Provably Robust DPO: Aligning Language Models with Noisy Feedback

Sayak Ray Chowdhury [*] [1] **Anush Kini** [*] [1] **Nagarajan Natarajan** [1]

Abstract

Learning from preference-based feedback has recently gained traction as a promising approach to align language models with human interests. While these aligned generative models have demonstrated impressive capabilities across various tasks, their dependence on high-quality human preference data poses a bottleneck in practical applications. Specifically, noisy (incorrect and ambiguous) preference pairs in the dataset might restrict the language models from capturing human intent accurately. While practitioners have recently proposed heuristics to mitigate the effect of noisy preferences, a complete theoretical understanding of their workings remain elusive. In this work, we aim to bridge this gap by introducing a general framework for policy optimization in the presence of random preference flips. We focus on the direct preference optimization (DPO) algorithm in particular since it assumes that preferences adhere to the Bradley-Terry-Luce (BTL) model, raising concerns about the impact of noisy data on the learned policy. We design a novel loss function, which de-bias the effect of noise on average, making a policy trained by minimizing that loss robust to the noise. Under log-linear parameterization of the policy class and assuming good feature coverage of the SFT policy, we prove that the sub-optimality gap of the proposed robust DPO (rDPO) policy compared to the optimal policy is of the order $O(\frac{1}{1-2\varepsilon}\sqrt{\frac{d}{n}})$, where $\varepsilon < 1/2$ is flip rate of labels, d is policy parameter dimension and n is size of dataset. Our experiments on IMDb sentiment generation and Anthropic's helpful-harmless dataset shows that rDPO is robust to noise in preference labels compared to vanilla DPO and other heuristics proposed by practitioners.

[*]Equal contribution [1]Microsoft Research, India. Correspondence to: Sayak Ray Chowdhury <t-sayakr@microsoft.com>.

Proceedings of the 41^{st} International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

1. Introduction

Reinforcement Learning from Human Feedback (RLHF) has proven highly effective in aligning Large Language Models (LLMs) with human preferences (Christiano et al., 2017; Stiennon et al., 2020; Ouyang et al., 2022). In the RLHF pipeline(Kaufmann et al., 2023), an LLM is first pretrained using supervised fine tuning to obtain a reference or SFT policy. A reward model is fit to a dataset of human preferences in the form of a classifier between preferred and rejected responses. Next, an LLM policy is trained using RL algorithms such as proximal policy optimization (PPO) to generate high-reward responses while minimizing a certain notion of divergence from the SFT policy.

While RLHF produces models (e.g. GPT4, Llama, Mistral etc.) with impressive capabilities across diverse tasks ranging from programming to creative writing, it introduces notable complexities into the training process (Zheng et al., 2023). It requires training two language models (one for reward and another for policy) and frequent sampling from the policy in the course of training. This demands significant compute and storage, often limiting the feasible size of a model. To get around these issues, the direct preference optimisation (DPO) method (Rafailov et al. (2023)) optimizes the language model policy directly from human preferences without learning a reward model explicitly and avoiding complexities of RL. Given a dataset of human preferences over model responses, DPO defines a certain binary-cross entropy loss, and implicitly optimizes the same objective as RLHF in the form of KL-regularized reward maximization.

A crucial ingredient governing the success of both RLHF and DPO is the quality of preference data. Gathering responses for a vast array of prompts is often inherently noisy (e.g., ambiguous preferences), which could derail policy training, with or without RL (Lambert et al., 2023; Bai et al., 2022b). We find empirical evidence that these algorithms are robust to noise in some scenarios (as also demonstrated by Rafailov et al. (2023); Ouyang et al. (2022)), even though they work under the assumption that the observed preferences adhere to an underlying sampling model (see Section 2). On the other hand, as we show via simple noise injection mechanisms on real-world datasets in Section 6, the performance of DPO drops significantly when the noise rates are high. We are not the first to identify or address this problem — Wang et al. (2024) demonstrate the sensitivity of reward training step in the RLHF pipeline to noisy preferences in real data; and design heuristics to mitigate the impact (discussed in Section 6). However, little is known about theory behind these heuristics, which could justify their performance in practice.

In this work, we focus on bridging this gap between theory and practice by introducing a general theoretical framework for learning from noisy preference data. We particularly focus on the DPO algorithm in the presence of random preference noise, where preferences are flipped with some (known) rate. We make the following contributions.

Novel loss function. We design a novel loss function by adapting the binary cross entropy (BCE) loss of DPO with the rate of label flips. We show that this loss is an unbiased estimate of the original BCE loss, which de-biases the effect of preference noise and makes the policy robust. We call it robust DPO (rDPO). Similar to DPO, our rDPO gradients on average increase the log probability of preferred answers relative to the rejected ones. But, unlike DPO, the importance weights in gradients are tuned to the noise level, which mitigate the effect of noisy preferences. Notably, our approach generalizes to reward training in RLHF and to other preference optimization methods (discussed in Section 5).

First theoretical guarantees. To the best of our knowledge, we are the first to provide theoretical guarantees for practical preference optimization algorithms. Under log-linear parameterization of the policy class, we show that estimation error of our rDPO policy compared to the optimal policy is at most $O(\frac{1}{1-2\varepsilon}\sqrt{\frac{d}{n}})$, where $\varepsilon \in [0, 1/2)$ is flip rate, d is dimension of policy parameter and n is number of preference samples. Under good coverage of the SFT policy over the feature space, the estimation error bound translates to a bound on the average reward obtained by our trained policy as compared to the optimal policy. Our results show that the additional cost of preference flips is a (multiplicative) factor of $O(\frac{1}{1-2\varepsilon})$. Along the way, setting $\varepsilon = 0$ in the above bound, we obtain the first performance bounds for DPO policy without label noise, which resolves an elusive theoretical gap in the understanding of practical algorithms for learning from human preferences.

Empirical evidence. On noisy preferences generated from sentiment generation on IMDb dataset (Maas et al., 2011) and on Anthropic's helpful-harmless (Bai et al., 2022a), we provide empirical evidence that shows performance of DPO degrades with the introduction of high noise in data. However, rDPO is robust to noise in preference labels compared to other baselines including DPO with label smoothing (Mitchell, 2023). Additionally, policies optimized using rDPO are consistently better than other methods across different sampling temperatures. All artifacts are made available at https://aka.ms/RobustDPO.

1.1. Related Work

Recognizing the storage and computational challenges in RLHF, several alternatives have been proposed. Each of these method work with different loss functions. While DPO optimizes BCE loss to learn the policy (Rafailov et al., 2023), SLiC uses hinge loss plus a regularization loss (Zhao et al., 2023), IPO uses square-loss (Azar et al., 2023), RRHF uses ranking loss plus SFT loss (Yuan et al., 2023) and RSO uses BCE loss plus a rejection sampling (Liu et al., 2023). While they have their own intricacies and differences, all are competitive with RLHF on standard language tasks.

A recent line of work provides theoretical guarantees on the performance of policy learned using preference-based RL algorithms (Pacchiano et al., 2021; Chen et al., 2022; Zhu et al., 2023; Zhan et al., 2023). All these works focus on guarantees in terms of regret bounds in the standard bandit or RL setting and they do not deal with the practical algorithms like RLHF or DPO. Zhu et al. (2024) considers the problem of reward overfitting in RLHF by replacing hard labels with soft ones. They do not consider model overfitting in the presence of noisy data.

There is a line of work in supervised (deep) learning literature that considers learning in the presence of label noise. Müller et al. (2019) study the effect of label smoothing to mitigate the overfitting problem under noisy data. Natarajan et al. (2013) consider binary classification with noisy labels, while Patrini et al. (2017) work on multi-label classification problems. They focus on bounding the excess population risk of trained classifiers under the clean distribution. In contrast, we aim to bound the estimation error of the trained policy, which brings out additional challenges in analysis.

2. Background and Problem Setup

Learning algorithms for conditional language generation from human feedback take a preference dataset $\mathcal{D} = (s_i, a_{w,i}, a_{l,i})_{i=1}^n$ of size n as input that distinguishes the better answer from the worse given the same prompt. First, a prompt is sampled from a distribution: $s \sim \rho$. Next, a pair of answers are sampled from a supervised fine tuned (SFT) policy: $a, a' \sim \pi_{\text{sft}}(\cdot|s)$. The response pairs are then presented to human labelers (or, an oracle) who express preferences for answers given prompt s, denoted as $a_w \succ a_l | s$. The preference distribution is typically expressed using a latent reward model $r^*(s, a)$ as

$$p^*_{s,a,a'} = \mathbb{P}[a \succ a'|s] = g(r^*(s,a) - r^*(s,a')) \ , \quad (1)$$

where $g : \mathbb{R} \to [0, 1]$ is a monotone non-decreasing function (with $g(z) = 1 - g(-z)$) that converts reward differences into winning probabilities. When g is the sigmoid function $\sigma(z) = \frac{1}{1+e^{-z}}$, we get the Bradley-Terry-Luce (BTL) model (Bradley & Terry, 1952; Luce, 2012).

Optimal Policy. Starting with a prompt distribution ρ and an SFT policy π_{sft}, the optimal language model policy π^* corresponding to the latent reward model r^* can be computed by maximizing the objective (Schulman et al., 2017)

$$J(\pi) = \mathbb{E}_{s\sim\rho, a\sim\pi(\cdot|s)}\left[r^*(s,a) - \beta \log \frac{\pi(a|s)}{\pi_{\text{sft}}(a|s)}\right] .$$

The optimal policy takes the form (Rafailov et al., 2023)

$$\pi^*(a|s) = \frac{1}{Z^*(s)}\pi_{\text{sft}}(a|s)\exp(r^*(s,a)/\beta) , \qquad (2)$$

where $Z^*(s) = \sum_{a\in\mathcal{A}} \pi_{\text{sft}}(a|s)\exp(r^*(s,a)/\beta)$ denotes the log-partition (normalizing) function. Here $\beta > 0$ is a parameter that governs the balance between exploitation and exploration. When $\beta \to 0$, all probability mass will concentrate on the response with highest reward (exploitation). On the other extreme, when $\beta \to \infty$, optimal policy will be the same as π_{sft} (exploration). The goal is to learn a policy from preference data that generates good reward.

Policy Estimation. Re-arranging (2), we get

$$r^*(s,a) = \beta \log \frac{\pi^*(a|s)}{\pi_0(a|s)} + \beta \log Z^*(s) . \qquad (3)$$

Then the true preference probabilities under the BTL model (1) can be expressed using the optimal and SFT policies as (Rafailov et al., 2023)

$$p^*_{s,a,a'} = \sigma\left(\beta \log \frac{\pi^*(a|s)}{\pi_{\text{sft}}(a|s)} - \beta \log \frac{\pi^*(a'|s)}{\pi_{\text{sft}}(a'|s)}\right) .$$

In this work, we consider parameterized policies π_θ, where $\theta \in \Theta \subset \mathbb{R}^d$ is a vector of dimension d. In practice, the most common policy classes are of the form

$$\Pi = \left\{\pi_\theta(a|s) = \frac{\exp(f_\theta(s,a))}{\sum_{a'\in\mathcal{A}}\exp(f_\theta(s,a'))}\right\} , \qquad (4)$$

where f_θ is a real-valued differentiable function. For example, the tabular softmax policy class is the one where $f_\theta(s,a) = \theta_{s,a}$. Typically, f_θ is either a linear function or a neural network. A linear f_θ can be expressed as $f_\theta(s,a) = \phi(s,a)^\top\theta$ using a feature map $\phi(s,a) \in \mathbb{R}^d$. In this case π_θ becomes a log-linear policy, i.e., $\log \pi_\theta(a|s) \propto \langle\theta, \phi(s,a)\rangle$. In case of language model policies, the feature map ϕ can be constructed by removing the last layer of the model, and θ correspond to the weights of the last layer.

Let θ^* and θ_0 denote the parameters corresponding to the optimal and SFT policies, respectively. Now, define the *preference score* of an answer a relative to another one a' given prompt s under policy π_θ as

$$h_\theta(s,a,a') = \widehat{r}_\theta(s,a) - \widehat{r}_\theta(s,a') , \qquad (5)$$

where $\widehat{r}_\theta(s,a) = \log \frac{\pi_\theta(a|s)}{\pi_{\theta_0}(a|s)}$ is an implicit reward defined by trained and SFT policies π_θ and π_{θ_0}. This lets us express, for any $\theta \in \Theta$, the predicted preference probabilities (we omit dependence on θ, θ_0 for brevity) as

$$p_{s,a,a'} = \mathbb{P}_\theta[a \succ a'|s] = \sigma(\beta h_\theta(s,a,a')) . \qquad (6)$$

In this notation, we have the true preference probabilities $p^*_{s,a,a'} = \sigma(\beta h_{\theta^*}(s,a,a'))$.

