SWEPO: SIMULTANEOUS WEIGHTED PREFERENCE OPTIMIZATION FOR GROUP CONTRASTIVE ALIGNMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Direct Preference Optimization (DPO) has proven effective in aligning large language models with human preferences but is often constrained to pairwise comparisons – overlooking additional positive and negative responses that are commonly available in real-world settings. We propose **Simultaneous Weighted Preference Optimization** (SWEPO), which incorporates multiple responses per query and prioritizes those that deviate most from the average reward. This deviation-based weighting focuses training on the most informative outliers, akin to a built-in curriculum. Theoretically, we prove that such multi-preference sampling lowers alignment bias, bounding the expected deviation from the true acceptable-response distribution at a rate of $\mathcal{O}\big(\frac{1}{\sqrt{k}}\big)$. Empirically, SWEPO outperforms state-of-the-art baselines on the Ultra-Feedback dataset and demonstrates substantial improvements over DPO and InfoNCA, yielding boosts of up to $\sim 4\%$ on length-controlled win-rate on *AlpacaEval*.

1 INTRODUCTION

Large language models (LLMs) are becoming integral to various applications that demand safe, reliable, and contextually appropriate outputs (Liu et al., 2023b; Ouyang et al., 2022; Christiano et al., 2017). To ensure these models operate in line with human values, *alignment* mechanisms such as Reinforcement Learning from Human Feedback (RLHF) (Christiano et al., 2017) and direct preference-based methods (Rafailov et al., 2024; Xu et al., 2024b) have been adopted. However, alignment remains difficult: human values are complex and heterogeneous, and even small misalignments can lead to undesired or harmful outputs (Wang et al., 2023). This underscores a pressing need for more robust alignment strategies—both in theoretical grounding and practical efficacy.

A popular approach for alignment is Direct Preference Optimization (DPO), which views human feedback as pairs of positive (preferred) and negative (disfavored) responses (Rafailov et al., 2024). While this pairwise paradigm simplifies training, it often *discards* additional responses that might be available in the data, thereby overlooking a wealth of potentially informative signals. Relying on only one positive versus one negative can induce *alignment bias* due to sample-specific quirks (Zhang et al., 2024; Cui et al., 2024; Tao et al., 2024). Moreover, comparing only two responses fails to capture the full spectrum of acceptable or unacceptable qualities that might exist for each query.

In reality, many scenarios intrinsically provide multiple candidate responses per query. For example, a single prompt may be answered by several annotators, each producing a distinct response (Liu et al., 2024b). Alternatively, one could generate multiple completions from a single LLM via different sampling seeds, or collect responses from multiple LLMs and aggregate the feedback (Cui et al., 2023). Some queries also naturally admit multiple correct answers, especially in open-ended or creative tasks (Yang et al., 2024). In such cases, restricting oneself to a single pairwise comparison not only discards valuable data but also risks overfitting to narrow definitions of what constitutes a "good" (or "bad") response, corresponding to a better understanding of the full reward preference distribution over the response space.

Moving from pairwise to *multi-preference* optimization is more than an incremental data usage improvement—it can systematically reduce alignment bias. As we show theoretically (Section 5), incorporating k positive and k negative responses per query shrinks the expected gap between learned preferences and the true acceptable-response distribution at an order of $\mathcal{O}(1/\sqrt{k})$. Intuitively, viewing multiple positive or negative samples helps identify recurring signals of quality (e.g., factual correctness, safety) or shortfalls (e.g., toxicity, hallucinations), stabilizing the learned alignment beyond any single example's idiosyncrasies (Bengio et al., 2009).

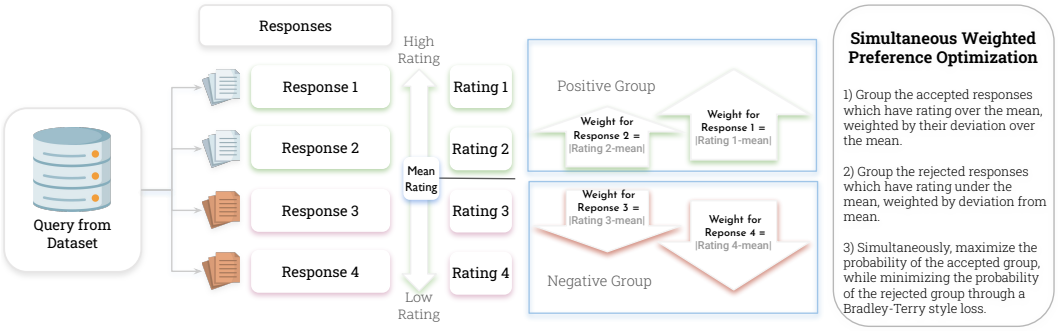

Figure 1: The workflow shows how responses from a multi-preference dataset are used for alignment. Each response (4 responses in the illustration) receives a rating, which is compared against the mean. We then calculate weights based on the absolute difference between each rating and the mean. Accepted responses (above mean) and rejected responses (below mean) are grouped separately, with their respective weights determining their influence in the optimization process. The final step employs a Bradley-Terry style loss function to simultaneously maximize the probability of the accepted group while minimizing the probability of the rejected group.

To fully exploit these multi-response settings, we propose *Simultaneous Weighted Preference Optimization* (SWEPO). As illustrated in Figure 1, SWEPO groups all responses for a given query into "accepted" (above-mean reward) and "rejected" (below-mean reward) categories (Cui et al., 2023), then assigns larger weights to the most deviant responses. These weights are used in a group contrastive loss that systematically boosts the probability of strongly positive responses while actively suppressing strongly negative ones. This design effectively functions like a built-in curriculum, letting the model learn most aggressively from outliers—either particularly good or particularly poor responses—while still incorporating medium-quality responses more moderately (Graves et al., 2017).

We validate SWEPO on real-world datasets where multiple responses per query are available, demonstrating state-of-the-art results compared to both standard DPO and existing multi-preference baselines (e.g., InfoNCA (Chen et al., 2024a)). Empirically, SWEPO significantly improves alignment metrics on AlpacaEval, Arena-Hard, and MT-Bench, confirming that multi-sample preference learning is not only theoretically justified but practically advantageous. The remainder of this paper is organized as follows: Section 3 defines the core notations and problem setup, Section 4 describes SWEPO in detail, Section 5 provides formal analysis of our approach, and Section 6.2 reports our experimental results.

1.1 Our Contributions

Our work provides three main contributions, detailed as follows:
1. Algorithmic Novelty: We propose a novel *rating-aware multi-preference* extension to Direct Preference Optimization. Specifically, we introduce **Simultaneous Weighted Preference Optimization** (SWEPO), which harnesses scalar rewards from multiple positive *and* negative responses per query. By assigning higher weights to responses that deviate more from the mean reward score, SWEPO effectively prioritizes the most informative outliers—akin to a built-in curriculum. This design stands in contrast to standard DPO, which relies on a single pairwise comparison and does not fully exploit the richer data distributions now accessible in modern preference datasets (see Section 4).

2. Theoretical Insights: We provide a rigorous analysis showing that our multi-preference optimization approach systematically reduces alignment bias compared to pairwise-only methods. In particular, we prove that sampling more positive and negative responses per query decreases the expected deviation from the true distribution of acceptable outputs (Section 5.1). Additionally, we compare SWEPO against InfoNCA (Chen et al., 2024a), offering a gradient-level examination (Appendix C and D) that highlights how our weighted contrastive objective naturally pushes negative-response probabilities to zero and maintains a curriculum-like focus on highly deviant samples.

3. State-of-the-Art Results: We validate SWEPO on multiple publicly recognized benchmarks—**AlpacaEval**, **Arena-Hard**, and **MT-Bench**—using various base models (e.g., Mistral-7B

and Llama-3-8B). Our experiments consistently show *state-of-the-art* performance across raw and length-controlled win-rate metrics. Compared to both DPO baselines and multi-reference methods such as InfoNCA, our method achieves the top results in all evaluations (Table 1). These results demonstrate that systematically leveraging multiple preferences via SWEPO yields markedly stronger alignment with human-valued responses.

2 Related Work

Here we give a short description of other preference optimization works and defer a more detailed literature review to Appendix A. DPO (Rafailov et al., 2024) extends early works in RLHF (PPO, TRPO) (Schulman et al., 2017) by incorporating a pairwise contrastive preference optimization.

Recent works extend pairwise preference optimization by incorporating diversified objectives or streamlining reward modeling. For instance, KTO and TDPO target response-level and token-level alignment without needing multiple positive samples per instruction (Ethayarajh et al., 2024; Zeng et al., 2024), while RAFT (Dong et al., 2023) and RRHF (Yuan et al., 2023) use list-wise or rank-based signals to refine preference supervision. Several methods (e.g., SPIN, CPO, ORPO, SimPO, R-DPO, LD-DPO) remove or modify the reference model, add additional regularizers to address length bias or data diversity, or unify preference optimization with supervised losses (Chen et al., 2024b; Xu et al., 2024a; Hong et al., 2024; Meng et al., 2024; Park et al., 2024; Liu et al., 2024a).

Multi-preference Optimization (MPO) has been considered by InfoNCA (Chen et al., 2024a), and we provide an alternative method for MPO. Both these works are enabled through the dataset provided by Ultra-Feedback (Cui et al., 2023).

3 Notations and Preliminaries

In this section, we establish the notations and preliminaries necessary for our proposed weighted multi-preference optimization method.

Let $\mathcal{X}$ denote the set of all possible queries, with $x \in \mathcal{X}$ representing a specific query. For each query x, let $\mathcal{Y}_x$ be the set of all potential responses. Our dataset $\mathcal{D}$ consists of N queries, where each query x is associated with n responses $\{y_i\}_{i=1}^n$ and corresponding reward scores $\{S_i\}_{i=1}^n$.

