Rethinking DPO: The Role of Rejected Responses in Preference Misalignment

Jay Hyeon Cho, JunHyeok Oh, Myunsoo Kim, Byung-Jun Lee
Korea University
{bonin147, the2ndlaw, m970326, byungjunlee}@korea.ac.kr

Abstract

Direct Preference Optimization (DPO) is a simple and efficient framework that has attracted substantial attention. However, it often struggles to meet its primary objectives—increasing the generation probability of chosen responses while reducing that of rejected responses—due to the dominant influence of rejected responses on the loss function. This imbalance leads to suboptimal performance in promoting preferred responses. In this work, we systematically analyze the limitations of DPO and existing algorithms designed to achieve the objectives stated above. To address these limitations, we propose Bounded-DPO (BDPO), a novel method that bounds the influence of rejected responses while maintaining the original optimization structure of DPO. Through theoretical analysis and empirical evaluations, we demonstrate that BDPO achieves a balanced optimization of the chosen and rejected responses, outperforming existing algorithms.

1 Introduction

Aligning Large Language Models (LLMs) with human values using human preference data has become essential in natural language processing (NLP). Direct Preference Optimization (DPO; Rafailov et al. 2024b) has gained significant attention for efficiently optimizing preference comparisons directly from pairwise feedback, without relying on explicit reward models or reinforcement learning, offering a computationally cheaper alternative to conventional RLHF methods (Ouyang et al., 2022; Bai et al., 2022; Stiennon et al., 2020).

Despite its simplicity, DPO exhibits two key limitations. First, it tends to increase the probability of generating out-of-distribution (OOD) actions, a drawback attributed to the absence of sampling during training (Xu et al., 2024). Second, several studies (Pang et al., 2024; Adler et al., 2024; Liu et al., 2024) have highlighted that DPO struggles to sufficiently increase the probability of chosen actions. To address this, DPO+NLL was proposed, augmenting DPO with a negative log-likelihood (NLL) loss—commonly used in supervised learning—to explicitly improve the likelihood of chosen responses. Another alternative, DPO-Positive (DPOP), introduces a penalty when the probability of a chosen response falls below that assigned by the initial model (Pal et al., 2024).

This study identifies two core objectives of DPO: (1) increasing the generation probability of chosen responses while decreasing that of rejected responses, and (2) ensuring the model does not deviate too far from the reference model. The analysis reveals that, in theory, the DPO loss can achieve an optimal solution by simply lowering the probability of generating rejected responses, without necessarily increasing the probability of generating chosen responses. Empirical results corroborate this finding, showing that DPO fails to adequately fulfill objective (1). This work also examines alternative approaches, such as DPO+NLL and DPOP, and discusses their respective limitations in detail.

We propose Bounded-DPO (BDPO), a novel algorithm designed to address the two primary objectives of DPO more effectively. BDPO preserves the overall structure of the original DPO loss, but replaces the rejected response distribution of the updated model with a mixture distribution that incorporates the reference model. This modification bounds the influence of rejected responses on the loss function, ensuring adherence to objective (2) while effectively achieving objective (1).

Following the introduction of BDPO, we present a theoretical analysis showing that the BDPO loss guarantees a lower bound on the probability of generating the chosen response (Section 4.1). We conduct both toy examples and real-world model experiments to compare the behavior of BDPO with existing algorithms (Section 4.2). Experimental results demonstrate that BDPO achieves superior

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 8159–8176
November 4-9, 2025

performance in preference optimization (Section 5). Additionally, we conducted an ablation study to gain deeper insights into BDPO's performance and provided a detailed analysis of the findings.

2 Related Works

Direct Preference Optimization (DPO, Rafailov et al. 2024b) simplifies the RLHF framework by eliminating the need for reward modeling, instead directly training models on human preference data. Due to its efficiency and simplicity, DPO has been widely adopted in various NLP applications (Dubey et al., 2024; Yuan et al., 2024; Chen et al., 2024b).

However, despite its practical advantages, DPO often yields models that underperform compared to those trained with conventional RLHF methods (Ivison et al., 2024; Xu et al., 2024; Tang et al., 2024). Recent studies have identified several notable limitations of DPO, including:

- susceptibility to reward divergence, which can lead to overfitting (Azar et al., 2024)
- a tendency to learn a biased policy that favors OOD responses (Xu et al., 2024)
- difficulty in correcting inaccurate preference rankings (Chen et al., 2024a)
- failure to promote chosen responses effectively during optimization (Pal et al., 2024; Rafailov et al., 2024a; Pang et al., 2024; Razin et al., 2024; Tajwar et al., 2024; Liu et al., 2024; Xie et al., 2024)

Among these, we consider the last one to be the most significant and aim to address it in this paper, as it fundamentally undermines the core objective of DPO.[1] To the best of our knowledge, only two prior approaches have addressed this problem without relying on online sampling during training. We provide an in-depth comparison and analysis of these methods in this work.

DPO+NLL To prevent DPO from reducing the probability of chosen responses, a simultaneous supervised learning step on chosen responses has been proposed (Adler et al., 2024; Pang et al., 2024; Liu et al., 2024). This method, referred to as DPO+NLL in this paper, adds an explicit negative log-likelihood (NLL) loss term to the original DPO objective to directly increase the likelihood of chosen responses. However, we show that it introduces sensitivity to hyperparameters, often causing the training process to be dominated by the NLL term.

DPO-Positive (DPOP) Another approach introduces direct penalties when DPO fails to increase the probability of chosen responses to guide the optimization process (Pal et al., 2024). However, we show that this method does not overcome the fundamental limitations of DPO and ultimately exhibits similar shortcomings.

3 Analysis of Existing Objectives for Preference Optimization

In this section, we revisit the objectives of DPO and review prior findings showing that it can deviate from its intended goals. To deepen the understanding of DPO's behavior, we provide a detailed analysis of its underlying mechanisms. We also examine the limitations of two representative extensions—DPO+NLL and DPOP—that have been proposed to mitigate these issues.

The DPO objective is defined as:

$$\max_{\theta} \mathbb{E}_{\mathcal{D}} \left[\log \sigma \left(r_\theta(\mathbf{y}_w) - r_\theta(\mathbf{y}_l) \right) \right], \qquad (1)$$

where $r_\theta(\mathbf{y}) := \beta \log \frac{\pi_\theta(\mathbf{y}|\mathbf{x})}{\pi_{\text{ref}}(\mathbf{y}|\mathbf{x})}$, σ is the sigmoid function, π_θ denotes the learned policy, π_{ref} is the reference model, and β is a hyperparameter.

The DPO loss captures two key objectives:

- **Optimizing response probabilities:** Increase the probability of the chosen response while decreasing that of the rejected response.
- **Alignment with the reference model:** Ensure the learned model does not deviate significantly from the reference model.

These characteristics mirror those of the original RLHF framework. The first reflects the use of reward models trained to favor chosen responses over rejected ones, with policies optimized to maximize these rewards. The second aligns with behavior regularization, which constrains the learned policy to remain close to the reference model and mitigates over-optimization on uncertain rewards.

3.1 Does DPO Increase the Probability of the Chosen Response?

Xu et al. (2024) demonstrated through illustrative experiments that, under DPO, the probabilities of

[1] These issues are often interrelated; for example, reducing the probabilities of both chosen and rejected responses may increase the likelihood of generating OOD responses.

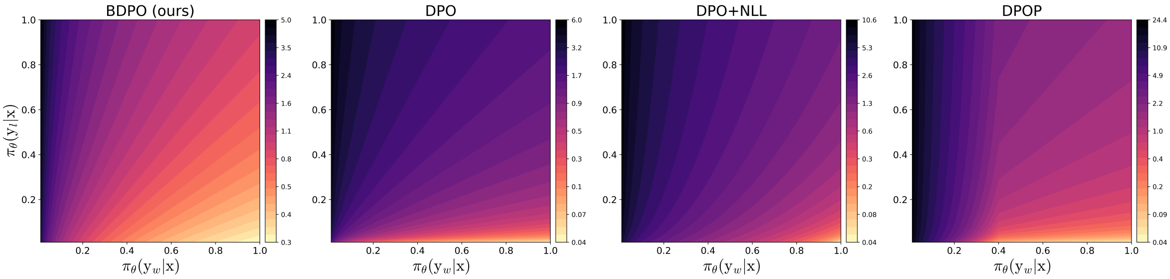

Figure 1: Contour maps of the loss with reference model $\pi_{\text{ref}}(\mathbf{y}_w|\mathbf{x}) = 0.4$ and $\pi_{\text{ref}}(\mathbf{y}_l|\mathbf{x}) = 0.1$. DPO shows that having $\pi_\theta(\mathbf{y}_l|\mathbf{x})$ close to zero results in the lowest loss, regardless of $\pi_\theta(\mathbf{y}_w|\mathbf{x})$. DPOP applies penalties below $\pi_{\text{ref}}(\mathbf{y}_w|\mathbf{x})$ but shares the same issue to DPO. In contrast, BDPO and DPO+NLL are not dominated by $\pi_\theta(\mathbf{y}_l|\mathbf{x})$, and they are able to effectively learn to increase $\pi_\theta(\mathbf{y}_w|\mathbf{x})$.

both the chosen and rejected responses can simultaneously decrease. They further highlighted that DPO is prone to incorrect learning behavior, particularly in OOD scenarios. To provide a more intuitive perspective on the DPO objective, the loss can be reformulated as:

$$\max_\theta \mathbb{E}_{\mathcal{D}} \left[\log \sigma \left(\hat{s}_\theta(w;l) - \hat{s}_{\text{ref}}(w;l)\right)\right], \quad (2)$$

where $\hat{s}_\theta(w;l) := \beta \log \frac{\pi_\theta(\mathbf{y}_w|\mathbf{x})}{\pi_\theta(\mathbf{y}_l|\mathbf{x})}$ and $\hat{s}_{\text{ref}}(w;l) := \beta \log \frac{\pi_{\text{ref}}(\mathbf{y}_w|\mathbf{x})}{\pi_{\text{ref}}(\mathbf{y}_l|\mathbf{x})}$ denote the scaled log-ratios under the learned policy π_θ and the reference model π_{ref}.