With preference probabilities expressed in terms of the optimal policy, the DPO algorithm (Rafailov et al., 2023) finds the maximum likelihood estimate (MLE) by minimizing the empirical BCE loss $\frac{1}{n}\sum_{i=1}^n \mathcal{L}(\theta; s, a_{w,i}, a_{l,i})$, where

$$\mathcal{L}(\theta; s, a_w, a_l) = -\log \sigma(\beta h_\theta(s,a_w,a_l)) . \qquad (7)$$

Technically, the minimizer of this loss is not strictly an MLE for the optimal policy parameter θ^* as the preference pairs are sampled from the SFT policy π_{θ_0}, but not from the policy to be estimated π_{θ^*}. In reality, however, it is challenging to obtain preference pairs directly sampled from π_{θ^*}.

Preference Noise. In this work, we model noise in the preferences via the standard random noise model (Natarajan et al., 2013; Wang et al., 2024; Mitchell, 2023), where the revealed preferences are true preferences flipped with a small probability $\varepsilon \in (0, 1/2)$, i.e.

$$\mathbb{P}_\varepsilon\left[(\widetilde{a}_{l,i}, \widetilde{a}_{w,i}) = (a_{w,i}, a_{l,i})|s_i\right] = \varepsilon . \qquad (8)$$

Let $\widetilde{\mathcal{D}} = (s_i, \widetilde{a}_{w,i}, \widetilde{a}_{l,i})_{i=1}^n$ denote the dataset of potentially noisy samples. These noisy samples are what the learning algorithm sees, i.e., $\widetilde{a}_{w,i}$ is seen to be preferred over $\widetilde{a}_{l,i}$.We will assume that the flip rate ε is known to the learner. In practice, we will tune the flip rate through cross-validation.

Performance Measure. Our goal is to learn a policy $\widehat{\pi}_n(a|s)$ (equivalently, a policy parameter $\widehat{\theta}_n$) from noisy preference data $\widetilde{\mathcal{D}}$ that generates maximum expected reward

$$r^*(\pi) = \mathbb{E}_{s\sim\rho, a\sim\pi(\cdot|s)}\left[r^*(s,a)\right] .$$

We measure performance of the learned policy using a sub-optimality gap from the optimal policy π^*, namely $r^*(\pi^*) - r^*(\widehat{\pi}_n)$. Ideally, we want the gap to go down to zero as $n \to \infty$ with a rate at least sub-linear in n. This is a standard measure of policy performance in the RL literature (Zhu et al., 2023; Qiao & Wang, 2022; Agarwal et al., 2021).

3. Our Approach: Robust DPO

We start with the BCE loss under noisy preferences and then approximate it with a conservative loss that practitioners have explored recently (Mitchell, 2023). Next, we discuss their drawback, which help us get intuition for a robust loss.

Given corrupted dataset $\widetilde{\mathcal{D}}$, one can use (7) to compute the MLE under noisy preferences by minimizing the loss

$$\mathcal{L}_\varepsilon(\theta; s, \widetilde{a}_w, \widetilde{a}_l) = -\log \mathbb{P}_{\theta,\varepsilon}[\widetilde{a}_w \succ \widetilde{a}_l|s] , \qquad (9)$$

where, for any (s,a,a') triplet, the predicted probabilities under noisy preferences are computed using (6) and (8):

$$\mathbb{P}_{\theta,\varepsilon}[a \succ a'|s] = (1-\varepsilon)\cdot \mathbb{P}_\theta[a \succ a'|s] + \varepsilon \cdot \mathbb{P}_\theta[a' \succ a|s]$$
$$= (1-\varepsilon)\cdot \sigma(\beta h_\theta(s,a,a')) + \varepsilon \cdot \sigma(\beta h_\theta(s,a',a)). \quad (10)$$

Now, using Jensen's inequality, one can obtain

$$\log \mathbb{P}_{\theta,\varepsilon}[\widetilde{a}_w \succ \widetilde{a}_l|s] \geq (1-\varepsilon)\cdot \log\sigma(\beta h_\theta(s,\widetilde{a}_w,\widetilde{a}_l)) + \varepsilon\cdot\log\sigma(\beta h_\theta(s,\widetilde{a}_l,\widetilde{a}_w)) .$$

Thus, one can upper bound (9) by a *conservative* loss

$$\begin{aligned}&\bar{\mathcal{L}}_\varepsilon(\theta; s,\widetilde{a}_w,\widetilde{a}_l)\\ &= -(1-\varepsilon)\log\sigma(\beta h_\theta(s,\widetilde{a}_w,\widetilde{a}_l)) - \varepsilon\log\sigma(\beta h_\theta(s,\widetilde{a}_l,\widetilde{a}_w))\\ &= (1-\varepsilon)\mathcal{L}(\theta;s,\widetilde{a}_w,\widetilde{a}_l) + \varepsilon\mathcal{L}(\theta;s,\widetilde{a}_l,\widetilde{a}_w) , \quad (11)\end{aligned}$$

which is simply a weighted sum of the DPO loss (7) under noisy preferences. Mitchell (2023) called this method conservative DPO (cDPO). This can also be motivated from the label smoothing technique (Müller et al., 2019) to mitigate over-fitting problem under noisy data. Notably, Wang et al. (2024) use exactly the same loss function to train the reward model for RLHF, and empirically show its superior performance over vanilla RLHF in the presence of noisy data. In our experiments, we call this method (when coupled with PPO for policy training) conservative PPO (cPPO).

3.1. An Unbiased Loss Function

The BCE loss (9) and the conservative loss (11) have a common drawback – both introduce bias in the DPO loss (7). This is due to the fact that

$$\mathbb{E}\left[\ell(\theta;s,\widetilde{a}_w,\widetilde{a}_l)\right] \neq \mathcal{L}(\theta;s,a_w,a_l),\ \ell\in\{\mathcal{L}_\varepsilon,\bar{\mathcal{L}}_\varepsilon\} .$$

It also holds that (Chowdhury et al., 2023)

$$\text{logit}(\mathbb{P}_{\theta,\varepsilon}[a\succ a'|s]) \neq \text{logit}(\mathbb{P}_\theta[a\succ a'|s]).$$

That is, the log-odds of preferring a over a' under noisy preferences is different from that without noise, which introduces a bias in preferences. Ideally, we want the logits to be same for both with and without noise. To this end, we define (un-normalized) preference probabilities

$$\widehat{\mathbb{P}}_{\theta,\varepsilon}[a\succ a'|s] = \frac{\sigma(\beta h_\theta(s,a,a'))^{(1-\varepsilon)}}{\sigma(\beta h_\theta(s,a',a))^{\varepsilon}} .$$

these have the same logits as those without noise, since

$$\begin{aligned}\text{logit}(\widehat{\mathbb{P}}_{\theta,\varepsilon}[a\succ a'|s]) &= \log\left(\frac{\sigma(\beta h_\theta(s,a,a'))}{\sigma(\beta h_\theta(s,a',a))}\right)\\ &= \text{logit}(\mathbb{P}_\theta[a\succ a'|s]).\end{aligned}$$

This motivates us to define the loss function:

$$\begin{aligned}\widehat{\mathcal{L}}_\varepsilon(\theta;s,\widetilde{a}_w,\widetilde{a}_l) &= -\frac{1}{1-2\varepsilon}\log\widehat{\mathbb{P}}_{\theta,\varepsilon}[\widetilde{a}_w\succ\widetilde{a}_l|s]\\ &= \frac{(1-\varepsilon)\mathcal{L}(\theta;s,\widetilde{a}_w,\widetilde{a}_l) - \varepsilon\mathcal{L}(\theta;s,\widetilde{a}_l,\widetilde{a}_w)}{1-2\varepsilon} .\end{aligned} \quad (12)$$

This loss is an unbiased estimator of the DPO loss (7) under noisy preferences as stated in the following lemma.

Lemma 3.1. *For any $\theta,\theta_0\in\mathbb{R}^d$, $\varepsilon\in(0,1/2)$, we have*

$$\mathbb{E}_\varepsilon\left[\widehat{\mathcal{L}}_\varepsilon(\theta;s,\widetilde{a}_w,\widetilde{a}_l)|a_w,a_l\right] = \mathcal{L}(\theta;s,a_w,a_l) .$$

This way, we learn a *good* estimate of the policy parameter in the presence of label noise by minimizing the sample average of the above *robust* (w.r.t. preference flips) loss:

$$\widehat{\theta}_n \in \text{argmin}_{\theta\in\Theta}\frac{1}{n}\sum_{i=1}^n \widehat{\mathcal{L}}_\varepsilon(\theta;s,\widetilde{a}_{w,i},\widetilde{a}_{l,i}) . \quad (13)$$

We call our method robust-DPO (or rDPO in short). Note that when preferences are clean (i.e. flip rate $\varepsilon=0$), the rDPO loss (12) reduces to the DPO loss (7), and hence our trained rDPO policy (13) coincides with the DPO policy of Rafailov et al. (2023).

Variance of rDPO loss. Along with unbiasedness, it is also desirable to have bounded variance of the estimator. To this end, consider the un-normalized rDPO loss $(1-2\varepsilon)\widehat{\mathcal{L}}_\varepsilon(\theta;s,\widetilde{a}_w,\widetilde{a}_l)$, which yields the same loss-minimizing policy as in (13). It has a variance $\varepsilon(1-\varepsilon)\left[\mathcal{L}(\theta;s,a_w,a_l)-\mathcal{L}(\theta;s,a_l,a_w)\right]^2$. For Neural policy class of the form (4) and for bounded f_θ, the variance is bounded by $C\varepsilon(1-\varepsilon)$ for some constant $C>0$. Since $\varepsilon\leq 1/2$, the variance is bounded by $C/4$.

3.2. Gradients of rDPO Loss

To further understand the mechanism of rDPO, let's now look at the gradients of its loss (12) and contrast that with that of DPO loss (7). The gradients of $\widehat{\mathcal{L}}_\varepsilon$ with respect to the parameters θ can be written as

$$\begin{aligned}&\nabla_\theta\widehat{\mathcal{L}}_\varepsilon(\theta;s,\widetilde{a}_w,\widetilde{a}_l)\\ &= \frac{(1-\varepsilon)\nabla_\theta\mathcal{L}(\theta;s,\widetilde{a}_w,\widetilde{a}_l) - \varepsilon\nabla_\theta\mathcal{L}(\theta;s,\widetilde{a}_l,\widetilde{a}_w)}{1-2\varepsilon}\\ &= -\beta\widehat{\zeta}_{\theta,\varepsilon}\left(\nabla_\theta\log\pi_\theta(\widetilde{a}_w|s) - \nabla_\theta\log\pi_\theta(\widetilde{a}_l|s)\right) . \quad (14)\end{aligned}$$

Here the weights in the gradients are given by

$$\widehat{\zeta}_{\theta,\varepsilon} = \underbrace{\frac{1-\varepsilon}{1-2\varepsilon}\sigma(\beta h_\theta(s,\widetilde{a}_l,\widetilde{a}_w))}_{\text{(I)}} + \underbrace{\frac{\varepsilon}{1-2\varepsilon}\sigma(\beta h_\theta(s,\widetilde{a}_w,\widetilde{a}_l))}_{\text{(II)}},$$

where $h_\theta(s,a,a')$ is the difference of implicit rewards $\widehat{r}_\theta$ of answers a and a' given prompt s; see (5). **Term (I)** puts higher weight when the implicit reward model $\widehat{r}_\theta$ orders the observed preferences incorrectly and scales it proportionally with the probability of no-flip. **Term (II)** puts higher weight when the implicit reward model $\widehat{r}_\theta$ orders the observed preferences correctly and scales it proportionally with the probability of flip. Both the terms together de-bias the effect of noise on average in observed preferences.

