REINFORCEMENT LEARNING FROM HUMAN RATINGS

by

DEVIN WHITE, B.S.

THESIS
Presented to the Graduate Faculty of
The University of Texas at San Antonio
In Partial Fulfillment
Of the Requirements
For the Degree of

MASTERS OF SCIENCE IN ARTIFICIAL INTELLIGENCE

COMMITTEE MEMBERS:
Yongcan Cao, Ph.D., Chair
Yufang Jin, Ph.D.
Dhireesha Kudithipudi, Ph.D.

THE UNIVERSITY OF TEXAS AT SAN ANTONIO
Klesse College of Engineering and Integrated Design
Department of Electical and Computer Engineering
December 2023

DEDICATION

I would like to dedicate this thesis to my mom and dad.

ACKNOWLEDGEMENTS

I would like to say a special thank you to Dr. Yongcan Cao, your guidance, help and support has shaped my career and future in so many ways and for that I am grateful. I would like to thank Dr. Dhireesha Kudithipudi and Dr. Yufang Jin for serving on my committee. I appreciate your support and insights. I would like to say a special thank you to Dr. Ellen Novoseller, Dr. Nicholas Waytowich and Dr. Vernon Lawhern for their incredible help and guidance throughout my graduate degree. Thank you to Dr. Don Peterson, I really appreciate all of your advice and help. I would like to give a huge thank you to all of my lab mates. Dr. Feng Tao, you helped immensely in my academic endeavors. I would also like to thank Mingkang Wu, Daniel Barron, Van Ngo, Gabriella Forbis, Peilang Li, Jeremy Cofield, Victor Osorio, Praveen Ranjan, Umer Siddique, and Mason Conkel for their all of their support. I would like to thank Aaron Fanous for his friendship and support. Most of all I would like to thank my parents for their immense support in my academic career and personal endeavors.

December 2023

REINFORCEMENT LEARNING FROM HUMAN RATINGS

Devin White, M.S.
The University of Texas at San Antonio, 2023

Supervising Professor: Yongcan Cao, Ph.D.

Reinforcement learning (RL), an important subject in artificial intelligence, focuses on learning control policies to tackle various tasks from Atari Games to autonomous driving and large language models. To derive control policies via RL, it is often required that reward functions are designed beforehand. Because reward functions, a map from states or state-action pairs to quantitative reward values, reflect both the task objectives and the underlying environments, it is often difficult and costly to design these reward functions. Instead of designing reward functions, an alternative approach is to learn reward functions from human guidance. In this thesis, we focus on developing a new rating-based reinforcement learning approach that uses human ratings to learn reward functions. Different from the existing human guidance methods, namely, preference-based and ranking-based reward learning paradigms which are based on human relative preferences over sample pairs, the proposed rating-based reinforcement learning approach is based on human evaluation of individual trajectories without relative comparisons between sample pairs. The rating-based reinforcement learning approach builds on a new prediction model for human ratings and a novel multi-class loss function to learn the reward functions, which are then used to learn RL-based control policies. We conduct experimental studies based on both synthetic ratings and real human ratings to evaluate the effectiveness and benefits of the new rating-based reinforcement learning approach.

TABLE OF CONTENTS

Acknowledgements . . . . **iii**

Abstract . . . . **iv**

List of Tables . . . . **vii**

List of Figures . . . . **viii**

Chapter 1: Introduction **1**

1.1 Background 1

1.2 Related Work 3

1.2.1 Inverse Reinforcement Learning (IRL) 3

1.2.2 Preference-based Reinforcement Learning (PbRL) 3

1.2.3 Combining Demonstrations and Preferences 4

1.2.4 TAMER-Style Feedback 4

Chapter 2: Learning from Human Ratings **5**

2.1 Problem Formulation 5

2.2 Main Approach 6

2.2.1 Modeling Reward and Return 6

2.2.2 Novel Rating-Based Cross-Entropy Loss Function 6

2.2.3 Modeling Human Rating Probabilities 7

2.2.4 Modeling Boundaries between Rating Categories 8

Chapter 3: Experiments Based on Synthetic Data **10**

3.1 Setup 10

3.2 Results 11

Chapter 4: Experiments Based on Real Human Data 12

4.1 Setup 12

4.2 rl-teacher 12

4.2.1 RbRL with Different Numbers of Rating Classes 12

4.2.2 Human User Study 13

4.2.3 Performance 14

4.2.4 User Questionnaire Results 17

4.3 BPref 18

4.3.1 Performance 19

4.4 Learning from Human Preferences 19

4.4.1 Human Time 20

4.4.2 Agent Performance 20

4.4.3 Different k Values 21

Chapter 5: Conclusion and Future Work 23

Appendix A: Questionnaire 24

A.1 Questionnaire 24

Bibliography 28

Vita

LIST OF TABLES

4.1 Physical meaning of rating classes for $n = 2, 3, \ldots, 8$ used in our human experiments. 13

4.2 Time it takes for a user to provide 1000 labels for different n compared to preferences 20

4.3 Mean reward over 20 runs for 1000 queries and 10 million timesteps of training. * denotes runs that have memory/implementation issues with respect to hardware. 21

4.4 Mean reward over 20 runs for different k values when $n = 2$. * denotes runs that have memory/implementation issues with respect to hardware. 22

LIST OF FIGURES

2.1 RbRL's modeled class probabilities $Q_\sigma(i)$. This figure illustrates Equation (2.3), evaluated with $n = 5$ classes, $k = 30$, and the rating category separation parameters hard-coded at $\{0, 0.2, 0.4, 0.6, 0.8, 1\}$. We can see that the probability of belonging to a class i is maximized when the normalized return $\tilde{R}(\sigma)$ is halfway between $\bar{R}_i$ and $\bar{R}_{i+1}$, and decreases as $\tilde{R}(\sigma)$ moves further from this point. 8

3.1 Performance of RbRL in synthetic experiments for different n, compared to PbRL: mean reward $\pm$ standard error over 10 experiment runs for Walker (left) and Quadruped (right). 11

4.1 RbRL performance for different n in a human experiment: performance in the Cheetah environment (mean $\pm$ standard error over 3 experiment runs). 13

4.2 Performance of RbRL in the human user study: Cheetah (left) and Swimmer (right). For non-expert users, the plots show mean $\pm$ standard error over 7 users. The expert results are each over a single experiment run. 15

4.3 RbRL performance for the top 3 (non-expert) user study participants: mean reward $\pm$ standard error over the 3 experiment runs each for Cheetah (left) and Swimmer (right). 16

4.4 Results of individual runs in the human user study for the Cheetah (Top) and Swimmer environments (Bottom) for PbRL (left) and RbRL (right) (7 users in each case). 16

4.5 Participants' responses to questions for RbRL and PbRL. The set of survey questions is detailed in the Appendix. The blue bar indicates the median and the edges depict the 1st quartile (left) and 3rd quartile (right). 17

4.6 Number of queries provided in 30 minutes across participants in our human user study (mean $\pm$ standard error). 18

4.7 Performance of RbRL in Human experiments for different n, compared to PbRL: for a single run for Walker (left) and Quadruped (right). 19

CHAPTER 1: INTRODUCTION

**This chapter has previously been published in the Many Facets of Preference-Based Learning Workshop at the Fortieth International Conference on Machine Learning (ICML). It was co-authored by Mingkang Wu, Dr. Ellen Novoseller, Dr. Vernon Lawhern, Dr. Nick Waytowich, and Dr. Yongcan Cao and has been reproduced with minor revisions.*

1.1 Background

With the development of deep neural network theory and the improvement of computation hardware, deep reinforcement learning (RL) has become capable of handling complex tasks with large state and/or action spaces (*e.g.*, Go and Atari games) and yielding human-level or better-than-human-level performance [27, 34]. Numerous approaches, such as DQN [27], DDPG [25], TRPO [28], PPO [33], and SAC [17] have been developed to address challenges such as stability, exploration, and convergence for potential adoption in applications [23] such as robotic control, autonomous driving, natural language processing, and gaming. Despite the important and fundamental advances behind these algorithms, one key obstacle for the wide application of deep RL is the requirement for task-specific prior knowledge. In other words, the learning process in deep RL requires knowledge of a reward function, which is often unavailable in practical applications. Hence, expert knowledge is required to design the reward function [29].

Although in some domains, human experts could design reward functions by hand, the associated cost is high because human experts need to understand the relationship between the mission objective and state-action values and may need to spend extensive time adjusting reward parameters and trade-offs not to encounter adverse behaviors such as reward hacking [1]. Another approach is to utilize qualitative human inputs *indirectly* to learn a reward function, such that humans guide reward function design *without* directly handcrafting the reward. Existing work on learning rewards for RL includes inverse reinforcement learning (IRL) [15, 30, 42], which infers rewards from human demonstrations, and preference-based reinforcement learning (PbRL) [12, 38], in which human users provide relative pairwise preferences over trajectories for reward function inference. Several methods also combine demonstrations and relative preferences, e.g. learning from pairwise preference over demonstrations [7, 8].

