Reward Learning through Ranking Mean Squared Error

Chaitanya Kharyal, Calarina Muslimani, Matthew E. Taylor

Keywords: Reinforcement learning, Reward Learning, Learning from feedback

Summary

Reward design remains a significant bottleneck in applying reinforcement learning (RL) to real-world problems. A popular alternative is reward learning, where reward functions are inferred from human feedback rather than manually specified. Recent work has proposed learning reward functions from human feedback in the form of ratings, rather than traditional binary preferences, enabling richer and potentially less cognitively demanding supervision. Building on this paradigm, we introduce **rMSE**, a new rating-based RL method that treats human-provided ratings as ordinal targets. Our approach learns from an offline dataset of trajectory–rating pairs, where each trajectory is labeled with a discrete rating (e.g., "bad", "neutral", "good"). At each training step, we sample one trajectory per rating class, compute their predicted returns under the learned reward model, and rank them using a differentiable sorting operator (i.e., soft ranks). We then optimize a mean squared error loss between the resulting soft ranks and the human ratings. Additionally, we incorporate a conservative regularization term to reduce overestimation on out-of-training-distribution actions. Through experiments with simulated human feedback, we demonstrate that rMSE can outperform another rating-based RL algorithm in Hungry-Thirsty and Lunar Lander. We also found that rMSE can learn reward functions that are more aligned to the simulated preferences than the baseline method. Through experiments with simulated feedback, we show that rMSE outperforms a recently proposed rating-based RL method in the Hungry-Thirsty and Lunar Lander domains. Additionally, rMSE learns reward functions that are better aligned with the simulated ratings.

Contribution(s)

1. We introduce **rMSE**, a novel rating-based RL method that treats human-provided ratings as ordinal targets and optimizes a ranking-based mean squared error loss.
   Context: None
2. In the Hungry-Thirsty and Lunar Lander environments, we demonstrate that rMSE can outperform RbRL, producing reward functions that result in better RL policies (i.e., policies that yield greater return under the environment reward). We also find that rMSE reward functions are slightly more aligned with the simulated ratings compared to RbRL.
   Context: We only compare our method to RbRL (White et al., 2024), as this is the only existing rating-based reinforcement learning approach.

Reward Learning through Ranking Mean Squared Error

Chaitanya Kharyal[1], **Calarina Muslimani**[1], **Matthew E. Taylor**[1,2]
kharyal,musliman,matthew.e.taylor@ualberta.ca

[1]**Department of Computing Science, University of Alberta, Canada**
[2]**Alberta Machine Intelligence Institute (Amii)**

Abstract

Reward design remains a significant bottleneck in applying reinforcement learning (RL) to real-world problems. A popular alternative is reward learning, where reward functions are inferred from human feedback rather than manually specified. Recent work has proposed learning reward functions from human feedback in the form of ratings, rather than traditional binary preferences, enabling richer and potentially less cognitively demanding supervision. Building on this paradigm, we introduce **rMSE**, a new rating-based RL method that treats human-provided ratings as ordinal targets. Our approach learns from an offline dataset of trajectory–rating pairs, where each trajectory is labeled with a discrete rating (e.g., "bad", "neutral", "good"). At each training step, we sample one trajectory per rating class, compute their predicted returns under the learned reward model, and rank them using a differentiable sorting operator (i.e., soft ranks). We then optimize a mean squared error loss between the resulting soft ranks and the human ratings. Additionally, we incorporate a conservative regularization term to reduce overestimation on out-of-training-distribution actions. Through experiments with simulated human feedback, we demonstrate that rMSE can outperform another rating-based RL algorithm in Hungry-Thirsty and Lunar Lander. We also found that rMSE can learn reward functions that are more aligned to the simulated preferences than the baseline method. Through experiments with simulated feedback, we show that rMSE outperforms a recently proposed rating-based RL method in the Hungry-Thirsty and Lunar Lander domains. Additionally, rMSE learns reward functions that are better aligned with the simulated ratings.

1 Introduction

Deep reinforcement learning (RL) has achieved remarkable success in games like Go, Chess, and Atari. However, a key feature of these environments is the presence of a well-defined reward function. In contrast, real-world environments often lack clean specifications, making reward design a significant bottleneck to deploying RL in complex, practical applications (Knox et al., 2023; Knox & MacGlashan, 2024). In practice, reward design is often an informal trial-and-error process where RL practitioners iteratively adjust a reward function until the RL agent exhibits acceptable behavior (Booth et al., 2023). This procedure can be error-prone, resulting in reward misspecification or reward overfitting. Reward misspecification arises when practitioners inadvertently define a reward function that does not align with the true task objective, leading the agent to learn undesirable or unintended behaviors (Skalse et al., 2022; Pan et al., 2022). Reward overfitting, on the other hand, occurs when reward functions are unintentionally over-engineered for a specific algorithm or environment setup, resulting in poor generalization to other RL algorithms or even slight variations of the environment (Booth et al., 2023). Notably, these challenges have been observed even in tabular

grid worlds, illustrating that reward design remains a core challenge, even in simple settings (Booth et al., 2023; Muslimani et al., 2025b).

