The Distributional Reward Critic Framework for Reinforcement Learning Under Perturbed Rewards

Xi Chen, Zhihui Zhu, Andrew Perrault

The Ohio State Univeristy
chen.10183@osu.edu, zhu.3440@osu.edu, perrault.17@osu.edu

Abstract

The reward signal plays a central role in defining the desired behaviors of agents in reinforcement learning (RL). Rewards collected from realistic environments could be perturbed, corrupted, or noisy due to an adversary, sensor error, or because they come from subjective human feedback. Thus, it is important to construct agents that can learn under such rewards. Existing methodologies for this problem make strong assumptions, including that the perturbation is known in advance, clean rewards are accessible, or that the perturbation preserves the optimal policy. We study a new, more general, class of unknown perturbations, and introduce a distributional reward critic framework for estimating reward distributions and perturbations during training. Our proposed methods are compatible with any RL algorithm. Despite their increased generality, we show that they achieve comparable or better rewards than existing methods in a variety of environments, including those with clean rewards. Under the challenging and generalized perturbations we study, we win/tie the highest return in 44/48 tested settings (compared to 11/48 for the best baseline). Our results broaden and deepen our ability to perform RL in reward-perturbed environments. If necessary, please check the full paper (https://arxiv.org/abs/2401.05710), including the Appendix.

Code — https://github.com/cx441000319/DRC

1 Introduction

The use of reward as an objective is a central feature of reinforcement learning (RL) that has been hypothesized to constitute a path to general intelligence (Silver et al. 2021). The reward is also the cause of a substantial amount of human effort associated with RL, from engineering to reduce difficulties caused by sparse, delayed, or misspecified rewards (Ng, Harada, and Russell 1999; Hadfield-Menell et al. 2017; Qian, Weng, and Tan 2023) to gathering large volumes of human-labeled rewards used for tuning large language models (LLMs) (Ouyang et al. 2022; Bai et al. 2022). Thus, the ability of RL algorithms to tolerate noisy, perturbed, or corrupted rewards is of general interest (Romoff et al. 2018; Wang, Liu, and Li 2020; Everitt et al. 2017; Moreno et al. 2006; Corazza, Gavran, and Neider 2022; Zheng, Liu, and Ni 2014).

Method	Required Info			Reward Estimator			Perturbation
	$\tilde{R}$	n_r	Reward Range	Discrete Rewards	Leverages (s,a,s')	Reduces Variance	Optimal-Policy-Changing
Standard	×	×	×	×	×	×	×
SR_W	✓	✓	✓	✓	×	×	✓
SR	×	✓	✓	✓	×	×	✓
RE	×	×	×	×	✓	✓	×
DRC	×	✓	✓	×	✓	✓	✓
GDRC	×	×	×	×	✓	✓	✓

Table 1: The methods proposed in this paper (in bold) have key advantages relative to past methods, which are limited by the amount of Required Info about the perturbation that is needed, the structural properties of the environment such as discrete rewards, and assumptions on the effect of the perturbation on the optimal policy. Leveraging (s, a, s') for reward estimation and variance reduction (of the estimated reward vs. true reward) are advantages of methods that construct explicit perturbation models. Blue is an advantage, and orange is a limitation.

Different reward correction methods have been proposed to enhance the performance of RL algorithms under reward perturbations by estimating the expectation of perturbed rewards (Romoff et al. 2018) or learning the perturbation model (Wang, Liu, and Li 2020). However, these existing approaches rely on strong assumptions. For instance, the Reward Estimation (RE) method (Romoff et al. 2018) assumes that the perturbation does not impact the optimal policy, a condition satisfied in limited cases, such as when the reward undergoes a positive affine transformation. On the other hand, the Surrogate Reward (SR) method (Wang, Liu, and Li 2020) can handle perturbations beyond affine transformations, but it presupposes that there are a finite number of possible reward values and a specific perturbation structure (see Section 2 for a more detailed discussion). Thus, the methods are individually limited in where they can be applied, and no method exists for certain problem-perturbation combinations (i.e., continuous reward environments where

the optimal policy is altered). Indeed, it is not clear that such a method can be constructed due to the structural difference between perturbations on a discrete-valued reward (e.g., binary success/failure) and those on a continuous-valued reward (e.g., velocity).

We show that an adaptive reward modeling approach, with appropriate design choices, can be used to unify the two scenarios, creating a universal framework for learning under perturbed rewards. We propose a family of *distributional* reward critic methods that redefine the problem of reward estimation as a classification task and infer the unknown perturbation structures without prior knowledge. Our contributions (Table 1) are summarized in three parts as follows.

The Distributional Reward Critic (DRC) Framework To recover from a changing optimal policy on the perturbed rewards, we begin by turning the reward regression into a classification problem using ordinal cross-entropy for estimating the reward distribution under the assumption of knowing the structure of the perturbation, which we call Generalized Confusion Matrix (GCM) perturbations. GCM perturbations generalize past reward perturbation distributions studied in the literature and allow for perturbations on both discrete- and continuous-valued rewards that alter the optimal policy, and, in addition, can approximate any distribution. However, DRC is limited by its assumed foreknowledge of the structural properties of the GCM perturbation.

General Distributional Reward Critic (GDRC) We devise a General Distributional Reward Critic (GDRC) method to simultaneously estimate the structure of the GCM during training. GDRC works by training an ensemble of DRCs and using the metrics from the training process to intelligently select the most credible DRC. We show in simulation that GDRC is effective at inferring the structure of unknown GCM perturbations.

