Machine Learning (2024) 113:1885–1901
https://doi.org/10.1007/s10994-022-06295-5

Learning state importance for preference-based reinforcement learning

Guoxi Zhang[1] · **Hisashi Kashima**[1,2]

Received: 30 May 2022 / Revised: 28 July 2022 / Accepted: 19 September 2022 /
Published online: 9 January 2023

Abstract
Preference-based reinforcement learning (PbRL) develops agents using human preferences. Due to its empirical success, it has prospect of benefiting human-centered applications. Meanwhile, previous work on PbRL overlooks interpretability, which is an indispensable element of ethical artificial intelligence (AI). While prior art for explainable AI offers some machinery, there lacks an approach to select samples to construct explanations. This becomes an issue for PbRL, as transitions relevant to task solving are often outnumbered by irrelevant ones. Thus, ad-hoc sample selection undermines the credibility of explanations. The present study proposes a framework for learning reward functions and state importance from preferences simultaneously. It offers a systematic approach for selecting samples when constructing explanations. Moreover, the present study proposes a perturbation analysis to evaluate the learned state importance quantitatively. Through experiments on discrete and continuous control tasks, the present study demonstrates the proposed framework's efficacy for providing interpretability without sacrificing task performance.

Keywords Interpretable reinforcement learning · Preference-based reinforcement learning · Human-in-the-loop reinforcement learning · Interpretability artificial intelligence

1 Introduction

Preference-based reinforcement learning (PbRL) (Akrour et al., 2011; Fürnkranz et al., 2012) is a reinforcement learning (RL) setting that develops agents using human preferences. Christiano et al. (2017) reported remarkable empirical results for tasks without reward functions–they were solved with less than 1% of interactions annotated. Besides,

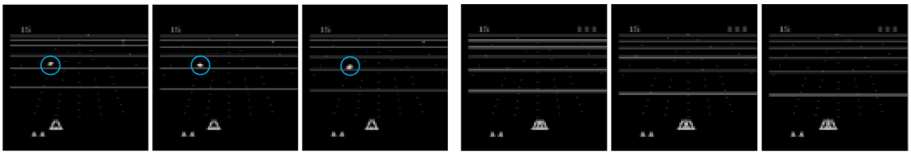
Editors: Yu-Feng Li, Prateek Jain.

✉ Guoxi Zhang
guoxi@ml.ist.i.kyoto-u.ac.jp

Hisashi Kashima
kashima@i.kyoto-u.ac.jp

[1] Graduate School of Informatics, Kyoto University, Yoshida-Honmachi, Kyoto 606-8501, Japan

[2] RIKEN Guardian Robot Project, Kyoto, Japan



(a) Incoming enemies (circled in blue). (b) Flying in open space.

Fig. 1 Examples for critical and non-critical states for the game *BeamRider*. An agent controls a spacecraft (shown at the bottom) in this game and fights with enemies. Case (**a**) is an example of critical states, as it shows an enemy that needs to be destroyed. Case (**b**) is an example of non-critical states, as no action needs to be taken when the agent is flying in open space (Color figure online)

due to its ability to comply with humans, PbRL is promising for human-centered tasks such as value alignment (Fisac et al., 2020) or shared autonomy (Reddy & Dragan, 2018).

The present study argues for interpretability as another first principle for PbRL. As pointed out by Glass et al. (2008), interpretability is crucial for earning users' trust. Without the ability to explain behaviors, an agent will appear to be an inaccessible contrived blackbox instead of something that admits interaction. Interpretability is even mandatory for ethical AI. Since 2017, the European Parliament and European Commission have launched a series of actions emphasizing the role of interpretability in intelligent systems. In April 2021, the European Commission released a new proposal for regulating AI systems that prescribes an obligatory requirement for transparency in high-risk systems. Moreover, interpretability facilitates developing agents. For example, with imperfect reward functions, Pan et al. (2022) observed that agents might learn unintended behaviors. In this case, an interpretable agent will allow practitioners to track down problems agilely.

Notwithstanding its significance, the previous framework for deep PbRL (Christiano et al., 2017; Brown et al., 2019; Shin et al., 2021) overlooks interpretability. It learns a deep reward function from human preferences and uses it in policy optimization. Although it is the core quantity that extracts knowledge from human preferences, the learned reward function remains a blackbox.

At first glance, general techniques for explainable AI (Linardatos et al., 2021), such as saliency maps, suffice for explaining reward functions. However, explanations are often generated using some samples. There lacks a systematic approach for sample selection, which compromises the credibility of explanations. For instance, suppose an agent needs to navigate through a maze. Then it is better to focus on paths leading to the goal position rather than paths leading to dead ends. Yet, a randomly selected sample is likely to be part of the latter. Explanations based on such a sample are thus not representative or meaningful.

The present study overcomes the sample selection issue with a novel reward learning framework, which simultaneously learns a reward function and a weighting network for states. The weighting network learns the importance of states for modeling preferences, which can be used for selecting samples to construct explanations. The key assumption is that in a long state sequence, only a few states are critical to preferences. Figure 1 shows an example for critical and non-critical states for the Atari game *BeamRider*. To be competitive, an agent needs to destroy enemy spacecrafts, so states containing enemy spacecrafts are critical. States representing open space, on the contrary, are not critical.

In specific, the proposed framework represents each trajectory as a weighted sum of representation vectors for states and uses a weighting network to compute the weights. ℓ_1 and ℓ_2 regularizations are imposed to the output of the weighting network to attain the assumed

sparsity. The proposed framework then models trajectory returns as inner products between a reward vector and trajectory representation vectors. Parameters of the entire model are learned by maximizing the likelihood of collected preferences.

A remaining question is how to evaluate the importance of states. In literature, approaches for interpretability are evaluated either with demonstrations or user studies. The former illustrates insights obtained from explanations, while the latter confirms if explanations are comprehensible. However, neither of them can answer whether the explanations are representative in a quantitative aspect. To address this drawback, the present study proposes a perturbation analysis. The core idea is to remove samples that appear to be important from expert demonstrations and train behavioral cloning (BC) agents on the remaining data. Then the performance drop of BC agents serves as an evaluation metric for sample importance.

The present study proceeds with experiments on 17 offline RL datasets. Firstly, a qualitative categorization of important states identified for the game *BeamRider* illustrates how the proposed framework helps to explain reward functions. Interestingly, the results indicate that the learned reward function was (a) uncertain about the exact start of events and (b) unable to discriminate temporally associated states. Both insights shed light on how one can improve PbRL agents. Moreover, the results for the proposed perturbation analysis confirm that, when compared to reward values, the inferred state weights are more indicative of critical states. Finally, there is a concern about the tradeoff between performance and interpretability (Puiutta & Veith, 2020). Our results show no significant performance loss for the proposed framework was observed, which corroborates its feasibility for complex control problems. The contributions of the present study are summarized as follows.