Comparison with DPO and cDPO. The weights in the gradients of cDPO loss $\bar{\mathcal{L}}_\varepsilon$ are

$$\bar{\zeta}_{\theta,\varepsilon} = (1-\varepsilon)\sigma(\beta h_\theta(s,\widetilde{a}_l,\widetilde{a}_w)) - \varepsilon\sigma(\beta h_\theta(s,\widetilde{a}_w,\widetilde{a}_l)) .$$

Meanwhile, the weights for the DPO loss gradients, if run on noisy preferences, are given by

$$\zeta_\theta = \sigma(\beta h_\theta(s, \widetilde{a}_l, \widetilde{a}_w)) = \sigma\left(\beta \widehat{r}_\theta(s, \widetilde{a}_l) - \beta \widehat{r}_\theta(s, \widetilde{a}_w)\right),$$

Lemma 3.2 (Gradient weights)**.** *For any $\varepsilon \in (0, 1/2)$, it holds that $\widehat{\zeta}_{\theta,\varepsilon} = \zeta_\theta + \frac{\varepsilon}{1-2\varepsilon}$ and $\zeta_\theta = \bar{\zeta}_{\theta,\varepsilon} + \varepsilon$.*

Consider the case, when there is no-flip, $(\widetilde{a}_w, \widetilde{a}_l) = (a_w, a_l)$. Observe from (14) that rDPO (also cDPO and DPO) gradients increase the likelihood of preferred answers and decreases that of dis-preferred ones. Since weights are higher for rDPO compared to DPO & cDPO (Lemma 3.2), this makes the parameter update for rDPO more aggressive than DPO & cDPO in the desired direction.

Now, for the case of preference flips, i.e., $(\widetilde{a}_w, \widetilde{a}_l) = (a_l, a_w)$, the gradients are not in the desired direction (increase likelihood of dis-preferred answers). Hence, rDPO updates will be more aggressive in the wrong direction than DPO & cDPO. However, since preferences are flipped with probability less than $1/2$, rDPO gradients will push the parameter updates in the correct direction faster than DPO & cDPO on average. This behavior is reflected in our experiments too - latent rewards of rDPO policy converges to that of the optimal policy much faster than DPO & cDPO policies; see Section 6.

4. Theoretical Analysis

Our method enjoys certain theoretical properties. By unbiasedness of $\widehat{\mathcal{L}}_\varepsilon$ (Lemma 3.1), we know that, for any fixed $\theta \in \Theta$, the empirical rDPO loss (12) converges to the population DPO loss $\mathbb{E}_{s,a_w,a_l}\left[\mathcal{L}(\theta; s, a_w, a_l)\right]$ even though the former is computed using noisy preferences whereas the latter depends on clean preferences. But the rDPO policy $\widehat{\pi}_n = \pi_{\widehat{\theta}_n}$ won't necessarily converge to the optimal policy π^* as preference pairs are sampled from the SFT policy π_{sft}, but not form π^* - an issue also shared by DPO policy (Liu et al., 2023). However, our end goal is to bound the sub-optimality gap of $\widehat{\pi}_n$. For this, we only need to characterize the estimation error of the learned policy parameter $\widehat{\theta}_n$ as function of number of samples n and flip rate ε.

4.1. Estimation Error

Under the BTL model (1), two reward functions from the same equivalence class[1] induce the same preference distribution and the same optimal policy (Rafailov et al., 2023). Due to this model under-specification and reward re-parameterization (3), we need to impose an identifiability constraint on the set of policy parameters Θ, namely $\Theta = \{\theta \in \mathbb{R}^d | \sum_{i=1}^d \theta_i = 0\}$ to achieve any guarantee on the estimation error. We also assume $\|\theta\| \le B, \forall \theta \in \Theta$. We give guarantees for Neural policy class of the form (4), i.e., when f_θ is a neural network parameterized by θ. We make a smoothness assumption on the policy class:

Assumption 4.1 (Smoothness)**.** For any $\theta \in \Theta$ and (s, a),

$$|f_\theta(s,a)| \le \alpha_0, \|\nabla f_\theta(s,a)\| \le \alpha_1, \nabla^2 f_\theta(s,a) \preccurlyeq \alpha_2 I .$$

The assumption ensures that implicit reward differences $h_\theta(s, a_w, a_l)$ are bounded, Lipschitz, and their gradients are also Lipschitz. This is quite common for establishing convergence for policy gradient methods (Agarwal et al., 2021). Log-linear policies ($f_\theta(s,a) = \theta^\top \phi(s,a)$), satisfy this assumption with $\alpha_0 = LB, \alpha_1 = L, \alpha_2 = 0$, where L is an upper bound on ℓ_2-norm of features $\phi(s,a)$.

The following result gives a guarantee on the estimation error in the semi-norm $\|\cdot\|_{\widehat{\Sigma}_\theta}$, which is expressed in terms of parameter dimension d and flip rate ε. Here, for any $\theta \in \mathbb{R}^d$, $\widehat{\Sigma}_\theta = \frac{1}{n}\sum_{i=1}^n x_i x_i^\top$ is the sample covariance matrix of gradients of implicit reward differences under true preferences, where $x_i = \nabla h_\theta(s_i, a_{w,i}, a_{l,i}) = \nabla f_\theta(s_i, a_{w,i}) - \nabla f_\theta(s_i, a_{l,i})$.

The error scales inversely with $\gamma\beta(1-2\varepsilon)$, where $\gamma \le \sigma'(\beta h_\theta(s, a_w, a_l))$ for all $\theta \in \Theta$ and for all preference samples (s, a_w, a_l). Here γ lower bounds the first derivative of the logistic function $\sigma(z_\theta; \beta, z_0) = \frac{1}{1+e^{-\beta(z_\theta - z_0)}}$, where $z_\theta = f_\theta(s, a_w) - f_\theta(s, a_l)$ and $z_0 = z_{\theta_0}$.

Theorem 4.2 (Estimation error of $\widehat{\theta}_n$)**.** *Let $\delta \in (0, 1], \varepsilon \in [0, 1/2), \lambda > 0$. Then, for Neural policy class (4) and under Assumption 4.1, with probability at least $1 - \delta$, we have*

$$\left\|\widehat{\theta}_n - \theta^*\right\|_{\widehat{\Sigma}_{\theta^*} + \lambda I} \le \frac{C}{\gamma\beta(1-2\varepsilon)} \cdot \sqrt{\frac{d + \log(1/\delta)}{n}} + C' \cdot B\sqrt{\lambda + \frac{\alpha_2}{\gamma\beta(1-2\varepsilon)} + \alpha_1\alpha_2 B},$$

where $\gamma = \frac{1}{2+e^{-4\beta\alpha_0}+e^{4\beta\alpha_0}}$, C, C' are absolute constants.

Several remarks are in order with this result. To keep the presentation simple, we consider log-linear policies in the following, where $\alpha_2 = 0$ and $x_i = \phi(s_i, a_{w,i}) - \phi(s_i, a_{l,i})$. In this case, $\widehat{\Sigma}_\theta$ is the covariance matrix of feature differences and independent of θ. We denote this by $\widehat{\Sigma}$ and get a high-probability error bound for log-linear policy class:

$$\left\|\widehat{\theta}_n - \theta^*\right\|_{\widehat{\Sigma} + \lambda I} = O\Big(\frac{1}{\gamma\beta(1-2\varepsilon)}\sqrt{\frac{d}{n}} + B\sqrt{\lambda}\Big). \quad (15)$$

Choice of Regularizer λ. When the feature covariance matrix $\widehat{\Sigma}$ is invertible, the above result holds for $\lambda = 0$. In this case, we will get a vanishing error-rate in the ℓ_2-norm

$$\left\|\widehat{\theta}_n - \theta^*\right\| = O\Big(\frac{1}{\sqrt{\lambda_{\min}(\Sigma)}}\frac{1}{\gamma\beta(1-2\varepsilon)}\sqrt{\frac{d}{n}}\Big). \quad (16)$$

[1]Two reward functions r_1, r_2 are equivalent iff $r_1(s,a) - r_2(s,a) = g(s)$ for some function g.

If this is not the case, $\widehat{\theta}_n$ won't necessarily converge to θ^*. But one might set $\lambda = O(d/n)$ to achieve a vanishing error in the semi-norm $\widehat{\Sigma}$ for log-linear policies. However, the error will not vanish for Neural policies (as $\alpha_2 \neq 0$).

Estimation Error of DPO Policy. As already mentioned, our rDPO policy (13) recovers the DPO policy under clean preferences. Thus, setting $\varepsilon = 0$ in Theorem 4.2, we get an error bound of order $O\big(\frac{1}{\gamma}\sqrt{d/n}\big)$ for the DPO policy. Therefore, as a by-product of our approach, we get the first error bound for the trained DPO policy of Rafailov et al. (2023), which could be of independent interest.

Effect of Noisy Preferences. When preferences are noisy (i.e. flip rate $\varepsilon > 0$), our rDPO policy achieves an error bound of order $O\big(\frac{1}{\gamma(1-2\varepsilon)}\sqrt{d/n}\big)$. Comparing this with the above error bound for DPO policy under clean preferences, we see that the cost of preference flips is a multiplicative factor of the order $\frac{1}{1-2\varepsilon}$ – the higher the (expected) number of preference flips, the higher the estimation error.

Effect of KL regularizer. Since $\gamma = O(1/e^{\beta})$, the dependence of estimation error on the KL regularizer β is of the order $g(\beta) = O(e^{\beta}/\beta)$. Hence our result won't no longer hold true when $\beta = 0$ (no regularization). In this case preference probabilities are exactly equal to $1/2$ (both actions are equally preferred), making learning impossible. Same is the case when $\beta \to \infty$ (full regularization) since one action will always be preferred over the other with probability 1, making the loss function degenerate. This points out the need for tuning β properly.

4.2. Performance Bounds of Learned Policy

In this Section, we discuss how the estimation error of $\widehat{\theta}_n$ relates to the sub-optimality gap of the policy $\widehat{\pi}_n$. We will consider log-linear policy class for ease of presentation.

It is well known that learning a near-optimal policy from an offline batch of data cannot be sample efficient without assuming the behavior policy (SFT in our case) has a good coverage over the feature space (Wang et al., 2020). To begin with, we define the population covariance matrix of centered features under a policy π:

$$\Sigma_\pi = \mathbb{E}\big[\phi(s,a)\phi(s,a)^\top\big] - \mathbb{E}[\phi(s,a)]\mathbb{E}[\phi(s,a)]^\top \ , \quad (17)$$

where the expectation is over random draws from $s \sim \rho, a \sim \pi(\cdot|s)$. Now, we define the condition number of Σ_π relative to $\Sigma_{\pi_{\rm sft}}$ (covariance matrix under SFT policy):

$$\forall \pi \in \Pi : \quad \kappa_\pi = \sup_{v \in \mathbb{R}^d} \frac{v^\top \Sigma_\pi v}{v^\top \Sigma_{\pi_{\rm sft}} v} = \frac{\lambda_{\max}(\Sigma_\pi)}{\lambda_{\min}(\Sigma_{\pi_{\rm sft}})} \ .$$

A small relative condition number helps to keep the ratio of maximum feature coverage of policy to be evaluated and minimum coverage of starting policy in check. Thus, it is important to have a good starting policy $\pi_{\rm sft}$ to ensure a small condition number. Roughly speaking, we desire an SFT policy which provides good coverage over the features.

Assumption 4.3 (Feature coverage)**.** The SFT policy satisfies the minimum eigenvalue condition: $\lambda_{\min}(\Sigma_{\pi_{\rm sft}}) > 0$.

Let $\kappa = \max_{\pi \in \Pi} \kappa_\pi$. The assumption ensures $\kappa < \infty$. The result below shows how estimation error and condition number determine the final performance of our learned policy.

Theorem 4.4 (Sub-optimality gap of $\widehat{\pi}_n$)**.** *Let $\delta \in (0,1]$ and $r^*(s,a) \le r_{\max}$ for all (s,a). Then, for log-linear policy class, we have with probability at least $1-\delta$:*

$$r^*(\pi^*) - r^*(\widehat{\pi}_n) \le r_{\max}\sqrt{\kappa/2}\,\Big\|\widehat{\theta}_n - \theta^*\Big\|_{\widehat{\Sigma}+\lambda I}$$

for $\lambda \ge C\sqrt{d\log(4d/\delta)/n}$, where C is a universal constant.

Now, plugging in the bound on estimation error (15) in Theorem 4.4, we get a sub-optimality gap of order $O\Big(\frac{\sqrt{\kappa}}{\gamma\beta(1-2\varepsilon)}\sqrt{\frac{d}{n}} + \frac{\sqrt{\kappa}d^{1/4}}{n^{1/4}}\Big)$. However, when sample feature covariance matrix $\widehat{\Sigma}$ is invertible, i.e. observed samples from SFT policy provide good coverage of the feature space, then we get $O\Big(\frac{\sqrt{\kappa}}{\gamma\beta(1-2\varepsilon)}\sqrt{\frac{d}{n}}\Big)$ suboptimality gap.