Experimental Evaluation Using an array of tasks and reward perturbation distributions, we compare DRC and GDRC to state-of-the-art methods for perturbed reward reinforcement learning. Under GCM perturbations, we win/tie (95% of the winner) the highest return in 44/48 sets (compared to 11/48 for the best baseline). Even under continuous perturbations with strict assumptions, we find that our methods are on par with existing ones that leverage strong assumptions about perturbations. Together, our results imply that GDRC is a strong tool for learning under broad perturbations. Moreover, we set up a series of studies to verify every decision we make when developing our methods.

2 Related Work

Reward perturbations in RL have been extensively studied (Romoff et al. 2018; Wang, Liu, and Li 2020; Rakhsha et al. 2020; Pattanaik et al. 2017; Pinto et al. 2017; Choromanski et al. 2020; Zhong, Wu, and Si 2023; Corazza, Gavran, and Neider 2022; Zhuang and Sui 2021; Hu et al. 2022; Ring and Orseau 2011; Hutter 2005; Amodei et al. 2016). We describe two of the most related approaches (Romoff et al. 2018; Wang, Liu, and Li 2020) in detail.

Reward Prediction By The Reward Estimation (RE) Method (Romoff et al. 2018) focus on variance reduction in the case of continuous perturbations that increase reward variance but do not change the optimal policy. This occurs, for example, when the reward perturbation applies a positive affine transformation, i.e., $\mathbb{E}[\tilde{R}(s,a)] = \omega_0 R(s,a) + \omega_1$ and $\omega_0 > 0$. In this setting, the perturbations can slow down the training and even destroy it when the critic is not trained with enough samples. They propose the Reward Estimation (RE) method by introducing a reward critic to predict rewards $\hat{R}(s,a)$ that aims to reduce the variance.

Perturbation Modeling By The Surrogate Reward (SR) Method (Wang, Liu, and Li 2020) study the setting where the rewards are discrete and perturbed by a confusion matrix C, where $C(i,j)$ represents the probability that reward R_i is perturbed into R_j. Wang et al. propose the Surrogate Reward (SR) method by inverting the confusion matrix $\hat{R} = C^{-1} \cdot R$, where vectorized R and $\hat{R}$ represent clean and predicted rewards, replacing each observed reward R_i with an unbiased estimate of the true reward $\hat{R}_i$. As their method assumes C to be estimated separately, they cannot leverage the structure of the reward signal in state-action-state space (i.e., that similar state-action pairs may have similar rewards.

In brief, our method addresses the limitations of both RE and SR, by handling a broader class of perturbations without assuming foreknowledge of its structure.

Distributional RL Our approaches are inspired by distributional RL (Dabney et al. 2018b; Bellemare, Dabney, and Munos 2017; Dabney et al. 2018a; Rowland et al. 2018), where the value function is modeled distributionally. We find that fixed-width regression (Bellemare, Dabney, and Munos 2017) is more appropriate than fixed-ratio for reward modeling for technical reasons. However, unlike standard fixed-width methods, we control the size and location of the fixed-width bins adaptively (which are standardly treated as hyperparameters in distributional RL). The result is a generic method that does not require tuning hyperparameters across tasks or perturbations, which is commonly a weakness of fixed-width compared to fixed-ratio methods.

3 Problem Statement

In Section 3.1, we describe the standard extension of MDPs to perturbed rewards. In Section 3.2, we introduce generalized confusion matrix (GCM) perturbations and show their significance and universality.

3.1 Perturbed Reward MDP

Let $\langle S, A, R, P, \gamma, \beta \rangle$ be a Markov Decision Process (MDP) (Puterman 2014), where S is the state space, A is the action space, $R : S \times A \times S \to \mathbb{R}$ is the reward function, $P : S \times A \to \Delta S$ is the transition function, $\beta \in \Delta S$ is the initial state distribution, and $\gamma \in [0,1]$ is the discount factor. We denote the state at timestep t as s_t, the action as a_t, and the reward as $r_t = R(s_t, a_t, s_{t+1})$.

We define a *perturbed reward MDP* of the form $\left\langle S, A, R, \tilde{R}, P, \gamma, d_0 \right\rangle$. The agent perceives perturbed re-

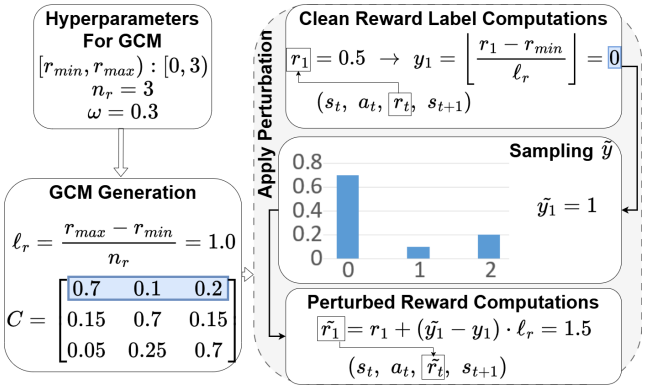


Figure 1: An example of GCM perturbation.

wards from $\tilde{R}$ instead of true rewards from R. Following the prior work (Romoff et al. 2018; Wang, Liu, and Li 2020), we assume that the perturbed reward is random and that its distribution depends on the true reward, i.e., $\tilde{r}_t \sim \tilde{R}(R(s_t, a_t, s_{t+1}))$. The objective is the same as in a standard MDP: we seek a policy $\pi : S \rightarrow \Delta A$ that maximizes the return, i.e., $\pi^* \in \arg\max_\pi \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{s+1})]$, but we only receive perturbed rewards.