1. To address the interpretability of PbRL and overcome the sample selection issue in particular, the present study proposes a novel framework for PbRL that learns rewards and state importance simultaneously.
2. A perturbation analysis is proposed to evaluate the importance of states quantitatively.
3. With a categorization of identified critical states, the present study confirms the efficacy of the proposed framework for explaining reward functions. It then evaluates the learned state importance using the proposed perturbation analysis. Finally, it verifies the task performance of the proposed framework.

The rest of this paper is organized as follows. Section 2 summarizes prior art for reward learning from human preferences, followed by Sect. 3 that formalizes the reward learning problem in an offline setting. Section 4 describes the proposed framework, and Sect. 5 presents empirical evaluations. Section 6 concludes this paper.

2 Related work

Learning from preference-based feedback has been a topic of reinforcemeng learning since last decade (Akrour et al., 2011; Fürnkranz et al., 2012; Wirth & Fürnkranz, 2013; Busa-Fekete et al., 2014). Wirth et al. (2017) presented a comprehensive survey for early algorithms. Recently, deep PbRL has demonstrated empirical success in Atari games (Christiano et al., 2017; Ibarz et al., 2018), locomotion tasks (Lee et al., 2021a) and navigation tasks (Shin et al., 2021). It is especially useful for human-in-the-loop applications such as value alignment (Fisac et al., 2020) or shared autonomy (Reddy &

Dragan, 2018) that requires adaptation to human feedback. Based on the Bradley-Terry (BT) model (Bradley & Terry, 1952), existing PbRL approaches relate preferences to trajectory returns via either trajectory features (Novoseller et al., 2020; Wirth et al., 2016) or sums of predicted rewards (Christiano et al., 2017; Ibarz et al., 2018).

Several approaches combine PbRL with other ideas. To name a few, Ibarz et al. (2018) proposed to combine PbRL with behavioral cloning. Lee et al. (2021b) suggested diversifying collected experience using intrinsic reward, and Brown et al. (2020) combined PbRL with Bayesian modeling and self-supervised pretraining. Besides, adaptive query selection also receives attention in literature (Sadigh et al., 2017; Biyik & Sadigh, 2018; Wilde et al., 2020; Biyik et al., 2020). However, interpretability receives little attention for PbRL. As the only exception, Bewley and Lecue (2022) proposed to use tree-based functions for reward functions, but this approach does not scale to complex high-dimensional tasks. In a separate line of work, Icarte et al. (2018) proposed to learn finite-state machines for reward functions, which was later extended to noisy reward setting (Corazza et al., 2022) and vision-based control (Camacho et al., 2021). Nevertheless, such a representation is not directly applicable to preference-based setting.

In general, one may interpret reward functions using similar techniques for interpreting policies, such as saliency maps (Atrey et al., 2020). Nevertheless, the issue of sample selection remains. Chances are that explanations are not representative of the agent's knowledge, if samples used by the explanations are selected arbitrarily. Moreover, the evaluation methods also do not cover sample importance aspect. In literature, approaches for interpretability are either illustrated with few examples (Bewley & Lecue, 2022; Beyret et al., 2019; Greydanus et al., 2018) or evaluated with user study (Madumal et al., 2020; Sequeira & Gervasio, 2020). Neither of them assesses the importance of samples used for explanations. The present study overcomes the two issues mentioned above with a novel reward learning framework and a perturbation analysis.

3 Problem setting

3.1 Markov decision process without reward

The present study models a task as a Markov decision process without reward (MDP/R) $\langle \mathcal{S}, \mathcal{A}, \mathrm{P}, \gamma \rangle$. $\mathcal{S}$ and $\mathcal{A}$ are the sets of states and actions, respectively. With slight abuse of terminology, throughout this paper states and observations are used interchangeably. P is the distribution of next state after taking some action at a state, and $\gamma \in (0, 1)$ is a discount factor.

MDP/R prescribes the following interaction protocol. As step t, the agent observes current state $s_t \in \mathcal{S}$ and selects a_t according to a policy $\pi : \mathcal{S} \to \mathcal{A}$. Based on (s_t, a_t), the environment decides next state s_{t+1}. Such interactions generate alternating sequences of states and actions $\tau = (s_1, a_1, s_2, a_2, \dots)$, which are called trajectories.

Unlike a Markov decision process, there is no reward function in MDP/R. The present study considers learning a reward function from human preference as follows, which enables one to apply off-policy learning algorithms.

3.2 The reward learning problem

A reward function $R : \mathcal{S} \to \mathbb{R}$ provides an agent with feedback for its decisions. Following previous works for PbRL (Ibarz et al., 2018; Brown et al., 2019), actions are omitted in R. This is a valid assumption for many tasks whose ground truth reward is also a function of states. For other tasks, extension for including actions is straightforward. With R, an agent can learn policies via maximizing the expected cumulative reward in a trajectory (i.e. return) $\mathbb{E}[\sum_{t=1}^{\infty} \gamma^{t-1} r_t]$.

An agent is provided with a collections of N trajectories $\{\tau_j\}_{j=1}^{N}$ and a set of M preferences Y. A preference sample $y^{(i)} \in Y$ can be written as $y^{(i)} = (\tau_1^{(i)}, \tau_2^{(i)}, c^{(i)})$, where $\tau_1^{(i)}$ and $\tau_2^{(i)}$ are two trajectories being compared, and $c^{(i)}$ is the preference label. $c^{(i)} = 1$ if $\tau_1^{(i)}$ is preferred over $\tau_2^{(i)}$, and $c^{(i)} = 0$ otherwise. In human-in-the-loop applications, preferences can be generated by human annotators. One may present annotators with pairs of trajectories and ask them to select the ones that are more favorable.

This setting differs from the existing PbRL setting (Christiano et al., 2017) in that the agent is not allowed to collect new trajectories or preferences. As a form of offline RL (Lange et al., 2012), it eliminates the need for online data acquisition. Moreover, this setting enables one to collect preferences at scale before reward learning. In the existing setting, the agent generates trajectories during policy learning, which means annotators have to be alongside the agent and provide preferences once trajectories are generated. On the contrary, in our setting all the preferences can be generated prior to reward learning using techniques such as crowdsourcing. This separation of preference collection and reward learning lowers the requirement for annotators and makes PbRL more accessible.