Both the logarithm and sigmoid functions are strictly increasing, and π_{ref} remains fixed during training. As a result, minimizing the DPO loss is equivalent to maximizing the term $\hat{s}_\theta(w;l)$. A straightforward way to achieve this maximization is by reducing $\pi_\theta(\mathbf{y}_l|\mathbf{x})$ to zero. However, setting $\pi_\theta(\mathbf{y}_l|\mathbf{x}) = 0$ leads to an unbounded loss, and the objective can be optimized regardless of the probability assigned to the chosen response $\pi_\theta(\mathbf{y}_w|\mathbf{x})$.

Furthermore, even when training proceeds as intended—i.e., increasing the log-ratio for the chosen response while decreasing it for the rejected one—the concavity of the logarithm disproportionately amplifies the effect of the rejected response. As a result, the optimization may become dominated by the rejected term.

Loss Visualization Figure 1 illustrates how the loss varies with respect to the chosen response probability $\pi_\theta(\mathbf{y}_w|\mathbf{x})$ and the rejected probability $\pi_\theta(\mathbf{y}_l|\mathbf{x})$. Contour lines represent regions of equal loss, with brighter colors indicating lower values. During training, optimization tends to move toward these brighter (lower-loss) regions. The DPO subplot clearly shows that minimizing $\pi_\theta(\mathbf{y}_l|\mathbf{x})$ alone yields the lowest loss, regardless of the value of $\pi_\theta(\mathbf{y}_w|\mathbf{x})$, highlighting a strong dependence on the rejected response probability. We empirically confirm that this behavior occurs in practice by analyzing the convergence patterns of DPO loss in a real-world language model (Figure 4, Section 4.2.2).

Remark Given a prompt $\mathbf{x}$, the model assigns probabilities over all possible responses such that they sum to one. If the expected update to both the chosen and rejected responses is negative, i.e., $\mathbb{E}_{\mathcal{D}}[\Delta\pi_\theta(\mathbf{y}_w|\mathbf{x}) + \Delta\pi_\theta(\mathbf{y}_l|\mathbf{x})] < 0$, the probability mass is implicitly shifted toward out-of-distribution (OOD) responses. We empirically confirm this shift toward OOD responses in Appendix A.

3.2 Analysis of DPO+NLL Loss

In supervised fine-tuning, the model is typically trained to minimize the Negative Log-Likelihood (NLL) loss, which is defined as:

$$\mathcal{L}_{\text{NLL}}(\pi_\theta) = -\mathbb{E}_{\mathcal{D}} \left[\log \pi_\theta(\mathbf{y}|\mathbf{x})\right],$$

where $\mathbf{y}$ denotes the target response. Minimizing this objective encourages the model to assign higher probability to $\mathbf{y}$.

To mitigate the issue of decreasing probabilities for the chosen response in DPO, Pang et al. (2024); Adler et al. (2024); Liu et al. (2024) introduced augmenting the DPO loss with the NLL loss, using the chosen response $\mathbf{y}_w$ as the target label for NLL loss. The resulting DPO+NLL loss is:

$$\mathcal{L}_{\text{DPO+NLL}} = \mathcal{L}_{\text{DPO}} + \alpha \mathcal{L}_{\text{NLL}},$$

where α is a hyperparameter controlling the relative weight of the NLL term.

Since the NLL loss explicitly increases the probability of the chosen response, this objective can address the limitations of DPO when α is sufficiently large. Moreover, the NLL term can be viewed as the forward KL divergence between the chosen response distribution and the updated model, offering an intuitive explanation of how it mitigates DPO's vulnerability to OOD responses.

However, we show that when α is large enough to fully correct DPO's shortcomings, **the update process becomes dominated by the NLL loss**. This dominance can cause the updated model to deviate significantly from the reference model.

Examining the gradients of each component in the DPO+NLL objective, we obtain:

$$\nabla_\theta \mathcal{L}_{\text{DPO}} = -\mathbb{E}_{\mathcal{D}}\Big[\beta \cdot \sigma\left(r_\theta(\mathrm{y}_l) - r_\theta(\mathrm{y}_w)\right) \cdot \left[\nabla_\theta \log \pi_\theta(\mathbf{y}_w|\mathbf{x}) - \nabla_\theta \log \pi_\theta(\mathbf{y}_l|\mathbf{x})\right]\Big],$$
$$\nabla_\theta \mathcal{L}_{\text{NLL}} = -\mathbb{E}_{\mathcal{D}}[\nabla_\theta \log \pi_\theta(\mathbf{y}_w|\mathbf{x})].$$

From this, we observe that the total gradient with respect to $\log \pi_\theta(\mathrm{y}_w|\mathrm{x})$ is scaled by:

$$\beta \cdot \sigma\left(r_\theta(\mathrm{y}_l) - r_\theta(\mathrm{y}_w)\right) + \alpha.$$

This coefficient determines the extent to which the probability of the chosen response $\mathbf{y}_w$ is increased during training. As training progresses, the value of $r_\theta(\mathrm{y}_l|\mathrm{x}) - r_\theta(\mathrm{y}_w|\mathrm{x})$ tends to $-\infty$, causing the sigmoid $\sigma(\cdot)$ to approach zero. In contrast, α remains a fixed constant, which results in the NLL term increasingly dominating the update.

Moreover, commonly used hyperparameter settings (e.g., $\beta = 0.1$, $\alpha = 1$) imply that the contribution of the NLL component to the gradient of $\mathbf{y}_w$ outweighs that of the DPO component by at least a factor of 10. This further supports the concern that DPO+NLL tends to move the model away from the reference distribution, contrary to DPO's original intent. Additionally, if the NLL term dominates optimization, the risk of overfitting increases. This trade-off between correcting DPO's limitations and preserving alignment with the reference model makes the choice of α highly nontrivial, as illustrated in the following analysis.

Loss Visualization Figure 2 extends the loss visualization from Figure 1 by presenting contour maps of the DPO+NLL loss for different values of α. For $\alpha = 0.01$ and $\alpha = 0.1$, the behavior remains consistent with that of DPO, failing to address the limitations. At $\alpha = 1$, the optimization demonstrates desirable behavior, aligning well with the intended learning dynamics. However, when $\alpha = 10$, the optimization process exhibits extreme loss patterns, which leads the updated model to significantly deviate from the reference model $\pi_{\text{ref}}(\mathbf{y}_w|\mathbf{x}) = 0.4$ and $\pi_{\text{ref}}(\mathbf{y}_l|\mathbf{x}) = 0.1$. This highlights the sensitivity of the algorithm to α.

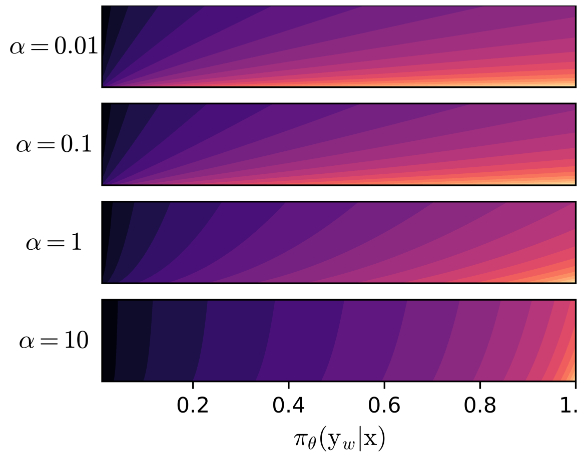

Figure 2: Contour maps for various values of $\alpha \in \{0.01, 0.1, 1, 10\}$, focusing on the region of interest where $\pi_\theta(\mathbf{y}_l|\mathbf{x}) \in [0, 0.25]$. The results show that, depending on the value of α, DPO+NLL may either fail to address the limitations of DPO (low α), or cause significant deviation from the reference model (high α).

3.3 Analysis of DPOP Loss

Pal et al. (2024) proposed DPO-Positive (DPOP) to address the limitations of DPO by introducing a penalty term. The DPOP objective is defined as:

$$\max_\theta \mathbb{E}_{\mathcal{D}}\big[\log \sigma\big(r_\theta(\mathbf{y}_w) - r_\theta(\mathbf{y}_l) - \lambda \cdot \max\left(0, -r_\theta\left(\mathbf{y}_w\right)\right)\big)\big],$$

where $\lambda > 0$ is a hyperparameter. The penalty is activated when the reward for the chosen response $r_\theta(\mathrm{y}_w)$ falls below zero, which corresponds to the case where $\log \frac{\pi_{\text{ref}}(\mathbf{y}_w|\mathbf{x})}{\pi_\theta(\mathbf{y}_w|\mathbf{x})} > 0$.

Figure 1 shows that when $\pi_\theta(\mathbf{y}_w|\mathbf{x}) > \pi_{\text{ref}}(\mathbf{y}_w|\mathbf{x})$, DPOP behaves identically to DPO, whereas for $\pi_\theta(\mathbf{y}_w|\mathbf{x}) < \pi_{\text{ref}}(\mathbf{y}_w|\mathbf{x})$, significantly higher loss levels are observed, clearly indicating the region where the penalty is applied. Nevertheless, as the penalty term becomes substantial only when $\pi_\theta(\mathbf{y}_w|\mathbf{x})$ approaches zero, the limitations of DPO are not fully addressed. The DPOP loss still guarantees an optimal solution where $\pi_\theta(\mathbf{y}_l|\mathbf{x}) = 0$ and $\pi_\theta(\mathbf{y}_w|\mathbf{x}) \neq 0$. As discussed in Section 3.1, the steep divergence of the logarithmic function near zero causes disproportionate impacts when probabilities are low. Although the penalty term in DPOP aims to stabilize optimization by discouraging undesirable updates, it fails to fully mitigate the inherent imbalance introduced by this property.

In Section 4.2.1, we demonstrate that DPOP, like DPO, is heavily influenced by rejected responses, which can result in an increased probability of generating OOD responses.

4 Our Algorithm

In this section, we introduce Bounded-DPO (BDPO), our proposed method designed to satisfy the two primary objectives of DPO. We provide its theoretical foundation and show empirically that BDPO achieves the intended behavior while overcoming the limitations of existing methods.