A popular alternative to manual reward engineering is *reward learning*, where reward functions are inferred from human feedback rather than explicitly designed. This feedback can take various forms, including scalar evaluations (Knox & Stone, 2009; Warnell et al., 2018), demonstrations (Arora & Doshi, 2021), or pairwise preferences over agent behaviors (Christiano et al., 2017b; Lee et al., 2021). In particular, learning from pairwise preferences known as, preference-based reinforcement learning has become widely used, as it may require less human effort and has been integral in the success of large language models (OpenAI, 2023).

Despite its success, learning from binary preferences can be limiting. For one, each binary comparison conveys only a single bit of information, which can lead to sample inefficiency. As a result, more human time may be required to collect a higher number of preferences, increasing the overall feedback burden. Moreover, such feedback is inherently relative. It indicates which behavior is preferred, but not by how much, nor whether either option is good in an absolute sense. For example, if both behaviors under comparison are poor, a human can at best indicate that they are equally preferable, but cannot express that both are low-quality overall.

Recent work has introduced a new paradigm known as rating-based reinforcement learning (RbRL) (White et al., 2024), in which humans provide discrete, multi-class ratings rather than binary preferences to guide reward learning. Instead of comparing two behaviors and selecting the preferred one, the human observes a single behavior at a time and assigns it a rating, typically on a fixed scale. This shift enables the collection of richer feedback, as ratings can capture both relative and absolute assessments of trajectory quality. Moreover, user studies have found that participants perceive rating-based feedback as less cognitively demanding than preference-based feedback, and report feeling more successful when completing tasks using ratings (White et al., 2024).

The advantages of RbRL motivate the development of more effective algorithms for learning from ratings. To this end, we propose a new rating-based reinforcement learning method, **rMSE**, which learns a reward function from an offline dataset of trajectories labeled with ordinal ratings (e.g., "bad", "neutral", or "good"). At each training step, we sample one trajectory per rating class, compute their predicted returns under the reward model, and rank them using a differentiable sorting operator (Blondel et al., 2020) (i.e., soft ranks). We then minimize a mean squared error loss between the resulting soft ranks and the human ratings. Additionally, we incorporate a conservative regularization term that penalizes high predicted rewards for state-action pairs not seen in the training data.

We evaluate rMSE on two environments: Hungry-Thirsty (Barto et al., 2009) and Lunar Lander (Brockman et al., 2016a). Our results show that rMSE learns reward models that lead to more performant RL policies than the RbRL baseline. Additionally, the reward models learned by rMSE are more closely aligned with the simulated feedback compared to those learned by RbRL. Overall, our work aims to advance research in rating-based RL by demonstrating how ordinal ratings can be used effectively to train reward models.

2 Related Work

Reward learning is a broad field in which reward functions are inferred from various forms of human feedback, including demonstrations, preferences, scalar evaluations, ratings, or combinations thereof. One common approach is inverse reinforcement learning (IRL), which learns a reward function such that the resulting policy produces behaviors similar to those in the provided demonstrations (Ng & Russell, 2000). Despite recent advances showing that reward functions can be recovered from suboptimal demonstrations (Shiarlis et al., 2016; Brown et al., 2019), it is still argued that providing demonstrations can be time-consuming and difficult in many settings.

As an alternative to demonstrations, other approaches rely on preference-based feedback (e.g., preference-based reinforcement learning). In this setting, human users typically provide binary

preferences over pairs of agent behaviors (Christiano et al., 2017a; Lee et al., 2021; Park et al., 2022; Liang et al., 2022; Hu et al., 2024; Muslimani et al., 2025a; Muslimani & Taylor, 2025). This form of supervision has gained recent popularity, as it is often considered more intuitive and less demanding than providing full demonstrations. However, binary preferences can be limited in the richness of information they convey. To address this, recent work has explored scaled preferences, where users indicate not just which behavior they prefer, but also the strength of that preference (e.g., on a scale from "strongly prefer A" to "strongly prefer B") (Wilde et al., 2021). These graded comparisons have been shown to outperform strict binary preferences, offering more informative supervision for reward learning.

Similarly, scalar feedback methods provide rich signals by allowing humans to rate behaviors directly. For example, the TAMER framework allowed humans to provide binary signals indicating whether a behavior was judged to be optimal (Knox & Stone, 2009; Warnell et al., 2018). Later work introduced reward sketching, where, for a given behavior, humans continuously provide a scalar signal indicating the agent's progress toward a goal (Cabi et al., 2020). Most similar to our approach is a recently proposed framework, rating-based reinforcement learning. It handles multi-class discrete ratings by using a novel cross-entropy loss formulation (White et al., 2024). In this setup, humans assign class labels such as "very good," "good," "bad," or "very bad" to entire trajectories, and these labels are then used to train a reward function.

