{X}_{R_m}(s_V))$
11: **if** $\mathscr{F}_A(\mathscr{X}_{R_m}(s_V)) = 1$ **then**
12: $\mathscr{C} = \mathscr{C} \cup \mathscr{X}_{R_m}(s_V)$
13: **else**
14: $\mathscr{C}_{s_V} = \mathscr{C}_{s_V} \cup \mathscr{X}_{R_m}(s_V)$
15: $\mathscr{C} = Get\_feedback(\mathscr{C}, s_V, Select\_queries(\mathscr{C}_{s_V}))$
16: $T_m[2] = Get\_Score(\mathscr{X}_{R_m}(S_V))$
17: **if** $T_m \geq T_b$ **then**
18: $R_b = R_m$
19: $T_b = T_m$
20: $\epsilon = \sum_{s_V} 1 - \mathscr{F}_A(\mathscr{X}_{R_m}(s_V))$
21: **return** R_b, T_b

5. Solution Approach

Our algorithm, *iterative learning and verification of reward* (ILV), is outlined in Algorithm 1. The input is a set of demonstrations, $\mathscr{D}$, and verification test states, S_V. We assume that the final reward is the MAP of the distribution (mean reward can be obtained by modifying Line 7). The algorithm first initializes an empty set of feedback $\mathscr{C}$, an empty list of candidate explanations $\mathscr{C}_{s_V}$ for each $s_V \in S_V$, best reward R_b randomly drawn from $\mathscr{R}$, and best test score T_b. The test scores consist of two numbers: the first number indicates *how likely* the demonstrations are under the reward model, and the second number indicates *how good* the model is in explaining the reward.

In each iteration, the current best reward model, denoted by R_m, is calculated as the MAP of $\mathscr{P}(R|\mathscr{D}, \mathscr{C})$ (Line 7). The corresponding test score of that model, denoted by T_m is initialized to a 2-D array that consists of the likelihood of data $\mathscr{D}$ under R_m, and zero to indicate that the correctness of the model has not been evaluated (Line 8). For each verification test state s_V, the reward value $R_m(s_V)$ and the corresponding explanation $\mathscr{X}_{R_m}(s_V)$ are shown to the demonstrator for approval (Line 10).

If approved, then $\mathscr{X}_{R_m}(s_V)$ is added to $\mathscr{C}$ (Lines 11-12). If disapproved, then the explanation is added to the candidate explanation set $\mathscr{C}_{s_V}$ and additional feedback from the demonstrator is requested, by selecting two explanations from $\mathscr{C}_{s_V}$ that have not been queried so far (Lines 14-15). The agent can also add additional explanations to $\mathscr{C}_{s_V}$ either by modifying existing ones or sampling additional models from $\mathscr{P}(R|\mathscr{D}, \mathscr{C})$ and generating their corresponding explanations for query selection. The demonstrator can provide no feedback ($\mathscr{C}$ unchanged), generate exact feature attribution, or provide pairwise preferences over explanations from $\mathscr{C}_{s_V}$.

A test score for $\mathscr{X}_{R_m}(s_V), \forall s_V \in S_V$ is added to T_m using the score function (Line 16). The score reflects the similarity of $\mathscr{X}_{R_m}(s_V)$ to human-generated explanations or their preferred explanations. Finally, R_b and T_b are updated based on the score and the algorithm ends by returning the best reward and best score, when all verification tests have been approved, $\epsilon = 0$ (Lines 17-20).

MAP Estimation The maximum a posteriori probability (MAP) is estimated as $\mathscr{P}(R|\mathscr{D}, \mathscr{C}) = \mathscr{P}(\mathscr{D}|R)\mathscr{P}_{\mathscr{C}}(R)$ where $\mathscr{P}_{\mathscr{C}}(R)$ is a prior over R defined by the feedback on explanations. The feedback on explanations is represented as a distribution because semantically it is a set of constraints over the model parameters. This also allows generalization to other IRL algorithms that use the Bayesian framework, as most algorithms differ only in their likelihood function.

5.1. Learning Linear Reward Using Explanations

This section analyzes our proposed method using linear reward models. A linear reward is described by a linear weighted combination over the n-size vector of features describing the state, $R(s) = \mathbf{w}^T\phi(s)$, $\mathbf{w} \in \mathbb{R}^n$. The corresponding explanations generated using GaE and LIME (LM) will produce the same output. Therefore, we only present results using GaE and saliency maps (SM).