The mean reward score for query x is calculated as:

$$S_{\text{mean}} = \frac{1}{n} \sum_{i=1}^{n} S_i. \tag{1}$$

The deviation of each response's reward score from the mean is:

$$\Delta S_i = S_i - S_{\text{mean}}. \tag{2}$$

We partition the responses into positive and negative sets:

$$Y^+ = \{y_i \mid \Delta S_i > 0\}, \tag{3}$$

$$Y^- = \{y_j \mid \Delta S_j \leq 0\}. \tag{4}$$

Weights are assigned based on the deviation, using an exponential function or a power function. For positive responses ($y_i \in Y^+$):

$$w_i^+ = \exp\left(\alpha \Delta S_i\right) \quad \text{or} \quad w_i^+ = \left(\Delta S_i\right)^p, \tag{5}$$

and for negative responses ($y_j \in Y^-$):

$$w_j^- = \exp\left(\alpha \left(-\Delta S_j\right)\right) \quad \text{or} \quad w_j^- = \left(-\Delta S_j\right)^p, \tag{6}$$

where $\alpha > 0$ is a scaling hyperparameter and $p \in \{0, 1, 2\}$.

The language model parameterized by θ provides the conditional probability $P_\theta(y \mid x)$ of generating response y given query x. The logit or score function is:

$$s_\theta(y \mid x) = \log\left(\frac{P_\theta(y \mid x)}{P_{\text{ref}}(y \mid x)}\right) \tag{7}$$

$$= \log P_\theta(y \mid x) - \log P_{\text{ref}}(y \mid x). \tag{8}$$

Algorithm 1 Simultaneous Weighted Preference Optimization (SWEPO)

1: **Input:** Initial model parameters θ_0, dataset $\mathcal{D}$ with n responses and reward scores per query, scaling hyperparameter α, power $p \in \{0, 1, 2\}$, iterations T
2: **Output:** Optimized model parameters θ_T
3: Initialize $\theta \leftarrow \theta_0$
4: **for** $t = 1$ **to** T **do**
5: **for all** query $x \in \mathcal{D}$ **do**
6: Compute S_{mean}, deviations ΔS_i, and partition responses into Y^+ and Y^-
7: Assign weights: $w_i^+ = (\Delta S_i)^p$, $w_j^- = (-\Delta S_j)^p$
8: Compute scores:
$s_\theta(y \mid x) = \log\left(P_\theta(y \mid x) - P_{\text{ref}}(y \mid x)\right)$
9: Compute modified scores: $s'_\theta(y \mid x) = s_\theta(y \mid x) + \alpha \Delta S$
10: **end for**
11: Compute loss:
$$L_{\text{weighted}}(\theta) \leftarrow -\log\left[\frac{\sum_{y \in Y^+} \exp\left(s'_\theta(y|x)\right)}{\sum_{y \in Y^+ \cup Y^-} \exp\left(s'_\theta(y|x)\right)}\right]$$
12: Update model parameters: $\theta \leftarrow \theta - \eta \nabla_\theta L_{\text{weighted}}(\theta)$
13: **end for**
14: **return** θ

Incorporating the weights into the probabilities, we have:

$$w_i \times \exp\left(s_\theta(y_i \mid x)\right) = \exp\left(\alpha \Delta S_i + s_\theta(y_i \mid x)\right), \tag{9}$$

which leads to the modified score:

$$s'_\theta(y_i \mid x) = s_\theta(y_i \mid x) + \alpha \Delta S_i. \tag{10}$$

The weighted contrastive loss function is defined as:

$$L_{\text{weighted}}(\theta) = -\log \frac{\sum\limits_{y \in Y^+} \exp\left(s'_\theta(y \mid x)\right)}{\sum\limits_{y \in Y} \exp\left(s'_\theta(y \mid x)\right)}, \tag{11}$$

where $Y = Y^+ \cup Y^-$.

4 Algorithm and Methodology

We present the *Simultaneous Weighted Preference Optimization* (SWEPO) algorithm, which aligns the language model with human preferences by incorporating multiple responses per query and weighting them based on their deviation from the mean reward score.

4.1 Algorithm 1 Description.

Line 1 specifies the initial settings (model parameters, dataset, hyperparameters) and defines the iteration budget. Line 2 clarifies that the goal is to return the updated parameters after training. Line 3 initializes the model from a chosen starting point, and line 4 begins the main optimization loop over the specified number of iterations.

Within each iteration, line 5 loops through all queries in the dataset. Line 6 computes the mean reward, identifies which responses exceed this mean, and partitions the responses into positive and negative sets. Line 7 assigns weights to these responses by exponentiating their deviations from the mean (with a chosen power) – this is often called the Advantage function in related literature. In lines 8 and 9, we calculate and scale the log-score to measure how the model's probability compares against a reference, wherein the reference model is the same model at initialization. In, line 11 we aggregate these measurements into the SWEPO loss, and line 12 updates the model parameters via a gradient step. Finally, line 14 returns the optimized parameters.

4.2 Weight Computation and Modified Scores

In the notations, we defined weights using an exponential function of the deviation ΔS_i. Specifically, the weight for each response is:

$$w_i = \exp\left(\alpha \Delta S_i\right), \tag{12}$$

for positive responses, and

$$w_j = \exp\left(\alpha\left(-\Delta S_j\right)\right), \tag{13}$$

for negative responses.

By incorporating these weights into the loss function, we observe that:

$$w_i \times \exp\left(s_\theta(y_i \mid x)\right) = \exp\left(\alpha \Delta S_i + s_\theta(y_i \mid x)\right). \tag{14}$$

This demonstrates that weighting the probabilities is equivalent to adjusting the logits by adding the scaled deviation. Thus, the modified score for each response becomes:

$$s'_\theta(y_i \mid x) = s_\theta(y_i \mid x) + \alpha \Delta S_i. \tag{15}$$

4.3 Generalization with Power p

In the algorithm, we generalize the weighting scheme by defining the weights as the p-th power of the deviation:

$$w_i^+ = \left(\Delta S_i\right)^p, \quad \text{for } y_i \in Y^+, \tag{16}$$

$$w_j^- = \left(-\Delta S_j\right)^p, \quad \text{for } y_j \in Y^-, \tag{17}$$

where $p \in \{0, 1, 2\}$. This allows flexibility in modifying the impact of the deviation on the weights. When $p = 0$, all weights are equal to 1, reducing the method to an unweighted loss.

5 Theoretical Analysis

In this section, we provide theoretical insights into why incorporating multiple preferences per query, as in our proposed SWEPO method, leads to better alignment with human values compared to methods that rely on pairwise preferences, such as Direct Preference Optimization (DPO). We also differentiate our SWEPO loss from the InfoNCA loss (Chen et al., 2024a) and discuss the implications for model optimization.

5.1 Bias Reduction through Multiple Preferences

A key motivation for using multiple preferences per query is to reduce *alignment bias*, which arises when sampling a limited subset of responses from the distribution of acceptable and suboptimal responses. We formalize this intuition by analyzing how the expected bias with respect to an attribute decreases as the number of samples increases.

To analyze biases, we introduce an **attribute function** $a(y) : \mathcal{Y}_x \to \mathbb{R}$, which maps responses to real numbers (e.g., response length, politeness).

The **expected attribute value over the model's distribution** is defined as:

$$\mu_\theta = \mathbb{E}_{x \sim \mathcal{X}}\left[\mathbb{E}_{y \sim P_\theta(\cdot|x)}\left[a(y)\right]\right], \tag{18}$$

where $P_\theta(\cdot \mid x)$ is the model's conditional distribution over responses given query x.

The **true expected attribute value over acceptable responses** is:

$$\mu_{\mathcal{A}} = \mathbb{E}_{x\sim\mathcal{X}}\left[\mathbb{E}_{y\sim\mathcal{A}_x}\left[a(y)\right]\right], \tag{19}$$

where $\mathcal{A}_x$ is the distribution of acceptable responses for query x.

The **bias** with respect to attribute a is then defined as:

$$B^{(k)} = \left|\mu_\theta^{(k)} - \mu_{\mathcal{A}}\right|, \tag{20}$$

where $\mu_\theta^{(k)}$ is the expected attribute value under the model after training with k positive and k negative samples per query.

5.1.1 Assumptions

We make the following assumptions:

- **Finite Variance:** The attribute $a(y)$ has finite variance over the acceptable response distribution $\mathcal{A}_x$ for each query x, i.e., $\mathrm{Var}_{y\sim\mathcal{A}_x}[a(y)] = \sigma^2_{\mathcal{A}_x} < \infty$.
- **Independent Sampling:** Responses are independently sampled from their respective distributions.
- **Model Capacity:** The model can represent the true distribution given sufficient data.
- **Uniform Bounded Variance:** There exists a constant $\sigma^2_{\max}$ such that $\sigma^2_{\mathcal{A}_x} \le \sigma^2_{\max}$ for all $x \in \mathcal{X}$.

Theorem 1. Under the stated assumptions, the expected bias $\mathbb{E}[B^{(k)}]$ decreases with the number of samples k as

$$\mathbb{E}[B^{(k)}] \le \frac{C}{\sqrt{k}}, \tag{21}$$

where $C = \sigma_{\max}$ is a constant depending on the maximum variance of $a(y)$ over acceptable responses.