Data efficiency of rDPO under a given noise level. We can obtain a bound on sample complexity of rDPO for a given noise level and permissible sub-optimality gap. For instance, if $\widehat{\Sigma}$ is invertible, then training rDPO on $n \ge \frac{\kappa d}{\Delta^2\gamma^2\beta^2(1-2\varepsilon)^2}$ samples, we can ensure a sub-optimality gap $\le \Delta$ for the aligned model. In contrast, when samples are clean ($\varepsilon = 0$), then training vanilla DPO on $n \ge \frac{\kappa d}{\Delta^2\gamma^2\beta^2}$ samples, we can ensure a suboptimality gap $\le \Delta$. Thus, under the presence of noise, rDPO needs roughly $\frac{1}{(1-2\varepsilon)^2}$ times the samples that DPO needs under clean data. The higher the noise level, the higher the number of samples needed for rDPO.

Dimension dependence in κ. It is reasonable to expect κ to be dimension dependent, but it doesn't necessarily depend on the size of the vocabulary. To see this, consider log-linear policies with bounded features $\|\phi(s,a)\| \le L$. In this case $\lambda_{\max}(\Sigma_\pi) \le L^2$ and thus $\kappa_\pi \le \frac{L^2}{\lambda_{\min}(\Sigma_{\pi_{\rm sft}})}$. Now, $\lambda_{\min}(\Sigma_\pi)$ depends implicitly on the dimension d of features $\phi(s,a)$ and it is reasonable to assume $\lambda_{\min}(\Sigma_{\pi_{\rm sft}}) = \Theta(L^2/d)$ (Wang et al., 2020). Thus it is always possible to have $\kappa = O(d)$ (Agarwal et al., 2021).

Margin Gap. A related performance measure is the *margin* under clean distribution. The margin of a policy π_θ is defined to be the average difference of implicit rewards $\widehat{r}_\theta(s,a) = \log\frac{\pi_\theta(a|s)}{\pi_{\rm sft}(a|s)}$ of chosen and rejected actions, i.e.,

$$\mathcal{M}(\pi_\theta) = \mathbb{E}_{s\sim\rho,(y_w,y_l)\sim\pi_{\rm sft}}\left[\widehat{r}_\theta(a_w|s) - \widehat{r}_\theta(a_l|s)\right] \ .$$

Then $\mathcal{M}(\pi^*) - \mathcal{M}(\widehat{\pi}_n)$ defines the margin gap of learned policy $\widehat{\pi}_n$ from the optimal policy π^*. This metric is quite commonly used by practitioners to demonstrate performance of learned policy (von Werra et al., 2020).

Lemma 4.5 (Margin gap)**.** *Assuming $\widehat{\Sigma}$ to be invertible for log-linear policy class, the margin gap of $\widehat{\pi}_n$ satisfies*

$$\mathcal{M}(\pi^*) - \mathcal{M}(\widehat{\pi}_n) = O\Big(\frac{1}{\lambda_{\min}(\widehat{\Sigma}^{1/2})}\frac{1}{\gamma\beta(1-2\varepsilon)}\sqrt{\frac{d}{n}}\Big) \,.$$

Since $\kappa = O(1/\lambda_{\min}(\Sigma_{\pi_{\text{sft}}}))$, comparing this result with sub-optimality bound from the above paragraph, we see that both margin and sub-optimality gaps are roughly of the same order when $\widehat{\Sigma}$ has good coverage. This is also reflected in our experiments, where we see strong correlation between evaluation accuracy (on clean data) and average reward performance for any policy; see Section 6.

Generalizing to Neural Policy Class. A similar reasoning as the above can be also used to establish a sub-optimality bound for neural policy class (4). Here the relative condition number needs to be defined using the covariance matrix for the features $f_\theta(s,a)$, which depend on θ, as opposed to the feature map $\phi(s,a)$ in the log-linear case. The rest follows with an appropriate adaptation of the results above.

5. Generalizations and Extensions

Our approach to mitigate the effect of noisy preferences in data is not limited to DPO algorithm and BTL preference model. It is a general framework that can be adapted to other preference optimizations methods (e.g. SLiC, IPO) and other preference models (e.g. probit, Placket-Luce). More importantly, since DPO implicitly learns a reward function $\widehat{r}_\theta$ as we have discussed above, our method seamlessly extends to the reward training stage of the RLHF pipeline, showing versatility of our proposed approach.

Reward training in RLHF. Let us consider parameterized reward models $r_\xi(s,a)$, where $\xi \in \mathbb{R}^d$ is a parameter vector. Let ξ^* be the parameter of the latent reward model $r^*(s,a)$. Then, from (1), the true preference probabilities following BTL model are given by

$$p^*_{s,a,a'} = \mathbb{P}_{\xi^*}[a \succ a'|s] = \sigma(r_{\xi^*}(s,a) - r_{\xi^*}(s,a')) \,.$$

Similar to (7), for any $\xi \in \mathbb{R}^d$, this yields the BCE loss for a preference pair (s,a_w,a_l):

$$\mathcal{L}(\xi; s, a_w, a_l) = -\log\sigma(r_\xi(s,a_w) - r_\xi(s,a_l)) \,. \quad (18)$$

Under our random noise model (8) with flip rate ε, for a potentially noisy data $(s,\widetilde{a}_w,\widetilde{a}_l)$, one can define a loss $\widehat{\mathcal{L}}_\varepsilon(\xi; s,\widetilde{a}_w,\widetilde{a}_l)$ using (12), which will be an unbiased estimate of (18) by Lemma 3.1. Thus, using a similar argument as in Section 3, a reward model trained by minimizing this loss will be robust to noisy preferences. This trained reward model can be then directly plugged into (2) to train a language model policy. In practice (2) is solved using PPO algorithm (Schulman et al., 2017). Thus, we call this entire procedure robust PPO (or rPPO in short).

Other Optimization Methods. Instead of the BCE loss (7), SLiC (Zhao et al., 2023) minimizes a hinge loss:

$$\mathcal{L}_{\text{hinge}}(\theta; s, a_w, a_l) = \max\{0, 1 - \beta h_\theta(s,a_w,a_l)\}$$

where $1/\beta$ acts as the margin (of miss-classification). IPO (Azar et al., 2023) minimizes the square loss:

$$\mathcal{L}_{\text{IPO}}(\theta; s, a_w, a_l) = (\beta h_\theta(s,a_w,a_l) - 1/2)^2 \,.$$

A potential advantage of IPO and SLiC over DPO is that these methods don't assume any preference model like BTL and could work with general preference probabilities. Under our random noise model (8), one can define robust counterparts of both $\mathcal{L}_{\text{hinge}}$ and $\mathcal{L}_{\text{IPO}}$ using (12). Lemma 3.1 will ensure these losses under noisy data $(\widetilde{a}_w,\widetilde{a}_l)$ are unbiased estimates of those under clean data (a_w,a_l), and will help one learn a robust policy for these loss functions. Thus our approach is also not to the BTL preference model.

Other Preference Models. Our results can be extended to any preference model of the form (1) if g is strongly log-concave, i.e., $-\frac{d^2}{dz^2}\log g(z) \ge \gamma > 0$ in a closed interval around $z = 0$. For example, in the probit (also known as Thurstone) model (Thurstone, 1927), g is the CDF of standard Gaussian distribution. Thus, for any θ, the preference probabilities are $\mathbb{P}_\theta[a \succ a'|s] = \Phi(\beta h_\theta(s,a,a'))$. Since Φ is strongly log-concave in Θ (Tsukida et al., 2011), one can derive similar performance bounds under probit model too.

For the Placket-Luce (PL) model (Plackett, 1975; Luce, 2012) for K-wise comparisons between actions. Let Π be the set of all permutations $\pi : [K] \to [K]$, that denotes a ranking given by an oracle over all K actions, where $a_{\pi(j)}$ denotes the j-th ranked action. Under the PL model, we define the loss of a permutation $\pi \in \Pi$ for a question s as

$$\mathcal{L}(\theta; s, \pi) = -\log\Big(\prod_{j=1}^{K}\frac{\exp(\widehat{r}_\theta(s,a_{\pi(j)}))}{\sum_{k'=j}^{K}\exp(\widehat{r}_\theta(s,a_{\pi(k')}))}\Big) \,.$$

Noisy preferences are obtained by perturbing the true ranking π to some other ranking $\widetilde{\pi}$ with probability $\frac{\varepsilon}{N-1}$, where N is the number of possible rankings (can be at most $K!$). Then, if we define the robust-loss for noisy ranking $\widetilde{\pi}$ as

$$\widehat{\mathcal{L}}_\varepsilon(\theta; s, \widetilde{\pi}) = \frac{(N-1-\varepsilon)\mathcal{L}(\theta; s,\widetilde{\pi}) - \varepsilon\sum_{\pi' \neq \widetilde{\pi}}\mathcal{L}(\theta; s, \pi')}{(1-\varepsilon)N - 1},$$

it will be an unbiased estimate of $\mathcal{L}(\theta; s, \pi)$. This would help us to learn a robust policy under PL feedback model.

6. Experiments

In this section, we provide details about baselines, datasets, and evaluation results. We empirically evaluate rDPO on two open-ended generation tasks similar to Rafailov et al. (2023): (i) **Controlled Sentiment Generation** and

(ii) **Single-turn Dialogue**. We compare rDPO with vanilla DPO and cDPO in both tasks. In the sentiment generation task, we also include SLiC (Zhao et al., 2023) and IPO (Azar et al., 2023) as baselines. Furthermore, we compare rPPO with vanilla PPO (RLHF) and cPPO in this task.

6.1. Controlled Sentiment Generation Task

In this experiment, each prompt s represents the prefix of a movie review from the IMDb dataset (Maas et al., 2011), and the task is to generate a review (action) $a \sim \pi(\cdot|s)$ with a positive sentiment. We extract the first 20 tokens from each review in the IMDb dataset as a prefix. Subsequently, we generate reviews using a gpt2-large model supervised fine-tuned on the IMDb dataset. We generate four reviews resulting in six preference pairs for each prefix. We employ siebert/sentiment-roberta-large-english[2] as the latent (ground-truth) reward model $r^*(s,a)$. To ensure that we have a clean dataset, we only retain preference triplets (s, a_w, a_l) where $r^*(s,a_w) - r^*(s,a_l) > \tau$ where $\tau = 0.1$ is a threshold chosen for this task. This resulted in a dataset with 12000 preference triplets of which 10000 were used to train the policy, and 2000 for evaluation.

Table 1. Mean reward $\pm$ Standard Deviation of actions generated by different methods after several steps of policy training on the IMDb dataset under noise level 0.4.

Steps	DPO (On clean data)	DPO	cDPO	IPO	SLiC	rDPO
200	0.99 ± 0.03	0.93 ± 0.26	0.84 ± 0.36	0.85 ± 0.35	0.94 ± 0.22	**0.99 ± 0.00**
400	0.99 ± 0.02	0.72 ± 0.43	0.82 ± 0.37	0.83 ± 0.37	0.88 ± 0.31	**0.99 ± 0.00**
600	0.99 ± 0.00	0.88 ± 0.32	0.82 ± 0.38	0.84 ± 0.36	0.90 ± 0.29	**0.99 ± 0.00**
800	0.99 ± 0.00	0.88 ± 0.32	0.83 ± 0.36	0.83 ± 0.37	0.89 ± 0.30	**0.99 ± 0.00**
1000	0.99 ± 0.02	0.88 ± 0.32	0.83 ± 0.37	0.82 ± 0.38	0.90 ± 0.29	**0.99 ± 0.00**

We then introduce noise into this dataset by randomly flipping preferences with a probability of $\varepsilon = 0.4$. For all methods, gpt2-large is employed as the initial policy. For methods in the DPO family (vanilla DPO, rDPO, cDPO), we optimized the policy for 1000 steps with batch size 16. We do the same for IPO and SLiC. For methods in the PPO family (vanilla PPO, rPPO, cPPO), we trained a reward model on preference data for 1000 steps with batch size 16 and performed policy optimization for 1 epoch over the entire train dataset.

Table 2. Mean reward $\pm$ Standard Deviation on IMDb dataset after policy optimization. The reward model is trained on 1000 steps for all baselines, followed by running PPO for 1 epoch.