For a linear reward, $\text{GaE}(R(s)) = \nabla_{\phi(s)} R_{\mathbf{w}}(s) = \nabla_{\phi(s)} \mathbf{w}^T\phi(s) = \mathbf{w}$, and SM is $|\mathbf{w}|$. Using Definition 5, we now show that feedback on GaE-based and SM-based explanations reduces reward ambiguity. We assume that the agent and the demonstrator share the same similarity measure, and discuss results with cosine similarity.

Proposition 1. *The complexity of removing reward ambiguity with Oracle-generated GaE explanations as feedback, $\mathscr{F}_O(\mathscr{X}(s_V)) = \mathscr{X}_O(s_V), \forall s_V \in S_V$, is $O(1)$.*

Proof Sketch. $\forall s \in S$, the explanation given by the oracle is $GaE(R_{\mathbf{w}^*}(s)) = \mathbf{w}^*$. Thus one oracle-generated GaE explanation is sufficient to reduce $|\mathscr{C}_{GaE}^{\mathscr{X}_O}(S_V)|$ to one. □

Though $\mathscr{X}_O$ is often difficult to obtain, it shows the best-case scenario for REVEALE to eliminate reward ambiguity. The significance of Proposition 1 is that it establishes a direct bridge between the feature attribution methods and reward learning.

Proposition 2. *To reduce reward function ambiguity by x% in expectation, it suffices to have preference feedback over a set of* $k = \log_2(1/(1-x/100))$ *randomly generated GaE explanation pairs.*

Proof Sketch. Consider a set of pairwise preference GaE feedback denoted by $\mathscr{C}^{\mathscr{X}_P}_{GaE}(S_V) = \{(e^1_1 \succ e^2_1), \dots, (e^1_m \succ e^2_m)\}$ where, (e^1_i, e^2_i) are candidate explanations for s_V. By Definition 4, the explanation-consistent reward set, $\Delta(\mathscr{C}^{\mathscr{X}_P}_{GaE}(S_V))$, is described by the half-space constraints:

$$\mathbf{w}^T(\hat{e}^1_i - \hat{e}^2_i) > 0,\ \forall (e^1_i \succ e^2_i) \in \mathscr{C}^{\mathscr{X}_P}_{GaE}(S_V),$$

where $\hat{e}^j_i$ is normalized vector of e^j_i. Each constraint in $\mathscr{C}^{\mathscr{X}_P}_{GaE}(S_V)$ enforces that the cosine similarity of $\mathbf{w}$ with e^1_i should be larger than the cosine similarity of $\mathbf{w}$ with e^2_i. We want to find the bound k on the size of $\mathscr{C}^{\mathscr{X}_P}_{GaE}(S_V)$, $|\mathscr{C}^{\mathscr{X}_P}_{GaE}(S_V)|$, that is sufficient for reducing the size of reward hypotheses by x%. Let $|\mathscr{R}| = J$. According to [27] $|\Delta(\mathscr{C}^{\mathscr{X}_P}_{GaE}(S_V))| = \frac{J}{2^k}$ in expectation where $|\mathscr{C}^{\mathscr{X}_P}_{GaE}(S_V)| = k$. Therefore, $x = (1 - \frac{1}{2^k}) * 100\%$ volume of $\mathscr{R}$ is removed in expectation using feedback over a set of $k = \log_2(1/(1-x/100))$ randomly generated GaE explanation pairs. □

Proposition 3. *It is sufficient to have O(1) Oracle-generated SM Feedback to reduce* $|\Delta(\mathscr{C}^{\mathscr{X}_O}_{SM}(S_V))| \leq 2^n$, $n = \dim(\mathbf{w})$.

Proof Sketch. $\forall s \in S$, the explanation given by the oracle is $GaE(R_{\mathbf{w}^*}(s)) = |\mathbf{w}^*|$. Then we can construct at most 2^n, $\mathbf{w}'s$ such that $|\mathbf{w}| = |\mathbf{w}^*|$ by taking $+, -$ sign combination of each element of $|\mathbf{w}^*|$. Therefore, $|\Delta(\mathscr{C}^{\mathscr{X}_O}_{SM}(S_V))| \leq 2^n$. □

Propositions 1 and 3 show that GaE can be more effective than SM in reducing reward ambiguity. Our empirical results show a similar trend for non-linear rewards.