Meanwhile, in the online PbRL setting reward functions are re-trained once new preferences are obtained. The agent is in fact repeatedly solving the learning problem considered here with a growing set of preferences. Thus, algorithms for the problem considered here also applies to the online setting.

The reward learning problem can be summarized as:

- Input: N trajectories $\{\tau_j\}_{j=1}^{N}$ and M preferences $Y = \{y^{(i)}\}_{i=1}^{M}$;
- Output: A reward function R.

4 Proposed approach

This section introduces how the proposed framework learns a reward function and a weighting network for states from the provided preferences. After that, it explains how the proposed analysis evaluates the importance of states.

4.1 Modeling preferences

Given $y^{(i)}$, the proposed framework first encodes the states of both trajectories as vectors in $\mathbb{R}^d$ with $f_e : \mathcal{S} \to \mathbb{R}^d$, where d is the dimension of this vector space. For tasks with image input, f_e can be parameterized with convolutional neural networks followed by fully-connected layers; for continuous input f_e may consist of several fully-connected layers. A weighting network $f_w : \mathbb{R}^d \to \mathbb{R}$ takes as input state representations and

outputs the weights of states. f_w is parameterized with fully-connected layers. Using f_e and f_w, a trajectory τ is encoded as $f_c(\tau)$ as:

$$f_c(\tau) = \sum_{s \in \tau} f_w(f_e(s)) f_e(s). \tag{1}$$

The return of τ, $G(\tau)$, is the inner product between a vector $\theta_R \in \mathbb{R}^d$ and $f_c(\tau)$:

$$G(\tau) = \theta_R^{\mathsf{T}} f_c(\tau). \tag{2}$$

The larger $|\, f_w(f_e(s)) \,|$ is, the more important state s is for $f_c(\tau)$ and $G(\tau)$. Similarly, the reward of a state s is the inner product between θ_R and the representation for s, multiplied by state importance: $R(s) = \theta_R^{\mathsf{T}} f_w(f_e(s)) f_e(s)$.

The proposed framework utilizes the BT model for modeling preferences. Specifically, the label in $y^{(j)}$ takes value one with probability:

$$\Pr(c^{(i)} = 1; \tau_1^{(i)}, \tau_2^{(i)}) = \frac{\exp(G(\tau_1^{(i)}))}{\exp(G(\tau_1^{(i)})) + \exp(G(\tau_2^{(i)}))}. \tag{3}$$

In other words, the larger $G(\tau_1^{(i)}) - G(\tau_2^{(i)})$ is, the higher probability that $\tau_1^{(i)}$ is preferred.

4.2 Learning rewards and weights

In the proposed framework, f_e, f_w and θ_R are learned by minimizing the following objective function:

$$L = L_{\text{ce}} + \lambda_1 L_1 + \lambda_2 L_2, \tag{4}$$

where $\lambda_1 \in \mathbb{R}$ and $\lambda_2 \in \mathbb{R}$ are hyper-parameters. L_{ce} is the cross entropy loss:

$$\begin{aligned} L_{ce} = -\frac{1}{M} \sum_{(\tau_1^{(i)}, \tau_2^{(i)}, c^{(i)}) \in Y} \Big[& c^{(i)} \log(\Pr(c^{(i)} = 1; \tau_1^{(i)}, \tau_2^{(i)})) \\ & + (1 - c^{(i)}) \log(\Pr(c^{(i)} = 0; \tau_1^{(i)}, \tau_2^{(i)})) \Big]. \end{aligned} \tag{5}$$

By minimizing L_{ce}, the proposed framework can learn a reward function that best explains the provided preferences. L_1 and L_2 are introduced to realize the assumption made for state importance–only few states are critical for preferences. L_1 penalizes the absolute values of state weights:

$$L_1 = \frac{1}{M} \sum_{(\tau_1^{(i)}, \tau_2^{(i)}, c^{(i)}) \in Y} \sum_{s \in \tau_1^{(i)}, \tau_2^{(i)}} |\, f_w(f_e(s)) \,| \,. \tag{6}$$

In consequence, only states that are relevant to preferences have weights deviate from zero. Irrelevant states will have weights close to zero. Meanwhile, L_1 treats each state independently. In many tasks of interests, critical states span multiple time steps. For example, in the example shown in Fig. 1a, it takes several time steps for the enemy spacecraft to approach the agent's ship. L_2 is proposed to capture this temporal structure.

$$L_2 = \frac{1}{M} \sum_{(\tau_1^{(i)}, \tau_2^{(i)}, c^{(i)}) \in Y} \sum_{s_t, s_{t+1} \in \tau_1^{(i)}, \tau_2^{(i)}} (f_w(f_e(s_t)) - f_w(f_e(s_{t+1})))^2 \tag{7}$$

This term penalizes the difference between weights of adjacent states. With both L_1 and L_2, only the states whose adjacent states are also critical have non-zero weights.

Preliminary experiments showed that when trained from scratch, optimizing Eq. 4 resulted in almost zero weights for most states. A possible reason is that the initial output of f_w is within a small range around zero since network parameters are initialized with random numbers close to zero. In this case, imposing L_1 and L_2 further restricts the output of f_w. The proposed framework overcomes this issue with a warming procedure for λ_1 and λ_2. When optimizing Eq. 4, the values of λ_1 and λ_2 are linearly increased to their final values. Initially, the values of λ_1 and λ_2 are small, so the reward network has full capacity for learning from preferences. As their values increase, the reward network is forced to prune away the contribution of some states in trajectory returns while maintaining modeling preferences. As a result, f_w learns to assign weights deviated from zero to critical states for preferences.

As a remark, the l_2-norm is not the only choice in Eq. 7. One may instead minimize the l_1-norm or the huber loss. In practice, for a specific application, the best choice depends on the structure of the ground-truth reward and should be chosen empirically. For illustration purpose, the present study uses the l_2-norm in Eq. 7.

4.3 Evaluating state weights

A remaining question is how to evaluate the state importance learned from preferences. As mentioned above, neither demonstrations nor user studies can answer if explanations are representative. To comprehensively evaluate the learned state importance, the present study proposes a new perturbation study for evaluating state importance.

First of all, it is worth discussing the notion of state importance. One candidate is the frequency of states. For two states $s, s' \in \mathcal{S}$, s is more important than s' if s appears more often in training trajectories. In consequence, explanations based on s better characterize patterns in data than those generated using s'. This perspective, however, lacks consideration for task performance. Often, RL agents are supposed to reach some particular states (e.g. the goal in a maze) or visit them as often as possible (e.g. destroying more enemy spacecrafts in *BeamRider*). It is these particular states that are critical to task performance, though they might be outnumbered by the non-critical ones. Explanations based on such states will reveal how a reward function facilitates the agent to solve the corresponding task.