Step	PPO (On clean data)	PPO	cPPO	rPPO
1000	0.99 ± 0.00	0.78 ± 0.41	0.87 ± 0.33	**0.94 ± 0.23**

For evaluation, we generate reviews using the final policy and computed rewards using the ground-truth reward model

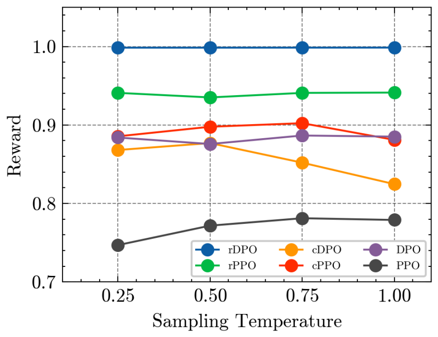

Figure 1. Mean reward on IMDb dataset at different sampling temperatures after 1000 steps.

Table 3. Percentage Improvement on win-rate vs chosen response over the initial SFT policy

Method	Improvement over SFT (%)	
	gpt2-large	Llama-2-7b
DPO ($\varepsilon = 0.0$)	22.20	45.78
cDPO ($\varepsilon = 0.1$)	18.34	39.16
rDPO ($\varepsilon = 0.1$)	**24.32**	**51.20**

r^*. The results are presented in Table 1 for the DPO family and in Table 2 for the PPO family. For reference, we also train DPO and PPO on clean data without any noise. We observe that the performance of DPO degrades with the introduction of high noise ($\varepsilon = 0.4$) in data. IPO and SLiC also suffers significantly due to noisy preferences. However, rDPO maintains performance across steps, which indicates its robustness to noise. We also observe that cDPO is not able to mitigate the effect of noise confirming the conclusions of Lemma 3.2. Similar observations are noticed for the PPO family. In Figure 1, we evaluate average rewards fetched by generations at different sampling temperatures. It is observed that rDPO and rPPO achieve the best reward by a significant margin compared to peers in their families.

6.2. Single-turn Dialogue task

In this experiment, each prompt s is a human query and each action a is a helpful response to s. We use the Anthropic helpful and harmless dataset (Bai et al., 2022a) as the preference data. We use a supervised fine-tuned gpt2-large model trained on a subset of the chosen preference data as the initial (SFT) policy. We first perform policy optimization using rDPO. As the true noise level in the dataset is unknown, we experiment with different values of $\varepsilon \in \{0.1, 0.2, 0.3, 0.4\}$. We plot the evaluation accuracy of the policy on a subset of the test set across different training steps. This is given by $\frac{1}{m}\sum_{i \in \mathcal{D}_{\text{test}}} \mathbb{1}(\widehat{r}_\theta(s_i, a_{w,i}) > \widehat{r}_\theta(s_i, a_{l,i}))$, where $\widehat{r}_\theta$ is the implicit reward defined by policy π_θ. We observed the best results with $\varepsilon = 0.1$. Subsequently, we train DPO and cDPO (with label-smoothing $\varepsilon = 0.1$) on the same data.

[2] huggingface.co/siebert/sentiment-roberta-large-english

In this experiment, as we do not have access to any latent reward model, we employ meta-llama/Llama-2-13b-chat-hf[3] to compute the win rate of policy generations against the chosen preferences on a representative subset of the test dataset. Next, to demonstrate that of our method generalizes to bigger models, we repeat this experiment with Llama-2-7b as the policy model and GPT-4 as the evaluation model. The win-rates for both experiments are tabulated in Table 3. In both cases, we observe that rDPO performs significantly better than DPO and cDPO.

7. Conclusion

We have studied the effect of noisy preferences in the final performance of language model policies. We have designed a robust loss function, which helps mitigate the effect of noise in the generations of the learned policy. We have proved first theoretical results to bound the sub-optimality gap of our robust policy. We have shown robustness of rDPO over a baseline method (DPO) and a label smoothing-based heuristic (cDPO) used by practitioners. It remains open to see how our method performs compared to other heuristics proposed in Wang et al. (2024) e.g. flipping some labels or adding an adaptive margin in the loss.

Acknowledgements

SRC would like to thank Xingyu Zhou and Gaurav Sinha for initial discussions about this work.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Agarwal, A., Kakade, S. M., Lee, J. D., and Mahajan, G. On the theory of policy gradient methods: Optimality, approximation, and distribution shift. *The Journal of Machine Learning Research*, 22(1):4431–4506, 2021.

Azar, M. G., Rowland, M., Piot, B., Guo, D., Calandriello, D., Valko, M., and Munos, R. A general theoretical paradigm to understand learning from human preferences. *arXiv preprint arXiv:2310.12036*, 2023.

Bai, Y., Jones, A., Ndousse, K., Askell, A., Chen, A., DasSarma, N., Drain, D., Fort, S., Ganguli, D., Henighan, T., Joseph, N., Kadavath, S., Kernion, J., Conerly, T., El-Showk, S., Elhage, N., Hatfield-Dodds, Z., Hernandez, D., Hume, T., Johnston, S., Kravec, S., Lovitt, L., Nanda, N., Olsson, C., Amodei, D., Brown, T., Clark, J., McCandlish, S., Olah, C., Mann, B., and Kaplan, J. Training a helpful and harmless assistant with reinforcement learning from human feedback, 2022a. URL https://arxiv.org/pdf/2204.05862.pdf.

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

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

Chen, X., Zhong, H., Yang, Z., Wang, Z., and Wang, L. Human-in-the-loop: Provably efficient preference-based reinforcement learning with general function approximation. In *International Conference on Machine Learning*, pp. 3773–3793. PMLR, 2022.

Chowdhury, S. R., Zhou, X., and Natarajan, N. Differentially private reward estimation with preference feedback. *arXiv preprint arXiv:2310.19733*, 2023.

Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S., and Amodei, D. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Hsu, D., Kakade, S., and Zhang, T. A tail inequality for quadratic forms of subgaussian random vectors. *Electronic Communications in Probability*, 17(none):1 – 6, 2012. doi: 10.1214/ECP.v17-2079. URL https://doi.org/10.1214/ECP.v17-2079.

Kaufmann, T., Weng, P., Bengs, V., and Hüllermeier, E. A survey of reinforcement learning from human feedback. *arXiv preprint arXiv:2312.14925*, 2023.

Lambert, N., Krendl Gilbert, T., and Zick, T. The history and risks of reinforcement learning and human feedback. *arXiv e-prints*, pp. arXiv–2310, 2023.

Liu, T., Zhao, Y., Joshi, R., Khalman, M., Saleh, M., Liu, P. J., and Liu, J. Statistical rejection sampling improves preference optimization. *arXiv preprint arXiv:2309.06657*, 2023.

Luce, R. D. *Individual choice behavior: A theoretical analysis*. Courier Corporation, 2012.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. Learning word vectors for sentiment analysis. In Lin, D., Matsumoto, Y., and Mihalcea, R. (eds.),

[3]huggingface.co/meta-llama/Llama-2-13b-chat-hf

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, pp. 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL https://aclanthology.org/P11-1015.

Mitchell, E. A note on dpo with noisy preferences and relationship to ipo, 2023. URL https://ericmitchell.ai/cdpo.pdf.

Müller, R., Kornblith, S., and Hinton, G. E. When does label smoothing help? *Advances in neural information processing systems*, 32, 2019.

Natarajan, N., Dhillon, I. S., Ravikumar, P. K., and Tewari, A. Learning with noisy labels. *Advances in neural information processing systems*, 26, 2013.

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

Pacchiano, A., Saha, A., and Lee, J. Dueling rl: reinforcement learning with trajectory preferences. *arXiv preprint arXiv:2111.04850*, 2021.

Patrini, G., Rozza, A., Krishna Menon, A., Nock, R., and Qu, L. Making deep neural networks robust to label noise: A loss correction approach. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1944–1952, 2017.

Plackett, R. L. The analysis of permutations. *Journal of the Royal Statistical Society Series C: Applied Statistics*, 24 (2):193–202, 1975.

Qiao, D. and Wang, Y.-X. Offline reinforcement learning with differential privacy. *arXiv preprint arXiv:2206.00810*, 2022.

Rafailov, R., Sharma, A., Mitchell, E., Ermon, S., Manning, C. D., and Finn, C. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*, 2023.

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

Stiennon, N., Ouyang, L., Wu, J., Ziegler, D., Lowe, R., Voss, C., Radford, A., Amodei, D., and Christiano, P. F. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33: 3008–3021, 2020.

Thurstone, L. L. A law of comparative judgment. *Psychological review*, 34(4):273, 1927.

Tropp, J. A. et al. An introduction to matrix concentration inequalities. *Foundations and Trends® in Machine Learning*, 8(1-2):1–230, 2015.

Tsukida, K., Gupta, M. R., et al. How to analyze paired comparison data. *Department of Electrical Engineering University of Washington, Tech. Rep. UWEETR-2011-0004*, 1, 2011.

von Werra, L., Belkada, Y., Tunstall, L., Beeching, E., Thrush, T., Lambert, N., and Huang, S. Trl: Transformer reinforcement learning. https://github.com/huggingface/trl, 2020.

Wang, B., Zheng, R., Chen, L., Liu, Y., Dou, S., Huang, C., Shen, W., Jin, S., Zhou, E., Shi, C., et al. Secrets of rlhf in large language models part ii: Reward modeling. *arXiv preprint arXiv:2401.06080*, 2024.

Wang, R., Foster, D. P., and Kakade, S. M. What are the statistical limits of offline rl with linear function approximation? *arXiv preprint arXiv:2010.11895*, 2020.

Yuan, Z., Yuan, H., Tan, C., Wang, W., Huang, S., and Huang, F. Rrhf: Rank responses to align language models with human feedback without tears. *arXiv preprint arXiv:2304.05302*, 2023.

Zhan, W., Uehara, M., Kallus, N., Lee, J. D., and Sun, W. Provable offline reinforcement learning with human feedback. *arXiv preprint arXiv:2305.14816*, 2023.

Zhao, Y., Joshi, R., Liu, T., Khalman, M., Saleh, M., and Liu, P. J. Slic-hf: Sequence likelihood calibration with human feedback. *arXiv preprint arXiv:2305.10425*, 2023.

Zheng, R., Dou, S., Gao, S., Hua, Y., Shen, W., Wang, B., Liu, Y., Jin, S., Liu, Q., Zhou, Y., et al. Secrets of rlhf in large language models part i: Ppo. *arXiv preprint arXiv:2307.04964*, 2023.

Zhu, B., Jiao, J., and Jordan, M. I. Principled reinforcement learning with human feedback from pairwise or k-wise comparisons. *arXiv preprint arXiv:2301.11270*, 2023.

Zhu, B., Jordan, M. I., and Jiao, J. Iterative data smoothing: Mitigating reward overfitting and overoptimization in rlhf. *arXiv preprint arXiv:2401.16335*, 2024.

Appendix

A. Missing Details

A.1. Proof of Lemma 3.1

It is easy to see that

$$\begin{aligned}&\mathbb{E}_\varepsilon\left[\widehat{\mathcal{L}}_\varepsilon(\theta; s, \widetilde{a}_w, \widetilde{a}_l) | a_w, a_l\right]\\&= \frac{(1-\varepsilon)^2\mathcal{L}(\theta; s, a_w, a_l) - \varepsilon(1-\varepsilon)\mathcal{L}(\theta; s, a_l, a_w)}{1-2\varepsilon} + \frac{\varepsilon(1-\varepsilon)\mathcal{L}(\theta; s, a_l, a_w) - \varepsilon^2\mathcal{L}(\theta; s, a_w, a_l)}{1-2\varepsilon}\\&= \mathcal{L}(\theta; s, a_w, a_l)\ .\end{aligned}$$

A.2. Variance of rDPO loss

First, define the un-normalized rDPO loss

$$\widetilde{\mathcal{L}}_\varepsilon(\theta; s, \widetilde{a}_w, \widetilde{a}_l) := (1-2\varepsilon)\widehat{\mathcal{L}}_\varepsilon(\theta; s, \widetilde{a}_w, \widetilde{a}_l) = (1-\varepsilon)\mathcal{L}(\theta; s, \widetilde{a}_w, \widetilde{a}_l) - \varepsilon\mathcal{L}(\theta; s, \widetilde{a}_l, \widetilde{a}_w)\ .$$

Its variance is given by

$$\mathrm{Var}\left[\widetilde{\mathcal{L}}_\varepsilon(\theta; s, \widetilde{a}_w, \widetilde{a}_l)\right] = \mathbb{E}\left[\widetilde{\mathcal{L}}_\varepsilon(\theta; s, \widetilde{a}_w, \widetilde{a}_l)^2\right] - \mathbb{E}\left[\widetilde{\mathcal{L}}_\varepsilon(\theta; s, \widetilde{a}_w, \widetilde{a}_l)\right]^2\ .$$