Prior Definition The prior $\mathscr{P}_{\mathscr{C}}(R)$ for explanation function $\mathscr{X}$ and feedback $\mathscr{F}$ is computed as:

$$\mathscr{P}_{\mathscr{C}}(R) \propto \mathscr{I}(\mathscr{C}^{\mathscr{F}}_{\mathscr{X}}(S_V)), \tag{1}$$

where $\mathscr{I}(.)$ is 1 if R satisfies all the constraints imposed by $\mathscr{C}^{\mathscr{F}}_{\mathscr{X}}(S_V)$ and 0 otherwise. Now, MAP of $\mathscr{P}(R|\mathscr{D}, \mathscr{C}) = \mathscr{P}(\mathscr{D}|R)\mathscr{P}_{\mathscr{C}}(R)$ can be estimated using an off-the-shelf Markov Chain Monte Carlo (MCMC) solver.

Generalization Notice that Equation 1 does not depend on the explanation generation mechanism, i.e. feature attribution. REVEALE can utilize any explanation generation method as long as $\mathscr{C}^{\mathscr{F}}_{\mathscr{X}}(.)$ can be represented for that method. Defining such constraints requires a measurement of similarity $\psi(.)$ between two explanations of similar representation (e.g. cosine distance). Though our discussion of the framework uses the definition of the likelihood function presented in B-REX [11], REVEALE can be used with other Bayesian IRL methods as well by replacing the definition of the likelihood function.

5.2. Deep REVEALE

Using MCMC for estimating MAP of $\mathscr{P}(R|\mathscr{D}, \mathscr{C})$ with $\mathscr{P}_{\mathscr{C}}(R)$ as in Equation 1 is inefficient in problems with large dimensions and feedback constraints because it can take a large number of steps to get a good estimation of the MAP. In addition, explanation feedback cannot be represented as linear constraints when reward functions are represented using neural networks. Hence, we present a method for calculating the priors as soft versions of the constraints discussed earlier. Note that when calculating the priors, the agent uses the previously collected feedback ($\mathscr{C}$) and the explanations generated. For oracle-generated explanations, $\mathscr{X}_O(s_V)$,

$$\mathscr{P}_{\mathscr{C}}(R) \propto \Big(\prod_{\mathscr{X}_O(s_V) \in \mathscr{C}^{\mathscr{X}_O}_{\mathscr{X}}} e^{\psi(\mathscr{X}_R(s_V), \mathscr{X}_O(s_V))} \Big)^{\lambda}, \tag{2}$$

where $\mathscr{X}_R(s_V)$ is the agent's explanation, $\mathscr{X}_O(s_V)$ is the oracle-generated explanation as feedback, $\psi(.)$ is a measurement of similarity, and $\lambda \in [0, \infty)$.

Similarly, for pairwise preferences, $\mathscr{F}_P(\mathscr{X}_1(s_V), \mathscr{X}_2(s_V)) = \mathscr{X}_P(s_V)$,

$$\mathscr{P}_{\mathscr{C}}(R) \propto \Big(\prod_{(\mathbf{x}^1_i \succ \mathbf{x}^2_i) \in \mathscr{C}^{\mathscr{X}_P}_{\mathscr{X}}} \frac{e^{\psi(\mathbf{x}^R_i, \mathbf{x}^1_i)}}{e^{\psi(\mathbf{x}^R_i, \mathbf{x}^1_i)} + e^{\psi(\mathbf{x}^R_i, \mathbf{x}^2_i)}} \Big)^{\lambda} \tag{3}$$

where $\mathbf{x}^R_i = \mathscr{X}_R(s_V)$ is the agent's explanation, $\mathbf{x}^1_i \succ \mathbf{x}^2_i$ denotes the human's preference, and $\lambda \in [0, \infty)$.

For the above priors, gradient-based MCMC optimization methods [31] work well in high dimensions and can be used to optimize Bayesian neural networks. In addition, $\mathscr{P}(R|\mathscr{D}, \mathscr{C})$ can be approximated using a gradient-based method with $-\log \mathscr{P}(R|\mathscr{D}, \mathscr{C})$ as loss functions. This loss function will decompose into two parts, one for the likelihood function and the other for the prior. The parameter λ can be adjusted to optimize these two parts simultaneously. Also notice that when λ is set to zero this becomes standard B-REX [11] and T-REX [6]. Optimizing this loss function requires calculating the gradient of the explanation function $\mathscr{X}$ with respect to state features. This can be automatically calculated through auto-diff libraries such as JAX [32].