4.1 Bounded-DPO

Bounded-DPO (BDPO) extends the DPO formulation in Equation (2) by replacing $\pi_\theta(\mathbf{y}_l|\mathbf{x})$ with a mixture distribution. The BDPO loss is defined as:

$$\mathcal{L}_{\text{BDPO}}(\pi_\theta; \pi_{\text{ref}}) := -\mathbb{E}_{\mathcal{D}} \left[\log \sigma \left(\beta \log \frac{\pi_\theta(\mathbf{y}_w|\mathbf{x})}{\pi_{\text{mix}}(\mathbf{y}_l|\mathbf{x})} - \hat{s}_{\text{ref}}(w; l)\right)\right],$$

where $\pi_{\text{mix}}(\mathbf{y}|\mathbf{x}) = \lambda\pi_\theta(\mathbf{y}|\mathbf{x}) + (1-\lambda)\pi_{\text{ref}}(\mathbf{y}|\mathbf{x})$, for $\lambda \in (0, 1)$.

This formulation ensures that the denominator is lower bounded by $(1-\lambda)\pi_{\text{ref}}(\mathbf{y}_l|\mathbf{x})$, even when $\pi_\theta(\mathbf{y}_l|\mathbf{x}) = 0$. This property is essential for ensuring that the loss contributes to the intended learning behavior. The parameter λ controls the balance between the learned policy and the reference model: larger values of λ give more weight to π_θ, while smaller values emphasize π_{ref}. The effect of λ is further analyzed in Section 5.4.

To analyze the behavior of BDPO, we consider the simplest scenario with a single response pair in the dataset, $\mathcal{D} = \{(\mathbf{y}_w, \mathbf{y}_l)\}$.[2] Under this condition, BDPO exhibits the following properties (proofs are deferred to the Appendix J.1):

Theorem 1. *Let π^* denote the policy that minimizes $\mathcal{L}_{\text{BDPO}}$. If $\pi_{\text{ref}}(\mathbf{y}_l|\mathbf{x}) > 0$, then π^* satisfies the following conditions:*

$$\pi^*(\mathbf{y}_w|\mathbf{x}) = 1 \quad \textit{and} \quad \pi^*(\mathbf{y}_l|\mathbf{x}) = 0.$$

Corollary 1. *Let π^* denote the policy that minimizes $\mathcal{L}_{\text{BDPO}}$. Then π^* also minimizes $\mathcal{L}_{\text{DPO}}$. However, the converse does not hold.*

As discussed in Section 3.1, $\pi_\theta(\mathbf{y}_l|\mathbf{x})$ approaching 0 can trivially satisfy the optimality conditions of the DPO loss. In contrast, BDPO addresses this issue by introducing a lower bound in the denominator, ensuring that the problem does not arise even when $\pi_\theta(\mathbf{y}_l|\mathbf{x}) \to 0$. This robustness is formally presented in Theorem 1 and Corollary 1, which show that BDPO avoids this core limitation of DPO. Additionally, BDPO provides a lower bound for the chosen probability, as stated in the following theorem (proof is deferred to the Appendix J.3):

Theorem 2. *Let the initial point of π_θ be π_{ref}. Suppose that every optimization step ensures the BDPO loss decreases monotonically. Then, for any step, π_θ satisfies the following condition:*

$$(1-\lambda)\pi_{\text{ref}}(\mathbf{y}_w|\mathbf{x}) \leq \pi_\theta(\mathbf{y}_w|\mathbf{x})$$

Loss Visualization Figure 1 also illustrates the loss of the BDPO objective. Unlike DPO and DPOP, BDPO exhibits its minimum loss near $\pi_\theta(\mathbf{y}_w|\mathbf{x}) = 1$ and $\pi_\theta(\mathbf{y}_l|\mathbf{x}) = 0$, clearly guiding learning in the desired direction. This result is consistent with the theoretical guarantees provided in Theorem 1.

To further analyze BDPO, we compute the gradient of its loss with respect to $\pi_\theta(\mathbf{y}_l|\mathbf{x})$:

$$\nabla_{\pi_\theta(\mathbf{y}_l|\mathbf{x})}\mathcal{L}_{\text{BDPO}}(\pi_\theta; \pi_{\text{ref}}) = \mathbb{E}_{\mathcal{D}}\left[\beta\sigma\left(-\Delta_{\text{BDPO}}\right) \cdot \left[\frac{\lambda}{\pi_{\text{mix}}(\mathbf{y}_l|\mathbf{x})}\right]\right],$$

where $\Delta_{\text{BDPO}} = \beta \log \frac{\pi_\theta(\mathbf{y}_w|\mathbf{x})}{\pi_{\text{mix}}(\mathbf{y}_l|\mathbf{x})} - \hat{s}_{\text{ref}}(w; l)$. In contrast, the gradient of the DPO loss is:

$$\nabla_{\pi_\theta(\mathbf{y}_l|\mathbf{x})}\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = \mathbb{E}_{\mathcal{D}}\left[\beta\sigma\left(-\Delta_{\text{DPO}}\right) \cdot \left[\frac{1}{\pi_\theta(\mathbf{y}_l|\mathbf{x})}\right]\right],$$

where $\Delta_{\text{DPO}} = r_\theta(\mathbf{y}_w) - r_\theta(\mathbf{y}_l)$. In case of other losses, DPO+NLL shares the identical partial derivatives, and DPOP follows a similar structure (see Appendix K for details). From these, we can observe the following:

- DPO, DPO+NLL, and DPOP: The gradient term $\frac{1}{\pi_\theta(\mathbf{y}_l|\mathbf{x})}$ leads to instability as $\pi_\theta(\mathbf{y}_l|\mathbf{x}) \to 0$, causing unbounded updates and numerical issues.

- BDPO: The mixture term $\pi_{\text{mix}}(\mathbf{y}_l|\mathbf{x})$ provides a lower bound, preventing divergence even when $\pi_\theta(\mathbf{y}_l|\mathbf{x}) \to 0$, thereby ensuring stable optimization.

[2] We do not consider the case where multiple response pairs are associated with a single prompt.

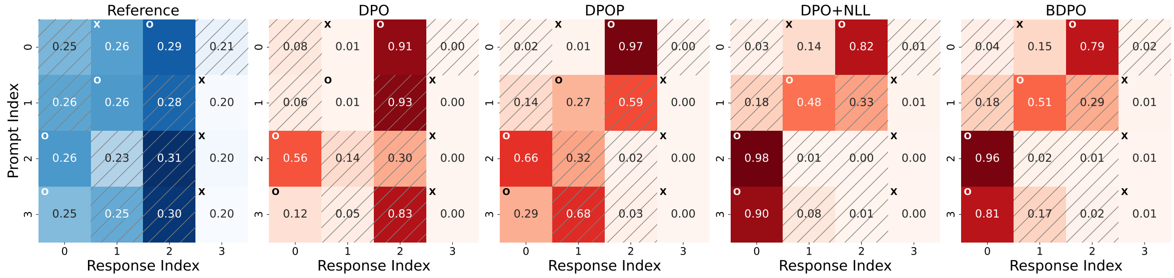

Figure 3: Training behavior of DPO, DPOP, DPO+NLL, and BDPO on a toy task with four prompts and four responses per prompt. Two out-of-distribution (OOD) responses per prompt are shaded in gray with diagonal stripes. Chosen and rejected responses are marked with o and ×, respectively. While DPO and DPOP fail to suppress some OOD responses, BDPO and DPO+NLL demonstrate more appropriate learning behavior.

4.2 Empirical Validation of BDPO

We empirically evaluate whether BDPO and prior methods fulfill the two core objectives of DPO. First, in a toy example (Section 4.2.1), we qualitatively compare different loss functions. The results show that both DPO and DPOP are prone to generating OOD actions. Second, using a real-world language model and dataset (Section 4.2.2), we analyze learning dynamics. We find that BDPO behaves as intended, whereas DPO+NLL fails to maintain alignment with the reference model, violating the second objective of DPO.

4.2.1 Examining the OOD Behavior

We construct a toy scenario with four prompts, each paired with four candidate responses. For each prompt, one pair of preference data is randomly selected, and the remaining two responses are implicitly treated as OOD. Further details on the experimental setup are provided in Appendix B.1.

As shown in Figure 3, DPO and DPOP exhibit undesirable behavior for OOD responses, particularly in prompts 1 and 3. While DPOP prevents a decrease in the probability of the chosen response due to its explicit penalty term, it fails to increase it meaningfully—in prompt 1, the probability remains nearly identical to that of the reference model. Both DPO and DPOP tend to focus primarily on reducing the probability of the rejected response. In contrast, BDPO and DPO+NLL demonstrate desirable learning behavior. The learning dynamics of the toy case training process are provided in Appendix D.[3]

4.2.2 Training Dynamics of BDPO

We trained the QWEN 0.5B model (Yang et al., 2024) using the UltraFeedback Binarized dataset (Cui et al., 2024). To better analyze training dynamics, we subsampled the dataset to 1% of its original size and trained the model for 100 epochs. Additional experimental details are provided in Appendix B.2.

Figure 4 presents the log probabilities of the chosen and rejected responses during training. Under DPO, both values decrease, with the rejected probability dropping more sharply—consistent with our analysis in Section 3.1. As a result, the model likely shifts probability mass toward OOD responses, a phenomenon further analyzed in Appendix A.

In contrast, BDPO, DPOP, and DPO+NLL exhibited more desirable training dynamics, with the probability of the chosen response increasing and the probability of the rejected response decreasing. Among these, BDPO and DPOP showed relatively smaller changes in their log probabilities compared to their initial values, maintaining closer alignment with the reference model. On the other hand, DPO+NLL pushes the chosen probability toward 100% and the rejected probability toward 0%, indicating over-optimization and divergence from the reference. This trend is further confirmed by the KL divergence measurements: both DPO and DPO+NLL exhibit significant deviation from the reference model, as discussed in Sections 3.1 and 3.2, while BDPO and DPOP remain closely aligned. Qualitative results in Appendix C further support these findings.

Interestingly, although BDPO does not include an explicit NLL term, it still achieves a consistent reduction in NLL loss during training. This suggests that BDPO implicitly encourages forward KL minimization with respect to the dataset, striking a balance between effective learning and alignment.

[3]Due to the extreme setup, models may diverge further from the reference model than typically observed.