Existing human-guided reward learning approaches have demonstrated effective performance in various tasks. However, they suffer from some key limitations. For example, IRL requires expert demonstrations and hence, cannot be directly applied to tasks that are difficult for humans to demonstrate. PbRL is a practical approach to learning rewards for RL, since it is straightforward for humans to provide accurate relative preference information. Yet, RL from pairwise preferences suffers from some key disadvantages. First, each pairwise preference provides only a single bit of information, which can result in sample inefficiency and require significant human time to gather sufficient data for training a well-performing policy. In addition, due to their binary nature, standard preference queries do not indicate how much better or worse one sample is than another. Furthermore, because preference queries are relative, they cannot directly provide a global view of each sample's absolute quality (good vs. bad); for instance, if all choices shown to the user are of poor quality, the user cannot say, "A is better than B, but they're both bad!". Thus, a PbRL algorithm may be more easily trapped in a local optimum, and cannot know to what extent its performance approaches the user's goal. Finally, PbRL methods often require strict preferences, such that comparisons between similar-quality or incomparable trajectories cannot be used in reward learning. While some works use weak preference queries [4, 5], in which the user can state that two choices are equally preferable, there is no way to specify the quality (good vs. bad) of such trajectories; thus, valuable information remains untapped.

The main contribution which will be discussed in this Thesis is the design of a new rating-based reinforcement learning (RbRL) approach. This approach learns a reward function via a multi-class representation of a users ratings.

RbRL differs from IRL and PbRL in that it leverages human ratings on individual samples, whereas IRL uses demonstrations and PbRL uses relative pairwise comparisons. In each query, RbRL displays one trajectory to a human and requests the human to provide a discrete rating.

The number of rating classes can be as low as two, e.g. "bad" and "good", and can be as high as desired. It is worth mentioning that ratings, a well-received concept by humans, provide more informative absolute evaluations of samples. In contrast, the pairwise comparisons used in PbRL can only provide a relative comparison between sample pairs. Hence, ratings provide rich evaluative information for reward function learning.

1.2 Related Work

1.2.1 Inverse Reinforcement Learning (IRL)

IRL is the study of problems and algorithms related to reward learning given demonstrations [2]. The main idea of IRL is to infer reward functions from demonstrations such that the learned reward functions generate control policies and samples that, via reinforcement learning, are similar to the demonstrations. Numerous IRL methods [10, 11, 15, 22, 30, 39, 42], such as maximum entropy IRL, Bayesian IRL, nonlinear IRL, and nonparametric Bayesian IRL, have been developed to infer reward functions. The need for demonstrations often makes these IRL methods costly since human experts are needed to provide demonstrations.

1.2.2 Preference-based Reinforcement Learning (PbRL)

Instead of requiring human demonstrations, PbRL [12, 18, 21, 24, 31, 38, 40, 41] leverages human pairwise preferences over trajectory pairs to learn reward functions. Querying humans for pairwise preferences rather than demonstrations can save dramatic human time. In addition, since human preference can be learned via, *e.g.*, adversarial neural networks [41], more human time can be saved by learning a good model to predict human preference. Another benefit of PbRL is that humans can provide preferences with respect to uncertainty to promote exploration [24]. Despite these benefits, PbRL can be ineffective, especially for complex environments, because pairwise preferences only provide relative information, while they cannot directly evaluate sample quality, unless sampled pairs are selected carefully to infer the global information. In practice, even if one sample is preferred over another, it does not mean that this sample is good. People can also have difficulty when comparing similar samples, thus taking more time and potentially yielding inaccurate preference labels. Notably, several works have sought to improve PbRL sample efficiency; for instance, PEBBLE [21] considers off-policy PbRL and SURF [31] explores data augmentations in PbRL. These contributions are orthogonal to ours, as they could straightforwardly be applied within our proposed RbRL framework.

1.2.3 Combining Demonstrations and Preferences

Several works infer reward functions via rankings over a pool of demonstrations [6–8], and have extrapolated better-than-demonstrator performance from the learned rewards. Notably, these methods do not require all demonstrations to be strictly ranked; rather, the reward learning process only requires access to a set of pairwise preferences over demonstrations. Meanwhile, some works have considered other methods of combining pairwise preferences and demonstrations, for instance first learning from demonstrations and then fine-tuning with preferences [3, 18].

1.2.4 TAMER-Style Feedback

In the TAMER framework [9, 19, 36], a person gives positive (encouraging) and negative (discouraging) feedback to an agent. This differs from our ratings-based learning setting, as the human gives feedback to specific states and actions, rather than to entire trajectories, and because the human decides when to provide each feedback instance, rather than the algorithm displaying queries to the human. Furthermore, TAMER-style methods generally take actions greedily with respect to the learned reward, which may not yield an optimal policy in continuous control settings such as we consider.

CHAPTER 2: LEARNING FROM HUMAN RATINGS

**This chapter has previously been published in the Many Facets of Preference-Based Learning Workshop at the Fortieth International Conference on Machine Learning (ICML). It was co-authored by Mingkang Wu, Dr. Ellen Novoseller, Dr. Vernon Lawhern, Dr. Nick Waytowich, and Dr. Yongcan Cao and has been reproduced with minor revisions.*

2.1 Problem Formulation

We consider a Markov Decision process without reward (MDP $\setminus$ R) augmented with ratings, which is a tuple of the form $(\mathcal{S}, \mathcal{A}, T, \rho, \gamma, n)$. Here, $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of possible actions, $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is a state transition probability function specifying the probability $p(s' \mid s, a)$ of reaching state $s' \in \mathcal{S}$ after taking action a in state s, $\rho : \mathcal{S} \to [0, 1]$ specifies the initial state distribution, γ is a discount factor, and n is the number of rating classes. The learning agent interacts with the environment through rollout trajectories, where a length-k trajectory segment takes the form $(s_1, a_1, s_2, a_2, \ldots, s_k, a_k)$. A *policy* π is a function that maps states to actions, such that $\pi(a \mid s)$ is the probability of taking action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$.

In traditional RL, the environment would receive a reward signal $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, mapping state-action pairs to a numerical reward, such that at time-step t, the algorithm receives a reward $r_t = r(s_t, a_t)$, where (s_t, a_t) is the state-action pair at time t. Accordingly, the standard RL problem can be formulated as a search for the optimal policy π^*, where $\pi^* = \arg\max_\pi \sum_{t=0}^{\infty} \mathbb{E}_{(s_t,a_t)\sim\rho_\pi} \Big[\gamma^t r(s_t, a_t)\Big]$, $a_t \sim \pi(\cdot|s_t)$, and ρ_π is a marginal state-action distribution given a policy π. Note that standard RL assumes the availability of the reward function r. When such a reward function is unavailable, standard RL and its variants may not be used to derive control policies. Instead, we assume that the user can assign any given trajectory segment $\tau = (s_1, a_1, \ldots, s_k, a_k)$ a rating in the set $\{0, 1, \ldots, n - 1\}$ indicating the quality of that segment, where 0 is the lowest possible rating, while $n - 1$ is the highest possible rating.

The algorithm presents a series of trajectory segments σ to the human and receives corresponding human ratings. Let $X := \{(\sigma_i, c_i)\}_{i=1}^{l}$ be the dataset of observed human ratings, where $c_i \in \{0, \ldots, n-1\}$ is the rating class assigned to segment σ_i, and l is the number of rated segments contained in X at the given point during learning.

Note that descriptive labels can also be given to the rating classes. For example, for $n = 4$ rating classes, we can call the rating class 0 “very bad”, the rating class 1 “bad”, the rating class 2 “good”, and the rating class 3 “very good”. With $n = 3$ rating classes, we can call the rating class 0 “ bad”, the rating class 1 “neutral”, and class 2 “good”.