Proof Sketch. The proof relies on the Central Limit Theorem and the fact that the standard error of the mean decreases with the square root of the number of samples. For each query x, the sample mean of the attribute over the k positive responses converges to the true mean $\mu_{\mathcal{A}_x}$ at a rate proportional to $1/\sqrt{k}$. Averaging over all queries, we obtain the bound on $\mathbb{E}[B^{(k)}]$. □

Corollary 1. As $k \to \infty$, the expected bias $\mathbb{E}[B^{(k)}]$ approaches zero:

$$\lim_{k\to\infty} \mathbb{E}[B^{(k)}] = 0. \tag{22}$$

Takeaway: Sampling multiple (y^+, y^-) pairs per query x, rather than single pairwise comparisons, enables better convergence to the true acceptable response distribution by averaging out sample-specific artifacts. This multi-sample approach improves robustness by allowing the model to identify consistent patterns of desirability across the response space rather than overfitting to individual sample characteristics, and this leads to a more consistent and stable model alignment.

We provide a deep-dive into this proof in Appendix B. Furthermore, Appendix C provides a comprehensive comparison between the Group Contrastive Loss and InfoNCA Loss, including detailed gradient analyses, and Appendix D offers a characterization of stationary points for both the InfoNCA and Weighted Contrastive Loss functions. We now offer two take-aways from our theorems.

Our loss function drives multiple undesirable response probabilities towards 0: Intuitively, as the probabilities of the negative samples approach zero, the weighted contributions of negative samples to both the numerator and denominator in L_{weighted} become negligible. The loss function simplifies to:

$$L_{\text{weighted}} = -\log\left(\frac{\sum\limits_{i\in Y^+} w_i e^{s_\theta(y_i|x)}}{\sum\limits_{j=1}^{K} w_j e^{s_\theta(y_j|x)}}\right) \approx -\log\left(\frac{\sum\limits_{i\in Y^+} w_i e^{s_\theta(y_i|x)}}{\sum\limits_{j\in Y^+} w_j e^{s_\theta(y_j|x)}}\right) = -\log 1 = 0.$$

This indicates that the loss (which is non-negative) vanishes when the model assigns negligible probabilities to negative samples, resulting in a model that is trained to avoid generating negative outputs.

Method	Mistral-Base (7B)				Llama-3-Base (8B)			
	AlpacaEval 2		Arena-Hard	MT-Bench	AlpacaEval 2		Arena-Hard	MT-Bench
	LC (%)	WR (%)	WR (%)	GPT-4	LC (%)	WR (%)	WR (%)	GPT-4
SFT	8.4	6.2	1.3	6.3	6.2	4.6	3.3	6.6
DPO	16.59	13.76	12.7	6.71	16.87	14.06	**18.5**	7.71
DPOx3	14.86	11.7	8.8	6.93	16.27	13.13	13.7	7.5
InfoNCA	14.76	10.79	9.7	7.04	15.89	12.9	14.8	7.56
SWEPO-1-vs-k (p = 0)	15.16	11.45	10.1	7.1	17.3	13.46	15.9	7.57
SWEPO-dynamic (p = 0)	18.35	14.37	**13.2**	7.18	18.36	15.06	18.4	7.53
SWEPO-dynamic (p = 1)	**20.32**	**14.94**	12.8	**7.25**	18.89	15.26	18.1	7.61
SWEPO-dynamic (p = 2)	18.04	13.91	11.8	7.19	**20.09**	**15.63**	**18.5**	**7.77**

Table 1: Comparison of preference optimization methods on AlpacaEval, Arena-Hard and MT-Bench benchmarks. LC-WR represents length-controlled win rate, and WR represents raw win rate. Best results are in **bold**, second-best are underlined. Our method (SWEPO) achieves SOTA performance across all metrics, with different variants achieving either best or second-best results consistently.

Weights create a Curriculum: In our loss function, the gradients are weighted by the weights w_i assigned to them. Therefore the high weight negative examples have probability sent to 0 first and later the lower weight ones. This has parallels with *curriculum learning* wherein, models are first exposed to more informative examples, gradually moving towards less informative ones. This serves as a built-in curriculum, guiding the model to focus gradients to the most instructive examples (high weight) examples first.

These theoretical insights provide strong justification for our proposed SWEPO method over traditional pairwise preference optimization methods. By using multiple weighted preferences, we achieve a more nuanced and effective alignment with human preferences.

6 EXPERIMENTS

6.1 EXPERIMENTAL SETUP

Model and Training Settings: For our experiments, we utilized the Ultrafeedback Dataset Cui et al. (2023), an instruction-following dataset annotated by GPT-4, containing approximately 64,000 instructions. Each instruction includes four responses generated by different language models, with GPT-4 assigning scalar scores on a scale of 0 to 10 for each response. Previous research has shown a strong correlation between these GPT-4 ratings and human annotations, establishing their reliability as a cost-effective alternative to human feedback.

Based on these scalar scores, we categorized the responses into two sets: chosen responses and rejected responses. This categorization was determined using the mean of the scalar scores. Responses scoring above the mean were classified as chosen, while the remaining responses were categorized as rejected.

Our training process aligns with the methodology outlined in Zephyr Tunstall et al. (2023). Initially, we fine-tune a base model, such as mistralai/Mistral-7B-v0.1 or meta-llama/Meta-Llama-3-8B, using the UltraChat-200k dataset Ding et al. (2023) to create a supervised fine-tuned (SFT) model. Subsequently, we apply preference optimization to the UltraFeedback dataset Cui et al. (2023), which consists of four answers per query. This approach ensures a high degree of transparency since the SFT models are derived from publicly available datasets. We applied our proposed preference optimization method, SWEPO, which consists of four configurations:1.) SWEPO-1-vs-k (p=0) 2.) SWEPO-dynamic (p=0) 3.) SWEPO-dynamic (p=1) and 4.) SWEPO-dynamic (p=2)

These configurations were designed to harness scalar scores effectively and improve model alignment. Our findings suggest that these setups significantly enhance performance, placing our models among the top contenders on the Alpaca leaderboard.

Evaluation Benchmarks: We evaluate our models using three widely recognized open-ended instruction-following benchmarks: MT-Bench Zheng et al. (2023), AlpacaEval 2, and Arena-Hard v0.1 Zheng et al. (2023). These benchmarks test the models' conversational versatility across a

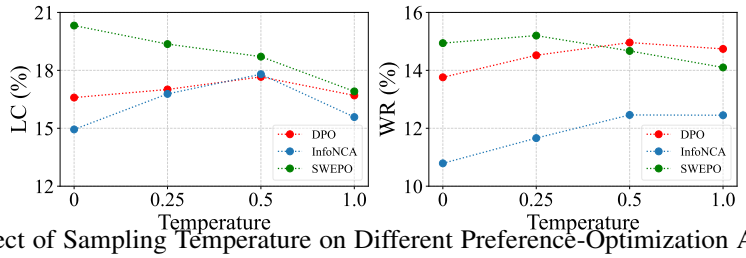

Figure 2: Effect of Sampling Temperature on Different Preference-Optimization Approaches for Mistral-Base (7B) on the AlpacaEval 2 Benchmark: (a) Length-Controlled Win Rate (LC) and (b) Overall Win Rate (WR).

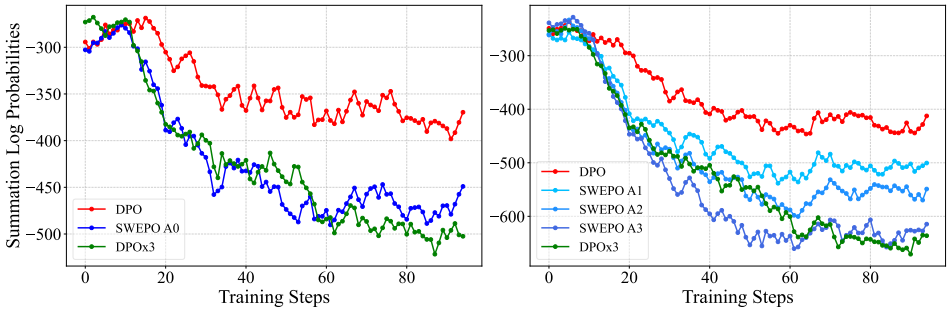

Figure 3: We highlight Avg. Perplexity per batch $\frac{1}{B}\sum_{b=1}^{B}\sum_{i=1}^{n_b}\log p_{b,i}$ for a.) Top-most Responses b.) Rest of the Responses for DPO, DPOx3 and SWEPO. We note that the while SWEPO decreases the probability of the chosen (top-rated) response, more than DPO, it also decreases the probability of negative responses. Furthermore, unlike running DPO three times (DPOx3), we get a separation between the post-training probabilities for the responses, based on their rating.

broad range of queries and are broadly utilized in the research community. AlpacaEval2 Dubois et al. (2024) includes 805 questions derived from five datasets, while MT-Bench spans eight categories with a total of 80 questions. Arena-Hard, a recently updated version of MT-Bench, focuses on 500 well-defined technical problem-solving queries. Scores are reported based on the evaluation protocols of each benchmark. For AlpacaEval2, both the raw win rate (WR) and the length-controlled win rate (LC) are reported, with LC specifically designed to mitigate biases related to model verbosity. For Arena-Hard, the win rate is reported relative to a baseline model. For MT-Bench, the average score is calculated using evaluations by GPT-4 as judge. For decoding details, we generate responses using both greedy decoding and multinomial sampling with temperatures of 0.2, 0.5, and 1.0. To address potential biases introduced by multinomial sampling at varying temperatures, we generate responses three times for each setting at different seed and average their performance across the datasets. We provide details regarding baselines in Appendix E

6.2 Experimental Results

SWEPO outperforms baseline preference-based methods: The results of our ablation study demonstrate that our proposed methods consistently outperform all baseline preference models. Specifically, our methods exhibit a significant performance improvement by achieving of 20.32% and 20.09% LC (%) and 14.94% and 15.63% WR (%) for mistral and llama respectively in the AlpacaEval2 benchmark when compared to the best-performing preference baseline, DPO. This substantial improvement underscores the effectiveness of our approach, SWEPO, which leverages its ability to fully exploit all the information available in the dataset. The enhanced utilization of score signals enables SWEPO to achieve superior performance, **highlighting** the importance of comprehensive information integration in preference-based models.