3 Problem Formulation

We consider a Markov decision process without rewards (MDP\R), where feedback is provided in the form of human ratings (White et al., 2024). Formally, MDP\R is defined as: $\mathcal{M} = (\mathcal{S}, \mathcal{A}, T, \rho, \gamma, n, \mathcal{D})$, where $\mathcal{S}$ and $\mathcal{A}$ are the state and action spaces, respectively, $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the transition probability function, ρ is the initial state distribution, and γ is the discount factor. The parameter n denotes the number of discrete rating classes, and $\mathcal{D}$ is an offline dataset of trajectories τ_i and their associated human ratings c_i. Specifically, a human observer watches each trajectory τ_i, where $\tau_i = (s_1^i, a_1^i, s_2^i, a_2^i, \dots)$, and assigns it a rating $c_i \in 0, 1, \dots, n-1$, where c_i indicates the perceived quality of the trajectory. A rating of 0 represents the lowest quality, while $n-1$ represents the highest. Note that the rating classes can also be assigned descriptive labels to aid interpretation. For instance, with $n = 3$ rating classes, the labels might be: 0 — "bad", 1 — "neutral", and 2 — "good". This process is repeated for all offline trajectories, and the resulting data is grouped by rating class. Specifically, for each rating class $k \in \{0, 1, \dots, n-1\}$, we define a subset $\mathcal{D}_k = \{\tau_i \mid (\tau_i, c_i) \in \mathcal{D},\ c_i = k\}$ containing all trajectories assigned to rating class k. The full dataset is then $\mathcal{D} = \bigcup_{k=0}^{n-1} \mathcal{D}_k$.

In the standard reinforcement learning setting, the key difference is that the MDP includes a reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, which provides a numerical reward for each state-action pair. This replaces the human ratings component (e.g., n, $\mathcal{D}$) used in our setting. The objective is then to find an optimal policy π^* that maximizes the expected discounted return, G_r, defined as: $\sum_t \gamma^t r(s_t, a_t)$. In contrast, our MDP\R setting lacks an engineered reward function, and thus the goal becomes two-fold: (1) to learn a parameterized reward function $\hat{r}_\theta$, with parameters θ, from human-provided ratings, and (2) to learn a policy that maximizes the expected discounted return with respect to $\hat{r}_\theta$.

4 Method

Given a set of rating classes $c_0, \dots, c_{n-1}$, where $i < j$ implies that trajectories in class c_i are rated lower than those in class c_j, we assume that for any $\tau_a \in c_i$ and $\tau_b \in c_j$, the (unobserved) return under the human's implicit reward function satisfies $G(\tau_a) < G(\tau_b)$. Here, $G(\tau) = \sum_{t:(s_t,a_t)\in\tau} \gamma^t r(s_t, a_t)$ denotes the discounted return of trajectory τ. Consequently, by sampling one trajectory from each class, we obtain a perfectly ordered ranking over n trajectories (where n is the number of rating classes).

We leverage this observation to define a novel ranking mean squared error (rMSE) objective over a set of trajectories, enabling us to learn a reward function $\hat{r}_\theta$ that estimates the reward function based on trajectory ratings. Finally, we use this learned reward function, in place of an engineered reward, to train an RL policy $\pi(a|s)$ to maximize the expected return, $\hat{G}_\theta$ denoted as $\mathbb{E}\left[\sum_t \gamma^t \hat{r}_\theta(s_t, a_t)\right]$.

4.1 Reward Learning with the Ranking Mean Squared Error Objective

To learn the reward function $\hat{r}_\theta$ from the dataset $\mathcal{D}$, we sample one trajectory τ_i from each class dataset $\mathcal{D}_k$. The return for each sampled trajectory is estimated as:

$$\hat{G}^i_\theta = \sum_{t,\ (s_t,a_t)\in\tau_{(i)}} \gamma^t \hat{r}_\theta(s_t, a_t) \tag{1}$$

We then rank these returns using a differentiable sorting algorithm (Blondel et al., 2020), yielding a soft rank $\hat{R}^i_\theta$ for each $\hat{G}^i_\theta$. For example, suppose we have predicted returns $\hat{G}_\theta = [3.2, 1.0, 4.5]$. The corresponding soft ranks might be $\hat{R}_\theta \approx [1.5, 3.0, 1.0]$, indicating that the third return (4.5) is ranked highest, the second return (1.0) is ranked lowest, and the first return (3.2) lies in between. Unlike hard sorting, the differentiable sort provides continuous (and potentially non-integer) ranks that allow gradients to flow through the ranking operation during optimization.

The rMSE loss is computed as the mean squared error between the soft rank and the rating class provided by the human:

$$\mathcal{L}_{\text{rMSE}} = \frac{1}{n}\sum_{i=0}^{n}\left(\hat{R}^i_\theta - c_i\right)^2 \tag{2}$$

Continuing the example, suppose the human-provided rating classes for the sampled trajectories are $c = [2.0, 3.0, 1.0]$. We compute the rMSE loss as:

$$\mathcal{L}_{\text{rMSE}} = \frac{1}{3}\left[(1.5-2.0)^2 + (3.0-3.0)^2 + (1.0-1.0)^2\right] \tag{3}$$

In this example, only the predicted rank for the first trajectory deviates slightly from the corresponding human rating, resulting in a relatively low loss. Since the soft ranks are differentiable with respect to the reward parameters θ, minimizing this loss allows the model to adjust $\hat{r}_\theta$ to better align with the human-provided ratings.