2.2 Main Approach

2.2.1 Modeling Reward and Return

Our approach learns a reward model $\hat{r} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ that predicts state-action rewards $\hat{r}(s, a)$. We further define $\hat{R}(\sigma) := \sum_{t=1}^{k} \gamma^t \hat{r}(s_t, a_t)$ as the cumulative discounted reward, or the *return*, of length-k trajectory segment σ. Larger $\hat{R}(\sigma)$ corresponds to a higher predicted human rating for segment σ. Next, we define $\tilde{R}(\sigma)$ as a function mapping a trajectory segment σ to an estimated total discounted reward, normalized to fall in the interval $[0, 1]$ based on the dataset of rated trajectory segments X:

$$\tilde{R}(\sigma) = \frac{\hat{R}(\sigma) - \min_{\sigma' \in X} \hat{R}(\sigma')}{\max_{\sigma' \in X} \hat{R}(\sigma') - \min_{\sigma' \in X} \hat{R}(\sigma')}. \tag{2.1}$$

2.2.2 Novel Rating-Based Cross-Entropy Loss Function

To construct a new (cross-entropy) loss function that can take multi-class human ratings as the input, we need to estimate the human’s rating class predictions. We here propose a new multi-class cross-entropy loss given by:

$$L(\hat{r}) = -\sum_{\sigma \in X} \left(\sum_{i=0}^{n-1} \mu_\sigma(i) \log \left(Q_\sigma(i) \right) \right), \tag{2.2}$$

where X is the collected dataset of user-labeled segments, $\mu_\sigma(i)$ is an indicator that equals 1 when the user assigns rating i to trajectory segment σ, and $Q_\sigma(i) \in [0, 1]$ is the estimated probability that the human assigns the segment σ to the ith rating class. Next, we will model the probabilities $Q_\sigma(i)$ of the human choosing each rating class. Notably, we do this *without* comparing the segment σ to other segments.

2.2.3 Modeling Human Rating Probabilities

We next describe our model for $Q_\sigma(i)$ based on the normalized predicted returns $\tilde{R}(\sigma)$. To model the probability that σ belongs to a particular class, we will first model separations between the rating classes in reward space.

We define rating class boundaries $\bar{R}_0, \bar{R}_1, \ldots, \bar{R}_n$ in the space of normalized trajectory returns such that $0 := \bar{R}_0 \leq \bar{R}_1 \leq \ldots \leq \bar{R}_n := 1$. Then, if a segment σ has normalized predicted return $\tilde{R}(\sigma)$ such that $\bar{R}_i \leq \tilde{R}(\sigma) \leq \bar{R}_{i+1}$, we wish to model that σ belongs to rating class i with the highest probability.

For example, when the total number of rating classes is $n = 4$, we aim to model the lower and upper return bounds for rating classes $0, 1, 2$, and 3, which could respectively correspond to "very bad", "bad", "good", and "very good". In this case, if $0 \leq \tilde{R}(\sigma) < \bar{R}_1$, then we would like our model to predict that σ most likely belongs to class 0 ("very bad"), while if $\bar{R}_2 \leq \tilde{R}(\sigma) < \bar{R}_3$, then our model should predict that σ most likely belongs to class 2 ("good").

We propose to model $Q_\sigma(i)$ given the normalized predicted returns $\tilde{R}(\sigma)$ and rating category separations $\bar{R}_i$:

$$Q_\sigma(i) = \frac{e^{-k(\tilde{R}(\sigma)-\bar{R}_i)(\tilde{R}(\sigma)-\bar{R}_{i+1})}}{\sum_{j=0}^{n-1} e^{-k(\tilde{R}(\sigma)-\bar{R}_j)(\tilde{R}(\sigma)-\bar{R}_{j+1})}}, \tag{2.3}$$

where k is a hyperparameter modeling the noisiness in the human labels, and the denominator ensures that $\sum_{i=0}^{n-1} Q_\sigma(i) = 1$, i.e. that the class probabilities sum to 1.

To gain intuition for Equation (2.3), note that when $\tilde{R}(\sigma) \in (\bar{R}_i, \bar{R}_{i+1})$, such that the predicted return falls within rating class i's predicted boundaries, then $-(\tilde{R}(\sigma) - \bar{R}_i)(\tilde{R}(\sigma) - \bar{R}_{i+1}) \geq 0$ while $-(\tilde{R}(\sigma) - \bar{R}_j)(\tilde{R}(\sigma) - \bar{R}_{j+1}) \leq 0$ for all $j \neq i$. This means that $Q_\sigma(i) \geq Q_\sigma(j),\ j \neq i$, so that the model assigns category i the highest class probability, as desired. Furthermore, we note that $Q_\sigma(i)$ is maximized when $\tilde{R}(\sigma) = \frac{1}{2}(\bar{R}_i + \bar{R}_{i+1})$, such that the predicted return falls directly in the center of category i's predicted range. As $\tilde{R}(\sigma)$ becomes increasingly further from $\frac{1}{2}(\bar{R}_i + \bar{R}_{i+1})$, the modeled probability $Q_\sigma(i)$ of class i monotonically decreases. These probability trends are illustrated in Figure 2.1. We next show how to compute the class boundaries $\bar{R}_i,\ i = 0, \ldots, n$.

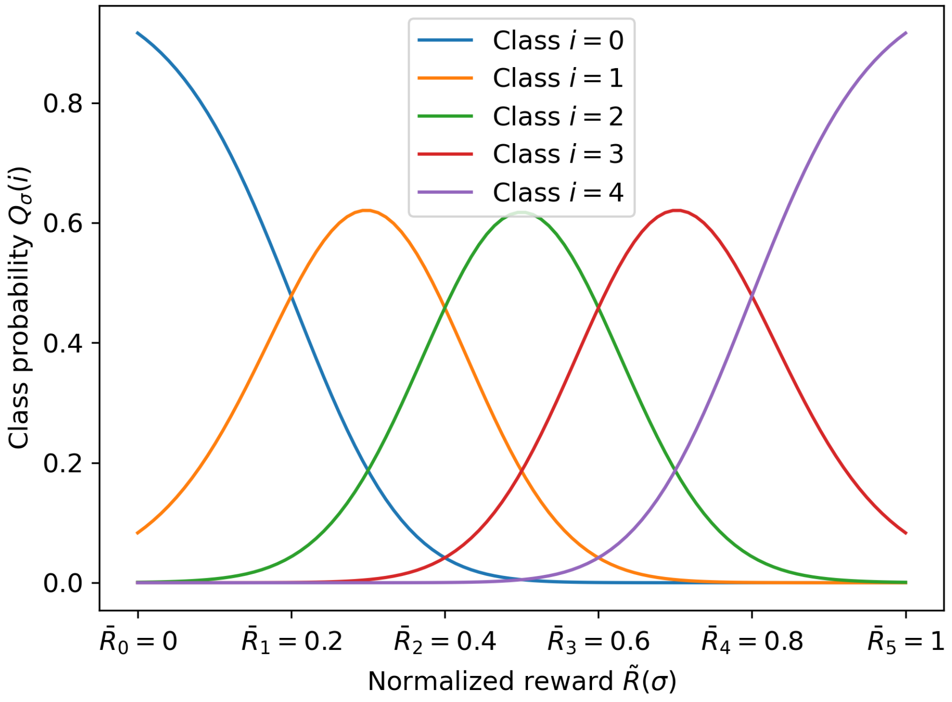


Figure 2.1: RbRL's modeled class probabilities $Q_\sigma(i)$. This figure illustrates Equation (2.3), evaluated with $n = 5$ classes, $k = 30$, and the rating category separation parameters hard-coded at $\{0, 0.2, 0.4, 0.6, 0.8, 1\}$. We can see that the probability of belonging to a class i is maximized when the normalized return $\tilde{R}(\sigma)$ is halfway between $\bar{R}_i$ and $\bar{R}_{i+1}$, and decreases as $\tilde{R}(\sigma)$ moves further from this point.

2.2.4 Modeling Boundaries between Rating Categories

Next, we discuss how to model the boundaries between rating categories, $0 =: \bar{R}_0 \leq \bar{R}_1 \leq \ldots \leq \bar{R}_n := 1$. We determine these boundary values based on the distribution of $\tilde{R}(\sigma)$ for the trajectory segments $\sigma \in X$ and the number of observed samples in X from each rating class. We select the $\bar{R}_i$ values such that the number of training data samples that the model assigns to each modeled rating class matches the number of samples in X that the human assigned to that rating class. Note that this does not require the predicted ratings based on $\tilde{R}(\sigma)$ to match the human ratings for σ in the training data X, but ensures that the proportions of segments assigned to each rating class matches that in the human dataset X. This matching in rating class proportions is desirable for learning an appropriate reward function based on human preference, since different humans could give ratings in significantly different proportions depending on their preferences and latent reward functions, as modeled by $\hat{R}$.