Method	LC (%)	WR (%)	Var (LC)	Var (WR)
DPO	14.86	12.71	1.96	1.32
SWEPO	**20.32**	**14.94**	**0.096**	**0.11**

Table 2: We sampled the datasets for DPO by changing seeds on Binarized Ultrafeedback dataset (Cui et al., 2023). The dataset for SWEPO stays the same as all the responses are considered. The variance on SWEPO is due to temperature sampling for the judge on AlpacaEval 2.0.

SWEPO vs. InfoNCA on Multi-preference Data We implemented the InfoNCA Chen et al. (2024a) baseline under our custom settings, adhering to the hyperparameters specified in their original method. The primary difference lies in the finetuning approach: while InfoNCA utilizes QLoRA, we opted for full finetuning of their model.

Using this approach and from table 1, we observed that SWEPO outperforms InfoNCA on downstream datasets such as AlpacaEval2, MTBench and Arena-Hard. This result underscores the significance of SWEPO and its clear advantage over InfoNCA.

Robustness of SWEPO to Response Selection To analyze the robustness of DPO and SWEPO to response selection, we evaluate the variance in performance in Mistral-Base (7B). As shown in Table 2, DPO demonstrates significant variability, with a variance of 1.96 in LC (%) and 1.32 in WR (%), caused by random sampling of rejected responses. This indicates DPO's sensitivity to sampling choices, potentially leading to inconsistent alignment performance.In contrast, SWEPO achieves consistent results across all evaluations, with very small variance in both LC (%) and WR (%). By leveraging all responses, **Key takeaway** SWEPO eliminates randomness in rejected response selection, ensuring robust performance.

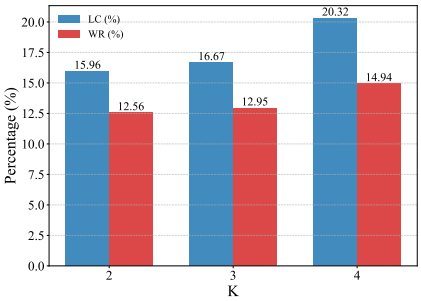

Table 3: Performance variation of SWEPO-Dynamic ($p = 1$) for Mistral-Base (7B) with different numbers of responses (K).

Importance of Suboptimal Responses: Suboptimal responses are also important. Previous practices always ensure selecting the responses with highest scalar score when constructing preference data. The assumption behind this strategy is that the dataset's best-performing response determines the upper limit of alignment performance.However, our experiments contradict this assumption. Results in table 1 indicate that extra suboptimal responses can also be advantageous for policy training. **Specifically, we observe** consistent performance improvements when increasing the number of data responses from K = 2 to K = 4 for SWEPO as shown in Figure 3.

Contrasting Many Positive Responses with Many Negative Responses vs. One Positive Response with Many Negative Responses: In this ablation study, we analyze the difference between contrasting multiple positive responses with multiple negative responses and contrasting a single positive response against all negative responses. Considering only one positive response in a one-vs-all-negative framework may lead to an incomplete representation, as other responses with high positive scores could also contribute meaningful signals. Treating these high-scoring responses as negatives may distort the learning process and hinder model performance. The same has been demonstrated in table 1 and we are getting significant improvement in performance over one-vs-all-negative framework.

Key Takeaway: By incorporating multiple positive responses, we better account for nuanced variations and alignments in the data, leading to more robust and generalizable model behavior. This approach ensures that all positively relevant responses are effectively utilized rather than being misclassified as negatives.

Effect of Logit Weighting on Group Contrastive Loss: In this ablation study, we explore the impact of weighting logits using two approaches: (1) the absolute deviation of scalar scores, and (2) the squared deviation of scalar scores from the mean score value. These weighting schemes are compared against unweighted logits in the context of group contrastive loss.

Our results in table 1 demonstrate a substanial performance improvement with the introduction of logit weighting. In the AlpacaEval2 benchmark, the SWEPO-based weighting scheme outperforms unweighted group contrastive loss, achieving an improvement of 1.98% and 1.73% in LC-WR and 0.57% and 0.57% in WR for Mistral-Base (8B) and Llama-3-Base (8B) . **This highlights** the effectiveness of incorporating deviation-based weighting mechanisms to enhance the model's capability in aligning scores with a contrastive loss objectives.

REFERENCES

Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 1, 2004.

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

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48, 2009.

Huayu Chen, Guande He, Lifan Yuan, Ganqu Cui, Hang Su, and Jun Zhu. Noise contrastive alignment of language models with explicit rewards. *arXiv preprint arXiv:2402.05369*, 2024a.

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

Pengyu Cheng, Yifan Yang, Jian Li, Yong Dai, and Nan Du. Adversarial preference optimization. *arXiv preprint arXiv:2311.08045*, 2023.

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

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Wei Zhu, Yuan Ni, Guotong Xie, Zhiyuan Liu, and Maosong Sun. Ultrafeedback: Boosting language models with high-quality feedback. *arXiv preprint arXiv:2310.01377*, 2023.

Justin Cui, Wei-Lin Chiang, Ion Stoica, and Cho-Jui Hsieh. Or-bench: An over-refusal benchmark for large language models. *arXiv preprint arXiv:2405.20947*, 2024.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. Enhancing chat language models by scaling high-quality instructional conversations. *arXiv preprint arXiv:2305.14233*, 2023.

Hanze Dong, Wei Xiong, Deepanshu Goyal, Yihan Zhang, Winnie Chow, Rui Pan, Shizhe Diao, Jipeng Zhang, Kashun Shum, and Tong Zhang. Raft: Reward ranked finetuning for generative foundation model alignment. *arXiv preprint arXiv:2304.06767*, 2023.

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B Hashimoto. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475*, 2024.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. Kto: Model alignment as prospect theoretic optimization. *arXiv preprint arXiv:2402.01306*, 2024.

Alex Graves, Marc G Bellemare, Jacob Menick, Remi Munos, and Koray Kavukcuoglu. Automated curriculum learning for neural networks. In *international conference on machine learning*, pp. 1311–1320. Pmlr, 2017.

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

Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 133–142, 2002.

Dahyun Kim, Yungi Kim, Wonho Song, Hyeonwoo Kim, Yunsu Kim, Sanghoon Kim, and Chanjun Park. sdpo: Don't use your data all at once. *arXiv preprint arXiv:2403.19270*, 2024.

Tassilo Klein and Moin Nabi. Contrastive perplexity for controlled generation: An application in detoxifying large language models. *arXiv preprint arXiv:2401.08491*, 2024.

Jie Liu, Zhanhui Zhou, Jiaheng Liu, Xingyuan Bu, Chao Yang, Han-Sen Zhong, and Wanli Ouyang. Iterative length-regularized direct preference optimization: A case study on improving 7b language models to gpt-4 level. *arXiv preprint arXiv:2406.11817*, 2024a.

Tianqi Liu, Yao Zhao, Rishabh Joshi, Misha Khalman, Mohammad Saleh, Peter J Liu, and Jialu Liu. Statistical rejection sampling improves preference optimization. *arXiv preprint arXiv:2309.06657*, 2023a.

Yang Liu, Yuanshun Yao, Jean-Francois Ton, Xiaoying Zhang, Ruocheng Guo, Hao Cheng, Yegor Klochkov, Muhammad Faaiz Taufiq, and Hang Li. Trustworthy llms: a survey and guideline for evaluating large language models' alignment. *arXiv preprint arXiv:2308.05374*, 2023b.

Yang Liu, Jiahuan Cao, Chongyu Liu, Kai Ding, and Lianwen Jin. Datasets for large language models: A comprehensive survey. *arXiv preprint arXiv:2402.18041*, 2024b.

Yu Meng, Mengzhou Xia, and Danqi Chen. Simpo: Simple preference optimization with a reference-free reward. *arXiv preprint arXiv:2405.14734*, 2024.

Andrew Y Ng, Stuart Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, pp. 2, 2000.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

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

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

Ryan Park, Rafael Rafailov, Stefano Ermon, and Chelsea Finn. Disentangling length from quality in direct preference optimization. *arXiv preprint arXiv:2403.19159*, 2024.

Biqing Qi, Pengfei Li, Fangyuan Li, Junqi Gao, Kaiyan Zhang, and Bowen Zhou. Online dpo: Online direct preference optimization with fast-slow chasing. *arXiv preprint arXiv:2406.05534*, 2024.

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

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*, 2015.

John Schulman. Trust region policy optimization. *arXiv preprint arXiv:1502.05477*, 2015.

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

Feifan Song, Bowen Yu, Minghao Li, Haiyang Yu, Fei Huang, Yongbin Li, and Houfeng Wang. Preference ranking optimization for human alignment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 18990–18998, 2024.

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

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

Yunhao Tang, Zhaohan Daniel Guo, Zeyu Zheng, Daniele Calandriello, Rémi Munos, Mark Rowland, Pierre Harvey Richemond, Michal Valko, Bernardo Ávila Pires, and Bilal Piot. Generalized preference optimization: A unified approach to offline alignment. *arXiv preprint arXiv:2402.05749*, 2024.

Yan Tao, Olga Viberg, Ryan S Baker, and René F Kizilcec. Cultural bias and cultural alignment of large language models. *PNAS nexus*, 3(9):pgae346, 2024.

Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro von Werra, Clémentine Fourrier, Nathan Habib, et al. Zephyr: Direct distillation of lm alignment. *arXiv preprint arXiv:2310.16944*, 2023.

Yuanhao Wang, Qinghua Liu, and Chi Jin. Is rlhf more difficult than standard rl? a theoretical perspective. *Advances in Neural Information Processing Systems*, 36:76006–76032, 2023.