However, collecting such critical states requires expertise for the corresponding task and massive annotation. Both requirements are infeasible in practice. As a surrogate, the present study considers a state of being important if it is indispensable for expert demonstrations. Then, removing such states should cause performance drop for behavioral-cloning agents. By measuring the performance drop, one can evaluate state importance from the perspective of task solving.

In specific, let $h : \mathcal{S} \times \mathcal{A} \to [0, +\infty)$ be an arbitrary function that maps states and actions to non-negative values. The higher $h(s, a)$ is, the more important this state-action pair is. The proposed analysis rolls out into several rounds. Firstly, rank state-action pairs in decreasing order according to $h(s, a)$. Then at each round, remove the top-k state-action pairs, and train a BC agent on the remaining. To prevent creating artificial transitions, after

removing a pair (s_t, a_t), set its preceding pair (s_{t-1}, a_{t-1}) as a terminating pair. The present study reports result for removing up to 40% of data, 10% each round.

As for alternative methods, the present study reports results for random removal and absolute-reward (AR) removal. With random removal, state-action pairs are removed uniformly at random. Its results illustrate to what extent training data size affects agents' performance. With AR removal, state-action pairs are removed according to the absolute value of rewards inferred from preferences.

With the same amount of data removed, the poorer BC agents perform, the better a method is in characterizing state importance. Moreover, the fewer data are removed for the same performance drop, the better the method is.

4.3.1 Remarks

- As a BC agent learns a mapping from states to actions in data, using it eliminates the effect of reward functions on task performance. Besides, this choice avoids potential influences from data removal on temporal structures.
- While the proposed reward learning algorithm applies to general trajectories, this perturbation analysis is limited to expert demonstrations. This is because it utilizes the drop in performance as a measure for state importance. Meanwhile, general trajectories, especially imperfect ones, contain "wrong" decisions. Removing these decisions might improve performance in regardless of state importance. In consequence, these "wrong" decisions complicates the discussion of the results, as one will not be able to attributes changes in performance to importance of removed states.

5 Experiments

5.1 Overview

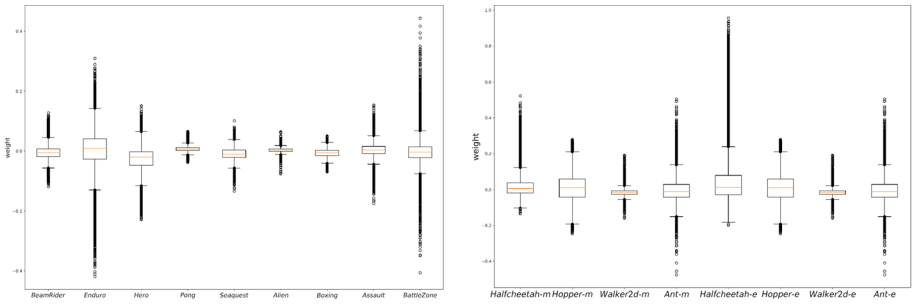
Experiments proceed in three steps. Firstly, this section reports findings and insights from a case study on *BeamRider*, which illustrates how the proposed framework helps to explain reward functions. Then, it uses the proposed analysis to evaluate whether the inferred weights correlated with the importance of states quantitatively, which complements the first evaluation. Finally, this section evaluates the task performance of the proposed framework, demonstrating its practical applicability.

5.2 Datasets

The present study utilizes the "medium" and "expert" version of trajectories collected for four continuous control tasks in the D4RL (Fu et al., 2020) datasets: *Hopper*, *Walker2d*, *Halfcheetah* and *Ant*. For each task, the "medium" version (labeled with "-m") contains trajectories collected by an online RL agent, and the "expert" version (labeled with "-e") contains expert demonstrations.

In addition, this study utilizes the DQN replay dataset (Agarwal et al., 2020). The data for nine games are included: *BeamRider, Enduro, Hero, Pong, Seaquest, Alien, Boxing, Assault* and *BattleZone*.

For each task (game), 5000 preference queries are generated as follows. First of all, 250 trajectories are sampled uniformly at random. Following the practice proposed by Ibarz



(a) State weights on Atari datasets. (b) State weights on D4RL datasets.

Fig. 2 Boxplots for state weights. The median, first quantile, and third quantile are in $[-0.1, 0.1]$ on every dataset, indicating many of the weights are very small. This confirms the effect of the L_1 term in Eq. 4 for selecting states

et al. (2018) and Brown et al. (2019), queries for preferences are made up with subsequences of trajectories. From each trajectory three subsequences of length 60 are sampled, yielding a pool of 750 subsequences. Then 5000 queries are randomly selected from them. Three labels are generated using the Bradley-Terry model and ground-truth rewards for each query.

5.3 Experiment design

5.3.1 Visualizing identified states

This part of the experiments presents an analysis for state weights. Figure 2 illustrates the distribution of states weights of the 750 training subsequences and 750 new subsequences.

Figures 3, 4 and 5 present in-depth qualitative analysis of state weights of 60 subsequences. These subsequences are sampled from the first ten trajectories for *BeamRider*. Half of them are used in reward learning, and the other half are unseen ones. In particular, Figs. 4 and 5 present a categorization of critical states identified from the same 60 subsequences. From each subsequence, the state with the largest $|f_w(f_e(s))|$ is visualized, together with the immediately preceding state and subsequent state. Then the authors inspected these visualizations and summarized seven success cases (Fig. 4) and three failure cases (Fig. 5). As will be clear later, these states form example-based explanations themselves, and they can be used to construct other types of explanations if necessary.

5.3.2 Task performance

This part of experiments compares task performance of the proposed framework with other reward learning methods. It utilizes the average test return of the same policy learning algorithm on inferred rewards as evaluation metric. For continuous control tasks, it utilizes the behavioral-regularized actor-critic algorithm (Wu et al., 2019), and for Atari games, it uses the quantile-regression DQN algorithm (Dabney et al., 2018).