6. Experimental Setup

We evaluate the effectiveness of learning aligned linear and non-linear rewards with REVEALE using three explanation generation techniques: gradient as explanations

(GaE), LIME, and saliency map (SM). Reward alignment is measured by (1) the accuracy of predicting the user's trajectory preferences in test environments, and (2) the average reward collected by executing the policy in the test environments trained using the learned reward. The results of our approach are compared with those of the policy with the true reward function (Optimal) and two recent IRL algorithms, B-REX [11] (for linear rewards) and T-REX [6] (for non-linear rewards). Hence, we use *REX* to denote B-REX for domains with linear reward and T-REX for non-linear reward.

We report results on five proof-of-concept domains, including three domains from the AI safety literature. Many of the domains we consider suffering from *spurious feature correlation*, where two or more state features always co-occur in the demonstrations and this correlation affects the learning process. Since such correlations are spurious during test time the agent encounters novel states. We use $\mathscr{P}_{\mathscr{C}}(R)$ as a test score for REVEALE. Verification states $\mathscr{S}_V$ are selected from $\mathscr{D}$. Explanations are randomly selected from $\mathscr{C}_{s_V}$ for querying feedback. Notice that neither the $\mathscr{D}$ nor the $\mathscr{S}_V$ contains novel states that the agent encounter during evaluations. As a result, evaluation performance is a good indicator of the generalizability of the methods. All algorithms were implemented by us and tested on a machine with 32 GB RAM and 12GB GPU. Values are averaged over 60 different random seeds. Experiments with non-linear rewards use a four-layer neural network with Relu activation.

LavaLand This domain introduced by Hadfield-Menell et al. [15] consists of a 'lava' feature that never appears in the demonstrations. As a result, the agent may not learn to avoid it when it navigates to a goal location, potentially resulting in unsafe behavior when deployed.

DogWalk This is the AV domain illustrated in Figure 1, where the AV must learn to stop for both pedestrians and dogs. Each state is represented by ⟨location, human, dog, bag⟩. This environment is an example of spurious feature correlation as humans and dogs occur together in the demonstrations.

WaterWorld This domain, based on [33], tests how the agent responds to a distribution shift. There are two types of surfaces in the problem: 'water' and 'ground'. We consider a linear reward, with a negative reward for stepping into the water. The demonstrations and the training environment have fixed water locations, but the test environments have scattered water locations. Reward ambiguity arises as the agent may not be able to distinguish if the negative reward is associated with the surface type or grid location.

Navigation (AVNav) This domain, designed by us, describes a safe route planning problem, where the demonstrations are preferences over different routes. Each state represents a road segment and is denoted by the tuple ⟨road segment length, average speed, #potholes, mobile network quality, and accident history in the segment⟩. The non-linear reward incentivizes the AV to navigate on safe (low pothole and low accident history) and comfortable (good mobile network) routes while reducing the time to the destination. However, the demonstration data contains spurious feature correlation as most roads that have good mobile networks also have bad accident history, leading to reward ambiguity.

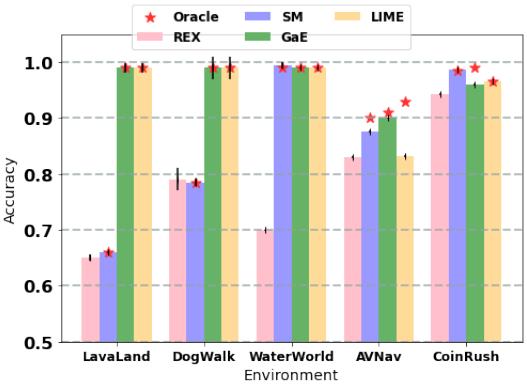

Figure 2: Preference prediction accuracy

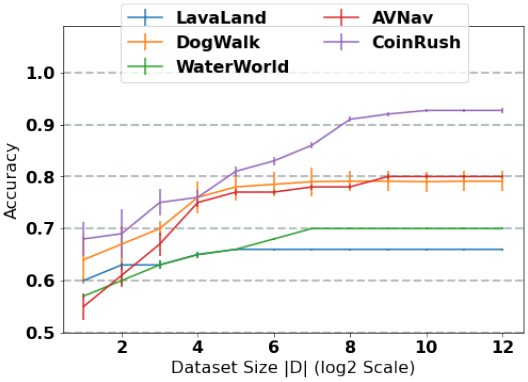

Figure 3: Dataset size vs accuracy for REX

CoinRush This domain is similar to the CoinRun environment described in [31]. The cells in the grid have coins, gold, or an enemy. The target is to gather as much gold and coins while avoiding the enemy. However, in the demonstration data, the enemy and coin/gold always have fixed colors (green and yellow respectively), resulting in spurious feature correlation. But in the test environment, they can have any color.