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

Haoran Xu, Amr Sharaf, Yunmo Chen, Weiting Tan, Lingfeng Shen, Benjamin Van Durme, Kenton Murray, and Young Jin Kim. Contrastive preference optimization: Pushing the boundaries of LLM performance in machine translation. *ArXiv*, abs/2401.08417, 2024a.

Shusheng Xu, Wei Fu, Jiaxuan Gao, Wenjie Ye, Weilin Liu, Zhiyu Mei, Guangju Wang, Chao Yu, and Yi Wu. Is dpo superior to ppo for llm alignment? a comprehensive study. *arXiv preprint arXiv:2404.10719*, 2024b.

Joshua C Yang, Marcin Korecki, Damian Dailisan, Carina I Hausladen, and Dirk Helbing. Llm voting: Human choices and ai collective decision making. *arXiv preprint arXiv:2402.01766*, 2024.

Weizhe Yuan, Ilia Kulikov, Ping Yu, Kyunghyun Cho, Sainbayar Sukhbaatar, Jason Weston, and Jing Xu. Following length constraints in instructions. *arXiv preprint arXiv:2406.17744*, 2024.

Zheng Yuan, Hongyi Yuan, Chuanqi Tan, Wei Wang, Songfang Huang, and Fei Huang. Rrhf: Rank responses to align language models with human feedback without tears. *arXiv preprint arXiv:2304.05302*, 2023.

Yongcheng Zeng, Guoqing Liu, Weiyu Ma, Ning Yang, Haifeng Zhang, and Jun Wang. Token-level direct preference optimization. *arXiv preprint arXiv:2404.11999*, 2024.

Xuanchang Zhang, Wei Xiong, Lichang Chen, Tianyi Zhou, Heng Huang, and Tong Zhang. From lists to emojis: How format bias affects model alignment. *arXiv preprint arXiv:2409.11704*, 2024.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623, 2023.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

SUPPLEMENTARY MATERIALS

These supplementary materials provide additional details, derivations, and experimental results for our paper. The appendix is organized as follows:

- Section A presents a detailed overview of related literature starting from the broader RLHF literature, before moving on to multi-preference optimization and our dataset.
- Section B presents a detailed bias analysis, demonstrating how incorporating multiple preferences reduces alignment bias.
- Section C provides a comprehensive comparison between the Group Contrastive Loss and InfoNCA Loss, including detailed gradient analyses.
- Section D offers a thorough characterization of stationary points for both the InfoNCA and Weighted Contrastive Loss functions.
- Section E describes the baselines used for comparison in our experimental evaluations, including various DPO implementations and alternative approaches.
- Section F provides the implementation details of the reward loss computation, including the actual code used in our experiments.

A RELATED WORK

We will start this literature survey with a high level overview of the RLHF literature and then going deeper into the area of preference, and then multi-preference optimization relevant to our work.

Broader RLHF Literature: Reinforcement Learning through Human feedback (RLHF) has emerged as a robust alignment algorithm for language models. The area broadly started of with works like Trust Region Policy Optimization (TRPO), and Proximal Policy Optimization (PPO) (Schulman, 2015; Schulman et al., 2017) which extend direct RL based methods by constraining the update space to within a trusted region and clipping policy updates to prevent instability respectively. Building upon earlier policy gradient methods (Sutton et al., 1999), PPO has been successfully applied to alignment tasks in Reinforcement Learning from Human Feedback (RLHF), allowing language models to produce outputs aligned with human preferences (Ziegler et al., 2019; Ouyang et al., 2022). Its simplicity and efficiency make it a standard approach for fine-tuning large-scale models. Prior to PPO, Trust Region Policy Optimization (TRPO) (Schulman, 2015) introduced constraints to improve learning stability, influencing the development of PPO. Early applications of policy gradient methods in natural language processing (Ranzato et al., 2015) demonstrated the potential of reinforcement learning for language model training.

Preference Optimization: Direct Preference Optimization (DPO) simplifies the alignment of language models by optimizing a contrastive loss directly over paired preference data, bypassing the intermediate step of reward modeling (Rafailov et al., 2024). Unlike RLHF, DPO does not require explicit reward functions, making it computationally efficient and suitable for limited preference datasets. Recent extensions of DPO, such as Identity Preference Optimization (IPO) (Azar et al., 2024), self-play preference optimization (Wu et al., 2024), preference ranking optimization (Song et al., 2024), rejection sampling optimization (Liu et al., 2023a), and generalized preference optimization (Tang et al., 2024) are amongst the other recent works improve on the DPO method.

Beyond the foundational pairwise approaches and their direct extensions, numerous recent works have proposed methods that adapt or refine DPO-like strategies, often eliminating the need for separate reward modeling or reference models.

Alternative Approaches Without Full Reward Modeling. Ethayarajh et al. (2024) propose **KTO**, a framework inspired by prospect theory that directly learns whether a response is globally

desirable or undesirable, thereby removing the requirement of having multiple positive examples per instruction. Zeng et al. (2024) focus on token-level alignment in **TDPO**, imposing forward KL divergence constraints for each token rather than solely for the final output. This fine-grained approach can mitigate the mode-collapse issues sometimes observed in sequence-level alignment. Meanwhile, Dong et al. (2023) introduce a list-wise method called **RAFT**, where the model finetunes on the best response from each sampled set of k candidates, iteratively converging toward an optimal subset policy. By contrast, Yuan et al. (2023) center on rank-based supervision through **RRHF**, which combines a rank loss with standard supervised signals to ensure the model maintains stronger probabilities on higher-ranked (i.e., better) responses and less on suboptimal responses.

Enhancing DPO with Additional Objectives and Training Schemes. Other works further modify or reinterpret the DPO loss to incorporate new constraints or to remove the need for a reference model. Chen et al. (2024b) propose **SPIN**, which treats the model as part of a two-player adversarial game, obviating separate reward modeling by training with a discriminator that distinguishes human from machine responses. **CPO** (Xu et al., 2024a) reworks the DPO objective by removing the reference-model term and adding a behavior cloning regularizer. Similarly, **ORPO** (Hong et al., 2024) folds preference optimization into a negative log-likelihood objective via an odds-ratio penalty, thereby unifying supervised fine-tuning (SFT) and preference training. In **SimPO**, Meng et al. (2024) remove the reference model and incorporate a length normalization to address verbosity issues that can skew preference data. Likewise, **R-DPO** (Park et al., 2024) and **LD-DPO** (Liu et al., 2024a) specifically tackle length bias by injecting additional regularizers or by explicitly separating length-based preferences from other factors. For instance, LD-DPO modifies the training set to handle length constraints, preventing performance drops on standard benchmarks while mitigating length exploitation in preference tasks.

Refining Training Regimens for Preference Data. A final family of works emphasizes how training procedures or data usage can be systematically improved. For instance, Kim et al. (2024) propose **sDPO**, a step-wise learning method partitioning preference data to stabilize training. **IRPO** (Pang et al., 2024) enhances chain-of-thought reasoning by incorporating a negative log-likelihood term for the chosen solution path, thus nudging LLMs toward robust multi-step reasoning. **OFS-DPO** (Qi et al., 2024) trains two LoRA modules at different paces—one faster, one slower—to sustain gradient momentum and to adapt more efficiently. Lastly, Yuan et al. (2024) tackle verbosity with **LIFT-DPO**, an approach that augments preference data with length-control instructions, ensuring that the model does not exploit response length to inflate its preference scores.

Multi-Preference Optimization: Traditional preference optimization methods, like DPO, consider pairwise comparisons. However, datasets such as UltraFeedback (Cui et al., 2023) highlight the necessity of multi-preference optimization. Multi-preference methods, such as InfoNCA (Chen et al., 2024a), leverage all available positive and negative responses simultaneously, reducing alignment bias and better approximating the true preference distribution. These methods mitigate limitations inherent to pairwise approaches by incorporating the diversity of acceptable and suboptimal responses. Earlier works in search have also used multiple user preferences to optimize models in various applications such as search (Joachims, 2002).

Reward Modeling in Preferences: Reward modeling is essential for translating qualitative human feedback into quantitative metrics that guide AI behavior optimization. Traditional methods, such as Reinforcement Learning from Human Feedback (RLHF), utilize reward models trained on human annotations to inform policy updates (Christiano et al., 2017; Stiennon et al., 2020). Early approaches like inverse reinforcement learning (Ng et al., 2000) and apprenticeship learning (Abbeel & Ng, 2004) demonstrated the feasibility of inferring reward functions from observed behaviors. Recent advancements have diversified reward modeling techniques. For instance, the Adversarial Preference Optimization (APO) framework employs adversarial training to adapt reward models to the evolving generation distribution of language models (Cheng et al., 2023).

Noise Contrastive Estimation and InfoNCA: Contrastive learning, particularly methods like InfoNCE (Oord et al., 2018), maximizes mutual information between positive samples while discriminating against negatives. In the language domain, Klein & Nabi (2024) leverage a perplexity-based contrastive objective to reduce toxic language generation while preserving the model's overall utility. InfoNCA adapts these principles for preference optimization, aligning responses with scalar rewards through noise-contrastive estimation (Chen et al., 2024a). Despite its strengths, InfoNCA can overem-

phasize less informative negative samples, which motivates methods like SWEPO that dynamically weigh responses based on deviation from the mean reward.

UltraFeedback Dataset: The UltraFeedback dataset (Cui et al., 2023) is a significant advancement in preference-based training resources. It comprises GPT-4 annotated feedback for over 64,000 instructions, including scalar reward evaluations. UltraFeedback has been pivotal in developing models like UltraLM-13B-PPO and UltraRM, which achieve state-of-the-art performance across benchmarks such as AlpacaEval. This dataset's granularity enables advanced preference optimization methods like SWEPO to leverage diverse response quality levels effectively.