The first reward learning method to compare is the one proposed by Christiano et al. (2017), which uses the sum of rewards as the return of a trajectory. As it is a direct utilization of the BT model for preferences, it is referred to as BT in the sequel. The second

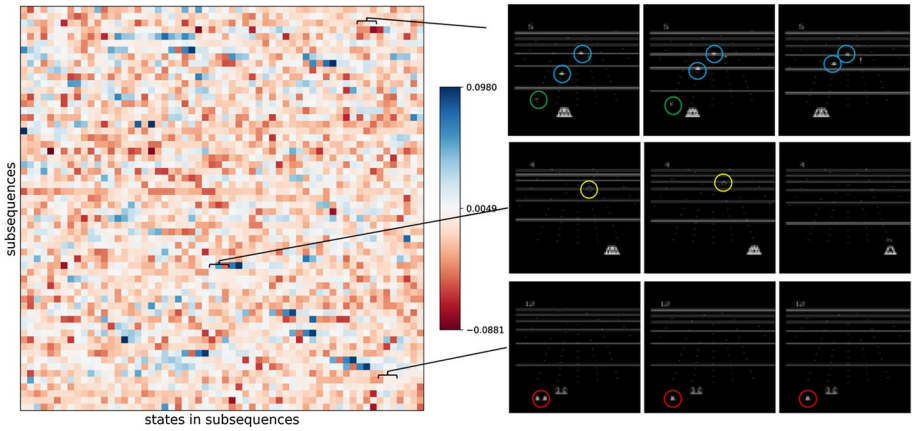


Fig. 3 Left: A heatmap for state weights on *BeamRider*, which confirms the effect of L_2. Upper right: an example for states with large absolute weights, in which the agent is close to an incoming missile (circled in green) in the presence of enemies (circled in blue). Its weights are the three cells starting from the 3rd row from the top and the 10th column from the right. Middle right: an example for states with weights close to zero, in which the agent launch missiles (circled in yellow) in open space. Its weights are the three cells starting from the 21st row from the bottom and 32nd row from the right. Bottom right: an example for transitioning from states with large absolute weights to states with small absolute weights, in which the agent lose a life (circled in red). Its weights are the three cells starting from 5th column from the right and the 5th column from the bottom (Color figure online)

method is T-REX (Brown et al., 2019) which utilizes the ranking over entire trajectories in reward learning. As the goal is to analyze performance for reward learning, approaches that combine PbRL with BC or intrinsic rewards are not included. Note that it is straightforward to combine the proposed framework with these ideas.

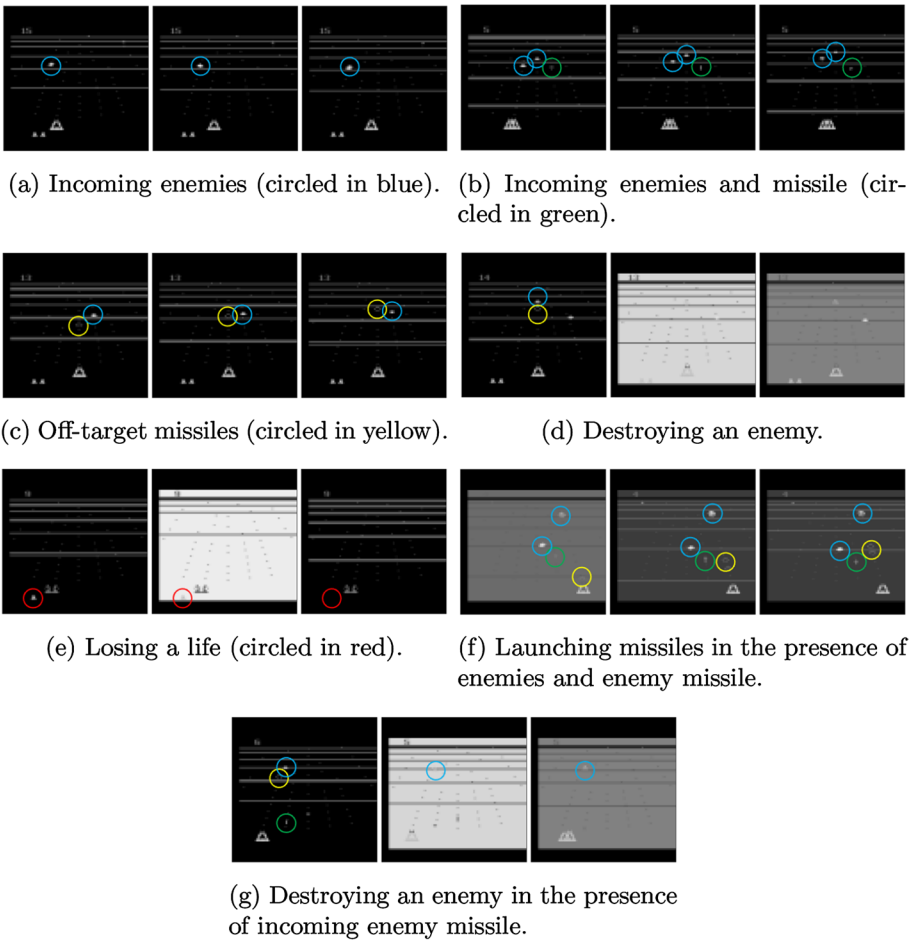
5.4 Technical details

For Atari games, the input to agents contains game scores. To prevent exploiting such information, game scores are masked out during reward learning and reward inference.

λ_1 and λ_2 control the level of sparsity. They are selected using grid search over $\{0.1, 0.01, 0.001\}$ and 10% of preferences as validation sets. After that, reward functions are trained with the selected hyperparameters and all of the preferences.

Experiments are repeated five times, and the average values of metrics and standard errors are reported. Details for hyper-parameters, including implementations, are available on our website.[1]

[1] https://altriaex.github.io/identify-critical-states/.



(a) Incoming enemies (circled in blue).

(b) Incoming enemies and missile (circled in green).

(c) Off-target missiles (circled in yellow).

(d) Destroying an enemy.

(e) Losing a life (circled in red).

(f) Launching missiles in the presence of enemies and enemy missile.

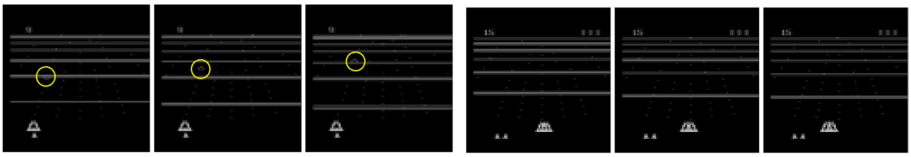
(g) Destroying an enemy in the presence of incoming enemy missile.