To define each $\bar{R}_i$ so that the number of samples in each modeled rating category reflects the numbers of ratings in the human data, we first sort the estimated returns $\tilde{R}(\sigma)$ for all $\sigma \in X$ from lowest to highest, and label these sorted estimates as $\tilde{R}_1 \leq \tilde{R}_2 \leq \cdots \leq \tilde{R}_l$, where l is the cardinality of X. Denoting via k_j the number of segments that the human assigned to rating class $j,\ j \in \{0, \cdots, n-1\}$, we can then model each category boundary $\bar{R}_i, i \notin \{0, n\}$ (since $\bar{R}_0 = 0$ and $\bar{R}_n = 1$ by definition), as follows:

$$\bar{R}_i = \frac{\tilde{R}_{k_i^{\text{cum}}} + \tilde{R}_{1+k_i^{\text{cum}}}}{2}, \quad i \in \{1, 2, \ldots, n-1\}, \tag{2.4}$$

where $k_i^{\text{cum}} := \sum_{j=0}^{i} k_j$ is the total number of segments that the human assigned to any rating category $j \leq i$. When the user has not assigned any ratings within a particular category, i.e., $k_i = 0$ for some i, then we define the upper bound for category i as $\bar{R}_{k_{i+1}} := \bar{R}_{k_i}$, such the lower and upper bounds for this rating category are identical.

This definition guarantees that when all normalized return predictions $\tilde{R}(\sigma), \sigma \in X$, are distinct, then our model places k_i segments within each interval $[\bar{R}_i, \bar{R}_{i+1})$ for $i < n-1$ and k_i segments in $[\bar{R}_i, \bar{R}_{i+1}]$ for $i = n-1$, and thus predicts that k_i segments have rating i.

CHAPTER 3: EXPERIMENTS BASED ON SYNTHETIC DATA

**This chapter has previously been published in the Many Facets of Preference-Based Learning Workshop at the Fortieth International Conference on Machine Learning (ICML). It was co-authored by Mingkang Wu, Dr. Ellen Novoseller, Dr. Vernon Lawhern, Dr. Nick Waytowich, and Dr. Yongcan Cao and has been reproduced with minor revisions.*

3.1 Setup

We conduct a set of synthetic experiments based on tests in [20]. Synthetic experiments are meant to be used for testing the performance of an algorithm where a computer with knowledge of the reward provides a label. For PbRL, synthetic preferences are given such that the segment with a higher true reward is preferred.

For RbRL, many changes must be made to create a synthetic label. As RbRL is based of a single segment we had to find a new way to provide labels. We tested 2 methods, a sliding window where the top n% were given a specific label and a threshold for the reward. We saw that the performance for the threshold outperformed the sliding window method. In order to do this we divided the total possible reward by the number of classes, if a segment was over the threshold then it would be given that rating. For example, if the total reward is 100 and there are 3 classes. If a segment has a reward of 25 it would be in class 1 (Bad), if a different segment had a reward of 53 then it would be in class 2 (Neutral) and if a segment had a reward of 90 it would be in class 3 (Good).

We tested 2 methods of how a synthetic labeler is shown a segment, Uniform and Disagreement sampling. Uniform sampling is a random sample of n number of segments from a larger set of samples. Disagreement sampling uses an ensemble of reward predictors to choose a segment. For PbRL, it does this by getting a predicted reward for a segment pair, and finding a predicted label probability. Then taking the standard deviation of this probability among the ensemble members and the segment with the highest standard deviation is shown to the user.

For RbRL, disagreement sampling also gets the predicted reward but now only of a single segment. Like in PbRL this uses a ensemble of reward predictors, so each segment will have the number of ensemble members of predicted rewards. In RbRL the segment with the highest standard deviation in this predicted reward is shown to the user.

For both PbRL and RbRL the same neural network structure was used for both the reward predictor and control policy with the same hyperparameters as [20].

3.2 Results

For synthetic testing, our primary results are from the [20] (BPref) code implementation. Figure 3.1 shows the performance of RbRL with different number of classes and performance of PbRL for 2 Environments: Walker and Quadruped. What we were able to see is that for Walker, performance with higher number of n performs similar if not better than PbRL (left image). It can be seen that for Quadruped, the lower number of performs similar or outperforms PbRL (right image). We believe that the performance for both environments could be improved if an ideal threshold was found for synthetic labels.

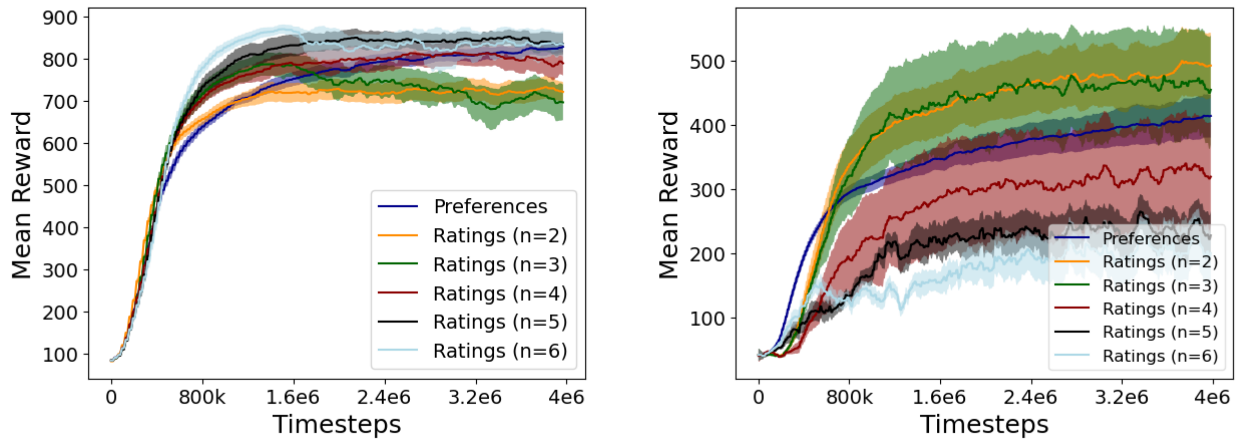


Figure 3.1: Performance of RbRL in synthetic experiments for different n, compared to PbRL: mean reward $\pm$ standard error over 10 experiment runs for Walker (left) and Quadruped (right).

CHAPTER 4: EXPERIMENTS BASED ON REAL HUMAN DATA

**This chapter has previously been published in the Many Facets of Preference-Based Learning Workshop at the Fortieth International Conference on Machine Learning (ICML). It was co-authored by Mingkang Wu, Dr. Ellen Novoseller, Dr. Vernon Lawhern, Dr. Nick Waytowich, and Dr. Yongcan Cao and has been reproduced with minor revisions and updated figures.*

4.1 Setup

We conducted a comprehensive set of tests based on human labels. There were 3 code bases used for these tests. The code bases are rl-teacher [12], BPref [20] and learning-from-human-preferences. learning-from-human-preferences is an implementation of [12]. The primary goal remains the same as in synthetic ratings, however a user is now providing labels rather than a computer.

4.2 rl-teacher

For rl-teacher, the setup for both RbRL and PbRL is the same. The only difference is that for PbRL a user is shown a segment pair and asked which segment they believe is preferred. In RbRL the user is shown a single segment and asked what they believe the performance is. For both RbRL and PbRL the segment lasts 2 seconds and is on a loop until the user provides a label. For all tests, the same neural network structures for both the reward predictor and control policy were used and the same hyperparameters.

4.2.1 RbRL with Different Numbers of Rating Classes

To evaluate the impact of the number of rating classes n on RbRL's performance we conducted tests in which a human expert (an author on the study) provided ratings from n = 2, ..., 8, on the Cheetah MuJoCo environment. For each class, 3 tests were run. Figure 4.1 shows the performance of RbRL for each class. We observe that for n =3, ..., 7 the performance is higher than that or n = 2 and 8. This is an expected behavior as when increasing the number of classes more information is provided to the network which explains why we see improvements with larger n but when n is too large, it is difficult for a user to provide a rating explaining why performance decreases with n = 8. In Table 4.1 you can see the physical meaning of for the different rating classes for n = 2, ..., 8.

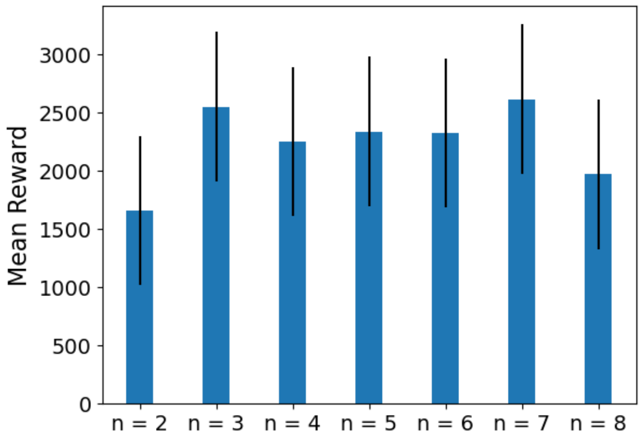


Figure 4.1: RbRL performance for different n in a human experiment: performance in the Cheetah environment (mean $\pm$ standard error over 3 experiment runs).