From Lemma 3.1, we have

$$\mathbb{E}\left[\widetilde{\mathcal{L}}_\varepsilon(\theta; s, \widetilde{a}_w, \widetilde{a}_l)\right] = (1-2\varepsilon)\mathcal{L}(\theta; s, a_w, a_l)\ .$$

Furthermore, we have

$$\begin{aligned}\mathbb{E}\left[\widetilde{\mathcal{L}}_\varepsilon(\theta; s, \widetilde{a}_w, \widetilde{a}_l)^2\right] = (1-\varepsilon)^2\mathbb{E}\left[\mathcal{L}(\theta; s, \widetilde{a}_w, \widetilde{a}_l)^2\right] + \varepsilon^2\mathbb{E}\left[\mathcal{L}(\theta; s, \widetilde{a}_l, \widetilde{a}_w)^2\right]\\ - 2\varepsilon(1-\varepsilon)\mathbb{E}\left[\mathcal{L}(\theta; s, \widetilde{a}_w, \widetilde{a}_l)\mathcal{L}(\theta; s, \widetilde{a}_l, \widetilde{a}_w)\right]\ .\end{aligned}$$

Now observe that

$$\begin{aligned}\mathbb{E}\left[\mathcal{L}(\theta; s, \widetilde{a}_w, \widetilde{a}_l)^2\right] &= (1-\varepsilon)\mathcal{L}(\theta; s, a_w, a_l)^2 + \varepsilon\mathcal{L}(\theta; s, a_l, a_w)^2\ ,\\ \mathbb{E}\left[\mathcal{L}(\theta; s, \widetilde{a}_l, \widetilde{a}_w)^2\right] &= (1-\varepsilon)\mathcal{L}(\theta; s, a_l, a_w)^2 + \varepsilon\mathcal{L}(\theta; s, a_w, a_l)^2\ ,\\ \mathbb{E}\left[\mathcal{L}(\theta; s, \widetilde{a}_w, \widetilde{a}_l)\mathcal{L}(\theta; s, \widetilde{a}_l, \widetilde{a}_w)\right] &= \mathcal{L}(\theta; s, a_w, a_l)\mathcal{L}(\theta; s, a_l, a_w)\ .\end{aligned}$$

Combining all these, we get

$$\begin{aligned}\mathbb{E}\left[\widetilde{\mathcal{L}}_\varepsilon(\theta; s, \widetilde{a}_w, \widetilde{a}_l)^2\right] = (1-3\varepsilon+3\varepsilon^2)\mathcal{L}(\theta; s, a_w, a_l)^2 + \varepsilon(1-\varepsilon)\mathcal{L}(\theta; s, a_l, a_w)^2\\ - 2\varepsilon(1-\varepsilon)\mathcal{L}(\theta; s, a_w, a_l)\mathcal{L}(\theta; s, a_l, a_w)\ .\end{aligned}$$

Therefore, the variance of the un-normalized rDPO loss is given by

$$\begin{aligned}&\mathrm{Var}\left[\widetilde{\mathcal{L}}_\varepsilon(\theta; s, \widetilde{a}_w, \widetilde{a}_l)\right]\\&= \left[(1-3\varepsilon+3\varepsilon^2) - (1-2\varepsilon)^2\right]\mathcal{L}(\theta; s, a_w, a_l)^2 + \varepsilon(1-\varepsilon)\mathcal{L}(\theta; s, a_l, a_w)^2 - 2\varepsilon(1-\varepsilon)\mathcal{L}(\theta; s, a_w, a_l)\mathcal{L}(\theta; s, a_l, a_w)\\&= \varepsilon(1-\varepsilon)\left[\mathcal{L}(\theta; s, a_w, a_l)^2 + \mathcal{L}(\theta; s, a_l, a_w)^2 - 2\mathcal{L}(\theta; s, a_w, a_l)\mathcal{L}(\theta; s, a_l, a_w)\right]\\&= \varepsilon(1-\varepsilon)\left[\mathcal{L}(\theta; s, a_w, a_l) - \mathcal{L}(\theta; s, a_l, a_w)\right]^2.\end{aligned}$$

A.3. Proof of Lemma 3.2

The gradients of the rDPO loss $\widehat{\mathcal{L}}_\varepsilon$ with respect to the parameters θ can be written as

$$\begin{aligned}\nabla_\theta\widehat{\mathcal{L}}_\varepsilon(\theta; s, \widetilde{a}_w, \widetilde{a}_l) &= \frac{(1-\varepsilon)\nabla_\theta\mathcal{L}(\theta; s, \widetilde{a}_w, \widetilde{a}_l) - \varepsilon\nabla_\theta\mathcal{L}(\theta; s, \widetilde{a}_l, \widetilde{a}_w)}{1-2\varepsilon}\\ &= -\beta\cdot\widehat{\zeta}_{\theta,\varepsilon}\cdot\left(\nabla_\theta\log\pi_\theta(\widetilde{a}_w|s) - \nabla_\theta\log\pi_\theta(\widetilde{a}_l|s)\right)\ ,\end{aligned}$$

where the weights $\widehat{\zeta}_{\theta,\varepsilon}$ are given by

$$\begin{aligned}\widehat{\zeta}_{\theta,\varepsilon} &= \frac{1-\varepsilon}{1-2\varepsilon}\sigma(\beta h_\theta(s,\widetilde{a}_l,\widetilde{a}_w)) + \frac{\varepsilon}{1-2\varepsilon}\sigma(\beta h_\theta(s,\widetilde{a}_w,\widetilde{a}_l)) \\ &= \frac{1-\varepsilon}{1-2\varepsilon} - \sigma(\beta h_\theta(s,\widetilde{a}_w,\widetilde{a}_l)) = \frac{\varepsilon}{1-2\varepsilon} + \sigma(\beta h_\theta(s,\widetilde{a}_l,\widetilde{a}_w)) = \zeta_\theta + \frac{\varepsilon}{1-2\varepsilon},\end{aligned}$$

where ζ_θ are the weights of DPO gradients.

The gradient of cDPO loss is given by

$$\begin{aligned}\nabla_\theta \bar{\mathcal{L}}_\varepsilon(\theta; s, \widetilde{a}_w, \widetilde{a}_l) &= (1-\varepsilon)\nabla_\theta \mathcal{L}(\theta; s, \widetilde{a}_w, \widetilde{a}_l) + \varepsilon \nabla_\theta \mathcal{L}(\theta; s, \widetilde{a}_l, \widetilde{a}_w) \\ &= -\beta \cdot \bar{\zeta}_{\theta,\varepsilon} \cdot \big(\nabla_\theta \log \pi_\theta(\widetilde{a}_w|s) - \nabla_\theta \log \pi_\theta(\widetilde{a}_l|s)\big) ,\end{aligned}$$

where the weights are $\bar{\zeta}_{\theta,\varepsilon} = (1-\varepsilon)\sigma(\beta h_\theta(s,\widetilde{a}_l,\widetilde{a}_w)) - \varepsilon\sigma(\beta h_\theta(s,\widetilde{a}_w,\widetilde{a}_l))$. It holds that

$$\bar{\zeta}_{\theta,\varepsilon} = \sigma(\beta h_\theta(s,\widetilde{a}_l,\widetilde{a}_w)) - \varepsilon = \zeta_\theta - \varepsilon = \widehat{\zeta}_{\theta,\varepsilon} - \frac{2\varepsilon(1-\varepsilon)}{1-2\varepsilon} .$$

A.4. Proof of Theorem 4.2

For the neural policy of the form (4), we have

$$h_\theta(s,a,a') = [f_\theta(s,a) - f_\theta(s,a')] - [f_{\theta_0}(s,a) - f_{\theta_0}(s,a')] .$$

Then from Assumption 4.1, we have

$$\begin{aligned} |h_\theta(s,a,a')| &\le |f_\theta(s,a) - f_{\theta_0}(s,a)| + |f_\theta(s,a') - f_{\theta_0}(s,a')| \le 2\alpha_0, \\ \|\nabla h_\theta(s,a,a')\| &= \|\nabla f_\theta(s,a) - \nabla f_\theta(s,a')\| \le 2\alpha_1 , \\ \big\|\nabla^2 h_\theta(s,a,a')\big\|_{\text{op}} &= \big\|\nabla^2 f_\theta(s,a) - \nabla^2 f_\theta(s,a')\big\|_{\text{op}} \le 2\alpha_2 . \end{aligned} \tag{19}$$

Now, we express the population DPO loss $\mathbb{E}_{s,a_w,a_l}\Big[\mathcal{L}(\theta; s, a_w, a_l)\Big]$ by incorporating preference probabilities $p^*_{s,a,a'}$ as

$$\mathcal{L}(\theta) = -\mathbb{E}_{s,a,a',y}\Big[- y \log \sigma(\beta h_\theta(s,a,a')) + (1-y)\log(1-\sigma(\beta h_\theta(s,a,a'))\Big] ,$$

where y is a Bernoulli random variable with mean $p^*_{s,a,a'} = \sigma(\beta h_{\theta^*}(s,a,a'))$.

Similarly, under the random noise model (8), let each $\widetilde{y}_i$ be Bernoulli distributed with probability $\mathbb{P}_{\theta^*,\varepsilon}[\widetilde{a}_{w,i} \succ \widetilde{a}_{l,i}|s_i]$, where $\mathbb{P}_{\theta,\varepsilon}[a \succ a'|s]$ is defined in (10).

Denote $z_i = (s_i, \widetilde{a}_{w,i}, \widetilde{a}_{l,i})$. Then, our de-biased loss function (12) can be re-written as[4]

$$\begin{aligned}\widehat{\mathcal{L}}_\varepsilon(\theta) = -\frac{1}{n}\sum_{i=1}^n \Bigg[& \mathbb{1}(\widetilde{y}_i = 1)\Big((1-\varepsilon)\log\sigma(\beta h_\theta(z_i)) - \varepsilon \log(1-\sigma(\beta h_\theta(z_i))\Big) \\ & + \mathbb{1}(\widetilde{y}_i = 0)\Big((1-\varepsilon)\log(1-\sigma(\beta h_\theta(z_i)) - \varepsilon\log\sigma(\beta h_\theta(z_i))\Big)\Bigg] .\end{aligned}$$

The gradient of the loss function is given by $\nabla \widehat{\mathcal{L}}_\varepsilon(\theta) = -\frac{\beta}{n}\sum_{i=1}^n V_{\theta,i}\nabla h_\theta(z_i) = -\frac{\beta}{n} Z_\theta^\top V_\theta$, where

$$V_{\theta,i} = \mathbb{1}(\widetilde{y}_i = 1)\left(\frac{\sigma'(\beta h_\theta(z_i))}{\sigma(\beta h_\theta(z_i))}(1-\varepsilon) + \frac{\sigma'(\beta h_\theta(z_i))}{1-\sigma(\beta h_\theta(z_i))}\varepsilon\right) - \mathbb{1}(\widetilde{y}_i = 0)\left(\frac{\sigma'(\beta h_\theta(z_i))}{1-\sigma(\beta h_\theta(z_i))}(1-\varepsilon) + \frac{\sigma'(\beta h_\theta(z_i))}{\sigma(\beta h_\theta(z_i))}\varepsilon\right) .$$

It holds that for $\theta = \theta^*$:

$$\begin{aligned}\mathbb{E}_\theta[V_{\theta,i}|z_i] = & \Big(\sigma(\beta h_\theta(z_i))(1-\varepsilon) + (1-\sigma(\beta h_\theta(z_i)))\varepsilon\Big)\left(\frac{\sigma'(\beta h_\theta(z_i))}{\sigma(\beta h_\theta(z_i))}(1-\varepsilon) + \frac{\sigma'(\beta h_\theta(z_i))}{1-\sigma(\beta h_\theta(z_i)}\varepsilon\right) \\ & - \Big((1-\sigma(\beta h_\theta(z_i)))(1-\varepsilon) + \sigma(\beta h_\theta(z_i))\varepsilon\Big)\left(\frac{\sigma'(\beta h_\theta(z_i))}{1-\sigma(\beta h_\theta(z_i))}(1-\varepsilon) + \frac{\sigma'(\beta h_\theta(z_i))}{\sigma(\beta h_\theta(z_i))}\varepsilon\right) \\ & = 0 .\end{aligned}$$

[4]We ignore the normalization by $1 - 2\varepsilon$, since it doesn't affect the minimizer of the loss.