See Figure 1 for an overview of the rMSE training process. Furthermore, inspired by offline RL approaches (Kumar et al., 2020; Li et al., 2021), we penalize high predicted rewards (under r_θ) for state-action pairs not present in the offline dataset, $\mathcal{D}$:

$$\mathcal{L}_{\text{OOD}} = \mathbb{E}_{s,a\sim p}\left[\hat{r}_\theta(s,a)\right] - \mathbb{E}_{s,a\sim\mathcal{D}}\left[\hat{r}_\theta(s,a)\right] \tag{4}$$

Here, p is a distribution used to sample out-of-distribution state-action pairs. In our experiments, we use a uniform distribution over the state-action space ($\mathcal{S} \times \mathcal{A}$) as p. The first term in $\mathcal{L}_{\text{OOD}}$ penalizes high predicted reward values for out-of-distribution pairs, while the second term prevents the learned reward function from collapsing to large negative values. Without the second term, the learned reward function could trivially assign large negative values to all the state-action pairs, including those in the dataset. Finally, the overall loss function is defined as:

$$\mathcal{L} = \mathcal{L}_{\text{rMSE}} + \beta\, \mathcal{L}_{\text{OOD}} \tag{5}$$

The advantage of using rMSE objective over the RbRL objective is multifold:

1. **Richer feedback for reward learning:** The rMSE objective provides more informative feedback during training. For instance, if a trajectory from rating class 0 (e.g., "very bad") is misclassified by the learned reward function as belonging to class 3 (e.g., "very good"), the associated error

should be larger than if it were misclassified as class 1 (e.g., "bad"). The rMSE loss captures this magnitude of error, while the multi-class cross-entropy loss used in RbRL does not since it treats every misclassification equally.

2. **Eliminating hyperparameters:** The RbRL objective requires specifying rating class boundaries b. A trajectory is assigned to class k only if its return lies between $b[k]$ and $b[k+1]$. Our approach avoids the need for such explicitly defined boundaries, thereby removing these hyperparameters. Instead, we introduce a single hyperparameter β to balance the loss terms in $\mathcal{L}$.
3. **Preserving within-class diversity:** The RbRL objective encourages all trajectories in a class to have predicted returns close to the midpoint $\frac{b[k]+b[k+1]}{2}$, thereby ignoring intra-class diversity. In contrast, the rMSE objective does not enforce such a constraint, allowing greater flexibility in modeling varied returns within the same class.

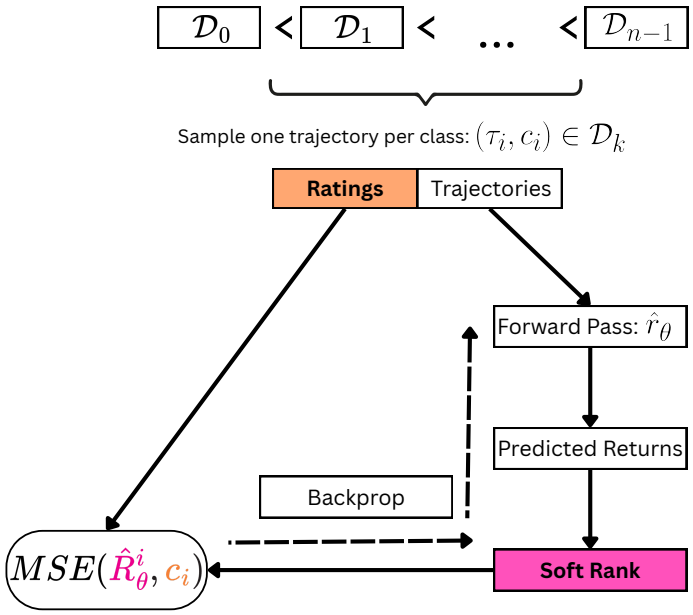


Figure 1: This figure illustrates the learning process of our proposed rMSE method. Given an offline dataset of trajectory–rating pairs, we sample one trajectory from each rating class. Using the current reward model $\hat{r}_\theta$, we compute the predicted return for each trajectory and apply a differentiable sorting algorithm to obtain soft ranks. Finally, we minimize the mean squared error between the soft ranks and the original human-provided ratings.

4.2 Batch Updates

While computing the loss using a single sampled trajectory per class dataset $\mathcal{D}_k$ provides a valid training signal, it can lead to a biased gradient estimate and hinder learning. To improve stability, we perform the soft ranking procedure B times per update step. In each iteration, we sample one trajectory per class dataset, compute predicted returns using $\hat{r}_\theta$, and apply the differentiable sorting algorithm (Blondel et al., 2020) to obtain soft ranks. The resulting B soft rank vectors (of size n) are then stacked to form a stacked soft ranks matrix. Correspondingly, we stack the human-provided class labels associated with each sampled trajectory into a ratings matrix. We compute the rMSE loss as the mean squared error between the stacked soft ranks and the stacked ratings.