Table 4.1: Physical meaning of rating classes for $n = 2, 3, \ldots, 8$ used in our human experiments.

Number of Rating Classes	Rating Classes
$n = 2$	“good", “bad"
$n = 3$	“good", "neutral","bad"
$n = 4$	“very good", “good", “bad", “very bad"
$n = 5$	“very good", “good", “neutral",“bad", “very bad"
$n = 6$	“very good", “good", “slightly good", “slightly bad",“bad", “very bad"
$n = 7$	“very good", “good", “slightly good", “neutral” “slightly bad",“bad", “very bad"
$n = 8$	“perfect”, “very good", “good", “slightly good", “neutral” “slightly bad",“bad", “very bad"

4.2.2 Human User Study

In addition to showing performance with different number of classes, we wanted to evaluate the effectiveness of RbRL for non-expert users. To do this we conducted an IRB-approved human user study. For this user study we conducted tests on 3 of the OpenAI Gym MuJoCo Environments which were also used in [12]: Swimmer, Hopper and Cheetah.

For the user study a total of 20 participants were recruited (7 for Swimmer, 7 for Cheetah, and 6 for Hopper). For Swimmer and Cheetah the environment goal was used as the users goal. For example, in Cheetah, the environment goal is for the agent to run as fast as possible to the right.

This is so we can see the true reward for the environment based on the users labels. For Hopper, we wanted to show that the agent could learn a desired policy for which there is no reward defined. To do this users provided labels to have the agent do a backflip.

Each of the 20 participants performed 2 tests, one using RbRL and one using PbRL. To eliminate potential bias, we assigned each participant a randomized order in which to perform the PbRL and RbRL experiment run. We also randomized the environment which a user was providing labels to.

As the participants are non-experts, each participant was shown a video showing the desired result as well as undesired results so that participants could better understand the task. After this, each user was given time to become familiar with the interface and ask any questions they may have. Once the participant was ready to perform the real test they were given 30 minutes to provide feedback. Once the 30 minutes had passed the user was given a questionnaire about the algorithm. The participant was then given a 10 minute break to minimize fatigue before performing the other test. The participant then would provide feedback for the other algorithm for 30 minutes. Then would fill out the same questionnaire as before about the specific algorithm. Once the user had completed this they were also given a final questionnaire comparing the 2 algorithms. The questionnaires can be seen in A.1. Policy and reward learning occurred during the 30 minutes in which the user was providing feedback and continued until 4 million time-steps was reached.

4.2.3 Performance

Figure 4.2 shows the performance of PbRL and RbRL across the seven participants for Cheetah and Swimmer as well as expert results. What we can see is that for both Swimmer and Cheetah, RbRL performs similar to PbRL for non-experts. In addition, we can see that RbRL outperforms PbRL in both environments. As well, we can see that for both expert and non-expert RbRL learns at a faster rate. Expert tests are tests which were done by an author on the study. For consistency, the same expert tested both PbRL and RbRL in each environment.

In 4.2 RbRL and PbRL both perform similarly, we observed that in the individual rewards seen

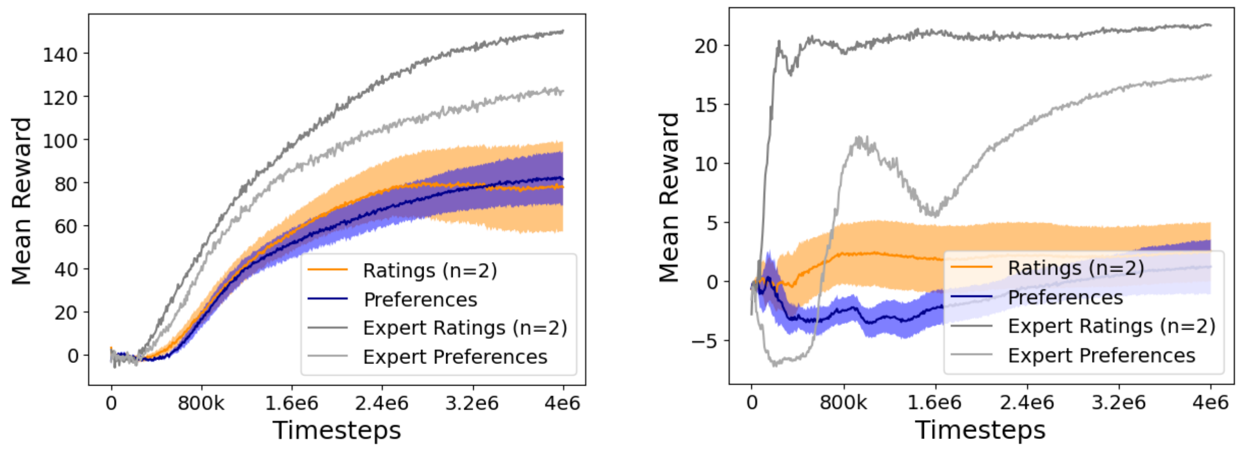


Figure 4.2: Performance of RbRL in the human user study: Cheetah (left) and Swimmer (right). For non-expert users, the plots show mean $\pm$ standard error over 7 users. The expert results are each over a single experiment run.

in Figure 4.4 that for both PbRL and RbRL there were outliers which lowered the mean. Due to this we checked the top 3 participants for both PbRL and RbRL to compare the best performance for both algorithms. This can be seen in Figure 4.3, this graph shows that for the top 3 participants RbRL can outperform PbRL in both environments. This shows us that for both the Cheetah and Swimmer Environments that RbRL can perform similar if not better than PbRL. To compare PbRL and RbRL on the Hopper backflip task, we ran the learned policies for the 6 participants to generate videos. Videos for the best learned policies from PbRL and RbRL can be found at https://bit.ly/3wgF5OH, and indicate that both RbRL and PbRL can learn to do a backflip.

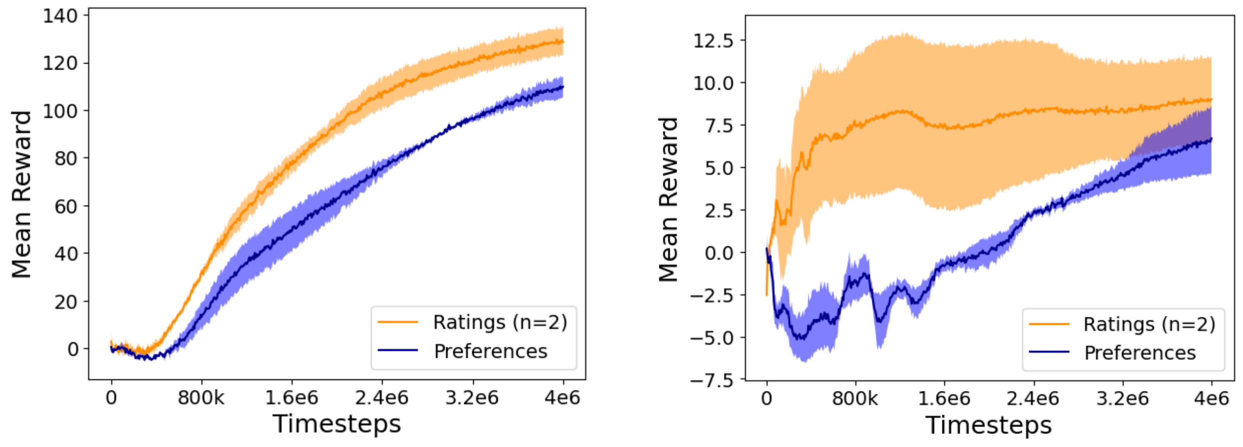


Figure 4.3: RbRL performance for the top 3 (non-expert) user study participants: mean reward ± standard error over the 3 experiment runs each for Cheetah (left) and Swimmer (right).