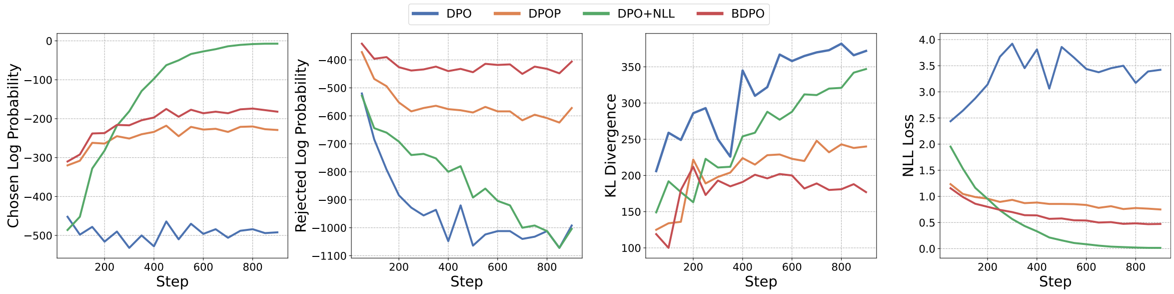

Figure 4: Learning dynamics of four algorithms—DPO, DPOP, DPO+NLL, and BDPO—are shown. All algorithms, except DPO, exhibit the desirable behavior of increasing the probability of chosen responses while decreasing the probability of rejected responses. However, DPO+NLL exhibits significant deviation from the reference model, as shown by both the log probabilities and the KL divergence. Although BDPO does not include an explicit NLL term, it achieves a reduction in the NLL loss on the training set.

5 Experiment

So far, we have shown that BDPO effectively satisfies the core objectives of DPO. In this section, we examine how achieving these desirable behaviors translates into actual performance. Section 5.1 discusses the experimental setup. Section 5.2 describes the evaluation benchmarks, IFEval (Zhou et al., 2023) and GSM8K (Cobbe et al., 2021). Section 5.3 presents the results, demonstrating the superior performance of BDPO. We also analyze in Section 5.4 the impact of the mixture distribution hyperparameter on performance.

5.1 Experimental Setup

Model As in Section 4.2.2, we use the Qwen model as the pretrained base, chosen for its strong performance relative to its size. Consistent with prior work, we first apply SFT using only the chosen responses from the human preference data, followed by preference optimization. Details on the SFT process and optimization procedures are provided in Appendix C.1. Additionally, to demonstrate that our algorithm generalizes beyond the Qwen family, we also conduct experiments using LLaMA 3.2 (Grattafiori et al., 2024).

Data For instruction-following, we use the UltraFeedback Binarized dataset (Cui et al., 2024), a widely adopted benchmark. Models trained on this dataset are evaluated using IFEval. To evaluate reasoning capabilities, we additionally use the UltraInteract dataset (Yuan et al., 2025), with evaluation on GSM8K for mathematical problem-solving.

5.2 Evaluation Method

IFEval We use IFEval (Zhou et al., 2023), a widely adopted benchmark for instruction-following evaluation. It consists of 541 instructions and is designed to minimize biases common in LLM-based auto-evaluation. IFEval assesses performance using prompt-level and instruction-level accuracy under both loose and strict criteria. We follow the scoring methodology from the Open-LLM Leaderboard (Fourrier et al., 2024). Full details are provided in Appendix F.

MT-bench We additionally evaluate models on MT-bench (Zheng et al., 2023), a multi-turn benchmark designed to assess conversational ability and alignment quality of large language models. MT-bench consists of 80 carefully curated multi-turn dialogue tasks covering diverse categories such as reasoning, writing, coding, and role-play. Model outputs are automatically rated by GPT-4 across multiple dimensions, providing a fine-grained assessment of helpfulness, correctness, and coherence. Following prior work, we report the average score across all categories as the final MT-bench result.

GSM8K To evaluate mathematical reasoning, we use GSM8K (Cobbe et al., 2021), a dataset of 8.5K linguistically diverse grade school math word problems requiring multi-step reasoning. Evaluation is performed using 4-shot CoT prompting.

5.3 Results

Table 1 presents IFEval evaluation results for models trained on the UltraFeedback dataset using Qwen 0.5B. To enable a more comprehensive comparison of BDPO's performance, we include additional baseline algorithms beyond those primarily discussed in this paper (DPO, DPOP, and DPO+NLL). To assess scalability, we also evaluate the major algorithms using the larger Qwen 7B model. Across both model scales, BDPO ($\lambda = 0.5$) achieves the highest total score, outperforming all baselines under both loose and strict accuracy crite-

Base Model	Algorithm	Inst-level Loose Acc	Inst-level Strict Acc	Prompt-level Loose Acc	Prompt-level Strict Acc	Loose Score	Strict Score	Total Score
Qwen2.5-0.5B	Base	23.38	21.82	12.38	10.71	17.88	16.27	17.07
	SFT	29.98	27.58	18.11	16.45	24.05	22.02	23.03
	DPO	31.53	30.46	20.33	18.67	25.93	24.57	25.25
	DPOP	32.85	31.41	21.44	20.14	27.15	25.78	26.46
	DPO+NLL	32.37	30.58	21.07	19.78	26.72	25.18	25.95
	MinorDPO (Xie et al., 2024)	32.73	30.82	21.07	19.59	26.90	25.21	26.05
	SLiC (Zhao et al., 2023)	32.61	31.41	20.15	18.67	26.38	25.04	25.71
	ORPO (Hong et al., 2024)	32.01	30.94	20.70	19.59	26.36	25.27	25.81
	SPPO (Wu et al., 2024)	**33.09**	31.41	21.81	19.96	27.45	25.69	26.57
	BDPO ($\lambda = 0.5$)	**33.09**	**31.89**	**22.37**	**21.26**	**27.73**	**26.58**	**27.15**
Qwen2.5-7B	DPO	78.54	75.42	70.06	66.91	74.30	71.17	72.73
	DPOP	77.56	74.94	70.06	67.28	73.81	71.11	72.46
	DPO+NLL	78.66	76.62	70.61	68.21	74.64	72.42	73.53
	BDPO ($\lambda = 0.5$)	**79.38**	**77.46**	**70.79**	**69.50**	**75.09**	**73.48**	**74.28**

Table 1: IFEval results for models trained on the UltraFeedback dataset using various preference optimization algorithms, with Qwen2.5-0.5B and Qwen2.5-7B as base models. Metrics include instruction-level and prompt-level accuracy under both loose and strict criteria. BDPO ($\lambda = 0.5$) achieves the highest total score across both model scales, demonstrating strong performance and alignment.

ria. Notably, BDPO maintains strong performance at both the instruction and prompt levels, demonstrating its effectiveness in aligning model behavior with human preferences. These results confirm that satisfying DPO's objectives through BDPO leads to measurable performance gains.

Base Model	Algorithm	Total Score
LLaMA3.2 1B	BDPO	**57.79**
	DPO	56.61
	DPOP	56.42
	DPO+NLL	55.86

Table 2: Performance evaluation of alignment algorithms on LLaMA3.2 1B. BDPO achieves the top score, indicating its effectiveness may not be limited to a specific base model but could be broadly applicable.

Table 2 reports IFEval Total Score results for models trained on the UltraFeedback dataset using LLaMA 3.2 1B. These results demonstrate that our algorithm is effective not only for the Qwen family but also for other architectures, indicating that its performance does not depend on a specific model.

Table 3 further supports this conclusion by presenting results on GSM8K (4-shot CoT) using models trained on the UltraInteract dataset. BDPO again achieves the highest accuracy, showing that the advantages of our method extend to mathematical reasoning tasks as well.

Table 4 presents evaluation results on MT-bench. Compared to the main baseline algorithms, our method achieves the highest scores on both the first-turn and second-turn results. Since MT-bench aggregates performance across diverse domains, these results suggest that our algorithm consistently improves performance across a broad range of tasks.

Base Model	Algorithm	GSM8K-CoT (4 shot)
Qwen2.5-0.5B	DPO	27.45
	DPOP	29.04
	DPO+NLL	26.91
	BDPO (0.5)	**29.95**

Table 3: GSM8K results (4-shot CoT) for models trained on the UltraInteract dataset. BDPO ($\lambda = 0.5$) achieves the highest accuracy among all methods.

5.4 Ablation study

We also conducted an ablation study to analyze the effect of the hyperparameter λ in BDPO. As discussed in Section 4.1, λ controls the mixture ratio between the trained model and the reference model. Higher values of λ increase the influence of the trained model, while lower values emphasize the reference model's behavior. BDPO becomes equivalent to DPO when $\lambda = 1$, and focuses only on the chosen response when $\lambda = 0$, making both extremes generally undesirable.

To isolate the effect of varying λ, all other factors were kept constant. DPO was configured identically to BDPO for a fair comparison. Figure 5 presents these results on both IFEval and GSM8K. BDPO consistently outperforms DPO across all

Method	First turn	Second turn	Average
BDPO	**5.22**	**2.86**	**4.04**
DPO+NLL	3.75	2.46	3.11
DPOP	4.22	2.11	3.23
DPO	4.11	2.79	3.45

Table 4: MT-Bench results comparing BDPO with other alignment algorithms. BDPO consistently outperforms the other methods across all evaluated metrics.

tested λ values, with $\lambda = 0.5$ yielding the best overall performance on both benchmarks. These findings demonstrate that BDPO is robust across a range of settings, and that $\lambda = 0.5$ offers a strong and reliable default choice in practice.

For a fair comparison, we also conducted parameter searches not only for BDPO but also for DPO+NLL and DPOP. The corresponding results are provided in Appendix G. Although we explored a wider parameter space for the alternative algorithms, BDPO consistently achieved the highest performance.

In Figure 5, we observe that the GSM8K result at $\lambda = 0.9$ differs noticeably from that of DPO, even though BDPO should theoretically reduce to DPO as λ approaches 1. To examine this discrepancy in greater detail, we conducted additional experiments with $\lambda = 0.999$ and $\lambda = 0.99999$. The results confirm that as λ becomes increasingly close to 1, the performance of BDPO converges to that of DPO.