Furthermore, we have

$$\begin{aligned}|V_{\theta,i}|_{\widetilde{y}_i=1} &= (1-\sigma(\beta h_\theta(z_i)))(1-\varepsilon) + \sigma(\beta h_\theta(z_i))\varepsilon =: \widetilde{p}_{i,0} \le 1,\\ |V_{\theta,i}|_{\widetilde{y}_i=0} &= \sigma(\beta h_\theta(z_i))(1-\varepsilon) + (1-\sigma(\beta h_\theta(z_i)))\varepsilon =: \widetilde{p}_{i,1} \le 1 .\end{aligned}$$

Therefore, it holds that $V_{\theta^*,i}$ is zero-mean and 1-sub-Gaussian under the conditional distribution $\mathbb{P}_{\theta^*}[\cdot|z_i]$.

Now the Hessian of the loss function is given by

$$\begin{aligned}\nabla^2 \widehat{\mathcal{L}}_\varepsilon(\theta) = \frac{1}{n}\sum_{i=1}^n \Bigg[&\mathbb{1}(\widetilde{y}_i = 1)\big(\varepsilon \nabla^2 \log(1-\sigma(\beta h_\theta(z_i))) - (1-\varepsilon)\nabla^2 \log \sigma(\beta h_\theta(z_i))\big)\\ &+ \mathbb{1}(\widetilde{y}_i = 0)\big(\varepsilon \nabla^2 \log \sigma(\beta h_\theta(z_i)) - (1-\varepsilon)\nabla^2 \log(1-\sigma(\beta h_\theta(z_i)))\big)\Bigg] ,\end{aligned}$$

where

$$\begin{aligned}\nabla^2 \log \sigma(\beta h_\theta(z_i)) &= \beta^2 \frac{\sigma''(\beta h_\theta(z_i))\sigma(\beta h_\theta(z_i)) - \sigma'(\beta h_\theta(z_i))^2}{\sigma(\beta h_\theta(z_i))^2} \nabla h_\theta(z_i)\nabla h_\theta(z_i)^\top\\ &\quad + \beta(1-\sigma(\beta h_\theta(z_i)))\nabla^2 h_\theta(z_i),\\ \nabla^2 \log(1-\sigma(\beta h_\theta(z_i))) &= -\beta^2 \frac{\sigma''(\beta h_\theta(z_i))(1-\sigma(\beta h_\theta(z_i))) + \sigma'(\beta h_\theta(z_i))^2}{(1-\sigma(\beta h_\theta(z_i)))^2} \nabla h_\theta(z_i)\nabla h_\theta(z_i)^\top\\ &\quad - \beta\sigma(\beta h_\theta(z_i))\nabla^2 h_\theta(z_i) .\end{aligned}$$

Using $\sigma''(z) = \sigma'(z)(1-2\sigma(z))$, we get

$$\begin{aligned}\nabla^2 \log \sigma(\beta h_\theta(z_i)) &= -\beta^2 \sigma'(\beta h_\theta(z_i))\nabla h_\theta(z_i)\nabla h_\theta(z_i)^\top + \beta(1-\sigma(\beta h_\theta(z_i))))\nabla^2 h_\theta(z_i)\\ \nabla^2 \log(1-\sigma(\beta h_\theta(z_i))) &= -\beta^2 \sigma'(\beta h_\theta(z_i))\nabla h_\theta(z_i)\nabla h_\theta(z_i)^\top - \beta\sigma(\beta h_\theta(z_i))\nabla^2 h_\theta(z_i) .\end{aligned}$$

Hence, the Hessian of the loss function takes the form

$$\begin{aligned}\nabla^2 \widehat{\mathcal{L}}_\varepsilon(\theta) &= (1-2\varepsilon)\beta^2 \frac{1}{n}\sum_{i=1}^n \sigma'(\beta h_\theta(z_i))\nabla h_\theta(z_i)\nabla h_\theta(z_i)^\top\\ &\quad - \frac{\beta}{n}\sum_{i=1}^n \mathbb{1}(\widetilde{y}_i = 1)\Big(\sigma(\beta h_\theta(z_i))\varepsilon + (1-\sigma(\beta h_\theta(z_i)))(1-\varepsilon)\Big)\nabla^2 h_\theta(z_i)\\ &\quad + \frac{\beta}{n}\sum_{i=1}^n \mathbb{1}(\widetilde{y}_i = 0)\Big(\sigma(\beta h_\theta(z_i))(1-\varepsilon) + (1-\sigma(\beta h_\theta(z_i)))\varepsilon\Big)\nabla^2 h_\theta(z_i)\\ &= \beta^2(1-2\varepsilon)\frac{1}{n}\sum_{i=1}^n \sigma'(\beta h_\theta(z_i))\nabla h_\theta(z_i)\nabla h_\theta(z_i)^\top\\ &\quad - \frac{\beta}{n}\sum_{i=1}^n \mathbb{1}(\widetilde{y}_i = 1)\widetilde{p}_{i,0}\nabla^2 h_\theta(z_i) + \frac{\beta}{n}\sum_{i=1}^n \mathbb{1}(\widetilde{y}_i = 0)\widetilde{p}_{i,1}\nabla^2 h_\theta(z_i)\\ &\geqslant \gamma\beta^2(1-2\varepsilon)\frac{1}{n}\sum_{i=1}^n \nabla h_\theta(z_i)\nabla h_\theta(z_i)^\top - 2\beta\alpha_2 I ,\end{aligned}$$

which holds by (19) and observing that $\sigma'(\beta h_\theta(z_i)) \ge \gamma$ for all $\theta \in \Theta$, where $\gamma = \frac{1}{2+\exp(-4\beta\alpha_0)+\exp(4\beta\alpha_0)}$, and due to the fact that $\varepsilon < 1/2$.

Defining $v_i = \nabla h_\theta(z_i) - \nabla h_{\theta^*}(z_i)$, we have

$$\begin{aligned}\nabla h_\theta(z_i)\nabla h_\theta(z_i)^\top &= \nabla h_{\theta^*}(z_i)\nabla h_{\theta^*}(z_i)^\top + \nabla h_{\theta^*}(z_i)v_i^\top + v_i\nabla h_{\theta^*}(z_i)^\top + v_i v_i^\top\\ &\succeq \nabla h_{\theta^*}(z_i)\nabla h_{\theta^*}(z_i)^\top + \nabla h_{\theta^*}(z_i)v_i^\top + v_i\nabla h_{\theta^*}(z_i)^\top .\end{aligned}$$

By (19) and noting that $\|\theta\| \leqslant B$, we have $\|\nabla h_{\theta^*}(z_i)\| \leqslant 2\alpha_1$ and $\|v_i\| \leqslant 2\alpha_2 \|\theta^* - \theta\| \le 2\alpha_2 B$. Then, using simple algebra, we have for all $u \in \mathbb{R}^d$:

$$u^\top \nabla^2 \widehat{\mathcal{L}}_\varepsilon(\theta) u \geqslant \frac{\gamma\beta^2(1-2\varepsilon)}{n} \left\| Z_{\theta^*} u \right\|^2 - 2\alpha_2(\beta + 2\gamma\beta^2(1-2\varepsilon)\alpha_1 B) \left\| u \right\|^2 \ .$$

Since $\theta^* \in \Theta$, introducing the error vector $\Delta = \widehat{\theta}_n - \theta^*$, we conclude that

$$\gamma\beta^2(1-2\varepsilon) \left\|\Delta\right\|^2_{\Sigma_{\theta^*}} \leqslant \left\| \nabla \widehat{\mathcal{L}}_\varepsilon(\theta^*) \right\|_{(\widehat{\Sigma}_{\theta^*}+\lambda I)^{-1}} \left\|\Delta\right\|_{(\widehat{\Sigma}_{\theta^*}+\lambda I)} + 2\alpha_2\beta(1+2\beta\gamma(1-2\varepsilon)\alpha_1 B) \left\|\Delta\right\|^2$$

for some $\lambda > 0$. Introducing $M_{\theta^*} = \frac{1}{n^2} Z_{\theta^*}(\widehat{\Sigma}_{\theta^*} + \lambda I)^{-1} Z_{\theta^*}^\top$, we now have $\left\| \nabla \widehat{\mathcal{L}}_\varepsilon(\theta^*) \right\|^2_{(\widehat{\Sigma}_{\theta^*}+\lambda I)^{-1}} = \beta^2 V_{\theta^*}^\top M_{\theta^*} V_{\theta^*}$. Then, the Bernstein's inequality for sub-Gaussian random variables in quadratic form (Hsu et al., 2012, Theorem 2.1) implies that with probability at least $1-\delta$,

$$\begin{aligned}\left\| \nabla \widehat{\mathcal{L}}_\varepsilon(\theta^*) \right\|^2_{(\widehat{\Sigma}_{\theta^*}+\lambda I)^{-1}} = \beta^2 V_{\theta^*}^\top M_{\theta^*} V_{\theta^*} &\leqslant \beta^2 \left(\operatorname{tr}(M_{\theta^*}) + 2\sqrt{\operatorname{tr}(M_{\theta^*}^\top M_{\theta^*}) \log(1/\delta)} + 2 \left\| M_{\theta^*} \right\| \log(1/\delta) \right) \\ &\leqslant C_1 \cdot \beta^2 \cdot \frac{d + \log(1/\delta)}{n}\end{aligned}$$

for some $C_1 > 0$. Here we have used that $\operatorname{tr}(M_{\theta^*}) \le d/n$, $\operatorname{tr}(M_{\theta^*}^\top M_{\theta^*}) \le d/n^2$ and $\|M_{\theta^*}\| \le 1/n$. Noting that $\|\Delta\| \leqslant B$, this gives us

$$\begin{aligned}\gamma\beta^2(1-2\varepsilon) \left\|\Delta\right\|^2_{\widehat{\Sigma}_{\theta^*}+\lambda I} &\leqslant \left\| \nabla \widehat{\mathcal{L}}_\varepsilon(\theta^*) \right\|_{(\Sigma_{\theta^*}+\lambda I)^{-1}} \left\|\Delta\right\|_{(\widehat{\Sigma}_{\theta^*}+\lambda I)} + (\lambda\gamma\beta^2(1-2\varepsilon) + 2\alpha_2\beta(1+2\beta\gamma(1-2\varepsilon)\alpha_1 B))B^2 \\ &\leqslant \sqrt{C_1 \cdot \beta^2 \cdot \frac{d+\log(1/\delta)}{n}} \left\|\Delta\right\|_{(\widehat{\Sigma}_{\theta^*}+\lambda I)} + (\lambda\gamma\beta^2(1-2\varepsilon) + 2\alpha_2\beta(1+2\beta\gamma(1-2\varepsilon)\alpha_1 B))B^2 \ .\end{aligned}$$

Solving for the above inequality, we get

$$\left\|\Delta\right\|_{(\widehat{\Sigma}_{\theta^*}+\lambda I)} \leqslant C_2 \cdot \sqrt{\frac{1}{\gamma^2\beta^2(1-2\varepsilon)^2} \cdot \frac{d+\log(1/\delta)}{n} + (\lambda + \frac{\alpha_2}{\gamma\beta(1-2\varepsilon)} + \alpha_1\alpha_2 B)B^2}$$

for some constant $C_2 > 0$. Hence, we get

$$\left\| \widehat{\theta}_n - \theta^* \right\|_{(\widehat{\Sigma}_{\theta^*}+\lambda I)} \leqslant \frac{C}{\gamma\beta(1-2\varepsilon)} \cdot \sqrt{\frac{d+\log(1/\delta)}{n}} + C' \cdot B\sqrt{\lambda + \frac{\alpha_2}{\gamma\beta(1-2\varepsilon)} + \alpha_1\alpha_2 B},$$

for some $C, C' > 0$. This completes our proof.