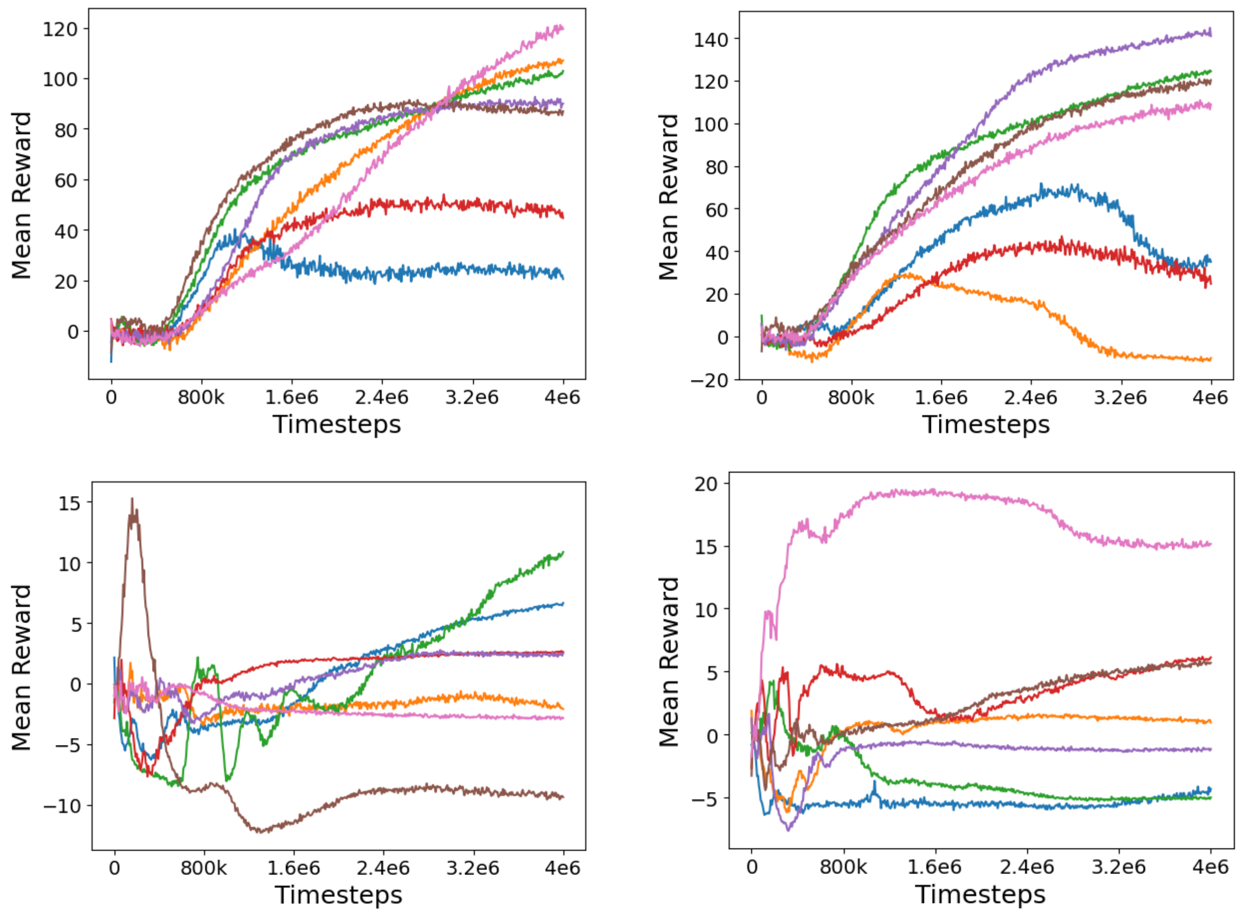
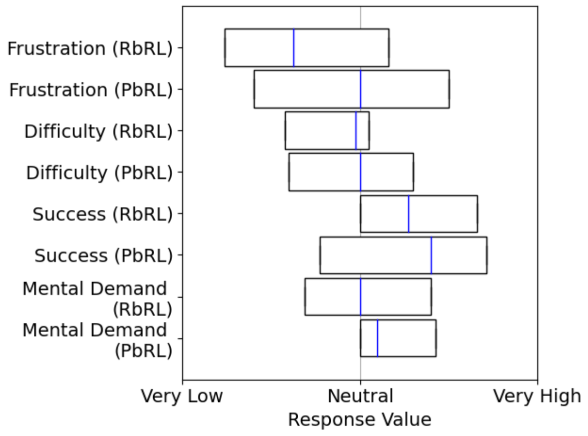


Figure 4.4: Results of individual runs in the human user study for the Cheetah (Top) and Swimmer environments (Bottom) for PbRL (left) and RbRL (right) (7 users in each case).

4.2.4 User Questionnaire Results

To understand how the non-expert users view their experience of giving rating and preferences, we conduct a post-experiment user questionnaire, shown in Appendix A.1. The questionnaire asked users for feedback about their experience providing feedback for both PbRL and RbRL, and their comparison between PbRL and RbRL. Figure 4.5 displays the normalized survey results from the 20 user study participants. In particular, the left subfigure of Figure 4.5 shows the participants' responses with respect to their separate opinions about PbRL and RbRL. These responses suggest that PbRL is more demanding and difficult than RbRL, leading users to feel more insecure and discouraged than when using RbRL. The right subfigure of Figure 4.5 shows the survey responses when users were asked to compare PbRL and RbRL; these results confirm the above findings and also show that users perceive themselves as completing the task more successfully when providing ratings (RbRL). One interesting observation is that the participants prefer RbRL and PbRL equally, which differs from the other findings. However, one participant stated that they preferred PbRL more because PbRL is more challenging. This suggests that "like" is a very subjective concept, making the responses for this question less informative than those for the other survey responses.



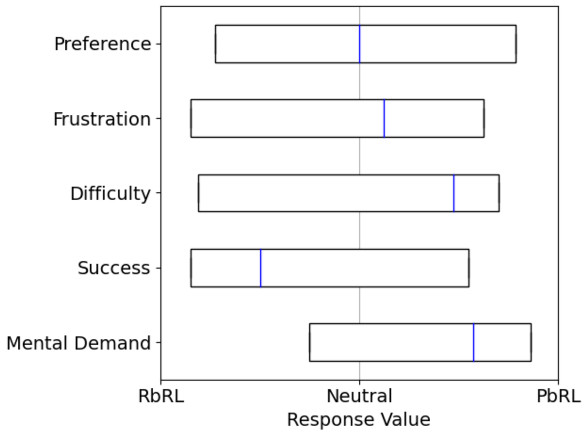


Figure 4.5: Participants' responses to questions for RbRL and PbRL. The set of survey questions is detailed in the Appendix. The blue bar indicates the median and the edges depict the 1st quartile (left) and 3rd quartile (right).

In addition to this, we also conducted a quantitative analysis of the human time effectiveness when humans were asked to give ratings and preferences. This can be seen in Figure 4.6, which shows the average number of human queries provided in 30 minutes for Cheetah, Swimmer, Hopper, and all three environments combined.

It can be observed that the participants can provide more ratings than pairwise preferences in all environments, indicating that it is easier and more efficient to provide ratings than to provide pairwise preferences. Which is further backed up by the feedback from the user questionnaire. On average, participants can provide approximately 14.03 ratings per minute, while they provide only 8.7 preferences per minute, which means that providing a preference requires 62% more time than providing a rating. For Cheetah, providing a preference requires 100%+ more time than providing a rating, which is mainly due to the need to compare video pairs that are very similar. For Swimmer and Hopper, the environments and goals are somewhat more complicated. Hence, providing ratings can be slightly more challenging, but is still easier than providing pairwise preferences.

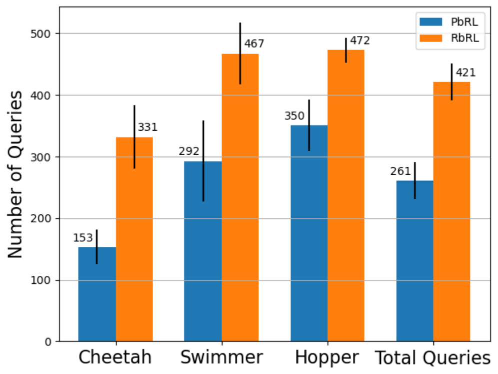


Figure 4.6: Number of queries provided in 30 minutes across participants in our human user study (mean ± standard error).

4.3 BPref

BPref [20] is a state-of-the-art Preference-based reinforcement learning benchmark code. The primary code base is based on synthetic feedback only. This works well if you are attempting to simulate human behaviour, our Rating-based Reinforcement Learning method however, can be difficult to define a synthetic label. Due to this, we created a simple interface which is similar to 4.4 for both Preferences and Ratings. For preferences, a segment pair is shown to the user and the user provides a preference. For ratings, a single segment is shown to the user and the user provides a rating. All background pipelines remain the same, and all hyperparameters remain the same between PbRL and RbRL.

4.3.1 Performance

In Figure 4.7 you can see initial results for both Preferences and Ratings. It can see from this figure that for Quadruped, RbRL can perform similar or outperform PbRL. In walker we can see that RbRL is unable to outperform PbRL, we believe that this is because it is a simpler task, this means that the more ratings a user has as an option, the more inconsistent their ratings can be.