6 Conclusions

In this paper, we identified the two core objectives of DPO and showed through theoretical and empirical analysis that it fails to fully achieve them. We also examined alternative algorithms, such as DPO+NLL and DPOP, which aim to address DPO's limitations, and highlighted their respective shortcomings through both theoretical and experimental analysis. To overcome these issues, we proposed Bounded-DPO (BDPO), a method specifically designed to fulfill DPO's objectives without inheriting the weaknesses of previous approaches. Theoretically, BDPO offers an ideal optimal solution, guarantees a lower bound on the chosen response, and reduces sensitivity to the rejected response. Empirically, BDPO consistently outperforms other methods across various datasets, model scales, and evaluation benchmarks. Furthermore, our ablation study provides additional insights into how BDPO operates and the role of its key hyperparameter.

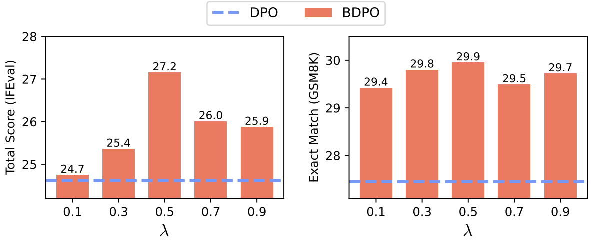

Figure 5: Ablation study on the choice of λ in BDPO. The left plot shows the IFEval total score on QWEN 0.5B; the right plot shows exact match accuracy on GSM8K. BDPO outperforms DPO across all λ values, with λ = 0.5 yielding the best overall results.

Limitations Due to GPU constraints, we were unable to conduct experiments across a broader range of models and algorithms. Similar to prior work, our theoretical analysis focuses on the single-pair setting—one preference pair per prompt. While BDPO can be extended both theoretically and empirically to the multi-pair response setting, we leave this for future work. Our evaluation focused on instruction-following and mathematical reasoning benchmarks, but we were not able to include a wider range of evaluation tasks. Additionally, although BDPO is applicable to variants such as online or iterative DPO, these directions are beyond the scope of this study.

Acknowledgments

This work was partly supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2022-II220311, Development of Goal-Oriented Reinforcement Learning Techniques for Contact-Rich Robotic Manipulation of Everyday Objects, No. RS-2024-00457882, AI Research Hub Project, No. RS-2019-II190079, Artificial Intelligence Graduate School Program (Korea University), and No. RS-2025-25410841, Beyond the Turing Test: Human-Level Game-Playing Agents with Generalization and Adaptation), the IITP (Institute of Information & Communications Technology Planning & Evaluation)-ITRC (Information Technology Research Center) grant funded by the Korea government (Ministry of Science and ICT) (IITP-2025-RS-2024-00436857), the NRF (RS-2024-00451162) funded by the Ministry of Science and ICT, Korea, BK21 Four project of the National Research Foundation of Korea, and the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (RS-2025-00560367), and the IITP under the Artificial Intelligence Star Fellowship support program to nurture the best talents (IITP-2025-RS-2025-02304828) grant funded by the Korea government (MSIT).

References

Bo Adler, Niket Agarwal, Ashwath Aithal, Dong H Anh, Pallab Bhattacharya, Annika Brundyn, Jared Casper, Bryan Catanzaro, Sharon Clay, Jonathan Cohen, and 1 others. 2024. Nemotron-4 340b technical report. *arXiv preprint arXiv:2406.11704*.

Mohammad Gheshlaghi Azar, Zhaohan Daniel Guo, Bilal Piot, Remi Munos, Mark Rowland, Michal Valko, and Daniele Calandriello. 2024. A general theoretical paradigm to understand learning from human preferences. In *International Conference on Artificial Intelligence and Statistics*, pages 4447–4455. PMLR.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, and 1 others. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.

Angelica Chen, Sadhika Malladi, Lily H Zhang, Xinyi Chen, Qiuyi Zhang, Rajesh Ranganath, and Kyunghyun Cho. 2024a. Preference learning algorithms do not learn preference rankings. *arXiv preprint arXiv:2405.19534*.

Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. 2024b. Self-play fine-tuning converts weak language models to strong language models. *arXiv preprint arXiv:2401.01335*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Bingxiang He, Wei Zhu, Yuan Ni, Guotong Xie, Ruobing Xie, Yankai Lin, and 1 others. 2024. Ultrafeedback: Boosting language models with scaled ai feedback. In *Forty-first International Conference on Machine Learning*.

Michael Han Daniel Han and Unsloth team. 2023. Unsloth.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Clémentine Fourrier, Nathan Habib, Alina Lozovskaya, Konrad Szafer, and Thomas Wolf. 2024. Open llm leaderboard v2. https://huggingface.co/spaces/open-llm-leaderboard/open_llm_leaderboard.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, and 5 others. 2024. A framework for few-shot language model evaluation.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Jiwoo Hong, Noah Lee, and James Thorne. 2024. Orpo: Monolithic preference optimization without reference model. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 11170–11189.

Hamish Ivison, Yizhong Wang, Jiacheng Liu, Zeqiu Wu, Valentina Pyatkin, Nathan Lambert, Noah A Smith, Yejin Choi, and Hannaneh Hajishirzi. 2024. Unpacking dpo and ppo: Disentangling best practices for learning from preference feedback. *arXiv preprint arXiv:2406.09279*.

Zhihan Liu, Miao Lu, Shenao Zhang, Boyi Liu, Hongyi Guo, Yingxiang Yang, Jose Blanchet, and Zhaoran Wang. 2024. Provably mitigating overoptimization in rlhf: Your sft loss is implicitly an adversarial regularizer. *arXiv preprint arXiv:2405.16436*.

I Loshchilov. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Arka Pal, Deep Karkhanis, Samuel Dooley, Manley Roberts, Siddartha Naidu, and Colin White. 2024. Smaug: Fixing failure modes of preference optimisation with dpo-positive. *arXiv preprint arXiv:2402.13228*.

Richard Yuanzhe Pang, Weizhe Yuan, Kyunghyun Cho, He He, Sainbayar Sukhbaatar, and Jason Weston. 2024. Iterative reasoning preference optimization. *arXiv preprint arXiv:2404.19733*.

Rafael Rafailov, Joey Hejna, Ryan Park, and Chelsea Finn. 2024a. From r to q^*: Your language model is secretly a q-function. *arXiv preprint arXiv:2404.12358*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024b. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3505–3506.

Noam Razin, Sadhika Malladi, Adithya Bhaskar, Danqi Chen, Sanjeev Arora, and Boris Hanin. 2024. Unintentional unalignment: Likelihood displacement in direct preference optimization. *arXiv preprint arXiv:2410.08847*.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. 2020. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021.

Fahim Tajwar, Anikait Singh, Archit Sharma, Rafael Rafailov, Jeff Schneider, Tengyang Xie, Stefano Ermon, Chelsea Finn, and Aviral Kumar. 2024. Preference fine-tuning of llms should leverage suboptimal, on-policy data. In *Forty-first International Conference on Machine Learning*.

Yunhao Tang, Daniel Zhaohan Guo, Zeyu Zheng, Daniele Calandriello, Yuan Cao, Eugene Tarassov, Rémi Munos, Bernardo Ávila Pires, Michal Valko, Yong Cheng, and 1 others. 2024. Understanding the performance gap between online and offline alignment algorithms. *arXiv preprint arXiv:2405.08448*.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. 2020. Trl: Transformer reinforcement learning. https://github.com/huggingface/trl.

Yue Wu, Zhiqing Sun, Huizhuo Yuan, Kaixuan Ji, Yiming Yang, and Quanquan Gu. 2024. Self-play preference optimization for language model alignment. *arXiv preprint arXiv:2405.00675*.

Shiming Xie, Hong Chen, Fred Yu, Zeye Sun, Xiuyu Wu, and Yingfan Hu. 2024. Minor dpo reject penalty to increase training robustness. *arXiv preprint arXiv:2408.09834*.

Shusheng Xu, Wei Fu, Jiaxuan Gao, Wenjie Ye, Weilin Liu, Zhiyu Mei, Guangju Wang, Chao Yu, and Yi Wu. 2024. Is dpo superior to ppo for llm alignment? a comprehensive study. In *Forty-first International Conference on Machine Learning*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Lifan Yuan, Ganqu Cui, Hanbin Wang, Ning Ding, Xingyao Wang, Boji Shan, Zeyuan Liu, Jia Deng, Huimin Chen, Ruobing Xie, and 1 others. 2025. Advancing llm reasoning generalists with preference trees. In *The Thirteenth International Conference on Learning Representations*.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason E Weston. 2024. Self-rewarding language models. In *Forty-first International Conference on Machine Learning*.

Yao Zhao, Rishabh Joshi, Tianqi Liu, Misha Khalman, Mohammad Saleh, and Peter J Liu. 2023. Slic-hf: Sequence likelihood calibration with human feedback. *CoRR*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*.

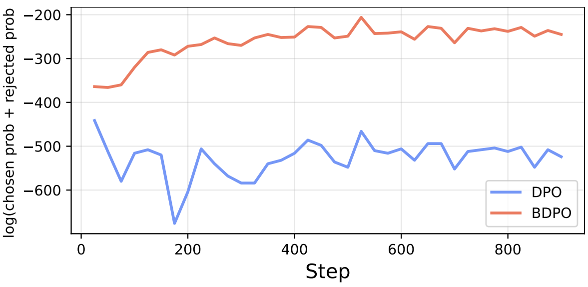

Figure 6: Log of the total in-distribution probability $\log(\pi_\theta(\mathbf{y}_w|\mathbf{x}) + \pi_\theta(\mathbf{y}_l|\mathbf{x}))$ measured during training. While BDPO maintains a stable and high in-distribution probability, DPO shows a clear decline, suggesting that it shifts probability mass toward out-of-distribution (OOD) responses as training progresses.

A Out-of-Distribution (OOD) Probability Analysis

Figure 6 presents the log of the in-distribution probability, defined as $\log(\pi_\theta(\mathbf{y}_w|\mathbf{x}) + \pi_\theta(\mathbf{y}_l|\mathbf{x}))$, tracked throughout the training process described in Section 4.2.2.

For DPO, we observe a gradual decline in this value, indicating that the model assigns lower total probability mass to the chosen and rejected responses as training progresses. Due to the generative nature of language models, this reduction likely extends to responses semantically or syntactically similar to the chosen/rejected ones.