B BIAS ANALYSIS:

In the first part of this section section, we analyze how the number of positive and negative examples per query, k, affects the bias with respect to an attribute $a(y)$. We provide a formal theorem establishing the relationship between bias and k, followed by a corollary discussing the behavior as $k \to \infty$.

The reason for this analysis is to show that multi-preference sampling of accepted and rejected answers from a distribution is better than using a single sample as DPO does. The more accepted and rejected samples you have, the lower the bias, provably.

B.1 ASSUMPTIONS

We make the following assumptions:

1. **Attribute Function**: Let $a(y) : \mathcal{Y}_x \to \mathbb{R}$ be an attribute function mapping responses to real numbers (e.g., response length).
2. **Finite Variance**: The attribute $a(y)$ has finite variance over the acceptable response distribution $\mathcal{A}_x$ for each query x, i.e., $\mathrm{Var}_{y \sim \mathcal{A}_x}[a(y)] = \sigma^2_{\mathcal{A}_x} < \infty$.
3. **Independent Sampling**: Responses are independently sampled from their respective distributions.
4. **Model Capacity**: The model can represent the true distribution given sufficient data.
5. **Uniform Bounded Variance**: There exists a constant $\sigma^2_{\max}$ such that $\sigma^2_{\mathcal{A}_x} \leq \sigma^2_{\max}$ for all $x \in \mathcal{X}$.

B.2 BIAS DEFINITION

The bias with respect to attribute a is defined as:

$$B^{(k)} = \left| \mu_\theta^{(k)} - \mu_{\mathcal{A}} \right|, \tag{23}$$

where:

- $\mu_\theta^{(k)}$ is the expected attribute value under the model after training with k positive and k negative samples per query.
- $\mu_{\mathcal{A}} = \mathbb{E}_{x \sim \mathcal{X}} \left[\mu_{\mathcal{A}_x} \right]$, with $\mu_{\mathcal{A}_x} = \mathbb{E}_{y \sim \mathcal{A}_x} \left[a(y) \right]$.

B.3 MAIN BIAS RESULT

Theorem 2. Under the stated assumptions, the expected bias $\mathbb{E}[B^{(k)}]$ decreases with the number of samples k as:

$$\mathbb{E}[B^{(k)}] \leq \frac{C}{\sqrt{k}}, \tag{24}$$

where $C = \sigma_{\max}$ is a constant depending on the maximum variance of $a(y)$ over the acceptable responses.

Proof. For each query x, consider the sample mean of the attribute over the k positive responses:

$$\overline{a}_x^{(k)} = \frac{1}{k}\sum_{i=1}^{k} a(y_{x,i}^+), \quad y_{x,i}^+ \sim \mathcal{A}_x. \tag{25}$$

Since the $y_{x,i}^+$ are independent and identically distributed samples from $\mathcal{A}_x$, the expected value and variance of $\overline{a}_x^{(k)}$ are:

$$\mathbb{E}\left[\overline{a}_x^{(k)}\right] = \mu_{\mathcal{A}_x}, \tag{26}$$

$$\mathrm{Var}\left(\overline{a}_x^{(k)}\right) = \frac{\sigma_{\mathcal{A}_x}^2}{k} \leq \frac{\sigma_{\max}^2}{k}. \tag{27}$$

Using the fact that for any random variable Z with finite variance, the expected absolute deviation from its mean satisfies:

$$\mathbb{E}\left[|Z - \mathbb{E}[Z]|\right] \leq \sqrt{\mathrm{Var}[Z]}, \tag{28}$$

we have:

$$\mathbb{E}\left[\left|\overline{a}_x^{(k)} - \mu_{\mathcal{A}_x}\right|\right] \leq \sqrt{\frac{\sigma_{\mathcal{A}_x}^2}{k}} \leq \frac{\sigma_{\max}}{\sqrt{k}}. \tag{29}$$

Averaging over all queries $x \in \mathcal{X}$:

$$\mathbb{E}_x\left[\left|\overline{a}_x^{(k)} - \mu_{\mathcal{A}_x}\right|\right] \leq \frac{\sigma_{\max}}{\sqrt{k}}. \tag{30}$$

Since $\mu_\theta^{(k)} = \mathbb{E}_x\left[\overline{a}_x^{(k)}\right]$ and $\mu_{\mathcal{A}} = \mathbb{E}_x\left[\mu_{\mathcal{A}_x}\right]$, the expected bias is:

$$\mathbb{E}[B^{(k)}] = \left|\mu_\theta^{(k)} - \mu_{\mathcal{A}}\right| = \left|\mathbb{E}_x\left[\overline{a}_x^{(k)} - \mu_{\mathcal{A}_x}\right]\right| \tag{31}$$

$$\leq \mathbb{E}_x\left[\left|\overline{a}_x^{(k)} - \mu_{\mathcal{A}_x}\right|\right] \leq \frac{\sigma_{\max}}{\sqrt{k}}. \tag{32}$$

Thus, the expected bias decreases with k as $\frac{1}{\sqrt{k}}$.

□

Corollary 2. As $k \to \infty$, the expected bias $\mathbb{E}[B^{(k)}]$ approaches zero:

$$\lim_{k\to\infty} \mathbb{E}[B^{(k)}] = 0. \tag{33}$$

Implications This theorem establishes a quantitative relationship between the bias $B^{(k)}$ and the number of samples k. It shows that incorporating multiple positive and negative responses per query reduces the bias with respect to attribute $a(y)$ at a rate proportional to $1/\sqrt{k}$. As k increases, the model's expected attribute value converges to the true expected attribute value over acceptable responses, leading to better alignment with human preferences.

C Differentiating the Group Contrastive Loss from InfoNCE Loss

In this subsection, we compare our proposed weighted contrastive loss function with the InfoNCA loss function. We present both loss functions, derive their gradients rigorously, and characterize their stationary points. Based on this characterization, we discuss the properties of the convergence points in terms of what the models learn and their alignment with human preferences.

C.1 Definitions of Loss Functions

InfoNCA Loss Function The InfoNCA loss function is defined as:

$$L_{\text{InfoNCA}} = -\sum_{i=1}^{K} p_i^{\text{target}} \log p_i^{\text{model}},$$

where p_i^{target} represents the target probability for the i-th response, calculated as

$$p_i^{\text{target}} = \frac{e^{r(x,y_i)/\alpha}}{\sum_{j=1}^{K} e^{r(x,y_j)/\alpha}},$$

and p_i^{model} denotes the model's predicted probability for the i-th response, given by

$$p_i^{\text{model}} = \frac{e^{s_\theta(y_i|x)}}{\sum_{j=1}^{K} e^{s_\theta(y_j|x)}}.$$

In this context, x is the instruction or prompt provided to the model, and $\{y_i\}_{i=1}^{K}$ represents a set of K responses generated for the instruction x. The term $r(x, y_i)$ is the reward associated with the response y_i, while $s_\theta(y_i \mid x) = \log\left(P_\theta(y_i \mid x)/P_{\text{ref}}(y_i \mid x)\right)$ is the score for response y_i. The parameter α serves as a temperature parameter that controls the influence of the reward, and K is the total number of responses considered for the instruction x.

Weighted Contrastive Loss Function Our proposed weighted contrastive loss function is expressed as:

$$L_{\text{weighted}} = -\log\left(\frac{\sum\limits_{i\in Y^+} w_i e^{s_\theta(y_i|x)}}{\sum\limits_{j=1}^{K} w_j e^{s_\theta(y_j|x)}}\right),$$

where Y^+ is the set of positive responses with rewards above the mean, defined as $Y^+ = \{y_i \mid S_i > S_{\text{mean}}\}$. Each response y_i is assigned a weight $w_i = e^{\alpha\delta_i}$, where δ_i is the deviation of the reward score S_i from the mean reward score S_{mean}. Specifically, $\delta_i = S_i - S_{\text{mean}}$ for responses in Y^+ and $\delta_i = S_{\text{mean}} - S_i$ for responses not in Y^+. The mean reward score S_{mean} is calculated as

$$S_{\text{mean}} = \frac{1}{K}\sum_{j=1}^{K} S_j,$$

where K is the total number of responses for the query x. The term $s_\theta(y_i \mid x)$ denotes the model's logit for response y_i, and α is a scaling hyperparameter that controls the influence of the deviation δ_i.

C.2 Gradient Analysis

To understand how each loss function influences the model during training, we derive the gradients with respect to the model logits $s_\theta(y_i \mid x)$ for both methods.

Gradient of InfoNCA Loss

Lemma 1. The gradient of the InfoNCA loss with respect to the model logits $s_\theta(y_i \mid x)$ is:

$$\frac{\partial L_{\text{InfoNCA}}}{\partial s_\theta(y_i \mid x)} = p_i^{\text{model}} - p_i^{\text{target}}. \tag{34}$$

Proof. The InfoNCA loss is:

$$L_{\text{InfoNCA}} = -\sum_{k=1}^{K} p_k^{\text{target}} \log p_k^{\text{model}}. \tag{35}$$

Our goal is to compute $\frac{\partial L_{\text{InfoNCA}}}{\partial s_\theta(y_i|x)}$.

Since p_k^{target} does not depend on $s_\theta(y_i \mid x)$ (the rewards are constants with respect to the model parameters), the derivative only affects the terms involving p_k^{model}.