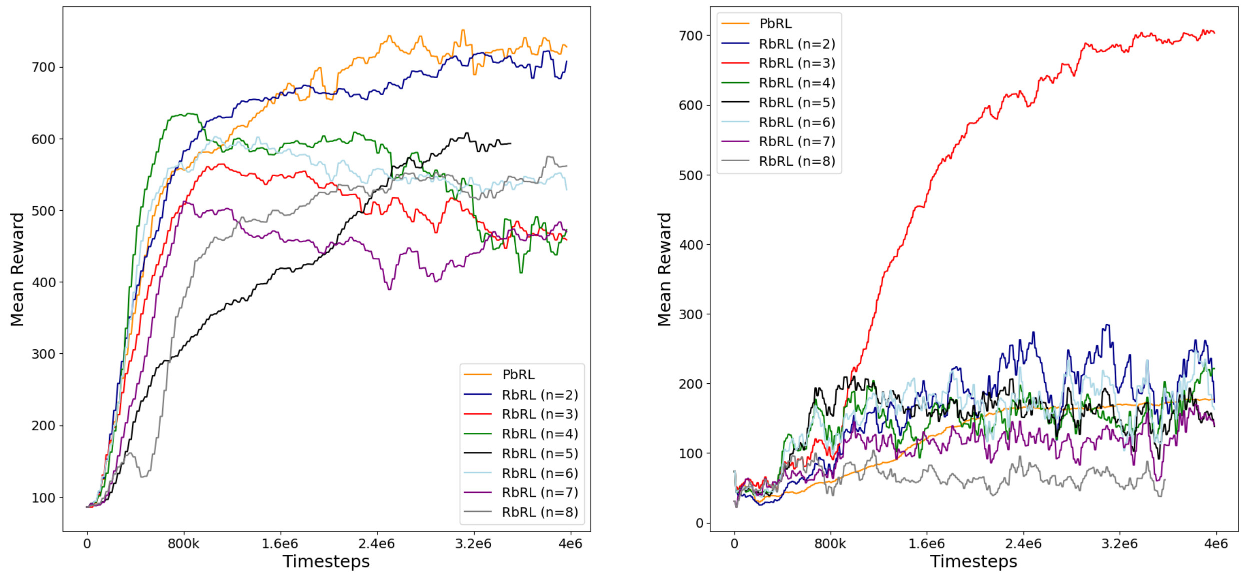


Figure 4.7: Performance of RbRL in Human experiments for different n, compared to PbRL: for a single run for Walker (left) and Quadruped (right).

4.4 Learning from Human Preferences

Preliminary tests were conducted using the learning-from-human-preferences repository. The setup for both RbRL and PbRL is the same. For this repository, some key features are not implemented, such as Batchnorm, L2 Normalization on the reward predictor model. Feedback is requested at a constant rate rather than on schedule, as well, the segments were chosen at random rather than by using disagreement. In these tests we were attempting to identify the potential in time savings as well as if there was any performance gain. For all tests, the same hyperparameters were used for both RbRL and PbRL. All tests done in this section were done by Authors.

4.4.1 Human Time

Our first objective is to quantify the time it takes for humans to provide a single rating (RbRL) and a single preference (PbRL) for various environments to evaluate which type of human feedback, rating vs preference, takes less time. For all tests in learning-from-human-preferences, Atari environments were used, similar to tests conducted in [12].

Just like in rl-teacher and BPref, a user is shown a segment which they provided feedback to. In Table 4.2 you can see the exact results for both PbRL and RbRL for the same number of labeled feedback. For this test, 1000 labels are provided to the network, these labels are provided at the beginning of training rather than throughout the training like in rl-teacher. What we can see from the results is that for n = 4, it takes roughly 2 to 3 seconds to provide a rating. However, for most environments the fewer the number of classes the less time it took to provide a label. For all n the user was able to provide the same number of labels in less time using RbRL compared to PbRL. This is further amplified by complex environments such as Private Eye and Pitfall. The average percentage time saving per human sample is 5%-50%.

Table 4.2: Time it takes for a user to provide 1000 labels for different n compared to preferences

Human time comparison for PbRL and RbRL				
Environment	PbRL	RbRL ($n = 4$)	RbRL ($n = 3$)	RbRL ($n = 2$)
Breakout	47:16	44:12	44:51	43:29
Sea Quest	58:14	54:51	46:45	40:52
Private Eye	1:34:24	58:32	52:58	55:16
Pitfall	1:50:18	52:47	52:45	42:23
Space Invaders	1:24:57	1:00:14	47:43	50:54
Pong	1:08:33	47:52	49:53	45:26

4.4.2 Agent Performance

Our second objective is observe the performance of RbRL in comparison to PbRL under the same conditions. For testing, the same 6 environments were used as in 4.4.1. In all testing, 1000 total labels are provided, 500 from a randomized policy rollout and 500 from the latest policy rollout. Table 4.3 shows the mean and maximum episode return of 20 runs for the given environments.

Table 4.3: Mean reward over 20 runs for 1000 queries and 10 million timesteps of training. * denotes runs that have memory/implementation issues with respect to hardware.

Episodic reward comparison for PbRL and RbRL				
Environment	PbRL	RbRL ($n = 4$)	RbRL ($n = 3$)	RbRL ($n = 2$)
Breakout (mean)	1.35	5.8	0.1	**10.85**
Breakout (max)	5	21	1	**46**
SeaQuest (mean)	7.9	8.75	8.45	**8.8**
SeaQuest (max)	10	10	10	**11**
Private Eye (mean)	0	**0.4**	-26.4*	-26.4*
Private Eye (max)	0	**1**	-25*	-25*
Pitfall (mean)	-2.9	-1.1	-0.7	**-0.3**
Pitfall (max)	0	0	0	**0**
Space Invaders (mean)	11.55	10.35	3.85	**12.2**
Space Invaders (max)	31	30	8	**31**
Pong (mean)	-20.7	-20.35	-20.5	**-20.25**
Pong (max)	-20	-19	-19	**-18**

4.4.3 Different k Values

As shown in (2.3), different k values will lead to different models of human rating probabilities for different rating classes. In particular, a larger k means that the rating probability leans more towards the set that the corresponding $\tilde{R}(\sigma)$ is in (Please check Subsection 2.2.3 for more details). However, if a very large k is selected, any mislabeling by humans can lead to a spike in the loss function, leading to volatility of reward and control policy learning. To obtain a quantitative view of the impact of k on learning performance, we conducted tests for 3 values of k. Table 4.4 shows the results from a run of 10 million time steps with a total of 1000 human ratings. 500 ratings were given based on an initial random policy rollout and the last 500 ratings were given based on the latest policy rollout. For this set of tests it can be seen that $k = 30$ had the best performance. This is the default value for k we used in all other implementation and tests for the rating-based reinforcement learning versions of rl-teacher, BPref, and learning-from-human-preferences.

Table 4.4: Mean reward over 20 runs for different k values when $n = 2$. * denotes runs that have memory/implementation issues with respect to hardware.

Different k values			
Environment	k = 10	k = 30	k = 50
Breakout	0.05	10.85	9.35
Sea Quest	6.5	8.8	9.3
Private Eye	0.0	-26.4*	-26.4*
Pitfall	N/A	-0.3	N/A
Space Invaders	21.85	12.2	14.55
Pong	-20.35	-20.25	-19.9

CHAPTER 5: CONCLUSION AND FUTURE WORK

In this thesis, we proposed a novel rating-based reinforcement learning framework that learns a reward function for training policies. The proposed approach is based on a new multi-class cross entropy loss function that learns the reward from ratings provided by users. Experiments for synthetic feedback show that rating-based reinforcement learning performs similar or even outperforms preference-based reinforcement learning for the state-of-the-art benchmark code base. Results from our IRB approved user study show similar or better performance when using ratings as compared to preferences. In addition, it is shown from the user study that users generally felt easier, less frustrating and less mentally demanding to provide ratings than preferences. Lastly, our results show that users are able to provide more ratings than preferences within the same amount of time.

We demonstrated that ratings can be useful in limited environments and applications, and expect more work to be done for more environments and applications. It is important to highlight that ratings may not be consistent during learning due to, e.g., users' attention span and fatigue level over time. Hence, developing a mechanism to quantify users' performance change over time along with workload level will be important for the development of cognitive-aware rating-based reinforcement learning approaches in real-world applications.

APPENDIX A: QUESTIONNAIRE

A.1 Questionnaire

The next 3 pages are the questionnaire the user was given after the respective test.

Appendix A.1.1

User Study Questionnaire

Instructions: Mark the bin best reflecting your response when instructed to do so.

Rating-based feedback:

1. How mentally demanding was the task?

Very low Neutral Very high

2. How successful were you in accomplishing what you were asked to do?

Very low Neutral Very high

3. How hard did you have to work to accomplish your level of performance?

Very low Neutral Very high

4. How insecure, discouraged, irritated, stressed, and annoyed were you?

Very low Neutral Very high

Appendix A.1.2

User Study Questionnaire

Instructions: Mark the bin best reflecting your response when instructed to do so.