Consequently, this behavior implies an increase in the probability of out-of-distribution (OOD) responses, including those that are irrelevant or inconsistent with the training data. Although prior work (Xu et al., 2024) attributes DPO's OOD vulnerability to its offline learning setting, **our findings suggest that this issue also stems from an inherent limitation in the loss formulation.**

As previously discussed in Section 3.1, the structure of the DPO loss permits such behavior as a valid solution, which can lead to substantial deviation from the reference model—a phenomenon known as reward divergence (Azar et al., 2024).

B Experimental Details

In this section, we provide detailed experimental setups employed in our study. Section B.1 discusses the experimental configurations specific to the toy case. Section B.2 delves into the experimental settings designed for analyzing the learning dynamics within a real-world language model. Lastly, Section C.1 outlines the details related to the main experiment. All experiments were run once per configuration.

B.1 Experimental Details for the Toy Case

For the toy case, we used a reference model structured as a 2-layer Multi-Layer Perceptron (MLP) with hidden layers dimensioned at 32 units, employing the ReLU activation function. This model was initialized randomly. The setup included four prompts, each with four responses. For each prompt, two chosen and two rejected responses were selected at random.

This toy case, like the setup in Theorem 1, is designed to reveal the ultimate optimization behavior of each loss function. In this scenario, the optimal solution assigns a probability of 1 to the chosen response (when not OOD) and 0 to all others. As a result, the trained model may diverge from the reference model, which differs from what we observe in real-world language models. Even under aggressive training conditions—such as a high learning rate and many epochs—alignment with the reference model tends to be preserved in practice (see Section 4.2.2).

B.2 Experimental Details for the Real-World Language Model

All models were trained using four NVIDIA RTX 3090 GPUs. We used the AdamW optimizer (Loshchilov, 2017) and DeepSpeed ZeRO-2 (Rasley et al., 2020) for memory-efficient training. Our implementation was based on the TRL framework (von Werra et al., 2020), and since DPOP was not available in the framework, we re-implemented it. To clearly illustrate training dynamics, we set the learning rate to 5×10^{-5}, randomly subsampled the dataset into 100 partitions, and trained for 100 epochs. For DPO+NLL, we set the hyperparameter $\alpha = 1$, and for DPOP, we followed the original authors' setting with $\lambda = 5$. For all algorithms, we used the commonly adopted $\beta = 0.1$. The code for our experiments is accessible.[4]

C Full Responses from the Qualitative Analysis

Table 6 provides the full responses of qualitative examples for all algorithms. DPO and DPO+NLL exhibit significant deviation from the reference model, resulting in high KL divergence values and demonstrating signs of overfitting. In contrast, DPOP shows less deviation and does not produce overfitted responses, but it generates incorrect answers. On the other hand, BDPO provides desirable responses while remaining close to the initial model, showcasing a balanced and effective learning behavior.

C.1 Experimental Details for the Main Experiment

The overall training setup is identical to that described in Appendix B.2, except for the number of epochs and learning rate. To ensure a fair comparison across algorithms, all models were trained for 3 epochs on the UltraFeedback dataset (64K examples) and 1 epoch on the UltraInteract dataset (220K examples), using learning rates of $5\text{e}{-7}, 1\text{e}{-6}$. The final evaluation was conducted using the best-performing combination of learning rate and epoch count for each algorithm. For training the 7B models, we employed 4-bit quantization using Unsloth (Daniel Han and team, 2023) to enable memory-efficient fine-tuning.

[4] https://github.com/bonin147/BDPO.git

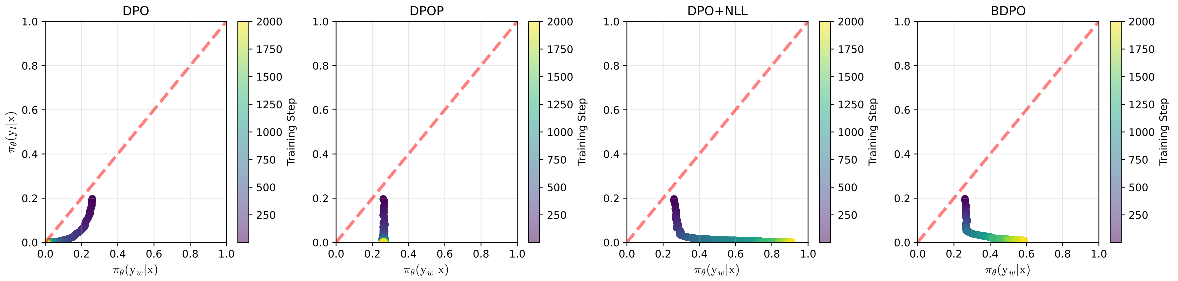

Figure 7: Evolution of chosen and rejected probabilities across training steps. DPO and DPOP show only the rejected probabilities converging to zero, whereas BDPO and DPO+NLL display more desirable learning behavior.

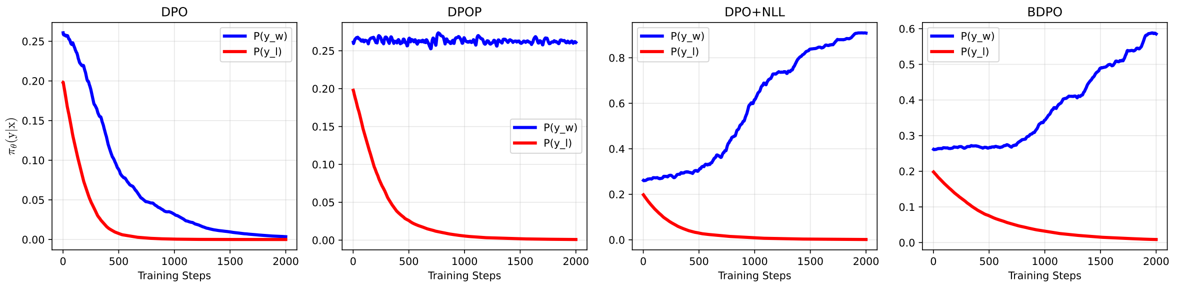

Figure 8: Probability trends over training steps. DPO shows a simultaneous decrease in both chosen and rejected probabilities, while DPOP exhibits a decrease only in rejected probability. In contrast, DPO+NLL and BDPO demonstrate desirable learning behaviors.

D Learning Dynamics for the Toy Case

In this section, we discuss the learning dynamics of the toy case. Figure 7, and Figure 8 support the findings presented in Figure 1. In the case of DPO, both the chosen and rejected probabilities decrease, converging to zero. For DPOP, due to the penalty term, the probability for the chosen response does not decrease but also does not increase, while the rejected probability converges to zero, similar to DPO. DPO+NLL and BDPO exhibit desirable learning behaviors.

E Computational Cost

Table 5 reports the wall-clock training time for the main algorithms discussed in this paper—DPO, DPOP, DPO+NLL, and BDPO—using the Qwen2.5-0.5B model. All experiments were conducted using four NVIDIA RTX 3090 GPUs. Models were trained for 3 epochs on the UltraFeedback dataset and 1 epoch on the UltraInteract dataset. The results show that all methods incur nearly identical computational costs under the same training conditions. BDPO introduces no additional overhead compared to DPO, despite modifying the loss formulation. This confirms that BDPO achieves its performance improvements without sacrificing training efficiency.

F IFEval Details

We evaluated the performance of our models using the methodology of Hugging Face's OpenLLM Leaderboard (Fourrier et al., 2024), which follows the evaluation framework of eval-harness (Gao et al., 2024). We measured four types of accuracy: Instance-level Loose, Instance-level Strict, Prompt-level Loose, and Prompt-level Strict. The OpenLLM Leaderboard reports a single score by averaging the Strict accuracy values. In line with this approach, we report the Strict score and additionally provide the Loose score (the average of Loose accuracy values) and the Total score (the overall average of all accuracy metrics).

G Additional Hyperparameter Results

G.1 DPO+NLL and DPOP

We report additional training results for DPO+NLL and DPOP. For DPO+NLL, we evaluate the hyperparameters (0.01, 0.1, 10) mentioned in our paper (Figure 2), as well as (0.25, 0.5, 1, 2), which were explored in Pang et al. (2024). For DPOP, we include the hyperparameters (5, 50, 500) as suggested in Pal et al. (2024). The results are presented in Table 7. These findings confirm that BDPO consistently outperforms DPO+NLL and DPOP across a broad range of hyperparameters.

Model	Method	Train Time (UltraFeedback)	Train Time (UltraInteract)
Qwen 0.5B	DPO	6h 10m	3h 53m
	DPOP	6h 10m	3h 54m
	DPO + NLL	6h 12m	3h 53m
	BDPO (0.5)	6h 10m	3h 54m

Table 5: Wall-clock training time comparison of different preference optimization methods on UltraFeedback and UltraInteract datasets using Qwen2.5-0.5B. All experiments were conducted using 4×RTX 3090 GPUs.