Fig. 4 Examples of identified critical states for *BeamRider*. Cases (**a**)–(**d**) are taken from different trajectories. Together, they depict critical behaviors required for this game: the agent needs to destroy enemies with its missiles. Case (**e**) covers the situation of losing a life, which is also critical to this game. Case (**f**) and case (**g**) cover complex situations where the agent needs to launch missiles while dodging enemy missiles (Color figure online)

5.5 Results

5.5.1 Visualizing identified states

Figure 2 shows box plots for state weight on all of the 17 datasets. The first and third quantile of state weights on all datasets are covered in $[-0.1, 0.1]$. This means that the weights of many states are very small, which confirms the effect of L_1. Figure 3 shows a heatmap for state weights and visualizations of states for *BeamRider*. Each row in the heatmap corresponds to a trajectory subsequence, and each cell in a row corresponds to the weight of a state. The left part of Fig. 3 shows that many consecutive states have similar weights, which confirms the effect of L_2 in smoothing state weights. The right part of Fig. 3 shows



(a) Launching missiles in open space. (b) Flying in open space.

(c) Screen flickering.

Fig. 5 Examples of identified states that are not critical. In case (**a**) and (**b**) the agent is in open space. Its behaviors are not critical to the game. Case (**c**) covers a case where the screen is flickering, and it constitutes 8.3% of samples in qualitative analysis. This error case might be explained by its resemblance with case (**d**) and (**e**) in Fig. 4

examples of consecutive states. The top three form an example whose weights deviate from zero, in which the agent is close to an incoming missile in the presence of enemies. This is indeed critical for this game, as the agent loses a life once hit by the missile. The middle part illustrates an example for states with weights close to zero. In this case the agent has launched a missile in open space, which is an irrelevant behavior for the game. The bottom part illustrates a transition from a state with large absolute weight to a state with small absolute weight. This example corresponds to the process of losing a life.

Figures 2 and 3 demonstrate that the proposed framework can select some states that are indeed critical. To further illustrate how the proposed framework facilitate interpreting reward functions, the present study reports a categorization of 60 identified states in Figures 4 and 5. All of the visualizations, as well as the related videos, can be found on our website.

Figure 4 shows the seven cases that are indeed critical. In case (a) there are incoming enemies. This case constitutes 11.6% of visualizations. Case (b) contains 3.3% of visualizations in which both incoming enemies and enemy missiles are present. Meanwhile, in case (c), the agent has launched missiles in the presence of enemies, and in case (d), an enemy is destroyed. They constitute 16.7% and 11.6% of the visualizations, respectively. Case (e) is the largest category (25%) in which the agent's ship is destroyed. Case (f) and case (g) are two more complex cases in which the agent has launched missile (f) and destroyed an enemy (g) in the presence of enemies and enemy missiles. Both of them constitute 3.3% of the visualizations. From simple situations to complex ones, these 74.8% of visualizations present a variety of states that are critical to the corresponding task. These states serve as example-based explanations that confirm the ability of the proposed framework for identifying critical states.

Meanwhile, Fig. 5 shows three failure cases. In these cases, the states have large absolute weights, but they are not important to the task from the authors' point of view. In case (a) (5%), the agent has launched a missile in open space. In case (b) (11.7%), the agent's ship is flying in open space. Both of them are not critical. Case (c) contains 8.3% of visualizations from which only screen flickering is observed.

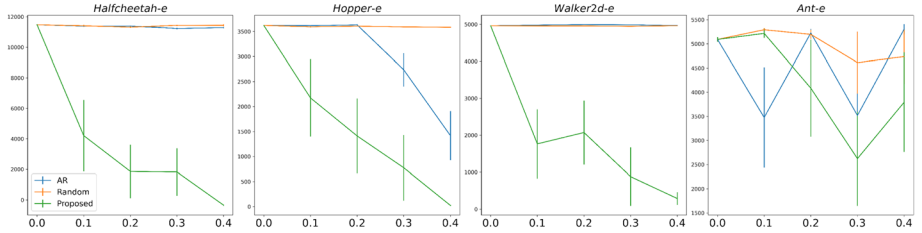


Fig. 6 Results for the perturbation analysis. The x-axis is the ratio of data removed using some method. The y-axis is the test return of BC agents. Lower test return indicates that the removed data are critical to the corresponding task, which means the corresponding method characterizes state importance better. Random removal has a negligible effect on agent performance. When compared to random removal, AR is effective only on *Halfcheetah* and *Walker2d*. Meanwhile, removing states using the proposed framework significantly affects the performance on all of the datasets (Color figure online)

Together, Figs. 4 and 5 show that critical states identified by the proposed method serve as sample-based explanations for reward functions. The following insights can be obtained by examining these samples, which shed light on how reward learning can be improved.

- The starts of events are not modeled accurately. For example, in case (e) of Fig. 4 the remaining lives decreased by one. It is more desirable that the reward function assigns high weights to the state in which enemy missiles just hit the agent.
- Temporal associations are not handled correctly. As shown in case (d) and (e) of Fig. 4, screen flickering is associated with the destruction of the agent's ship or enemies. This explains why case (c) in Fig. 5 is considered as important.

5.5.2 Evaluating state weights

Figure 6 shows results for the proposed perturbation analysis on the four continuous control tasks. As the Atari replay dataset does not provide expert demonstrations, results for Atari games are not included. A point (x, y) in these figures means that BC agents' test return after removing the x ratio of data is y, and the error bars are standard errors.

Initially, consider the performance of random removal (shown in orange). For *Halfcheetah, Walker2d* and *Hopper*, even with only 60% of expert demonstrations, the BC agents do not have performance loss. For *Ant*, removing 40% of data causes about 10% of performance drop. These results can be dissected from two aspects. The fact that BC agents remain competitive even with 60% of data eliminates the possibility of contributing performance drop to reduced training data sizes. Moreover, these results confirm that constructing explanations with random samples can be misleading, as the sampled states are likely to be irrelevant.