Preference-based feedback:

1. How mentally demanding was the task?

Very low Neutral Very high

2. How successful were you in accomplishing what you were asked to do?

Very low Neutral Very high

3. How hard did you have to work to accomplish your level of performance?

Very low Neutral Very high

4. How insecure, discouraged, irritated, stressed, and annoyed were you?

Very low Neutral Very high

Appendix A.1.3

User Study Questionnaire

Instructions: Mark the bin best reflecting your response when instructed to do so.

Comparison of preference-based and ratings-based feedback:

1. Which was more mentally demanding?

Ratings Neutral Preferences

2. With which were you able to better accomplish the task?

Ratings Neutral Preferences

3. With which was it more difficult to accomplish the task?

Ratings Neutral Preferences

4. Which caused you to be more insecure, discouraged, irritated, stressed, and annoyed?

Ratings Neutral Preferences

5. Which did you like more?

Ratings Neutral Preferences

BIBLIOGRAPHY

[1] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565*, 2016.

[2] Saurabh Arora and Prashant Doshi. A survey of inverse reinforcement learning: Challenges, methods and progress. *Artificial Intelligence*, 297:103500, 2021.

[3] Erdem Bıyık, Dylan P Losey, Malayandi Palan, Nicholas C Landolfi, Gleb Shevchuk, and Dorsa Sadigh. Learning reward functions from diverse sources of human feedback: Optimally integrating demonstrations and preferences. *The International Journal of Robotics Research*, 41(1):45–67, 2022.

[4] Erdem Bıyık, Malayandi Palan, Nicholas C Landolfi, Dylan P Losey, Dorsa Sadigh, et al. Asking easy questions: A user-friendly approach to active reward learning. In *Conference on Robot Learning*, pages 1177–1190, 2020.

[5] Erdem Bıyık, Aditi Talati, and Dorsa Sadigh. APReL: A library for active preference-based reward learning algorithms. In *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 613–617, 2022.

[6] Daniel Brown, Russell Coleman, Ravi Srinivasan, and Scott Niekum. Safe imitation learning via fast Bayesian reward inference from preferences. In *International Conference on Machine Learning*, pages 1165–1177, 2020.

[7] Daniel Brown, Wonjoon Goo, Prabhat Nagarajan, and Scott Niekum. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In *Proceedings of the International Conference on Machine Learning*, pages 783–792, 2019.

[8] Daniel S Brown, Wonjoon Goo, and Scott Niekum. Better-than-demonstrator imitation learning via automatically-ranked demonstrations. In *Conference on Robot Learning*, pages 330–359, 2020.

[9] Carlos Celemin and Javier Ruiz-del Solar. COACH: Learning continuous actions from corrective advice communicated by humans. In *2015 International Conference on Advanced Robotics (ICAR)*, pages 581–586, 2015.

[10] Jaedeug Choi and Kee-Eung Kim. Map inference for Bayesian inverse reinforcement learning. *Advances in Neural Information Processing Systems*, 24, 2011.

[11] Jaedeug Choi and Kee-Eung Kim. Nonparametric Bayesian inverse reinforcement learning for multiple reward functions. *Advances in Neural Information Processing Systems*, 25, 2012.

[12] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

[13] Christian Daniel, Malte Viering, Jan Metz, Oliver Kroemer, and Jan Peters. Active reward learning. In *Robotics: Science and Systems*, volume 98, 2014.

[14] Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. Go-explore: a new approach for hard-exploration problems. *arXiv preprint arXiv:1901.10995*, 2019.

[15] Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *Proceedings of the International Conference on Machine Learning*, pages 49–58, 2016.

[16] Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adverserial inverse reinforcement learning. In *International Conference on Learning Representations*, 2018.

[17] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the International Conference on Machine Learning*, pages 1861–1870, 2018.

[18] Borja Ibarz, Jan Leike, Tobias Pohlen, Geoffrey Irving, Shane Legg, and Dario Amodei. Reward learning from human preferences and demonstrations in Atari. *Advances in Neural Information Processing Systems*, 31, 2018.

[19] W Bradley Knox and Peter Stone. Interactively shaping agents via human reinforcement: The TAMER framework. In *Proceedings of the fifth international conference on Knowledge capture*, pages 9–16, 2009.

[20] Kimin Lee, Laura Smith, Anca Dragan, and Pieter Abbeel. B-Pref: Benchmarking preference-based reinforcement learning. *Advances in Neural Information Processing Systems*, 2021.

[21] Kimin Lee, Laura M Smith, and Pieter Abbeel. PEBBLE: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training. In *International Conference on Machine Learning*, pages 6152–6163, 2021.

[22] Sergey Levine, Zoran Popovic, and Vladlen Koltun. Nonlinear inverse reinforcement learning with Gaussian processes. *Advances in Neural Information Processing Systems*, 24, 2011.

[23] Yuxi Li. Reinforcement learning applications. *arXiv preprint arXiv:1908.06973*, 2019.

[24] Xinran Liang, Katherine Shu, Kimin Lee, and Pieter Abbeel. Reward uncertainty for exploration in preference-based reinforcement learning. In *International Conference on Learning Representation*, 2022.

[25] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *International Conference on Learning Representations*, 2015.

[26] Marlos C Machado, Marc G Bellemare, Erik Talvitie, Joel Veness, Matthew Hausknecht, and Michael Bowling. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research*, 61:523–562, 2018.

[27] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[28] John Schulman; Sergey Levine; Pieter Abbeel; Michael Jordan; Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897, 2015.

[29] Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the International Conference on Machine Learning*, volume 99, pages 278–287, 1999.

[30] Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, page 2, 2000.

[31] Jongjin Park, Younggyo Seo, Jinwoo Shin, Honglak Lee, Pieter Abbeel, and Kimin Lee. SURF: Semi-supervised reward learning with data augmentation for feedback-efficient preference-based reinforcement learning. In *International Conference on Learning Representations*, 2022.

[32] Dean A Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural Computation*, 3(1):88–97, 1991.

[33] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[34] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484, 2016.

[35] Andrew Szot, Amy Zhang, Dhruv Batra, Zsolt Kira, and Franziska Meier. BC-IRL: Learning generalizable reward functions from demonstrations. In *The Eleventh International Conference on Learning Representations*, 2022.

[36] Garrett Warnell, Nicholas Waytowich, Vernon Lawhern, and Peter Stone. Deep TAMER: Interactive agent shaping in high-dimensional state spaces. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[37] Paul Weng. Markov decision processes with ordinal rewards: Reference point-based preferences. In *Twenty-First International Conference on Automated Planning and Scheduling*, 2011.

[38] Christian Wirth, Riad Akrour, Gerhard Neumann, Johannes Fürnkranz, et al. A survey of preference-based reinforcement learning methods. *Journal of Machine Learning Research*, 18(136):1–46, 2017.

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

[40] Yichong Xu, Ruosong Wang, Lin Yang, Aarti Singh, and Artur Dubrawski. Preference-based reinforcement learning with finite-time guarantees. *Advances in Neural Information Processing Systems*, 33:18784–18794, 2020.

[41] Huixin Zhan, Feng Tao, and Yongcan Cao. Human-guided robot behavior learning: A GAN-assisted preference-based reinforcement learning approach. *IEEE Robotics and Automation Letters*, 6(2):3545–3552, 2021.

[42] Brian D Ziebart, Andrew Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2008.

VITA

Devin White began his academic career at the Alam Colleges in San Antonio Texas in 2016. In 2018, he transferred to the University of Texas at San Antonio (UTSA) to study electrical and computer engineering. Over the summer of 2021 Devin was a part of the National Science Foundation Research Experience for Undergraduates (NSF REU) at UTSA. This was his introduction to artificial intelligence (AI) and its many topics. He continued to do research in human-guided reinforcement learning throughout his undergraduate. In the fall of 2021 Devin received the outstanding undergraduate reward for his work in research as well as mentorship in the area. In addition, in the fall of 2021, Devin was apart of a team SDQ Engineering which came first place at the UTSA tech symposium. He graduated Cum Laude with his Bachelors in electrical and computer engineering in the fall of 2021. After graduate Devin was a technical lab assistant in the unmanned systems lab where he continued working on research and mentoring students in AI. In the fall of 2022 he was a member of the first cohort of students in the new Masters in Artificial Intelligence at the UTSA. Throughout his masters, he was a member of a team called Silent Siren in the UTSA Draper Data Science Business Plan Competition. They worked on creating an application and edge device which would inform drivers who are deaf or hard of hearing about emergency vehicle sirens. At this competition, they received honorable mention and a bonus.