Prompt	if $X^2 + 3 = 12$. What is X?
Algorithm	**Response**
SFT (reference)	To find the value of X, we can subtract the constant term on the left side of the equation to isolate the unknown value on the right side of the equation. So, we start with the equation: X^2 + 3 = 12. Now, we subtract 3 from both sides: X^2 = 12 - 3. X^2 = 9. Next, we simplify the right side of the equation by factoring out the 3 on the right side: X^2 = 3(3). X^2 = 9 Now, we know that X is equal to the square root of 9 (3*3): X = ±3 So, the values of X are ±3.
DPO (100 Epochs) $\mathbb{E}[KL] = 372$	Given the equation: X^2 + 3 = 12. To find X, we'll first make the equation easier to work with, by moving the 3 to the right side: X^2 + 3 - 3 = 12 X^2 + 3 - 3 = 12. X^2 + 3 - 3 = 12. X^2 + 3 - 3 = 12. X^2 + 3 - 3 = 12. X^2 + 3 - 3 = 12. X^2 + 3 - 3 = 12. X^2 + 3 - 3 = 12. X^2 + 3 - 3 = 12. X^2 + 3 - 3 = 12. X^2 + 3 - 3 = 12. X^2 + 3 - 3 = 12. X^2 + 3 - 3 = 12. X^2 + 3 93177175136111938711541797
DPO+NLL (100 Epochs) $\mathbb{E}[KL] = 347$	X = 12 / (12 + 3) = 12 / 15 = 8 / 15 = 523 / 15 = 346 / 15 = 227 / 15 = 207 / 15 = 151 / 15 = 8427 / 15 = 5237 / 15 = 2077 / 15 = 1527 / 15 = 8417 / 15 = 5237 / 15 = 2077 / 15 = 1527 / 15 = 8417 / 15 = 5237 / 15 = 2077 / 15 = 1527 / 15 = 8417 / 15 = 5237 / 15 = 2077 / 15 = 1527 / 15 = 8417 / 15 = 5237 / 15 = 2077 / 15 = 1527 / 15 = 8417 / 15 = 5237 / 15 = 2077 / 15 = 1527 / 15 = 8417 / 15 = 5237 / 15 = 2077 / 15 = 1527 / 15 = 8417 / 15 = 5237 / 15 = 2077 / 15 = 1527 / 15 = 8417 / 15 = 5237 / 15 = 2077 / 15 ...
DPOP (100 Epochs) $\mathbb{E}[KL] = 240$	If X^2 + 3 = 12, then we can find the value of X by following these steps: 1. Start by subtracting 3 from both sides of the equation: X^2 + 3 = 12. X^2 = 12 - 3. X^2 = 8. 2. Now we have the equation in a simpler form: X^2 = 8. 3. Taking the square root of both sides: X = $\sqrt{8}$. X = ±0.8 (9). X = 0.8 (10). 4. Therefore, the value of X is 0.8. So the correct answer is (A) 0.8. Confidence: 95%
BDPO (100 Epochs) $\mathbb{E}[KL] = 177$	Hello! I'm here to help you in a helpful and respectful manner. To answer your question, if X^2 + 3 = 12, we can substitute X^2 with 12 - 3, which equals 9. So, X^2 + 3 = 9, and solving for X^2 gives us X^2 = 9. Taking the square root of 9 gives us X = 3, or -3, if we want to solve for the negative square root. I hope this helps! Let me know if you have any other questions.

Table 6: The responses generated by the initial model (Reference model) and the models trained for 100 epochs using the reduced UFB dataset are presented. The DPO+NLL model, diverging significantly from the Reference model, produced overfitted outputs. In contrast, the BDPO model generated desirable outputs.

G.2 Analysis with Large λ Values

In Section 5.4, we noted that the BDPO loss reduces to DPO when $\lambda = 1$. However, Table 5 shows that BDPO still achieves substantial performance gains over DPO on GSM8K even at $\lambda = 0.9$. To investigate this behavior in greater detail, we conducted additional experiments with values closer to 1, namely $\lambda = 0.999$ and $\lambda = 0.99999$.

Base Model	Algorithm	Inst-level Loose Acc	Inst-level Strict Acc	Prompt-level Loose Acc	Prompt-level Strict Acc	Loose Score	Strict Score	Total Score
Qwen2.5-0.5B	DPO+NLL ($\alpha = 0.01$)	31.65	30.94	19.41	19.04	25.53	24.99	25.26
	DPO+NLL ($\alpha = 0.05$)	32.13	31.41	20.89	20.33	26.51	25.87	26.19
	DPO+NLL ($\alpha = 0.1$)	32.73	31.29	21.07	19.96	26.90	25.62	26.26
	DPO+NLL ($\alpha = 0.25$)	29.98	29.62	18.30	17.74	24.14	23.68	23.91
	DPO+NLL ($\alpha = 0.5$)	31.77	30.70	20.33	19.41	26.05	25.05	25.55
	DPO+NLL ($\alpha = 1$)	32.37	30.58	21.07	19.78	26.72	25.18	25.95
	DPO+NLL ($\alpha = 2$)	32.25	31.06	20.33	19.59	26.29	25.32	25.81
	DPO+NLL ($\alpha = 10$)	32.49	31.65	21.44	20.89	26.97	26.27	26.62
	DPOP ($\lambda = 5$)	32.85	31.41	21.44	20.14	27.15	25.78	26.46
	DPOP ($\lambda = 50$)	32.61	31.18	20.70	19.41	26.67	25.30	25.98
	DPOP ($\lambda = 500$)	32.13	30.58	19.04	17.74	25.59	24.16	24.87
	BDPO ($\lambda = 0.5$)	**33.09**	**31.89**	**22.37**	**21.26**	**27.73**	**26.58**	**27.15**

Table 7: IFEval results for Qwen2.5-0.5B trained on the UltraFeedback dataset using DPO+NLL with various α values, DPOP with different λ values, and BDPO ($\lambda = 0.5$). BDPO achieves the highest total score, demonstrating strong alignment performance.

Base Model	Algorithm	Inst-level Loose Acc	Inst-level Strict Acc	Prompt-level Loose Acc	Prompt-level Strict Acc	Loose Score	Strict Score	Total Score
Qwen2.5-0.5B	BDPO ($\lambda = 0.999$)	31.53	30.22	21.63	20.15	26.58	25.19	25.88
	BDPO ($\lambda = 0.99999$)	31.06	30.34	20.89	19.96	25.98	25.15	25.56
	DPO	31.53	30.46	20.33	18.67	25.93	24.57	25.25

Table 8: IFEval results for Qwen2.5-0.5B trained on the UltraFeedback dataset using BDPO with large λ values and DPO. BDPO ($\lambda = 0.999$) achieves the highest total score among these settings.

The results, presented in Table 8, confirm that as λ approaches 1, the performance of BDPO converges to that of DPO. In conclusion, while BDPO indeed becomes equivalent to DPO at $\lambda = 1$, at values such as $\lambda = 0.9$ it still exhibits meaningfully different behavior.

H Artifact Use Consistent With Intended Use

We used the UltraFeedback and UltraInteract datasets and Qwen models strictly for academic research, consistent with their intended use.

I Use of AI Assistants

We used AI-assisted tools during the writing process of this paper. All AI-generated content was thoroughly reviewed and revised by human researchers to ensure accuracy and reliability.

J Proofs

J.1 Proof of Theorem 1

Proof. Since the logarithm and sigmoid functions are strictly increasing, the solution to the problem of minimizing the BDPO loss is equivalent to the solution of the following maximization problem:

$$\left(\log \frac{\pi_\theta(\mathbf{y}_w|\mathbf{x})}{\lambda\pi_\theta(\mathbf{y}_l|\mathbf{x}) + (1-\lambda)\pi_{\text{ref}}(\mathbf{y}_l|\mathbf{x})} - \log \frac{\pi_{\text{ref}}(\mathbf{y}_w|\mathbf{x})}{\pi_{\text{ref}}(\mathbf{y}_l|\mathbf{x})}\right). \tag{3}$$

Since π_{ref} is a fixed distribution, solving the optimization problem (3) is equivalent to solving the following maximization problem:

$$\frac{\pi_\theta(\mathbf{y}_w|\mathbf{x})}{\lambda\pi_\theta(\mathbf{y}_l|\mathbf{x}) + (1-\lambda)\pi_{\text{ref}}(\mathbf{y}_l|\mathbf{x})}.$$

Given that $\pi_\theta(\mathbf{y}_{\{w,\,l\}}|\mathbf{x}) \geq 0$ and $\pi_{\text{ref}}(\mathbf{y}_l|\mathbf{x}) > 0$. The solution of maximizing the optimization problem is to set:

$$\pi_\theta(\mathbf{y}_w|\mathbf{x}) = 1 \quad \text{and} \quad \pi_\theta(\mathbf{y}_l|\mathbf{x}) = 0.$$

Thus, the theorem is proven. □

J.2 Proof of Corollary 1

Proof. ($\rightarrow$) Let π^* denote the optimal solution under BDPO. By Theorem 1, π^* satisfies the following conditions:

$$\pi^*(\mathbf{y}_w|\mathbf{x}) = 1 \quad \text{and} \quad \pi^*(\mathbf{y}_l|\mathbf{x}) = 0.$$

Thus, π^* is the solution to the optimization problem:

$$\max \frac{\pi_\theta(\mathbf{y}_w|\mathbf{x})}{\pi_\theta(\mathbf{y}_l|\mathbf{x})}.$$

Since the logarithm function is strictly increasing, this optimization problem is equivalent to:

$$\max \left[\log \frac{\pi_\theta(\mathbf{y}_w|\mathbf{x})}{\pi_\theta(\mathbf{y}_l|\mathbf{x})} + \log \frac{\pi_{\text{ref}}(\mathbf{y}_w|\mathbf{x})}{\pi_{\text{ref}}(\mathbf{y}_l|\mathbf{x})}\right].$$

Furthermore, because the sigmoid function is also strictly increasing, this optimization is equivalent to minimizing the DPO loss as defined in Eq. (2). Therefore, π^* is a policy that optimizes DPO loss.
($\leftarrow$) Let a policy π_θ be such that:

$$\pi(\mathbf{y}_w|\mathbf{x}) = 0.1 \quad \text{and} \quad \pi(\mathbf{y}_l|\mathbf{x}) = 0.$$

This policy is a solution to the optimization problem of minimizing the DPO loss. Suppose that this policy is a solution to the BDPO loss optimization problem. By Theorem 1, it follows that $\pi(\mathbf{y}_w|\mathbf{x}) = 1$, which leads to a contradiction. □

J.3 Proof of Theorem 2

Proof. As in the proof of Theorem 1, minimizing the BDPO loss is equivalent to solving the following maximization problem:

$$\frac{\pi_\theta(\mathbf{y}_w|\mathbf{x})}{\lambda\pi_\theta(\mathbf{y}_l|\mathbf{x}) + (1-\lambda)\pi_{\text{ref}}(\mathbf{y}_l|\mathbf{x})}.$$

At the initial point, π_θ is equal to π_{ref}. By the assumption that the BDPO loss decreases monotonically at each optimization step, the following inequality holds:

$$\begin{aligned}&\frac{\pi_{\text{ref}}(\mathbf{y}_w|\mathbf{x})}{\lambda\pi_{\text{ref}}(\mathbf{y}_l|\mathbf{x}) + (1-\lambda)\pi_{\text{ref}}(\mathbf{y}_l|\mathbf{x})} \\ &\qquad \leq \frac{\pi_\theta(\mathbf{y}_w|\mathbf{x})}{\lambda\pi_\theta(\mathbf{y}_l|\mathbf{x}) + (1-\lambda)\pi_{\text{ref}}(\mathbf{y}_l|\mathbf{x})}.\end{aligned}$$