5 Experimental Details

In this section, we describe our experimental setup, including the environments used for evaluation, the procedure for collecting trajectories, and the method for generating simulated ratings. We then outline the training and evaluation procedures.

5.1 Environments

To evaluate the effectiveness of rMSE, we consider two environments: the Hungry-Thirsty domain (Barto et al., 2009) and OpenAI Gym's Lunar Lander (Brockman et al., 2016b).

Hungry-Thirsty. This environment consists of a 4×4 grid with food and water randomly placed in the corners. The state space includes the agent's position and two binary indicators for hunger and thirst. The action space comprises six discrete actions: moving in four cardinal directions, eating, and drinking. Hunger occurs if the agent has not eaten in the previous timestep, and thirst arises with 10% probability at each step. The eat action only succeeds if the agent is on the food location and is not thirsty, while the drink action succeeds only at the water location. The agent receives a reward of +1 every timestep it is not hungry. Each episode lasts for 200 timesteps.

Lunar Lander. This classic control task involves landing a spacecraft between two designated flags. The 8-dimensional state space includes the lander's position, velocity, angle, angular velocity, and two ground contact indicators. The action space includes four discrete actions: firing the left, right, or main engine, or taking no action. The environment features a highly engineered reward function composed of multiple handcrafted terms. An episode is considered successful if the total return exceeds 200.

5.2 Offline Trajectories and Ratings

To obtain the offline trajectory dataset for each environment, we train RL agents using the environment's native reward function for 3 seeds. For the Hungry-Thirsty domain, we train a Q-learning agent (Watkins & Dayan, 1992) for 10,000 episodes, storing every 10th episode trajectory. For Lunar Lander, we train a D3QN agent (Wang et al., 2016) with prioritized experience replay (PER) (Schaul et al., 2015) for 1,000 episodes, storing all trajectories. In both cases, we also record the ground-truth return for each trajectory computed using the environment's reward function.

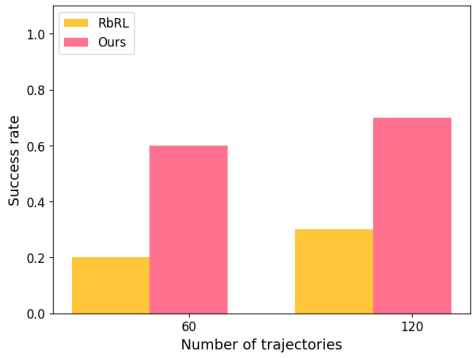
To systematically evaluate performance, we use a simulated teacher that assigns scalar ratings to each trajectory based on its ground-truth return. Specifically, we define a set of return thresholds (or bin boundaries), and each trajectory is labeled according to the return bin it falls into. For the Hungry-Thirsty domain, we use four bins with boundaries defined as $b = [0, 1, 19, 49, 200]$, such that any trajectory τ with return $b[k] \leq G(\tau) < b[k+1]$ is assigned to rating class $c_i = k$. Similarly, trajectories from the Lunar Lander environment are divided into six bins with boundaries: $b = [-300, -200, -100, 0, 100, 200, 300]$. These trajectories, along with their ground-truth rating classes c_i form our data-collection $\mathcal{C}$. From this collection, we construct the dataset used for reward learning. Specifically, we sample N/n trajectories from each rating class and store them in $\mathcal{D}_k$, the subset corresponding to class k. The final dataset is then defined as $\mathcal{D} = \bigcup_{k=0}^{n-1} \mathcal{D}_k$, resulting in a total of N trajectories.

5.3 Training and Evaluation

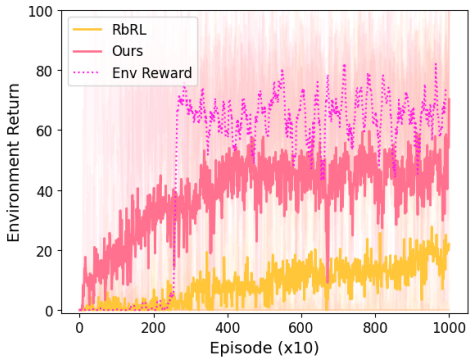
We begin by training the reward models using the offline dataset $\mathcal{D}$. For both environments, we evaluate performance at two dataset sizes: 60 and 120 trajectories (and ratings) for Hungry-Thirsty and 96 and 498 for Lunar Lander. The reward function is approximated using a small neural network, which is updated for U steps using a batch size of B. Details of the network architecture and training hyperparameters are provided in the supplementary material (Section A). To evaluate the effectiveness of rMSE, we compare it against RbRL (White et al., 2024), using the same training procedure for both approaches. Note that for rMSE, batch size B involves sampling B trajectories

from each of the n rating classes, resulting in a total of $B \cdot n$ trajectories per batch. To allow for a fair comparison, we also use a batch size of $B \cdot n$ for the baseline from the same dataset $\mathcal{D}$. Lastly, as an oracle, we consider the environment reward function.