3.2 Generalized Confusion Matrix (GCM) Perturbations

To effectively handle perturbed rewards, we rely on an appropriate model to capture the perturbations. The Surrogate Reward (SR) method (Wang, Liu, and Li 2020) models the reward perturbation by perturbing discrete rewards with a confusion matrix C, where $C(i, j)$ represents the probability that reward R_i is perturbed into R_j. We develop a perturbation model that preserves the continuous nature of the reward space. Specifically, we propose a generalized confusion matrix (GCM) perturbation model and provide a motivating example in Fig. 1. The parameters of a GCM are the reward interval length $\ell_r = (r_{\max} - r_{\min})/n_r$, the minimum and maximum rewards $r_{\min}$ and $r_{\max}$, and the associated confusion matrix C. To sample from a GCM with input reward r_t, we convert r_t into its reward label y_t with $y_t = \lfloor (r_t - r_{min})/l_r \rfloor$. Then, we use the probability distribution $C(y_t, \cdot)$ to sample perturbed label $\tilde{y}_t$ and shift r_t by the signed distance between y_t and $\tilde{y}_t$, i.e., $\tilde{r}_t = r_t + \ell_r(\tilde{y}_t - y_t)$.

We study GCM perturbations for three reasons. First, GCM perturbations naturally handle continuous rewards without sparsifying the perturbed rewards (i.e., the perturbed rewards are themselves continuous). Second, GCM can represent perturbations that can change the optimal policy, i.e., the optimal policy for the perturbed rewards is different than the optimal policy for the clean rewards. This expressivity comes from their ability to represent perturbations with arbitrary probability density functions (PDF) for each reward interval—each row is a perturbation PDF. Third, GCMs can approximate any continuous perturbation with a bounded error that diminishes as reward interval number n_r as shown in Proposition 1, whose proof is in the Appendix.

Proposition 1. *Consider continuous perturbations that for each reward $r \in [r_{\min}, r_{\max})$, it can be perturbed to $\bar{r} \in [r_{\min}, r_{\max})$. Our GCM represents $\bar{r}$ with $\tilde{r}$ that satisfies $|\tilde{r} - \bar{r}| \leq \frac{r_{\max} - r_{\min}}{n_r}$.*

We now turn our attention to developing a method that can learn effectively under a GCM-perturbed reward signal.

4 The Distributional Reward Critic Architecture

We begin in Sec. 4.1 with the simpler case where the GCM interval length, number of intervals, and minimum and maximum are all known (i.e., all the GCM parameters except for the confusion matrix C). Then, we study the more general case where these parameters must be inferred during training in Sec. 4.2.

4.1 Distributional Reward Critic (DRC)

To learn effectively under a GCM perturbation, we want to introduce a network to model the reward distribution for each state-action-state. We view the reward estimation as a classification problem by turning the reward range $[r_{\min}, r_{\max}]$ into n_r intervals. Therefore, the problem we want to resolve could be expressed as: given input $(s_t, a_t, s_{t+1}, \tilde{r}_t)$, predict the distribution of label $\tilde{y}_t = \text{floor}\big((\tilde{r}_t - r_{\min})/\ell_r\big)$ where $\ell_r = (r_{\max} - r_{\min})/n_r$ is the reward interval length. We do so by training a classification model $\hat{R}_\theta : S \times A \times S \rightarrow \Delta Y$ where $Y = (0, 1, \ldots, n_r - 1)$.

When we train a classification model like $\hat{R}_\theta$, the most common loss is cross-entropy (CE). However, CE discards information about the order of reward labels. This is especially critical in RL because, as the reward distribution shifts during training, it is critical to be able to quickly identify that certain rare reward values are superior to those seen thus far. When CE is used as a loss function, $\hat{R}_\theta$ tends to classify unseen samples as belonging to the most common class that is seen thus far (as the loss incurred by predicting any incorrect class is equivalent). We thus propose ordinal cross-entropy (OCE) instead[1]:

$$OCE = \sum_t (1 + \frac{|\hat{y}_t - \tilde{y}_t|}{n_r - 1}) \cdot H(\tilde{y}_t, \hat{R}_\theta(s_t, a_t, s_{t+1})), \quad \text{where } \hat{y}_t = \arg\max_{y \in Y} \hat{R}_\theta(s_t, a_t, s_{t+1}) \tag{1}$$

Using OCE as the loss function, the order among rewards is preserved naturally when we are in the discrete label space. In addition, when the observed reward label $\tilde{y}_t$ is far from the predicted reward label $\hat{y}_t$, the cross-entropy term $H(\tilde{y}_t, \hat{R}_\theta(s_t, a_t, s_{t+1}))$ will be assigned more weight based on their distance. Therefore, misclassified rare samples can receive a large weight during training if the distance is large.

We propose our Distributional Reward Critic (DRC) method in Fig. 2. DRC estimates the reward distribution online during RL training using OCE as the loss. The rewards that are received by the RL algorithm are replaced with the current predictions of the reward critic after they are updated based on the rewards observed for the current epoch. In experiments, we see that DRC is an effective approach across

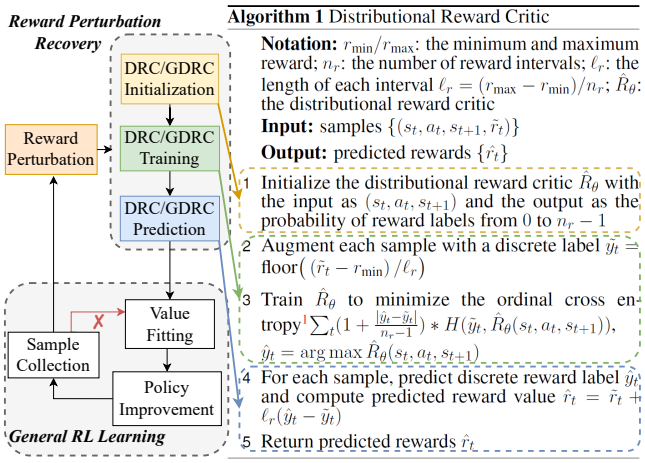


Figure 2: The pipeline of the whole process and the pseudocode of distributional reward methods.

environments and that the use of OCE rather than CE is a critical choice.