In WaterWorld and CoindRush, the ambiguity is about which feature should get attribution. In other domains, the ambiguity is about which feature to attribute and whether their attribution should be positive or negative. AVNav and CoinRush have a non-linear reward structure, while the other problems use linear rewards.

Models	LavaLand	DogWalk	WaterWorld	AVNav	CoinRush
REX	-6.50 ± 0.26 [−8.5]	-6.10 ± 0.29 [−24]	-7.10 ± 0.35 [−20]	-29.30 ± 0.23 [−31.5]	33.30 ± 0.73 [14.8]
GaE ($\mathcal{X}_O$)	-2.50 ± 0.00 [−2.5]	-2.60 ± 0.00 [−2.6]	2.00 ± 0.00 [2.0]	-25.97 ± 0.21 [−27.9]	35.00 ± 0.00 [35.0]
GaE ($\mathcal{X}_P$)	-2.50 ± 0.00 [−2.5]	-2.60 ± 0.00 [−2.6]	2.00 ± 0.00 [2.0]	-26.25 ± 0.20 [−27.9]	35.00 ± 0.00 [35.0]
SM ($\mathcal{X}_O$)	-5.50 ± 0.35 [−8.5]	-13.50 ± 0.41 [−18]	2.00 ± 0.00 [2.0]	-25.94 ± 0.21 [−27.9]	35.00 ± 0.00 [35.0]
SM ($\mathcal{X}_P$)	-5.50 ± 0.33 [−8.5]	-13.50 ± 0.39 [−24]	2.00 ± 0.00 [2.0]	-26.63 ± 0.24 [−27.9]	35.00 ± 0.00 [35.0]
LIME ($\mathcal{X}_O$)	-2.50 ± 0.00 [−2.5]	-2.60 ± 0.00 [−2.6]	2.00 ± 0.00 [2.0]	-26.37 ± 0.20 [−27.9]	35.00 ± 0.00 [35.0]
LIME ($\mathcal{X}_P$)	-2.50 ± 0.00 [−2.5]	-2.60 ± 0.00 [−2.6]	2.00 ± 0.00 [2.0]	-29.20 ± 0.22 [−31.5]	32.00 ± 0.93 [14.8]
Optimal	**−2.50**	**−2.60**	**2.00**	**−24.00**	**35.00**

Table 1
Average ± standard error and worst case reward (in brackets) achieved by executing policies with the learned reward models.

6.1. Results and Discussion

Prediction accuracy Figure 2 shows the average prediction accuracy tested on 2000 pairs of trajectories. For training, we use 256 demonstrations and 64 preference feedback over pair of explanations for domains with linear reward and 1024 demonstrations and 256 preference feedback over pair of explanations for non-linear reward. The red star over each bar represents the accuracy of the corresponding explanation method when exact Oracle explanations were given instead of preference feedback.

In every domain, except CoinRush, REVEALE with GaE explanations achieves the highest accuracy and matches the accuracy of prediction based on human-generated explanations. In LavaLand and DogWalk, SM identified that 'lava' and 'dogs' are important features, respectively, but could not identify whether they should be positively attributed because it uses the absolute value of the gradient. B-REX also suffers from this drawback. In domains where the ambiguity is about which feature should be attributed, such as location or surface type in WaterWorld, SM performs comparably to other approaches. However, B-REX often associated the reward with location, instead of surface type. Overall, our results indicate that REVEALE with any explanation method performs better than REX.

In all five environments, all the approaches, including REX, achieve near-optimal prediction accuracy on the demonstration dataset used to learn the reward. The performance degrades in the test scenarios because the agents encounter novel states that did not occur in the demonstration data. As evident from Figure 2, REVEALE improves the prediction performance significantly in such cases. In the absence of prior knowledge about novel situations, it might not be possible to predict how the agent will perform just by assessing the agent's reward/policy/value function in states that appear in the training environment. However, examining the consistency of the agent's explanations in states that appear in training data allows the demonstrator to infer its behavior in many novel situations. This allows the demonstrator to provide valuable feedback, which the agent uses to reduce reward ambiguity in novel situations.