After training the reward model $\hat{r}_\theta$, we train an RL agent on an unseen environment seed using $\hat{r}_\theta$ as the reward (i.e., the evaluation environment seed is held out from reward model training). For the Hungry-Thirsty and Lunar Lander domains, we train a Q-learning agent and a D3QN agent with PER for 10,000 and 4,500 episodes, respectively. However, any discrete action RL algorithm can be used. We repeat this two-stage process — training a new reward model on a freshly sampled dataset $\mathcal{D}$, followed by training a new RL agent using the learned reward — for 10 random seeds to account for variability in both reward learning and policy optimization.

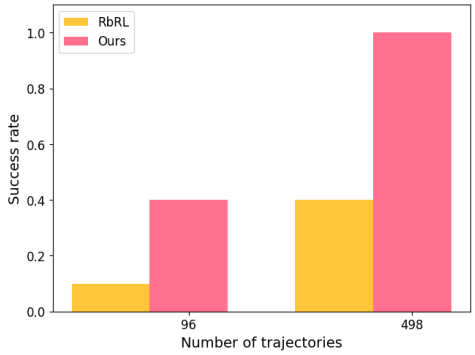


(a) Average success rate for rMSE objective versus RbRL baseline for 60 and 120 trajectories.

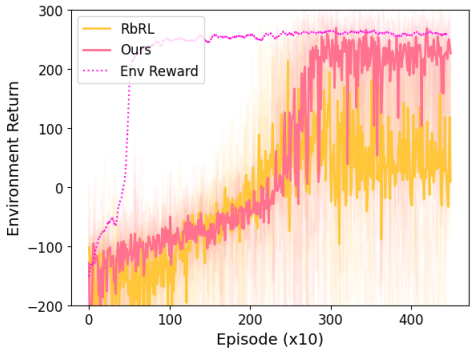


(b) Mean return computed using the environment's reward function versus number of episodes while training a Q-learning agent using: (1) environment reward, (2) rMSE objective, and (3) RbRL baseline for $N = 120$ trajectories.

Figure 2: Results for the Hungry Thirsty domain.



(a) Success rate for rMSE objective versus RbRL baseline for 96 and 498 trajectories.



(b) Median return computed using the environment's reward function versus number of episodes while training a D3QN agent using: (1) environment reward, (2) rMSE objective, and (3) RbRL baseline for $N = 498$ trajectories.

Figure 3: Results for the Lunar Lander environment.

We evaluate the learned reward functions in two ways. First, we assess how well an RL agent can learn from the reward model. The agent's performance is measured using the environment's ground-truth reward function. A learned reward is considered successful if the fully trained agent achieves an average return (computed over 100 evaluation episodes) greater than a predefined threshold. The

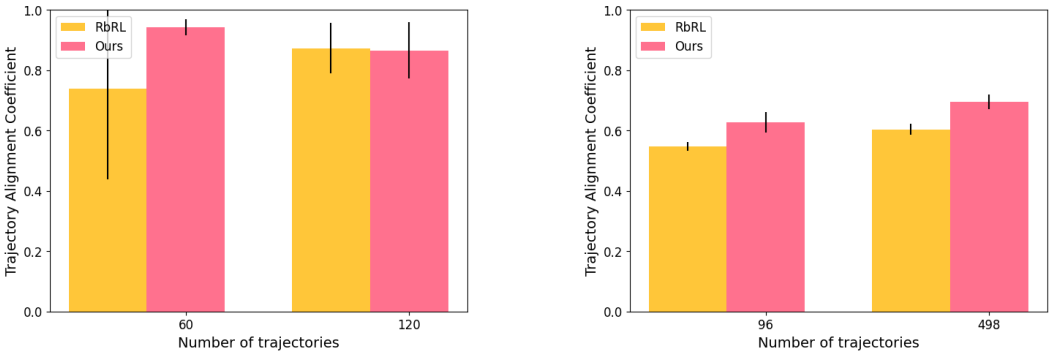
threshold is set to 50 for Hungry-Thirsty and 200 for Lunar Lander. For each method, we report the average success rate across the 10 runs. We also plot the learning curves, showing individual runs in a light color and the median (Lunar Lander) or mean (Hungry-Thirsty) in a darker color. We report the median for Lunar Lander, as there was high variance in the baseline results. The plot for mean performance can be found in the supplementary material, Section B, Figure 5. Second, we evaluate the alignment between the learned reward functions and the environment reward function using the Trajectory Alignment Coefficient (Muslimani et al., 2025b). This metric quantifies the similarity in trajectory rankings induced by two reward functions. Specifically, we use the trajectories from the data collection $\mathcal{C}$, comparing the simulated rankings derived from environment returns to the rankings induced by the learned reward model. A higher Trajectory Alignment Coefficient indicates closer alignment between the learned reward model and the simulated human feedback. We report the average Trajectory Alignment Coefficient (across the 10 runs) along with the standard error.

6 Results

Now that we have described the experimental setup, we will present the results and show that our rMSE objective optimization can outperform the RbRL baseline.