However, DRC will only be practically applicable in environments with a known structure of GCM. Next, we study the simultaneous estimation of all GCM parameters.

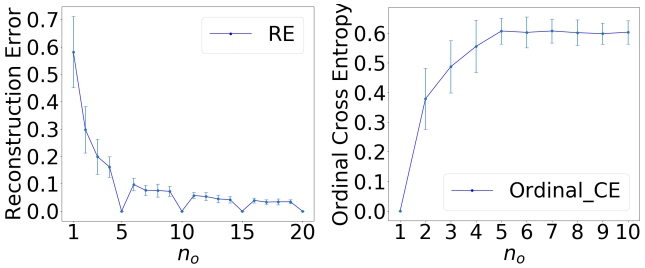
4.2 General Distributional Reward Critic (GDRC)

In this section, we first study the influence of the discretization parameter n_o of DRC on the reward prediction error, an unobservable metric termed Reconstruction Error, which provides insights into what n_o is preferred. Then, we study how to use the observable ordinary cross-entropy loss to guide the selection of n_o. At last, we show how to estimate the reward range online while allowing for reward distribution shifts during training.

Impact Of n_o On Reconstruction Error For DRC Depending on the number of bins n_o used in the reward critic, it may be impossible to recover the true reward even in the infinite sample case. We define *Reconstruction Error* as $\text{ERROR}_r(\hat{R}_\theta) = \frac{1}{|T|}\sum_{t\in T}|\hat{r}_t - r_t|$, where T represents the number of samples. Reconstruction Error prevents exact recovery of the reward when $n_o \neq a \cdot n_r (a \in \mathbb{Z}^+)$ as there is now an irreducible source of error caused by the misalignment of the intervals in the reward critic compared to the perturbation.

To illustrate Reconstruction Error intuitively, we divide the range $[r_{\min}, r_{\max}]$ into n_o intervals of length ℓ_o. Given an observed reward $\tilde{r} = r + \ell_r(\tilde{y} - y)$ perturbed from r, we compute its predicted reward $\hat{r} = \tilde{r} + \ell_o(\hat{y}_o - \tilde{y}_o)$, where $\hat{y}_o = \arg\max_{y_o \in Y_o} \hat{R}_\theta(s, a, s')$. Even if $\hat{R}_\theta$ is sufficiently expressive and trained well, $\hat{r}$ is not a correct prediction of r due to the misalignment between $\ell_r(\tilde{y} - y)$ and $\ell_o(\hat{y}_o - \tilde{y}_o)$.

[1]With slight abuse of notation, $H(\tilde{y}_t, \hat{R}_\theta(s_t, a_t, s_{t+1}))$ denotes the cross entropy between $\hat{R}_\theta(s_t, a_t, s_{t+1})$ and the distribution with all zero probabilities except for the $\tilde{y}_t$-th being 1 (i.e., the one-hot vector version of $\tilde{y}_t$).



(a) The reconstruction error initially decreases as n_o increases, reaches 0 at $n_o = n_r$, and then oscillates.

(b) The minimum ordinal cross-entropy of the reward critic increases as n_o increases until n_o reaches n_r.

Figure 3: Illustration of reconstruction error and cross entropy as n_o varies in simulation environments where $n_r = 5$.

In general, when $n_o < n_r$, Reconstruction Error becomes more pronounced as n_o decreases because of the large granularity of ℓ_o. When $n_o > n_r$, Reconstruction Error still exists except for the case that n_o is a multiple of n_r, but is less significant. Fig. 3a shows the impact n_o on Reconstruction Error when a large number of samples are available for training. A detailed discussion is in the Appendix.

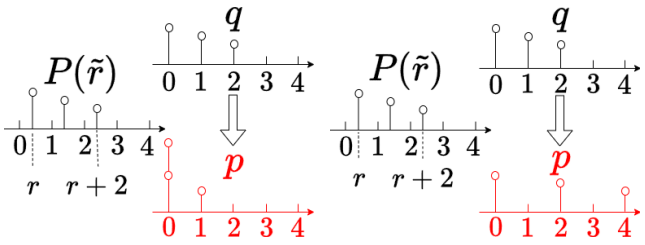
With infinite samples, a large n_o is preferred to achieve a small Reconstruction Error. Without this condition, there is a tradeoff—a larger n_o leads to more overfitting because of limited samples, but a small n_o results in more Reconstruction Error. We aim to set $n_o = n_r$ as it achieves zero Reconstruction Error and requires the least samples, and we show the ordinal cross-entropy is an accessible metric to help infer it in the next part.

Leveraging The Ordinal Cross-Entropy To Select n_o During training, we can view the ordinal cross-entropy (OCE) loss of the distributional reward critic. We turn our attention to the dynamics of this loss as n_o changes, and we will show that it can be used to estimate n_r.

We begin with a motivating example of intuition. For simplicity, we study the relation between cross-entropy (CE) and n_o because the best prediction of the reward critic would be the reward label distribution using CE, and that relation also applies to OCE.

For example, $n_r = 3$, $[r_{\min}, r_{\max}) = [0, 3)$, and we have many samples $(s_t, a_t, s_{t+1}, \tilde{r}_t)$ whose true rewards are all the same value $r \in [0, 1)$. We define the probability of their perturbed rewards $\tilde{r}$ as $\mathbb{P}(\tilde{r})$, where $\tilde{r} = r + m$, $m \in (0, 1, 2)$ and $\mathbb{P}(\tilde{r}) = C(0, \cdot)$. The probability of perturbed reward labels discretized by n_r intervals is denoted as q, where $q_m = \mathbb{P}(\tilde{r} = r + m)$. Then we use n_o intervals to fit q, whose reward label distribution is represented by p. Here, the question turns into how the entropy $H(p)$ changes as n_o changes. There are two circumstances shown in Fig. 4:

- When $n_o < n_r$, more q_m get combined together for smaller n_o, resulting in a smaller entropy $H(p) = -\sum_{k=1}^{n_o} p_k \log p_k$. For instance, combing $q_0, q_1 > 0$ together leads to a reduction as $-(q_0 + q_1)\log(q_0 + q_1) < -(q_0 \log q_0 + q_1 \log q_1)$.