Effect of number of demonstrations We also test the effect of #demonstrations on the prediction accuracy of REX (Figure 3), with the size of $\mathcal{D}$ ranging between 2 to 2048. We observe no improvement in the accuracy of B-REX beyond 128 demonstrations and 1024 for T-REX, which indicates that the approach is unable to eliminate reward ambiguity even if the number of demonstrations increases. This is because the additional trajectories do not encode any information about novel situations the agent may encounter when deployed. Therefore, the performance does not improve in the test cases.

Average and worse case reward Table 1 shows the average and worst reward obtained with different approaches in test environments. We report the worst-case reward since it provides insights into the degree of unsafe behavior that may arise when the reward is not well-aligned. We evaluate the effectiveness of each explanation method using both types of feedback: Oracle-generated $\mathcal{X}_O$ and pairwise preferences $\mathcal{X}_P$. We also report the average reward obtained with the true reward function in each setting, denoted by Optimal. Our results show that REVEALE with GaE using $\mathcal{X}_P$ feedback performs better on most domains. SM outperforms the other approaches only when the ambiguity was about whether the reward should be associated with a feature or location. That is, SM often identifies the magnitude of correlation but struggles to refine whether it is positive or negative, often associating incorrectly. LIME performs similarly to GaE, when feedback is $\mathcal{X}_O$ but performs relatively poorly when feedback is $\mathcal{X}_P$. This is because LIME works with a large set of states in the neighborhood of the input states, unlike GaE and SM, which only work with a single state. Therefore when exact inputs are given, LIME works very well. With $\mathcal{X}_P$ feedback, the error can propagate to many states causing worse performance than GaE. Overall, our results show that REVEALE can learn and generalize reward that is better aligned than the existing approaches.

7. Summary and Future Work

This paper presents a general interpretable reward learning and verification framework to ensure that the learned reward is aligned with that of the demonstrator's intent. The results demonstrate the benefits of our approach in learning the intended reward, thereby supporting the safe deployment of RL agents in the real world. In the future, we aim to develop techniques to automatically identify critical states for verification, and integrate active learning methods [34] to optimize queries.

Acknowledgments

This work was supported in part by the National Science Foundation Grants IIS-1954782, IIS-2205153, NSF and USDA-NIFA award number 2021-67021-35344.

References

[1] S. Zilberstein, Building strong semi-autonomous systems, in: Proceedings of the 29th AAAI Conference on Artificial Intelligence, 2015, pp. 4088–4092.

[2] B. D. Argall, S. Chernova, M. Veloso, B. Browning, A survey of robot learning from demonstration, Robotics and Autonomous Systems 57 (2009) 469–483.

[3] R. S. Sutton, A. G. Barto, Reinforcement Learning: An Introduction, MIT press Cambridge, 1998.

[4] A. Y. Ng, S. J. Russell, Algorithms for inverse reinforcement learning., in: Proceedings of the 17th International Conference on Machine Learning, 2000, pp. 663–670.

[5] B. D. Ziebart, Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy, Carnegie Mellon University, 2010.

[6] D. S. Brown, W. Goo, N. Prabhat, S. Niekum, Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations, in: Proceedings of the 36th International Conference on Machine Learning, 2019, pp. 783–792.

[7] C. Basu, M. Singhal, A. D. Dragan, Learning from richer human guidance: Augmenting comparison-based learning with feature queries, in: Proceedings of the 13th International Conference on Human-Robot Interaction, 2018, pp. 132–140.

[8] D. S. Brown, J. J. Schneider, S. Niekum, Value alignment verification, in: Proceedings of the 38th International Conference on Machine Learning, 2021, pp. 1105–1115.

[9] P. Abbeel, A. Y. Ng, Apprenticeship learning via inverse reinforcement learning, in: Proceedings of the 21st International Conference on Machine learning, 2004.

[10] B. D. Ziebart, A. L. Maas, J. A. Bagnell, A. K. Dey, Maximum entropy inverse reinforcement learning, in: Proceedings of the 23rd AAAI Conference on Artificial Intelligence, 2008, pp. 1433–1438.

[11] D. S. Brown, R. Coleman, R. Srinivasan, S. Niekum, Safe imitation learning via fast Bayesian reward inference from preferences, in: Proceedings of the 37th International Conference on Machine Learning, 2020, pp. 1165–1177.