Now consider results for AR, which is shown as blue curves. For *Halfcheetah* and *Walker2d*, removing data according AR has little effect. It becomes effective on *Hopper* only after 20% of data are removed. Recall that AR uses the absolute values of inferred rewards as a score for state importance. These results mean that states with either large positive or negative rewards are not indispensable for the corresponding tasks. As our analysis only uses expert demonstrations, a possible explanation is that such states represent task accomplishment and are not part of decision sequences that lead to task accomplishment. Hence, constructing

Table 1 Test return of PbRL agents on 17 datasets. The proposed framework outperforms BT on nine datasets. Among the rest eight datasets, its performance gap against BT is small except for *Seaquest*, *Hopper-m*, *Walker2d-e* and *Ant-m*. Numbers shown in bold indicate the corresponding method outperforms others

Task	Proposed	BT	T-REX
BeamRider	4438.18 ± 796.92	**4681.45 ± 949.61**	305.00 ± 128.74
Enduro	**915.13 ± 167.41**	628.44 ± 135.80	56.94 ± 31.72
Hero	**4397.80 ± 2046.37**	1043.79 ± 543.10	770.22 ± 421.79
Pong	**−19.86 ± 0.71**	−19.89 ± 1.03	− 20.88 ± 0.04
Seaquest	107.78 ± 49.38	**155.82 ± 33.57**	51.32 ± 26.33
Alien	164.14 ± 67.37	166.06 ± 33.11	**317.13 ± 77.25**
Boxing	**21.06 ± 13.77**	− 9.08 ± 4.51	− 8.25 ± 2.67
Assault	**133.34 ± 57.32**	84.10 ± 44.49	126.91 ± 89.24
BattleZone	**4727.37 ± 1733.43**	4175.15 ± 859.19	4194.85 ± 1655.48
Hopper-m	1589.10 ± 126.70	1729.52 ± 17.42	**1731.06 ± 40.98**
Hopper-e	**3604.00 ± 8.27**	3515.32 ± 38.06	3577.56 ± 25.57
Walker2d-m	3404.37 ± 54.85	**3426.19 ± 59.82**	3270.97 ± 41.99
Walker2d-e	4963.82 ± 6.12	**4971.17 ± 7.71**	3977.50 ± 892.78
Ant-m	3110.78 ± 115.75	3287.12 ± 111.55	**3441.29 ± 169.01**
Ant-e	**5208.21 ± 103.82**	4951.52 ± 172.17	4941.51 ± 155.29
Halfcheetah-m	**5374.33 ± 29.16**	5123.57 ± 17.28	5247.13 ± 11.15
Halfcheetah-e	11299.84 ± 148.04	11252.16 ± 176.39	**11492.95 ± 21.43**

explanations using such states is not likely to provide meaningful information on how agents solve tasks.

Meanwhile, removing samples using the proposed framework (shown in green) is effective for all four tasks. For *Halfcheetah, Walker2d* and *Hopper*, the performance of BC agents drops significantly even when only 10% of data are removed. This means that the proposed framework can accurately identify the critical states for these tasks.

5.5.3 Task performance

A remaining question is how an RL agent performs when trained on rewards learned with the proposed framework. Table 1 answers this question. On these 17 datasets, the proposed framework outperforms BT on *Enduro, Hero, Pong, Boxing, Assault, BattleZone, Hopper-e, Ant-e* and *Halfcheetah-m*. Among the rest eight datasets, it has small performance gaps against BT except for *Seaquest, Hopper-m, Walker2d-e* and *Ant-m*. These results indicate that the proposed framework might perform worse than BT when the ground-truth reward is dense, which means the sparsity assumption might need revision. Nevertheless, these results mean that the proposed framework does not incur performance loss on a wide range of tasks while providing interpretability.

6 Conclusion

While PbRL demonstrates good empirical performance with a limited amount of annotation, little attention has been paid to the interpretability of learned reward functions. In particular, due to the immense size of training data, explanations based on ad-hoc samples might not represent the whole data distribution. The present study addresses the interpretability of PbRL and the sample selection issue in particular. A framework is proposed to jointly learn a reward function and a weighting network for states from preferences, which can be used to select samples for explanation purposes. Structural sparsity is imposed on the output of the weighting network to identify states that are highly relevant to preferences. On *BeamRider*, an Atari game, a categorization of 60 identified states illustrates how the proposed framework help to interpret reward functions. Moreover, previous works on explainable RL evaluate methods with demonstrations on a few examples or user studies. Neither of them is capable of illustrating whether the explanations are representative. The present study proposes a new perturbation analysis to evaluate such an aspect quantitatively. Our results show that the weights inferred by the proposed framework are more effective as an indicator than reward values for state importance. Last but not least, through experiments on 17 datasets, the present study also shows that no significant performance loss is observed for the proposed framework on most datasets, which demonstrates its applicability.

As for future direction, it is of interest to combine the proposed framework with other forms of human supervision in training RL agents. Improving its performance on tasks with dense rewards is also worth investigating.

Author Contributions All the authors contributed to the study conception. The design and analysis were by Guoxi Zhang. The first draft of the manuscript was written by Guoxi Zhang, and all the authors commented on previous versions of the manuscript. All the authors read and approved the final manuscript.

Funding Hisashi Kashima is supported by the JST CREST (Grant No.: JPMJCR21D1).

Data availability This work utilizes data published by previous work, and the sources are clearly cited.

Code availability This work utilizes implementations released by existing research, and the sources are clearly cited. The implementation of the proposed method is released.

Declarations

Conflict of interest Not Applicable.

Ethics approval Not Applicable.

Consent to participate Not Applicable.

Consent for publication Not Applicable.

References

Agarwal, R., Schuurmans, D., & Norouzi, M. (2020). An optimistic perspective on offline reinforcement learning. In *Proceedings of the thirty-seventh international conference on machine learning*. PMLR, pp. 104–114.

Akrour, R., Schoenauer, M., & Sebag, M. (2011). Preference-based policy learning. In *Machine learning and knowledge discovery in databases.* Berlin, Heidelberg, Athens, Greece, pp. 12–27.

Atrey, A., Clary, K., & Jensen, D. (2020). Exploratory not explanatory: Counterfactual analysis of saliency maps for deep reinforcement learning. In *Proceedings of the international conference on learning representations*. Virtual.

Bewley, T., & Lecue, F. (2022). Interpretable preference-based reinforcement learning with tree-structured reward functions. In *Proceedings of the twenty-first international conference on autonomous agents and multiagent systems*. International Foundation for Autonomous Agents and Multiagent System, Virtual, pp. 118–126.

Beyret, B., Shafti, A., & Faisal, A. A. (2019). Dot-to-dot: Explainable hierarchical reinforcement learning for robotic manipulation. In *2019 IEEE/RSJ international conference on intelligent robots and systems.* IEEE, Macau, China, pp. 5014–5019.

Biyik, E., & Sadigh, D. (2018). Batch active preference-based learning of reward functions. In *Proceedings of the second conference on robot learning*. PMLR, Auckland, New Zealand, pp. 519–528.

Biyik, E., Huynh, N., Kochenderfer, M. J., et al. (2020). Active preference-based gaussian process regression for reward learning. In *Robotics: Science and Systems XVI*, Virtual.

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

Brown, D., Goo, W., Nagarajan, P., et al. (2019). Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In *Proceedings of the thirty-six international conference on machine learning.* PMLR, Long Beach, CA, USA, pp. 783–792.