(a) How p fits q when $n_o < n_r$ (b) How p fits q when $n_o \geqslant n_r$

Figure 4: Illustration of cross-entropy for different n_o.

- When $n_o > n_r$, there are n_r dimensions with non-zero probabilities and $n_o - n_r$ dimensions with zero probabilities. Hence, $H(p) = -\sum_{k=1}^{n_o} p_k \log p_k = -\sum_{m=1}^{n_r} q_m \log q_m$ remains constant with respect to n_o

The combination and the separation of impulses with CE are also the case for OCE. Therefore, our intuition is that OCE also increases first when $n_o < n_r$ and then stays still when $n_o \geq n_r$. We verify the intuition by running a simple simulation test where we use a large number of samples to predict the perturbed reward label distribution q using OCE in the cases $n_r = 5$ and $\omega = 0.1$. The result for each n_o is averaged over 20 trails with different perturbed reward label distributions. As shown in Fig. 3b, the minimum OCE first increases when $n_o < n_r$ until it reaches n_r, which verifies our intuition above.

We run experiments using our Distributional Reward Critic (DRC) method with different n_o to confirm the relation between OCE and n_o experimentally. According to the results in Fig. 7b, we can clearly tell with the same TotalEnvInteracts (training steps), OCE first increases as n_o increases and then oscillates when $n_o \geq n_r$.

Based on our analysis, we conclude there is a significant relation between OCE and n_o: OCE first increases as n_o increases and then oscillates when $n_o \geq n_r$. As we aim to set n_o to be n_r to achieve the smallest Reconstruction Error, we propose our General Distributional Reward Critic (GDRC) method, which trains an ensemble of reward critics $\left\{\hat{R}_\theta^{(n_o)}\right\}$ with uniform perturbation discretizations $n_o \in N_o$ to identify n_r by referring to ordinal cross-entropy. We use these reward critics to vote on where the increasing rate of ordinal cross-entropy starts increasing. We use the critic who has received the most votes with a discount factor for reward prediction in each epoch.

Handling The Unknown Reward Range For the case of an unknown reward range and an unknown number of intervals, we use the GDRC from the previous part plus an addition that updates the intervals based on the observed rewards seen in the latest 20 epochs. We create variables r_{emin} and r_{emax} to store the 1% percentile and 99% of the observed rewards across the samples in the latest 20 epochs as the minimum and maximum rewards, which excludes the possible influence of outlier perturbed rewards because of long-tail perturbations. This choice turns out to be important—keeping samples for too long slows training under reward distribution shift. We study the impact of this choice in Sec. 5.5.

We provide pseudocode for GDRC in the Appendix.

5 Experimental Results

In this section, we demonstrate that DRC and GDRC methods outperform existing approaches by attaining higher clean rewards and exhibiting broader applicability. Sec. 5.1 introduces the setups first. In Sec. 5.2 and 5.6, we experiment under Generalized Confusion Matrix (GCM) and continuous perturbations. The influence of n_o on Reconstruction Error and ordinal cross-entropy is substantiated in Sec. 5.3. In Sec. 5.4 and 5.5, we do ablation studies to the critical decisions while developing DRC and GDRC. We include hyperparameters and training curves in the Appendix.

5.1 Experimental Setup

Algorithms The methods introduced for perturbed rewards in this paper and previous work can be applied to any RL algorithm. Thus, we compare all methods as applied to some popular algorithms such as PPO (Schulman et al. 2017), DDPG (Lillicrap et al. 2015), and DQN (Mnih et al. 2013), covering on-policy and off-policy algorithms. The baseline methods other than the original algorithms are the Reward Estimation (RE) (Romoff et al. 2018) method and Surrogate Reward (SR/SR_W) (Wang, Liu, and Li 2020) methods (SR estimates the confusion matrix and SR_W receives the true confusion matrix as input). Our proposed methods are Distributional Reward Critic **(DRC)** and General Distributional Reward Critic **(GDRC)**.

Environments First, we consider complex Mujoco environments: Hopper, HalfCheetah, Walker2d, and Reacher (Todorov, Erez, and Tassa 2012) (the environments tested by RE), where SR cannot be applied due to the complexity of the state-action space, but our methods can, demonstrating their broader applicability. Then we consider two discrete control tasks, Pendulum and CartPole (the environments tested by SR and SR_W), and show that our performance is also dominant in these settings.

Perturbations We test two kinds of perturbations: Generalized Confusion Matrix (GCM) and continuous perturbations. For GCM perturbations, we vary two parameters: the number of intervals n_r and the perturbation ratio ω. An ω proportion of samples in the interval containing the true reward are perturbed into other intervals at random. For continuous perturbations, we test the same perturbation as (Romoff et al. 2018). These are zero-mean additive Gaussian noise, the "Uniform" perturbation where the reward is sampled uniformly from $[-1, 1]$ with a probability of ω and is unaltered otherwise, and the "Reward Uniform" perturbation, adjusting the range of the uniform distribution to r_{min} and r_{max} (the minimum and maximum reward achievable in each environment). Each method has its episodic reward averaged over 50 and 20 seeds under GCM and continuous perturbations respectively, and +/- shows the standard error in the Appendix.