[12] M. Palan, N. C. Landolfi, G. Shevchuk, D. Sadigh, Learning reward functions by integrating human demonstrations and preferences, in: Proceedings of Robotics: Science and Systems XV, 2019.

[13] D. Ramachandran, E. Amir, Bayesian inverse reinforcement learning, in: Proceedings of the 20th International Joint Conference on Artificial intelligence, 2007, pp. 2586–2591.

[14] D. Bogdoll, S. Guneshka, J. M. Zollner, One ontology to rule them all: Corner case scenarios for autonomous driving, ArXiv abs/2209.00342 (2022).

[15] D. Hadfield-Menell, S. Milli, P. Abbeel, S. J. Russell, A. Dragan, Inverse reward design, in: Advances in Neural Information Processing Systems, 2017.

[16] M. P. Linegang, H. A. Stoner, M. J. Patterson, B. D. Seppelt, J. D. Hoffman, Z. B. Crittendon, J. D. Lee, Human-automation collaboration in dynamic mission planning: A challenge requiring an ecological approach, Proceedings of the Human Factors and Ergonomics Society Annual Meeting 50 (2006) 2482–2486.

[17] B. Hayes, J. A. Shah, Improving robot controller transparency through autonomous policy explanation, in: Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction, 2017, pp. 303–312.

[18] M. T. Ribeiro, S. Singh, C. Guestrin, "Why should I trust you?": Explaining the predictions of any classifier, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 1135–1144.

[19] S. B. Nashed, S. Mahmud, C. V. Goldman, S. Zilberstein, A unifying framework for causal explanation of sequential decision making, ArXiv abs/2205.15462 (2022).

[20] S. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, 2017. arXiv:1705.07874.

[21] J. Tayyub, M. Sarmad, N. Schönborn, Explaining deep neural networks for point clouds using gradient-based visualisations, arXiv preprint arXiv:2207.12984 (2022).

[22] K. Simonyan, A. Vedaldi, A. Zisserman, Deep inside convolutional networks: Visualising image classification models and saliency maps, CoRR abs/1312.6034 (2014).

[23] T. Chakraborti, S. Sreedharan, Y. Zhang, S. Kambhampati, Plan explanations as model reconciliation: Moving beyond explanation as soliloquy, arXiv preprint arXiv:1701.08317 (2017).
[24] O. Amir, F. Doshi-Velez, D. Sarne, Summarizing agent strategies, Autonomous Agents and Multi-Agent Systems 33 (2019) 628–644.
[25] S. H. Huang, K. Bhatia, P. Abbeel, A. D. Dragan, Establishing appropriate trust via critical states, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 2018, pp. 3929–3936.
[26] A. Tabrez, S. Agrawal, B. Hayes, Explanation-based reward coaching to improve human performance via reinforcement learning, in: Proceedings of the 14th ACM/IEEE International Conference on Human-Robot Interaction, 2020, p. 249–257.
[27] D. S. Brown, W. Goo, S. Niekum, Better-than-demonstrator imitation learning via automaticaly-ranked demonstrations, in: Proceedings of the 3rd Annual Conference on Robot Learning, 2019, pp. 330–359.
[28] R. Ramakrishnan, E. Kamar, D. Dey, J. Shah, E. Horvitz, Discovering blind spots in reinforcement learning, in: Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems, 2018, pp. 1017–1025.
[29] C. Molnar, Interpretable Machine Learning: A Guide For Making Black Box Models Explainable, Lulu.com, 2022.
[30] I. Lage, D. Lifschitz, F. Doshi-Velez, O. Amir, Exploring computational user models for agent policy summarization, in: Proceedings of the 28th International Joint Conference on Artificial Intelligence, 2019, pp. 1401–1407.
[31] L. Langosco, J. Koch, L. Sharkey, J. Pfau, D. Krueger, Goal misgeneralization in deep reinforcement learning, in: Proceedings of the 39th International Conference on Machine Learning, 2022, pp. 12004–12019.
[32] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, Q. Zhang, JAX: Composable transformations of python+numpy programs, http://github.com/google/jax, 2018.
[33] J. Leike, M. Martic, V. Krakovna, P. A. Ortega, T. Everitt, A. Lefrancq, L. Orseau, S. Legg, AI safety gridworlds, ArXiv abs/1711.09883 (2017).
[34] B. Settles, Active learning, Synthesis Lectures on Artificial Intelligence and Machine Learning 6 (2012) 1–114.