First, express $\log p_k^{\text{model}}$ explicitly:

$$\log p_k^{\text{model}} = s_\theta(y_k \mid x) - \log \left(\sum_{j=1}^{K} e^{s_\theta(y_j|x)} \right). \tag{36}$$

Now, compute the derivative of $\log p_k^{\text{model}}$ with respect to $s_\theta(y_i \mid x)$:

$$\frac{\partial \log p_k^{\text{model}}}{\partial s_\theta(y_i \mid x)} = \frac{\partial s_\theta(y_k \mid x)}{\partial s_\theta(y_i \mid x)} - \frac{\partial}{\partial s_\theta(y_i \mid x)} \log \left(\sum_{j=1}^{K} e^{s_\theta(y_j|x)} \right). \tag{37}$$

Compute each term separately.

First term:

$$\frac{\partial s_\theta(y_k \mid x)}{\partial s_\theta(y_i \mid x)} = \delta_{ik}, \tag{38}$$

where δ_{ik} is the Kronecker delta, equal to 1 if $i = k$ and 0 otherwise.

Second term:

Let $Z = \sum_{j=1}^{K} e^{s_\theta(y_j|x)}$. Then,

$$\frac{\partial}{\partial s_\theta(y_i \mid x)} \log Z = \frac{1}{Z} \frac{\partial Z}{\partial s_\theta(y_i \mid x)}. \tag{39}$$

Compute $\frac{\partial Z}{\partial s_\theta(y_i|x)}$:

$$\frac{\partial Z}{\partial s_\theta(y_i \mid x)} = e^{s_\theta(y_i|x)}. \tag{40}$$

Therefore,

$$\frac{\partial}{\partial s_\theta(y_i \mid x)} \log Z = \frac{e^{s_\theta(y_i|x)}}{Z} = p_i^{\text{model}}. \tag{41}$$

Putting it all together:

$$\frac{\partial \log p_k^{\text{model}}}{\partial s_\theta(y_i \mid x)} = \delta_{ik} - p_i^{\text{model}}. \tag{42}$$

Now, compute the gradient of the loss:

$$\frac{\partial L_{\text{InfoNCA}}}{\partial s_\theta(y_i \mid x)} = -\sum_{k=1}^{K} p_k^{\text{target}} \frac{\partial \log p_k^{\text{model}}}{\partial s_\theta(y_i \mid x)} \tag{43}$$

$$= -\sum_{k=1}^{K} p_k^{\text{target}} \left(\delta_{ik} - p_i^{\text{model}}\right) \tag{44}$$

$$= -\left(p_i^{\text{target}} - p_i^{\text{model}} \sum_{k=1}^{K} p_k^{\text{target}}\right). \tag{45}$$

Since $\sum_{k=1}^{K} p_k^{\text{target}} = 1$, we have:

$$\sum_{k=1}^{K} p_k^{\text{target}} = 1 \implies \sum_{k=1}^{K} p_k^{\text{target}} = 1. \tag{46}$$

Therefore,

$$\frac{\partial L_{\text{InfoNCA}}}{\partial s_\theta(y_i \mid x)} = -\left(p_i^{\text{target}} - p_i^{\text{model}} \cdot 1\right) = p_i^{\text{model}} - p_i^{\text{target}}. \tag{47}$$

□

Gradient of Weighted Contrastive Loss

Lemma 2. The gradient of the weighted contrastive loss with respect to the model logits $s_\theta(y_i \mid x)$ is:

$$\frac{\partial L_{\text{weighted}}}{\partial s_\theta(y_i \mid x)} = p_i^{\text{weighted}} - p_i^{\text{pos}} \tag{48}$$

where:

$$p_i^{\text{weighted}} = \frac{w_i e^{s_\theta(y_i|x)}}{\sum_{j=1}^{K} w_j e^{s_\theta(y_j|x)}}, \qquad p_i^{\text{pos}} = \frac{w_i e^{s_\theta(y_i|x)}}{\sum_{k \in Y^+} w_k e^{s_\theta(y_k|x)}} \cdot \mathbb{I}_{y_i \in Y^+}, \tag{49}$$

and $\mathbb{I}_{y_i \in Y^+}$ is the indicator function, equal to 1 if $y_i \in Y^+$ and 0 otherwise.

Proof. Let us denote:

$$A = \sum_{k \in Y^+} w_k e^{s_\theta(y_k|x)}, \quad Z = \sum_{j=1}^{K} w_j e^{s_\theta(y_j|x)}. \tag{50}$$

The weighted contrastive loss is:

$$L_{\text{weighted}} = -\log\left(\frac{A}{Z}\right) = -\log A + \log Z. \tag{51}$$

Compute the derivative with respect to $s_\theta(y_i \mid x)$:

$$\frac{\partial L_{\text{weighted}}}{\partial s_\theta(y_i \mid x)} = -\frac{1}{A}\frac{\partial A}{\partial s_\theta(y_i \mid x)} + \frac{1}{Z}\frac{\partial Z}{\partial s_\theta(y_i \mid x)}. \tag{52}$$

Compute $\frac{\partial A}{\partial s_\theta(y_i|x)}$:

$$\frac{\partial A}{\partial s_\theta(y_i \mid x)} = w_i e^{s_\theta(y_i|x)} \cdot \mathbb{I}_{y_i \in Y^+}. \tag{53}$$

Compute $\frac{\partial Z}{\partial s_\theta(y_i|x)}$:

$$\frac{\partial Z}{\partial s_\theta(y_i \mid x)} = w_i e^{s_\theta(y_i|x)}. \tag{54}$$

Substitute back into the gradient:

$$\frac{\partial L_{\text{weighted}}}{\partial s_\theta(y_i \mid x)} = -\frac{1}{A} w_i e^{s_\theta(y_i|x)} \cdot \mathbb{I}_{y_i \in Y^+} + \frac{1}{Z} w_i e^{s_\theta(y_i|x)} \tag{55}$$

$$= w_i e^{s_\theta(y_i|x)} \left(\frac{1}{Z} - \frac{\mathbb{I}_{y_i \in Y^+}}{A} \right). \tag{56}$$

Recognize that:

$$p_i^{\text{weighted}} = \frac{w_i e^{s_\theta(y_i|x)}}{Z}, \quad p_i^{\text{pos}} = \frac{w_i e^{s_\theta(y_i|x)}}{A} \cdot \mathbb{I}_{y_i \in Y^+}. \tag{57}$$

Therefore:

$$\frac{\partial L_{\text{weighted}}}{\partial s_\theta(y_i \mid x)} = p_i^{\text{weighted}} - p_i^{\text{pos}}. \tag{58}$$

Since $p_i^{\text{pos}} = 0$ when $y_i \notin Y^+$, we have:

$$\frac{\partial L_{\text{weighted}}}{\partial s_\theta(y_i \mid x)} = \begin{cases} p_i^{\text{weighted}} - p_i^{\text{pos}}, & \text{if } y_i \in Y^+, \\ p_i^{\text{weighted}} - 0 = p_i^{\text{weighted}}, & \text{if } y_i \in Y^-. \end{cases} \tag{59}$$

However, this suggests that the gradient is always positive for negative examples. In other words, given w_i and Z are positive, $e^{s_\theta(y_i|x)}$ keeps increasing. But note that $s_\theta(y_i \mid x) = -\log\left(P_\theta(y_i \mid x)\right)$. Hence $\frac{1}{P_\theta(y_i|x)}$ keeps increasing implying that $P_\theta(y_i \mid x)$ keeps decreasing. i.e. at the stationary point, $P_\theta(y_i \mid x) \to 0$ for all negative examples, $y_i \in Y^-$.

Now let us examine the positive examples. The gradient simplifies to $w_i e^{s_\theta(y_i|x)} \left(\frac{1}{Z} - \frac{1}{A}\right)$. Since $Z \geq A$, $\frac{1}{Z} \leq \frac{1}{A}$. Hence the gradient term with respect to $s_\theta(y_i \mid x)$ is negative. Notice that $e^{s_\theta(y_i|x)} = \frac{1}{P_\theta(y_i|x)}$. A negative gradient implies that $\frac{1}{P_\theta(y_i|x)}$ decreases, implying that $P_\theta(y_i \mid x)$ increases for all positive examples $y_i \in Y^+$.

□

We now provide the gradients directly in terms of $P_\theta(y_j \mid x)$ instead of the scores $s_\theta(y_j \mid x)$, for easy interpretibility in terms of the probabilities.

Lemma 3. Let the weighted contrastive loss be defined as:

$$L_{\text{weighted}} = -\log\left(\frac{V}{U}\right) = -\log V + \log U,$$

where

$$U = \sum_{j=1}^{K} u_j P_\theta(y_j \mid x), \quad V = \sum_{i \in Y^+} u_i P_\theta(y_i \mid x),$$

and

$$u_i = \frac{w_i}{P_{\text{ref}}(y_i \mid x)},$$

with $w_i = e^{\alpha \delta_i}$, $P_\theta(y_i \mid x)$ being the model probability for response y_i, and $P_{\text{ref}}(y_i \mid x)$ being the reference model probability.

Then, the gradient of the weighted contrastive loss with respect to $P_\theta(y_i \mid x)$ is given by:

- For positive examples ($y_i \in Y^+$):

$$\frac{\partial L_{\text{weighted}}}{\partial P_\theta(y_i \mid x)} = u_i \left(\frac{1}{U} - \frac{1}{V}\right), \tag{60}$$

- For negative examples ($y_i \notin Y^+$):

$$\frac{\partial L_{\text{weighted}}}{\partial P_\theta(y_i \mid x)} = \frac{u_i}{U}. \tag{61}$$

Proof. Using the score function $s_\theta(y_i \mid x) = \log\left(\frac{P_\theta(y_i|x)}{P_{\text{ref}}(y_i|x)}\right)$, we have $e^{s_\theta(y_i|x)} = \frac{P_\theta(y_i \mid x)}{P_{\text{ref}}(y_i \mid x)}$.