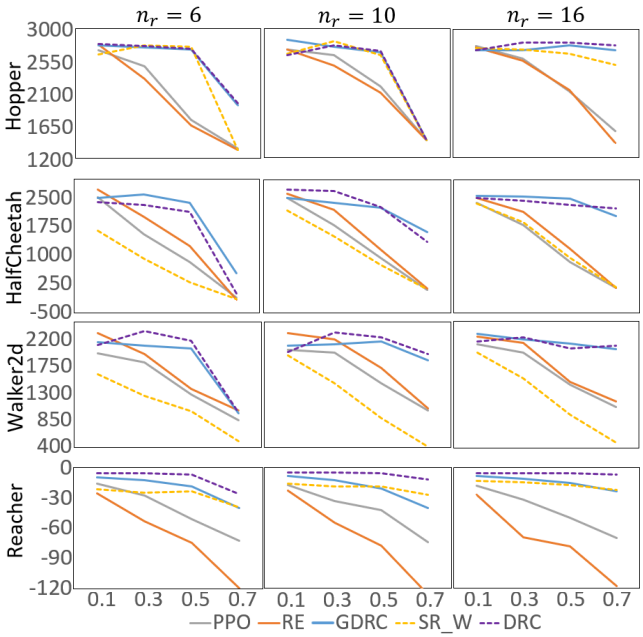


Figure 5: The results of Mujoco environments under GCM perturbations. Solid line methods, which are of greater interest, can be applied without any prior information. **DRC** and **GDRC** are our methods. The x-axis represents perturbation ratio ω, and the y-axis represents performance.

5.2 Under GCM Perturbations

Mujoco Environments There are two kinds of methods in Fig. 5, the methods (dashed lines) requiring prior information about perturbations and the ones (solid lines) without any prior knowledge, leading to two comparison groups: DRC vs. PPO, RE, and SR_W, and GDRC vs. PPO and RE. Overall, we can clearly see the outperformance of our DRC and GDRC compared with their baseline methods. To summarize, DRC wins/ties (95% of the winning performance) the best performance in 44/48 sets (compared to 11/48 for the best baseline RE), and GDRC wins/ties in 44/48 sets (compared to 12/44 for the best baseline RE), of which the second comparison is our focus. Both DRC and GDRC demonstrate comprehensive robustness across environments, n_r, and ω.

All methods perform worse as we increase ω. Other than that, varying n_r also influences their performance. There is a structural reason for this: GCMs with smaller n_r have less evenly distributed rewards outside the clean interval, making it harder to denoise the signal.

SR_W performs well in Hopper and Reacher, but worse than PPO in the HalfCheetah and Walker2d. This is perhaps surprising as it has access to the ground truth reward perturbation. We hypothesize two reasons why it can be beaten. First, the estimate of reward it provides $\hat{R} = C^{-1} \cdot R$ is conditioned only on the observed reward. This means it is not able to do additional denoising using (s, a, s'). Second, they introduce a hyperparameter rescaling the estimates, which is not tunable and has a large impact on performance across environments.

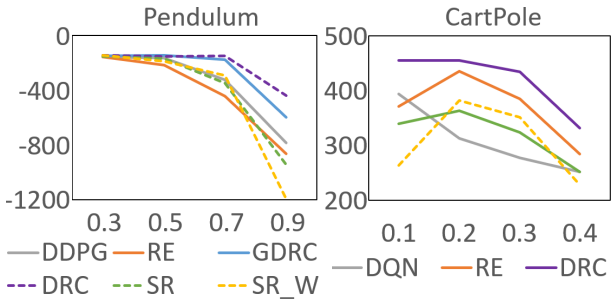


Figure 6: The results of discrete control tasks. Solid line methods, which are of greater interest, can be applied without any prior information. **DRC** and **GDRC** are our methods. The x-axis represents perturbation ratio ω and the y-axis represents performance.

We find that RE's performance is generally similar on average to PPO in these experiments. While RE can reduce reward variance, it also has to learn the reward structure. We hypothesize that the cost of learning the reward structure is sometimes overcome by the benefit of variance reduction and sometimes not.

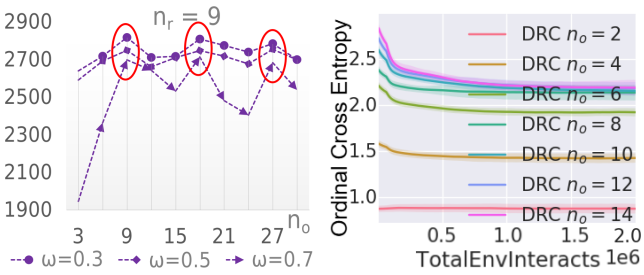
Discrete Control Tasks We run experiments on two discrete control tasks in Fig. 6. Following the settings in (Wang, Liu, and Li 2020), the reward range $[-17, 0)$ is discretized into $n_r = 17$ bins in Pendulum. In CartPole, apart from $+1$ rewards, -1 rewards are introduced for perturbations by (Wang, Liu, and Li 2020). As CartPole is an environment with discrete rewards, DRC and SR can work without prior knowledge. As depicted in Fig. 6, DRC is always the best performing for all levels of perturbations, with GDRC as the strongest performer among the methods without prior knowledge in Pendulum across all ω.

5.3 Impact of n_o on Reconstruction Error And Ordinal Cross-Entropy

In Fig. 7, we study the impact of n_o on Reconstruction Error (reflected as performance) and ordinal cross-entropy (OCE), experimentally verifying our propositions about the relation between n_o and Reconstruction Error and OCE in Fig. 3a, 3b and Sec. 4.2, and thus supporting our design choices for GDRC.