Brown, D., Coleman, R., Srinivasan, R., et al. (2020). Safe imitation learning via fast Bayesian reward inference from preferences. In *Proceedings of the thirty-seventh international conference on machine learning*. PMLR, Virtual, pp. 1165–1177.

Busa-Fekete, R., Szörényi, B., Weng, P., et al. (2014). Preference-based reinforcement learning: Evolutionary direct policy search using a preference-based racing algorithm. *Machine Learning, 97*(3), 327–351.

Camacho, A., Varley, J., Zeng, A., et al. (2021). Reward machines for vision-based robotic manipulation. In *Proceedings of the 2021 IEEE international conference on robotics and automation*, Xi'an, China, pp. 14284–14290.

Christiano, P. F., Leike, J., Brown, T., et al. (2017). Deep reinforcement learning from human preferences. In *Advances in neural information processing systems 30*. Curran Associates, Inc., Long Beach, CA, USA, pp. 4302–4310.

Corazza, J., Gavran, I., & Neider, D. (2022). Reinforcement learning with stochastic reward machines. In *Proceedings of the thirty-sixth AAAI conference on artificial intelligence.* AAAI Press, Virtual, pp. 6429–6436.

Dabney, W., Rowland, M., Bellemare, M. G., et al. (2018). Distributional reinforcement learning with quantile regression. In *Proceedings of the thirty-second AAAI conference on artificial intelligence.* AAAI Press, New Orleans, Louisiana, USA, pp. 2892–2901.

Fisac, J. F., Gates, M. A., Hamrick, J. B., et al. (2020). Pragmatic-pedagogic value alignment. In *Proceedings of the eighteenth international symposium on robotics research*. Springer International Publishing, Puerto Varas, Chile, pp 49–57.

Fu, J., Kumar, A., & Nachum, O., et al. (2020). D4rl: Datasets for deep data-driven reinforcement learning. arXiv:2004.07219.

Fürnkranz, J., Hüllermeier, E., Cheng, W., et al. (2012). Preference-based reinforcement learning: A formal framework and a policy iteration algorithm. *Machine Learning, 89*(1), 123–156.

Glass, A., McGuinness, D. L., & Wolverton, M. (2008). Toward establishing trust in adaptive agents. In *Proceedings of the thirteenth international conference on intelligent user interfaces.* Association for Computing Machinery, Gran Canaria, Spain, pp. 227–236.

Greydanus, S., Koul, A., Dodge, J., et al. (2018). Visualizing and understanding Atari agents. In *Proceedings of the thirty-fifth international conference on machine learning*. PMLR, Stockholm, Sweden, pp. 1792–1801.

Ibarz, B., Leike, J., Pohlen, T., et al. (2018). Reward learning from human preferences and demonstrations in Atari. In *Advances in neural information processing systems*. Curran Associates Inc., Montréal, Canada, pp. 8022–8034.

Lange, S., Gabel, T., & Riedmiller, M. (2012). Batch reinforcement learning. In *Reinforcement learning: State-of-the-art*. Springer, Berlin, Heidelberg, Germany, pp. 45–73.

Lee, K., Smith, L., Dragan, A., et al. (2021a). B-pref: Benchmarking preference-based reinforcement learning. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks, Virtual.*

Lee, K., Smith, L. M., Abbeel, P. (2021b). Pebble: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training. In *Proceedings of the thirty-eighth international conference on machine learning*. PMLR, Virtual, pp. 6152–6163.

Linardatos, P., Papastefanopoulos, V., & Kotsiantis, S. (2021). Explainable AI: A review of machine learning interpretability methods. *Entropy, 23*(1), 18.

Madumal, P., Miller, T., Sonenberg, L., et al. (2020). Explainable reinforcement learning through a causal lens. In *Proceedings of the thirty-fourth AAAI conference on artificial intelligence*. AAAI Press, New York, NY, USA, pp. 2493–2500.

Novoseller, E., Wei, Y., Sui, Y., et al. (2020). Dueling posterior sampling for preference-based reinforcement learning. In *Proceedings of the thirty-sixth conference on uncertainty in artificial intelligence*. PMLR, Virtual, pp. 1029–1038.

Pan, A., Bhatia, K., & Steinhardt, J. (2022). The effects of reward misspecification: Mapping and mitigating misaligned models. In *Proceedings of the International conference on learning representations.* Virtual.

Puiutta, E., Veith, E. M., et al. (2020). Explainable reinforcement learning: A survey. In *Machine learning and knowledge extraction*. Springer International Publishing, Dublin, Ireland, pp. 77–95.

Reddy, S., Dragan, A. D., & Levine, S. (2018). Shared autonomy via deep reinforcement learning. In *Proceedings of the robotics: Science and systems XIV*. Pittsburgh, PA, USA.

Sadigh, D., Dragan, A. D., Sastry, S., et al. (2017). Active preference-based learning of reward functions. In *Proceedings of robotics: Science and systems XIII*. Cambridge, MA, USA.

Sequeira, P., & Gervasio, M. (2020). Interestingness elements for explainable reinforcement learning: Understanding agents' capabilities and limitations. *Artificial Intelligence, 288*(103), 367.

Shin, D., & Brown, D. (2021). Offline preference-based apprenticeship learning. In *Workshop on human-AI collaboration in sequential decision-making at the thirty-eighth international conference on machine learning*. Virtual.

Icarte, R. T., Klassen, T., Valenzano, R., et al. (2018). Using reward machines for high-level task specification and decomposition in reinforcement learning. In *Proceedings of the thirty-fifth international conference on machine learning*. PMLR, Stockholm, Sweden, pp. 2112–2121.

Wilde, N., Kulic, D., & Smith, S. L. (2020). Active preference learning using maximum regret. In *IEEE/RSJ international conference on intelligent robots and systems*. IEEE, Las Vegas, NV, USA, pp. 10952–10959.

Wirth, C., & Fürnkranz, J. (2013). A policy iteration algorithm for learning from preference-based feedback. In *Advances in intelligent data analysis XII*. Springer Berlin Heidelberg, London, UK, pp. 427–437.

Wirth, C., Fürnkranz, J., Neumann, G. (2016). Model-free preference-based reinforcement learning. In *Proceedings of the thirtieth AAAI conference on artificial intelligence*. AAAI Press, Phoenix, AZ, USA, pp. 2222–2228.

Wirth, C., Akrour, R., Neumann, G., et al. (2017). A survey of preference-based reinforcement learning methods. *Journal of Machine Learning Research, 18*(136), 1–46.

Wu, Y., Tucker, G., & Nachum, O. (2019). Behavior regularized offline reinforcement learning. arXiv preprint arXiv:1911.11361.