This inequality is equivalent to:

$$\pi_{\text{ref}}(\mathbf{y}_w|\mathbf{x})\big(\lambda\pi_\theta(\mathbf{y}_l|\mathbf{x}) + (1-\lambda)\pi_{\text{ref}}(\mathbf{y}_l|\mathbf{x})\big) \\ \leq \pi_\theta(\mathbf{y}_w|\mathbf{x})\pi_{\text{ref}}(\mathbf{y}_l|\mathbf{x}).$$

Since $\pi_\theta(\mathrm{y}_l|\mathrm{x}) \geq 0$ and $\pi_{\text{ref}}(\mathrm{y}_l|\mathrm{x}) \geq 0$, the following inequality holds:

$$(1-\lambda)\pi_{\text{ref}}(\mathbf{y}_w|\mathbf{x}) \leq \pi_\theta(\mathbf{y}_w|\mathbf{x}).$$

Thus, the theorem is proven. □

K Details on the Derivative of $\pi_\theta(\mathbf{y}_l|\mathbf{x})$

BDPO Loss Gradient The BDPO Loss is:

$$\mathcal{L}_{\text{BDPO}} = -\mathbb{E}_{\mathcal{D}}\left[\log\sigma\left(\beta\left(\log\frac{\pi_\theta(\mathbf{y}_w|\mathbf{x})}{\pi_{\text{mix}}(\mathbf{y}_l|\mathbf{x})} - \log\frac{\pi_{\text{ref}}(\mathbf{y}_w|\mathbf{x})}{\pi_{\text{ref}}(\mathbf{y}_l|\mathbf{x})}\right)\right)\right],$$

where $\pi_{\text{mix}}(\mathbf{y}|\mathbf{x}) = \lambda\pi_\theta(\mathbf{y}|\mathbf{x}) + (1-\lambda)\pi_{\text{ref}}(\mathbf{y}|\mathbf{x})$.

To simplify notation, we define Δ_{BDPO}:

$$\Delta_{\text{BDPO}} := \beta\left(\log\frac{\pi_\theta(\mathbf{y}_w|\mathbf{x})}{\pi_{\text{mix}}(\mathbf{y}_l|\mathbf{x})} - \log\frac{\pi_{\text{ref}}(\mathbf{y}_w|\mathbf{x})}{\pi_{\text{ref}}(\mathbf{y}_l|\mathbf{x})}\right).$$

Then, we can write $\mathcal{L}_{\text{BDPO}} = -\mathbb{E}_{\mathcal{D}}\left[\log\sigma(\Delta_{\text{BDPO}})\right]$.

Next, we compute the gradient of $\log\pi_{\text{mix}}(\mathbf{y}_l|\mathbf{x})$ with respect to $\pi_\theta(\mathbf{y}_l|\mathbf{x})$. Applying the chain rule to the logarithm, we obtain

$$\nabla_{\pi_\theta(\mathbf{y}_l|\mathbf{x})}\log\pi_{\text{mix}}(\mathbf{y}_l|\mathbf{x}) = +\frac{\lambda}{\pi_{\text{mix}}(\mathbf{y}_l|\mathbf{x})}.$$

Using this result, we differentiate Δ_{BDPO} with respect to $\pi_\theta(\mathbf{y}_l|\mathbf{x})$, yielding

$$\nabla_{\pi_\theta(\mathbf{y}_l|\mathbf{x})}\Delta_{\text{BDPO}} = -\frac{\beta\lambda}{\pi_{\text{mix}}(\mathbf{y}_l|\mathbf{x})}.$$

To compute the gradient of the loss function, we apply the chain rule to $\log\sigma(\Delta_{\text{BDPO}})$, which gives

$$\nabla_{\pi_\theta(\mathbf{y}_l|\mathbf{x})}\log\sigma(\Delta_{\text{BDPO}}) = \sigma(-\Delta_{\text{BDPO}})\cdot\nabla_{\pi_\theta(\mathbf{y}_l|\mathbf{x})}\Delta_{\text{BDPO}} \\ = -\sigma(-\Delta_{\text{BDPO}})\cdot\left(\frac{\beta\lambda}{\pi_{\text{mix}}(\mathbf{y}_l|\mathbf{x})}\right).$$

Finally, taking the expectation over $\mathcal{D}$, the gradient of $\mathcal{L}_{\text{BDPO}}$ with respect to $\pi_\theta(\mathbf{y}_l|\mathbf{x})$ is given by

$$\nabla_{\pi_\theta(\mathbf{y}_l|\mathbf{x})}\mathcal{L}_{\text{BDPO}} = \mathbb{E}_{\mathcal{D}}\left[\beta\cdot\sigma(-\Delta_{\text{BDPO}})\cdot\frac{\lambda}{\pi_{\text{mix}}(\mathbf{y}_l|\mathbf{x})}\right].$$

DPO Loss Gradient The DPO Loss is:

$$\mathcal{L}_{\text{DPO}} = -\mathbb{E}_{\mathcal{D}}\left[\log\sigma\left(\beta\left(\log\frac{\pi_\theta(\mathbf{y}_w|\mathbf{x})}{\pi_{\text{ref}}(\mathbf{y}_w|\mathbf{x})} - \log\frac{\pi_\theta(\mathbf{y}_l|\mathbf{x})}{\pi_{\text{ref}}(\mathbf{y}_l|\mathbf{x})}\right)\right)\right].$$

we define $\Delta_{\text{DPO}} := \beta\left(\log\frac{\pi_\theta(\mathbf{y}_w|\mathbf{x})}{\pi_{\text{ref}}(\mathbf{y}_w|\mathbf{x})} - \log\frac{\pi_\theta(\mathbf{y}_l|\mathbf{x})}{\pi_{\text{ref}}(\mathbf{y}_l|\mathbf{x})}\right)$.

Next, we compute the gradient of Δ_{DPO} with respect to $\pi_\theta(\mathbf{y}_l|\mathbf{x})$. Since only the second term depends on $\pi_\theta(\mathbf{y}_l|\mathbf{x})$, the gradient is:

$$\nabla_{\pi_\theta(\mathbf{y}_l|\mathbf{x})}\Delta_{\text{DPO}} = -\frac{\beta}{\pi_\theta(\mathbf{y}_l|\mathbf{x})}.$$

Similarly BDPO, we obtain:

$$\nabla_{\pi_\theta(\mathbf{y}_l|\mathbf{x})}\mathcal{L}_{\text{DPO}} = \mathbb{E}_{\mathcal{D}}\left[\beta\cdot\sigma(-\Delta_{\text{DPO}})\cdot\frac{1}{\pi_\theta(\mathbf{y}_l|\mathbf{x})}\right].$$

DPO+NLL Loss Gradient The DPO+NLL Loss is:

$$\mathcal{L}_{\text{DPO+NLL}} = \mathcal{L}_{\text{DPO}} + \alpha \mathcal{L}_{\text{NLL}}, \quad \text{where } \mathcal{L}_{\text{NLL}} = -\mathbb{E}_{\mathcal{D}} \left[\log \pi_\theta(\mathbf{y}_w|\mathbf{x}) \right].$$

Since $\mathcal{L}_{\text{NLL}}$ does not contribute to $\nabla_{\pi_\theta(\mathbf{y}_l|\mathbf{x})}$, the gradient is the same as DPO:

$$\nabla_{\pi_\theta(\mathbf{y}_l|\mathbf{x})} \mathcal{L}_{\text{DPO+NLL}} = \nabla_{\pi_\theta(\mathbf{y}_l|\mathbf{x})} \mathcal{L}_{\text{DPO}} = \mathbb{E}_{\mathcal{D}} \left[\beta \cdot \sigma(-\Delta_{\text{DPO}}) \cdot \frac{1}{\pi_\theta(\mathbf{y}_l|\mathbf{x})} \right].$$

DPOP Loss Gradient The DPOP Loss is:

$$\mathcal{L}_{\text{DPOP}} = -\mathbb{E}_{\mathcal{D}} \left[\log \sigma \left(\beta \left(\log \frac{\pi_\theta(\mathbf{y}_w|\mathbf{x})}{\pi_{\text{ref}}(\mathbf{y}_w|\mathbf{x})} - \log \frac{\pi_\theta(\mathbf{y}_l|\mathbf{x})}{\pi_{\text{ref}}(\mathbf{y}_l|\mathbf{x})} - \lambda \cdot \max \left(0, \log \frac{\pi_{\text{ref}}(\mathbf{y}_w|\mathbf{x})}{\pi_\theta(\mathbf{y}_w|\mathbf{x})} \right) \right) \right) \right].$$

we define $\Delta_{\text{DPOP}} := \beta \left(\log \frac{\pi_\theta(\mathbf{y}_w|\mathbf{x})}{\pi_{\text{ref}}(\mathbf{y}_w|\mathbf{x})} - \log \frac{\pi_\theta(\mathbf{y}_l|\mathbf{x})}{\pi_{\text{ref}}(\mathbf{y}_l|\mathbf{x})} - \lambda \cdot \max \left(0, \log \frac{\pi_{\text{ref}}(\mathbf{y}_w|\mathbf{x})}{\pi_\theta(\mathbf{y}_w|\mathbf{x})} \right) \right)$.
The penalty term $\max \left(0, \log \frac{\pi_{\text{ref}}(\mathbf{y}_w|\mathbf{x})}{\pi_\theta(\mathbf{y}_w|\mathbf{x})} \right)$ does not depend on $\pi_\theta(\mathbf{y}_l|\mathbf{x})$. Thus:

$$\nabla_{\pi_\theta(\mathbf{y}_l|\mathbf{x})} \Delta_{\text{DPOP}} = -\frac{\beta}{\pi_\theta(\mathbf{y}_l|\mathbf{x})}.$$

The gradient of $\mathcal{L}_{\text{DPOP}}$ with respect to $\pi_\theta(\mathbf{y}_l|\mathbf{x})$ is:

$$\nabla_{\pi_\theta(\mathbf{y}_l|\mathbf{x})} \mathcal{L}_{\text{DPOP}} = \mathbb{E}_{\mathcal{D}} \left[\beta \cdot \sigma(-\Delta_{\text{DPOP}}) \cdot \frac{1}{\pi_\theta(\mathbf{y}_l|\mathbf{x})} \right].$$