Fig. 7a compares the performance of DRC as n_o varies in Hopper when $n_r = 9$. As we analyze in Sec. 4.2, Reconstruction Error turns zero when n_o is a multiple of n_r. For $n_o = a \cdot n_r (a \in \mathbb{Z}^+)$ and any ω, we can observe the peak performance of DRC as circled. Fig. 7a supports our strategy of aiming for $n_o = n_r$ of GDRC using ordinal cross-entropy.

In Fig. 7b, we study the empirical impact of n_o on ordinal cross-entropy (OCE) throughout the training. If we read Fig. 7b from the bottom to the top at a certain training step, we can tell OCE increases rapidly for small n_o and stops increasing around $n_o = n_r$, replicating the simulation results in Fig. 3b in real experiments. This suggests that OCE is a reliable metric for selecting n_o as discussed in Sec. 4.2.



(a) The performance of DRC for different n_o in Hopper, where $n_r = 9$.

(b) OCE during the training using DRC with different n_o, where $n_r = 10$.

Figure 7: Performance and OCE as n_o varies in Hopper.

5.4 Ordinal Cross-Entropy Matters For DRC

In Table 2, we compare the performance of DRC using OCE and DRC_CE using CE, which is the only difference between the two methods. The settings are the same as the ones in Sec. 5.2. DRC always performs better than DRC_CE, and the gap is especially large in HalfCheetah and Reacher. In Hopper and Walker2d, the episode is cut when the current state is unhealthy (for example, upside down), but this is not the case in HalfCheetah and Reacher, where the episode always continues to the step limit. The result is that "good" reward samples are very rare in early episodes of HalfCheetah and Reacher. We hypothesize that CE overfits to the dominant samples, but in OCE, the rare samples receive higher weights and are thus predicted more accurately. The figure in the Appendix confirms this in HalfCheetah. The dominant reward bin $y_t = 2$ is predicted well by both DRC and DRC_CE. However, DRC_CE struggles to classify any other samples correctly, whereas DRC quickly learns to distinguish $y_t = 1, 2, 3$, despite the low frequencies of $y_t = 1, 3$ initially.

5.5 The Reward Range Matters For GDRC

A key design choice for GDRC is how to estimate the reward range. Perhaps the most intuitive choice is to store all the perturbed rewards in a buffer and choose some statistics of the buffer (e.g., percentiles) to estimate the range. However, the challenge is the reward distribution can shift dramatically across epochs, so perhaps old samples could be detrimental by not allowing the range to shift up as the policy improves. Our strategy is to store the samples in a sliding window manner, meaning we only use the samples collected in the latest 20 epochs to decide $[r_{emin}, r_{emax}]$. The performance of GDRC over 50 seeds in HalfCheetah with clean reward of storing all samples vs. storing the latest 20 epochs of samples is **1667.9** and **2698.0**. The dynamics of the maximum of the reward range estimate in HalfCheetah are shown in the Appendix. Early training generates a large number of low-reward samples (as episodes always reach the step limit). These low-reward samples must be discarded to allow the estimated max reward to increase.

Method	Hopper	HalfCheetah	Walker2d	Reacher
DRC_CE	2198.8	851.5	1747.6	-12.2
DRC	**2677.5**	**2127.3**	**2058.6**	**-7.8**

Table 2: The average performance of DRC using OCE and DRC_CE using CE over $n_r = 6, 10, 16$ and $\omega = 0.1, 0.3, 0.5, 0.7$

Perturbation	Method	Hopper	HalfCheetah	Walker2d	Reacher
Gaussian	PPO	**2699.2**	2736.3	2048.8	**-10.1**
	RE	**2708.7**	**2871.6**	**2084.3**	-17.0
	GDRC	**2822.9**	**2798.9**	**2101.2**	-10.9
Uniform	PPO	**2743.9**	**2778.6**	1955.0	-7.7
	RE	2575.3	**2658.6**	**2211.3**	-7.2
	GDRC	**2736.8**	2332.2	**2274.5**	**-6.2**
Reward Range Uniform	PPO	**2648.6**	2030.4	1929.2	-31.4
	RE	2571.6	**2323.6**	**2235.3**	-43.5
	GDRC	**2721.4**	**2387.7**	**2237.9**	**-8.4**

Table 3: The performance under continuous perturbations.

Method	Hopper	HalfCheetah	Walker2d	Reacher
PPO	2122.6	**2621.2**	**2222.3**	**-4.9**
RE	**2689.2**	**2736.3**	**2237.2**	-5.5
GDRC	**2753.7**	**2698.0**	**2128.8**	**-4.9**

Table 4: The average performance in clean environments.

5.6 Under Continuous Perturbations

In Table 3, we compare GRDC to PPO and RE under continuous perturbations (SR_W and DRC cannot work outside GCM settings). We test $\sigma = 1.0, 1.5, 2.0$ for Gaussian perturbations and $\omega = 0.1, 0.2, 0.3, 0.4$ for Uniform and Reward Uniform perturbations. GRDC wins/ties in 10/12, whereas RE wins/ties in 7/12 cases. Even under non-GCM perturbations, GDRC has a small edge over RE, especially targeting this kind of perturbation by making the stringent assumption that $\mathbb{E}(\tilde{r}) = \omega_0 \cdot r + \omega_1 (\omega_0 > 0)$. We attribute GDRC's advantage to its potential ability to decide the best n_o to differentiate the true rewards according to Lipschitz for different continuous perturbations. We also notice that the GDRC is more stable across perturbations and environments than baselines.

5.7 In Clean Environments

In Table 4, we compare GDRC with PPO and RE in clean environments. All three methods perform comparably, with PPO somewhat worse on Hopper and RE slighly worse on Reacher. This shows that the reward signal can be learned in a way that does not appear to interfere with policy learning. In Hopper, a mild amount of reward noise appears to help PPO explore—performance is better in Fig. 5 and Table 3 (this phenomena has been observed consistently by other researchers including (Wang, Liu, and Li 2020)).