RL Performance: We report the performance of our algorithm compared to the RbRL baseline in Figures 2 and 3. As shown in Figures 2a and 3a, rMSE achieves significantly higher success rates than RbRL, even with substantially fewer trajectories. For example, in the Hungry-Thirsty domain, rMSE trained with only 60 trajectories outperforms RbRL trained with 120. Similarly, in the Lunar Lander environment, rMSE using only 96 trajectories performs on par with RbRL trained on 498. Finally, Figures 2b and 3b illustrate that RL agents trained using reward functions learned via rMSE exhibit faster learning compared to those trained with RbRL.

Reward Alignment: Figure 4 shows that in both domains, our rMSE objective leads to a slight improvement in the Trajectory Alignment Coefficient of the resulting reward functions compared to the RbRL objective. This indicates that, even with a small number of trajectories, rMSE aligns slightly better with the preferences captured in the data collection $\mathcal{C}$.



(a) Hungry Thirsty.

(b) Lunar Lander.

Figure 4: Mean Trajectory Alignment Coefficient $\pm$ standard error for reward functions learned using RbRL and rMSE objectives.

7 Conclusion

In this work, we proposed **rMSE**, a new method for reward learning from multi-class human ratings. Unlike traditional preference-based approaches, rMSE treats ratings as ordinal feedback and optimizes a rank-based mean squared error loss. This enables the reward model to make better use of the rating structure while learning from offline trajectory data. This approach builds on the growing appeal of rating-based reinforcement learning, which offers a promising alternative to pairwise

preferences by possibly reducing cognitive load and enabling richer supervision. Ratings allow humans to evaluate behaviors individually and express both relative and absolute judgments of quality. Through experiments on the Hungry-Thirsty and Lunar Lander environments, we showed that rMSE outperforms the existing rating-based RL baseline, producing reward models that lead to more performant policies and better alignment with the underlying feedback. In future work, we plan to extend our method to an online setting, where human ratings are collected in real time and used to iteratively improve both the reward model and the RL policy. Additionally, we aim to conduct a human subject study to evaluate the effectiveness of our approach using real human feedback.

8 Acknowledgments

Part of this work has taken place in the Intelligent Robot Learning Lab at the University of Alberta, which is supported in part by research grants from Alberta Innovates; Alberta Machine Intelligence Institute (Amii); a Canada CIFAR AI Chair, Amii; Digital Research Alliance of Canada; Mitacs; and the National Science and Engineering Research Council.

References

Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.

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

Andrew G. Barto, Richard L. Lewis, and Satinder Singh. Where do rewards come from. 2009. URL https://api.semanticscholar.org/CorpusID:14951500.

Mathieu Blondel, Olivier Teboul, Quentin Berthet, and Josip Djolonga. Fast differentiable sorting and ranking. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

Serena Booth, W. Bradley Knox, Julie Shah, Scott Niekum, Peter Stone, and Alessandro Allievi. The Perils of Trial-and-Error Reward Design: Misdesign through Overfitting and Invalid Task Specifications. In *AAAI Conference on Artificial Intelligence*, 2023.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. *arXiv:1606.01540 [cs]*, June 2016a. arXiv: 1606.01540.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016b. URL http://arxiv.org/abs/1606.01540. cite arxiv:1606.01540.

Daniel Brown, Wonjoon Goo, Prabhat Nagarajan, and Scott Niekum. Extrapolating Beyond Suboptimal Demonstrations via Inverse Reinforcement Learning from Observations . In *International Conference on Machine Learning*, 2019.

Serkan Cabi, Sergio Gómez Colmenarejo, Alexander Novikov, Ksenia Konyushkova, Scott Reed, Rae Jeong, Konrad Zołna, Yusuf Aytar, David Budden, Mel Vecerik, Oleg Sushkov, David Barker, Jonathan Scholz, Misha Denil, Nando de Freitas, and Ziyu Wang. Scaling data-driven robotics with reward sketching and batch reinforcement learning. In *Robotics: Science and Systems*, 2020.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep Reinforcement Learning from Human Preferences. In *Neural Information Processing Systems*, 2017a.

Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *International Conference on Neural Information Processing Systems*, 2017b.

Xiao Hu, Jianxiong Li, Xianyuan Zhan, Qing-Shan Jia, and Ya-Qin Zhang. Query-policy misalignment in preference-based reinforcement learning. In *International Conference on Learning Representations*, 2024.

W. Bradley Knox and James MacGlashan. How to Specify Reinforcement Learning Objectives. In *Finding the Frame: An RLC Workshop for Examining Conceptual Frameworks at the Reinforcement Learning Conference*, 2024.

W Bradley Knox and Peter Stone. Interactively Shaping Agents via Human Reinforcement. In *International Conference on Knowledge Capture*, 2009.

W. Bradley Knox, Alessandro Allievi, Holger Banzhaf, Felix Schmitt, and Peter Stone. Reward (Mis)design for Autonomous Driving. *Artificial Intelligence*, 316(103829), 2023.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Kimin Lee, Laura Smith, and Pieter Abbeel. PEBBLE: Feedback-Efficient Interactive Reinforcement Learning via Relabeling Experience and Unsupervised Pre-training . In *International Conference on Machine Learning*, 2021.