A.5. Proof of Theorem 4.4

Define the population covariance matrix of centered gradients of the function $f_\theta(s,a)$ under policy π:

$$\Sigma_\pi = \mathbb{E}_{s\sim\rho, a\sim\pi(\cdot|s)} \left[g_\theta(s,a) g_\theta(s,a)^\top \right] , \tag{20}$$

where $g_\theta(s,a) = \nabla f_\theta(s,a) - \mathbb{E}_{a'\sim\pi(\cdot|s)}[\nabla f_\theta(s,a')]$ denotes the centered features. For log-linear policies, $\nabla f_\theta(s,a) = \phi(s,a)$ and $g_\theta(s,a) = \phi(s,a) - \mathbb{E}_\theta[\phi(s,a')]$, which gives

$$\Sigma_\pi = \mathbb{E}_{s\sim\rho, a\sim\pi(\cdot|s)} \left[\phi(s,a)\phi(s,a)^\top \right] - \mathbb{E}_{s\sim\rho, a\sim\pi(\cdot|s)} \left[\phi(s,a) \right] \mathbb{E}_{s\sim\rho, a\sim\pi(\cdot|s)} \left[\phi(s,a) \right]^\top \ .$$

Define sample covariance and population matrix of feature differences under clean data $\mathcal{D}$;

$$\widehat{\Sigma} = \frac{1}{n}\sum_{i=1}^n \left(\phi(s_i,a_{w,i}) - \phi(s_i,a_{l,i})\right)\left(\phi(s_i,a_{w,i}) - \phi(s_i,a_{l,i})\right)^\top ,$$

$$\Sigma_{\pi,\text{diff}} = \mathbb{E}_{s\sim\rho, a,a'\sim\pi(\cdot|s)} \left[\left(\phi(s,a) - \phi(s,a')\right)\left(\phi(s,a) - \phi(s,a')\right)^\top \right] \ .$$

Since a, a' are independent samples from policy $\pi(\cdot|s)$, it holds that

$$\Sigma_{\pi,\text{diff}} = 2\Sigma_\pi$$

Since $(a_{w,i}, a_{li,i})$ are independent samples from SFT policy $\pi_{\text{sft}}(\cdot|s)$, by matrix concentration inequality (Tropp et al., 2015), we have the following lemma.

Lemma A.1. *With probability at least* $1-\delta$*, for some universal constant* C*, we have*

$$\left\|\widehat{\Sigma}-\Sigma_{\pi_{\mathrm{sft}},\mathrm{diff}}\right\|_2 \le C\sqrt{d\log(4d/\delta)/n}\,.$$

This implies, for $\lambda \ge C\sqrt{d\log(4d/\delta)/n}$, with probability at least $1-\delta$:

$$\begin{aligned}\widehat{\Sigma}+\lambda I &\succeq \Sigma_{\pi_{\mathrm{sft}},\mathrm{diff}}+\lambda I - C\sqrt{d\log(4d/\delta)/n}\\ &\succeq \Sigma_{\pi_{\mathrm{sft}},\mathrm{diff}} = 2\Sigma_{\pi_{\mathrm{sft}}}\,. \end{aligned}\tag{21}$$

Now, we bound the sub-optimality gap conditioned on this high-confidence event. Since $r^*(s,a)\le r_{\max}$ for all (s,a), we have the sub-optimality gap:

$$\begin{aligned} r^*(\pi^*)-r^*(\widehat{\pi}_n) &= \mathbb{E}_{s\sim\rho,a\sim\pi^*(\cdot|s)}\left[r^*(s,a)\right]-\mathbb{E}_{s\sim\rho,a\sim\widehat{\pi}_n(\cdot|s)}\left[r^*(s,a)\right]\\ &\le r_{\max}\mathbb{E}_{s\sim\rho}\left[\mathtt{TV}\left(\pi^*(\cdot|s),\widehat{\pi}_n(\cdot|s)\right)\right]\\ &\le r_{\max}\left[\mathbb{E}_{s\sim\rho}\sqrt{2\,\mathtt{KL}\left(\pi^*(\cdot|s),\widehat{\pi}_n(\cdot|s)\right)}\right]\\ &\le r_{\max}\sqrt{2\mathbb{E}_{s\sim\rho}\left[\mathtt{KL}\left(\pi^*(\cdot|s),\widehat{\pi}_n(\cdot|s)\right)\right]}\,,\end{aligned}$$

where the second step follows from Pinsker's inequality and the last step is due to Jensen's inequality.

Since the neural policy class (4) belongs to the exponential family of distributions, it holds that $\mathtt{KL}\left(\pi_\theta(\cdot|s),\pi_{\theta'}(\cdot|s)\right)=\mathcal{B}_{\mathcal{L}_s}(\theta',\theta)$, where $\mathcal{B}_{\mathcal{L}_s}$ is the Bregman divergence with potential function $\mathcal{L}_s(\theta)=\log\sum_{a'\in\mathcal{A}}f_\theta(s,a')$. It is defined as

$$\mathcal{B}_{\mathcal{L}_s}(\theta',\theta)\overset{\mathrm{def}}{=}\mathcal{L}_s(\theta')-\mathcal{L}_s(\theta)-\langle\theta'-\theta,\nabla\mathcal{L}_s(\theta)\rangle\,.$$

Therefore, we get

$$\mathtt{KL}\left(\pi^*(\cdot|s),\widehat{\pi}_n(\cdot|s)\right)=\mathcal{L}_s(\widehat{\theta}_n)-\mathcal{L}_s(\theta^*)-\langle\widehat{\theta}_n-\theta^*,\nabla\mathcal{L}_s(\theta^*)\rangle=\frac{1}{2}(\widehat{\theta}_n-\theta^*)^\top\nabla^2\mathcal{L}_s(\theta)(\widehat{\theta}_n-\theta^*)$$

for some $\theta\in\{t\theta^*+(1-t)\widehat{\theta}_n : t\in[0,1]\}$ using Taylor's approximation.

Now, for log-linear policy, we have $\mathbb{E}_{s\sim\rho}\left[\nabla^2\mathcal{L}_s(\theta)\right]=\Sigma_{\pi_\theta}$. Then, we can upper bound the sub-optimality gap using relative condition number κ as

$$\begin{aligned} r^*(\pi^*)-r^*(\widehat{\pi}_n) &\le r_{\max}\left\|\widehat{\theta}_n-\theta^*\right\|_{\Sigma_{\pi_\theta}}\\ &= r_{\max}\left\|\widehat{\theta}_n-\theta^*\right\|_{\widehat{\Sigma}+\lambda I}\sqrt{\frac{(\widehat{\theta}_n-\theta^*)^\top\Sigma_{\pi_\theta}(\widehat{\theta}_n-\theta^*)}{(\widehat{\theta}_n-\theta^*)^\top(\widehat{\Sigma}+\lambda I)(\widehat{\theta}_n-\theta^*)}}\\ &\le \frac{r_{\max}}{\sqrt{2}}\left\|\widehat{\theta}_n-\theta^*\right\|_{\widehat{\Sigma}+\lambda I}\sqrt{\frac{(\widehat{\theta}_n-\theta^*)^\top\Sigma_{\pi_\theta}(\widehat{\theta}_n-\theta^*)}{(\widehat{\theta}_n-\theta^*)^\top\Sigma_{\pi_{\mathrm{sft}}}(\widehat{\theta}_n-\theta^*)}}\\ &\le \frac{r_{\max}}{\sqrt{2}}\left\|\widehat{\theta}_n-\theta^*\right\|_{\widehat{\Sigma}+\lambda I}\sqrt{\sup_{v\in\mathbb{R}^d}\frac{v^\top\Sigma_{\pi_\theta}v}{v^\top\Sigma_{\pi_{\mathrm{sft}}}v}}\\ &= \frac{r_{\max}\sqrt{\kappa_{\pi_\theta}}}{\sqrt{2}}\left\|\widehat{\theta}_n-\theta^*\right\|_{\widehat{\Sigma}+\lambda I}\le\frac{r_{\max}\sqrt{\kappa}}{\sqrt{2}}\left\|\widehat{\theta}_n-\theta^*\right\|_{\widehat{\Sigma}+\lambda I}\,.\end{aligned}$$

Here, the third step follows from (21), the fifth step holds by definition of (relative) condition number and in the final step, we use that $\kappa=\max_{\pi\in\Pi}\kappa_\pi$. This completes our proof.

A.6. Proof of Lemma 4.5

Recall that $\widehat{r}_\theta(s,a) = \log \frac{\pi_\theta(a|s)}{\pi_{\text{sft}}(a|s)}$ denotes the implicit reward defined by trained and SFT policies π_θ and π_{sft}. Then, we have the expected margin gap under clean distribution

$$\begin{aligned}
\mathcal{M}(\pi^*) - \mathcal{M}(\widehat{\pi}_n) &= \mathbb{E}_{s\sim\rho,(a_w,a_l)\sim\pi_{\text{sft}}}\left[[\widehat{r}_{\theta^*}(a_w|s) - \widehat{r}_{\theta^*}(a_l|s)] - [\widehat{r}_{\widehat{\theta}_n}(a_w|s) - \widehat{r}_{\widehat{\theta}_n}(a_l|s)]\right] \\
&= \mathbb{E}_{s\sim\rho,(a_w,a_l)\sim\pi_{\text{sft}}}\left[\log\frac{\pi_{\theta^*}(a_w|s)}{\pi_{\theta^*}(a_l|s)} - \log\frac{\pi_{\widehat{\theta}_n}(a_w|s)}{\pi_{\widehat{\theta}_n}(a_l|s)}\right] \\
&= \mathbb{E}_{s\sim\rho,(a_w,a_l)\sim\pi_{\text{sft}}}\left[[f_{\theta^*}(s,a_w) - f_{\theta^*}(s,a_l)] - [f_{\widehat{\theta}_n}(s,a_w) - f_{\widehat{\theta}_n}(s,a_l)]\right] \\
&= \mathbb{E}_{s\sim\rho,(a_w,a_l)\sim\pi_{\text{sft}}}\left[[f_{\theta^*}(s,a_w) - f_{\widehat{\theta}_n}(s,a_w)] - [f_{\theta^*}(s,a_l) - f_{\widehat{\theta}_n}(s,a_l)]\right] \\
&\le \mathbb{E}_{s\sim\rho,(a_w,a_l)\sim\pi_{\text{sft}}}\left[\left|f_{\theta^*}(s,a_w) - f_{\widehat{\theta}_n}(s,a_w)\right| + \left|f_{\theta^*}(s,a_l) - f_{\widehat{\theta}_n}(s,a_l)\right|\right] \\
&\le 2\alpha_1 \left\|\theta^* - \widehat{\theta}_n\right\| ,
\end{aligned}$$

where the final step follows from Assumption 4.1. Now, assuming $\widehat{\Sigma}$ to be invertible for log-linear policies, we get from (15):

$$\left\|\widehat{\theta}_n - \theta^*\right\|_{\widehat{\Sigma}} = O\Big(\frac{1}{\sqrt{\lambda_{\min}(\widehat{\Sigma})}}\frac{1}{\gamma\beta(1-2\varepsilon)}\sqrt{\frac{d}{n}}\Big) .$$

Setting $\alpha_1 = LB$ for log-linear policies, we obtain

$$\mathcal{M}(\pi^*) - \mathcal{M}(\widehat{\pi}_n) = O\Big(\frac{1}{\sqrt{\lambda_{\min}(\widehat{\Sigma})}}\frac{2LB}{\gamma\beta(1-2\varepsilon)}\sqrt{\frac{d}{n}}\Big) ,$$

which completes our proof.

B. Implementation and Hyperparameter Details

PyTorch code for the Robust DPO loss is provided below. We use the same notation as in (Rafailov et al., 2023).

```
import torch.nn.functional as F

def robust_dpo_loss(pi_logps, ref_logps, yw_idxs, yl_idxs, beta, epsilon):
    """
    pi_logps: policy logprobs, shape (B,)
    ref_logps: reference model logprobs, shape (B,)
    yw_idxs: chosen completion indices in [0, B-1], shape (T,)
    yl_idxs: rejected completion indices in [0, B-1], shape (T,)
    beta: temperature controlling strength of KL penalty
    epsilon: Preference noise(0-0.5) in the data
    """
    pi_yw_logps, pi_yl_logps = pi_logps[yw_idxs], pi_logps[yl_idxs]
    ref_yw_logps, ref_yl_logps = ref_logps[yw_idxs], ref_logps[yl_idxs]

    pi_logratios = pi_yw_logps - pi_yl_logps
    ref_logratios = ref_yw_logps - ref_yl_logps
    logits = pi_logratios - ref_logratios

    losses = (
        -F.logsigmoid(beta * logits) * (1 - epsilon)
        + F.logsigmoid(-beta * logits) * epsilon
    ) / (1 - 2 * epsilon)

    return losses
```

The hyperparameters for the experiments are outlined in Table 4 and Table 5. Any hyperparameters not explicitly mentioned use the default values in the TRL[5] library.

Table 4. Hyperparameters used for methods in the DPO Family

Parameter	Value
beta	0.1
learning rate	0.001
batch size	16
max length	512
max prompt length	128

Table 5. Hyperparameters used for methods in the PPO Family

Model	Parameter	Value
Reward Model	learning rate	1.41 x 10^{-5}
	batch size	16
PPO	learning rate	1.41 x 10^{-5}
	batch size	16

[5] huggingface.co/docs/trl/index