Acknowledgements

We thank the Ohio Supercomputer Center (Center 1987) for providing the computational resources for this work.

References

Amodei, D.; Olah, C.; Steinhardt, J.; Christiano, P.; Schulman, J.; and Mané, D. 2016. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565*.

Bai, Y.; Jones, A.; Ndousse, K.; Askell, A.; Chen, A.; DasSarma, N.; Drain, D.; Fort, S.; Ganguli, D.; Henighan, T.; et al. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.

Bellemare, M. G.; Dabney, W.; and Munos, R. 2017. A distributional perspective on reinforcement learning. In *International conference on machine learning*, 449–458. PMLR.

Center, O. S. 1987. Ohio Supercomputer Center.

Choromanski, K.; Pacchiano, A.; Parker-Holder, J.; Tang, Y.; Jain, D.; Yang, Y.; Iscen, A.; Hsu, J.; and Sindhwani, V. 2020. Provably robust blackbox optimization for reinforcement learning. In *Conference on Robot Learning*, 683–696. PMLR.

Corazza, J.; Gavran, I.; and Neider, D. 2022. Reinforcement learning with stochastic reward machines. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 6429–6436.

Dabney, W.; Ostrovski, G.; Silver, D.; and Munos, R. 2018a. Implicit quantile networks for distributional reinforcement learning. In *International conference on machine learning*, 1096–1105. PMLR.

Dabney, W.; Rowland, M.; Bellemare, M.; and Munos, R. 2018b. Distributional reinforcement learning with quantile regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Everitt, T.; Krakovna, V.; Orseau, L.; Hutter, M.; and Legg, S. 2017. Reinforcement learning with a corrupted reward channel. *arXiv preprint arXiv:1705.08417*.

Hadfield-Menell, D.; Milli, S.; Abbeel, P.; Russell, S. J.; and Dragan, A. 2017. Inverse reward design. *Advances in neural information processing systems*, 30.

Hu, J.; Sun, Y.; Chen, H.; Huang, S.; Chang, Y.; Sun, L.; et al. 2022. Distributional Reward Estimation for Effective Multi-Agent Deep Reinforcement Learning. *Advances in Neural Information Processing Systems*, 35: 12619–12632.

Hutter, M. 2005. *Universal artificial intelligence: Sequential decisions based on algorithmic probability*. Springer Science & Business Media.

Lillicrap, T. P.; Hunt, J. J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; and Wierstra, D. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Moreno, A.; Martín, J. D.; Soria, E.; Magdalena, R.; and Martínez, M. 2006. Noisy reinforcements in reinforcement learning: some case studies based on gridworlds. In *Proceedings of the 6th WSEAS international conference on applied computer science*, 296–300.

Ng, A. Y.; Harada, D.; and Russell, S. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, 278–287. Citeseer.

Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35: 27730–27744.

Pattanaik, A.; Tang, Z.; Liu, S.; Bommannan, G.; and Chowdhary, G. 2017. Robust deep reinforcement learning with adversarial attacks. *arXiv preprint arXiv:1712.03632*.

Pinto, L.; Davidson, J.; Sukthankar, R.; and Gupta, A. 2017. Robust adversarial reinforcement learning. In *International Conference on Machine Learning*, 2817–2826. PMLR.

Puterman, M. L. 2014. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

Qian, J.; Weng, P.; and Tan, C. 2023. Learning Rewards to Optimize Global Performance Metrics in Deep Reinforcement Learning. *arXiv preprint arXiv:2303.09027*.

Rakhsha, A.; Radanovic, G.; Devidze, R.; Zhu, X.; and Singla, A. 2020. Policy teaching via environment poisoning: Training-time adversarial attacks against reinforcement learning. In *International Conference on Machine Learning*, 7974–7984. PMLR.

Ring, M.; and Orseau, L. 2011. Delusion, survival, and intelligent agents. In *Artificial General Intelligence: 4th International Conference, AGI 2011, Mountain View, CA, USA, August 3-6, 2011. Proceedings 4*, 11–20. Springer.

Romoff, J.; Henderson, P.; Piché, A.; Francois-Lavet, V.; and Pineau, J. 2018. Reward estimation for variance reduction in deep reinforcement learning. In *Proceedings of The 2nd Conference on Robot Learning*.

Rowland, M.; Bellemare, M.; Dabney, W.; Munos, R.; and Teh, Y. W. 2018. An analysis of categorical distributional reinforcement learning. In *International Conference on Artificial Intelligence and Statistics*, 29–37. PMLR.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Silver, D.; Singh, S.; Precup, D.; and Sutton, R. S. 2021. Reward is enough. *Artificial Intelligence*, 299: 103535.

Todorov, E.; Erez, T.; and Tassa, Y. 2012. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, 5026–5033. IEEE.

Wang, J.; Liu, Y.; and Li, B. 2020. Reinforcement learning with perturbed rewards. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, 6202–6209.

Zheng, J.; Liu, S.; and Ni, L. M. 2014. Robust bayesian inverse reinforcement learning with sparse behavior noise. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28.

Zhong, J.; Wu, R.; and Si, J. 2023. A Long N-step Surrogate Stage Reward for Deep Reinforcement Learning. In Oh, A.; Neumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; and

Levine, S., eds., *Advances in Neural Information Processing Systems*, volume 36, 12733–12745. Curran Associates, Inc.

Zhuang, V.; and Sui, Y. 2021. No-regret reinforcement learning with heavy-tailed rewards. In *International Conference on Artificial Intelligence and Statistics*, 3385–3393. PMLR.