Jinning Li, Chen Tang, Masayoshi Tomizuka, and Wei Zhan. Dealing with the unknown: Pessimistic offline reinforcement learning. In *Conference on Robot Learning*, 2021. URL https://api.semanticscholar.org/CorpusID:237263626.

Xinran Liang, Katherine Shu, Kimin Lee, and Pieter Abbeel. Reward Uncertainty for Exploration in Preference-based Reinforcement Learning. In *International Conference on Learning Representations*, 2022.

Calarina Muslimani and Matthew E. Taylor. Leveraging sub-optimal data for human-in-the-loop reinforcement learning. In *International Conference on Learning Representations*, 2025.

Calarina Muslimani, Bram Grooten, Deepak Ranganatha Sastry Mamillapalli, Mykola Pechenizkiy, Decebal Constantin Mocanu, and Matthew E. Taylor. Boosting robustness in preference-based reinforcement learning with dynamic sparsity. In *International Conference on Autonomous Agents and Multiagent Systems (Extended Abstract)*, 2025a.

Calarina Muslimani, Kerrick Johnstonbaugh, Suyog Chandramouli, Serena Booth, W Bradley Knox, and Matthew E. Taylor. Towards improving reward design in rl: A reward alignment metric for rl practitioners. In *Reinforcement Learning Conference*, 2025b.

Andrew Y. Ng and Stuart Russell. Algorithms for inverse reinforement learning. In *International Conference on Machine Learning*, 2000.

OpenAI. GPT-4 Technical Report. 2023. URL: https://arxiv.org/abs/2303.08774.

Alexander Pan, Kush Bhatia, and Jacob Steinhardt. The Effects of Reward Misspecification: Mapping and Mitigating Misaligned Models. In *International Conference on Learning Representations*, 2022.

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.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *CoRR*, abs/1511.05952, 2015. URL https://api.semanticscholar.org/CorpusID:13022595.

Kyriacos Shiarlis, Joao Messias, and Shimon Whiteson. Inverse Reinforcement Learning from Failure. In *International Conference on Autonomous Agents and Multiagent Systems*, 2016.

Joar Skalse, Nikolaus Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and Characterizing Reward Gaming. In *Neural Information Processing Systems*, 2022.

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Freitas. Dueling network architectures for deep reinforcement learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pp. 1995–2003. JMLR.org, 2016.

Garrett Warnell, Nicholas Waytowich, Vernon Lawhern, and Peter Stone. Deep TAMER: Interactive Agent Shaping in High-Dimensional State Spaces. In *AAAI Conference on Artificial Intelligence*, 2018.

Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3):279–292, May 1992. ISSN 1573-0565. DOI: 10.1007/BF00992698. URL https://doi.org/10.1007/BF00992698.

Devin White, Mingkang Wu, Ellen Novoseller, Vernon J Lawhern, Nicholas Waytowich, and Yongcan Cao. Rating-based Reinforcement Learning. In *AAAI Conference on Artificial Intelligence*, 2024.

Nils Wilde, Erdem Bıyık, Dorsa Sadigh, and Stephen L Smith. Learning Reward Functions from Scale Feedback. In *Conference on Robot Learning*, 2021.

Supplementary Materials

The following content was not necessarily subject to peer review.

A Neural Network Architecture and Hyperparameters

We use two neural network architectures for learning the reward functions. The **Mini** model consists of one hidden layer with 5 neurons and ReLU activation Agarap (2018). The output layer uses a Tanh activation. The **Medium** model has two hidden layers with 10 neurons each, with ReLU activation applied after each layer.

Hyperparameter	Hungry Thirsty $N = 60$	Hungry Thirsty $N = 120$	Lunar Lander $N = 96$	Lunar Lander $N = 498$
Reward Training Hyperparameters				
β	0.1			
Model	Mini	Mini	Medium	Medium
B	15	30	16	64
U	10000	10000	3000	10000
D3QN Training Hyperparameters				
lr	0.0001			
γ	0.99			
Replay size	10,000			
Batch size	128			
τ	0.01			
priority α	0.6			
priority β	0.4			
priority ϵ	0.01			
priority increment	0.00001			

Table 1: Hyperparameters

B Supplementary Figures

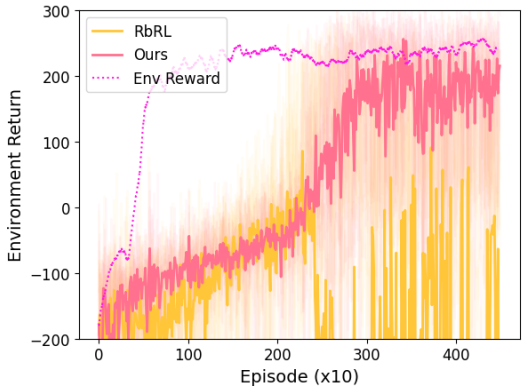


Figure 5: Average return computed using the environment's reward function versus number of episodes while training a D3QN agent using: (1) environment reward, (2) **rMSE** objective, and (3) **RbRL** baseline for $N = 498$ trajectories.