The weighted contrastive loss becomes:

$$L_{\text{weighted}} = -\log\left(\frac{\sum_{i \in Y^+} w_i e^{s_\theta(y_i|x)}}{\sum_{j=1}^{K} w_j e^{s_\theta(y_j|x)}}\right) = -\log\left(\frac{\sum_{i \in Y^+} w_i \frac{P_\theta(y_i \mid x)}{P_{\text{ref}}(y_i \mid x)}}{\sum_{j=1}^{K} w_j \frac{P_\theta(y_j \mid x)}{P_{\text{ref}}(y_j \mid x)}}\right) = -\log\left(\frac{V}{U}\right),$$

where $u_i = \frac{w_i}{P_{\text{ref}}(y_i \mid x)}$, $V = \sum_{i \in Y^+} u_i P_\theta(y_i \mid x)$, and $U = \sum_{j=1}^{K} u_j P_\theta(y_j \mid x)$.

We compute the gradient of L_{weighted} with respect to $P_\theta(y_i \mid x)$:

$$\frac{\partial L_{\text{weighted}}}{\partial P_\theta(y_i \mid x)} = -\frac{1}{V} \cdot \frac{\partial V}{\partial P_\theta(y_i \mid x)} + \frac{1}{U} \cdot \frac{\partial U}{\partial P_\theta(y_i \mid x)}.$$

Case 1: For $y_i \in Y^+$:

$$\frac{\partial V}{\partial P_\theta(y_i \mid x)} = u_i, \quad \frac{\partial U}{\partial P_\theta(y_i \mid x)} = u_i.$$

Thus,

$$\frac{\partial L_{\text{weighted}}}{\partial P_\theta(y_i \mid x)} = -\frac{u_i}{V} + \frac{u_i}{U} = u_i \left(\frac{1}{U} - \frac{1}{V}\right).$$

Case 2: For $y_i \notin Y^+$:

$$\frac{\partial V}{\partial P_\theta(y_i \mid x)} = 0, \quad \frac{\partial U}{\partial P_\theta(y_i \mid x)} = u_i.$$

Thus,

$$\frac{\partial L_{\text{weighted}}}{\partial P_\theta(y_i \mid x)} = 0 + \frac{u_i}{U} = \frac{u_i}{U}.$$

□

Corollary 3. The sign of the gradient indicates the optimization direction:

- For positive examples ($y_i \in Y^+$), since $V \leq U$, we have $\frac{1}{U} - \frac{1}{V} \leq 0$. Therefore, the gradient $\frac{\partial L_{\text{weighted}}}{\partial P_\theta(y_i \mid x)} \leq 0$, and minimizing L_{weighted} involves **increasing** $P_\theta(y_i \mid x)$.
- For negative examples ($y_i \notin Y^+$), the gradient $\frac{\partial L_{\text{weighted}}}{\partial P_\theta(y_i \mid x)} > 0$, and minimizing L_{weighted} involves **decreasing** $P_\theta(y_i \mid x)$.

Proof. As established in the lemma:

For positive examples ($y_i \in Y^+$): Since $V = \sum_{i \in Y^+} u_i P_\theta(y_i \mid x)$ and $U = V + \sum_{j \notin Y^+} u_j P_\theta(y_j \mid x)$, it follows that $V \leq U$ and thus $\frac{1}{U} - \frac{1}{V} \leq 0$.

Therefore, the gradient:

$$\frac{\partial L_{\text{weighted}}}{\partial P_\theta(y_i \mid x)} = u_i \left(\frac{1}{U} - \frac{1}{V} \right) \leq 0.$$

A negative gradient indicates that increasing $P_\theta(y_i \mid x)$ will decrease L_{weighted}. Hence, to minimize the loss, we should **increase** $P_\theta(y_i \mid x)$ for positive examples.

For negative examples ($y_i \notin Y^+$): The gradient is:

$$\frac{\partial L_{\text{weighted}}}{\partial P_\theta(y_i \mid x)} = \frac{u_i}{U} > 0,$$

since $u_i > 0$ and $U > 0$. A positive gradient indicates that decreasing $P_\theta(y_i \mid x)$ will decrease L_{weighted}. Therefore, to minimize the loss, we should **decrease** $P_\theta(y_i \mid x)$ for negative examples. □

D Characterization of Stationary Points

We now characterize the stationary points of both loss functions.

D.1 Stationary Points of the InfoNCA Loss Function

Theorem 3. For the InfoNCA loss, the stationary points occur when:

$$p_i^{\text{model}} = p_i^{\text{target}}, \quad \forall i \in \{1, \dots, K\}. \tag{62}$$

Proof. Stationary points are defined by the condition:

$$\frac{\partial L_{\text{InfoNCA}}}{\partial s_\theta(y_i \mid x)} = 0, \quad \forall i. \tag{63}$$

From the gradient:

$$\frac{\partial L_{\text{InfoNCA}}}{\partial s_\theta(y_i \mid x)} = p_i^{\text{model}} - p_i^{\text{target}}, \tag{64}$$

setting the gradient to zero yields:

$$p_i^{\text{model}} = p_i^{\text{target}}, \quad \forall i. \tag{65}$$

□

Remark 1. This stationary point is suboptimal because p_i^{model} expands to:

$$p_i^{\text{model}} = \frac{e^{\log P_\theta(y_i|x) - \log P_{\text{ref}}(y_i|x)}}{\sum_{j=1}^{K} e^{\log P_\theta(y_j|x) - \log P_{\text{ref}}(y_j|x)}}$$

Rather than equating the soft-max of the difference between $\log P_\theta(y_i|x)$ and $\log P_{\text{ref}}(y_i|x)$ to p_i^{target}, optimality may require directly setting $\log P_\theta(y|x)$ to match the softmax of the target scores.

D.2 Stationary Points of the Weighted Contrastive Loss under Simplifying Assumptions

Lemma 4. Consider the weighted contrastive loss function in a simplified scenario with the following conditions: There are N^+ positive examples, each with weight w^+, and N^- negative examples, each with weight w^-. All positive examples have the same score $s^{(t)}$ at iteration t, and all negative examples have the same score $s^{(t)}$ at iteration t. Then, the update rule for the score $s^{(t)}$ of the positive examples at iteration $t+1$ is given by

$$s^{(t+1)} = s^{(t)} + \eta \left(\frac{N^- w^-}{N^+(N^+ w^+ + N^- w^-)} \right), \tag{66}$$

where η is the learning rate.

Proof. Let Y^+ denote the set of positive examples and Y^- the set of negative examples, with N^+ and N^- examples respectively for a total of $K = N^+ + N^-$ examples. With weights w^+ and w^- assigned to positive and negative examples respectively, and logits $s^{(t)}$ for both classes at timestep t, the weighted contrastive loss function is defined as:

$$L_{\text{weighted}}(\theta) = -\log \left(\frac{\sum\limits_{i \in Y^+} w_i e^{s_i}}{\sum\limits_{j=1}^{K} w_j e^{s_j}} \right), \tag{67}$$

where $w_i = w^+$ and $s_i = s^{(t)}$ for $i \in Y^+$, and $w_j = w^-$ and $s_j = s^{(t)}$ for $j \in Y^-$.

Compute the numerator A and the denominator Z of the loss function:

$$A = \sum_{i \in Y^+} w_i e^{s_i} = N^+ w^+ e^{s^{(t)}}, \tag{68}$$

$$Z = \sum_{j=1}^{K} w_j e^{s_j} = N^+ w^+ e^{s^{(t)}} + N^- w^- e^{s^{(t)}} = e^{s^{(t)}}(N^+ w^+ + N^- w^-). \tag{69}$$

For positive examples $i \in Y^+$, the weighted probability p_i^{weighted} and the positive probability p_i^{pos} are:

$$p_i^{\text{weighted}} = \frac{w^+ e^{s^{(t)}}}{Z} = \frac{w^+}{N^+ w^+ + N^- w^-}, \tag{70}$$

$$p_i^{\text{pos}} = \frac{w^+ e^{s^{(t)}}}{A} = \frac{w^+}{N^+ w^+} = \frac{1}{N^+}. \tag{71}$$

The gradient of the loss with respect to $s^{(t)}$ for positive examples is:

$$\frac{\partial L_{\text{weighted}}}{\partial s^{(t)}} = p_i^{\text{weighted}} - p_i^{\text{pos}} = \frac{w^+}{N^+ w^+ + N^- w^-} - \frac{1}{N^+}. \tag{72}$$

To simplify this expression, we find a common denominator $D = N^+(N^+w^+ + N^-w^-)$:

$$\frac{\partial L_{\text{weighted}}}{\partial s^{(t)}} = \frac{w^+N^+ - (N^+w^+ + N^-w^-)}{D} \tag{73}$$

$$= \frac{w^+N^+ - N^+w^+ - N^-w^-}{N^+(N^+w^+ + N^-w^-)} \tag{74}$$

$$= \frac{-N^-w^-}{N^+(N^+w^+ + N^-w^-)}. \tag{75}$$

The update rule for $s^{(t)}$ is then:

$$s^{(t+1)} = s^{(t)} - \eta \frac{\partial L_{\text{weighted}}}{\partial s^{(t)}} = s^{(t)} + \eta \left(\frac{N^-w^-}{N^+(N^+w^+ + N^-w^-)} \right). \tag{76}$$

This completes the proof. □

Corollary 4. Assuming the initial scores are zero ($s^{(0)} = 0$), the score $s^{(t)}$ of the positive examples at iteration t is given by

$$s^{(t)} = t\eta \left(\frac{N^-w^-}{N^+(N^+w^+ + N^-w^-)} \right). \tag{77}$$

Proof. From the update rule established in the lemma,

$$s^{(t+1)} = s^{(t)} + c, \tag{78}$$

where

$$c = \eta \left(\frac{N^-w^-}{N^+(N^+w^+ + N^-w^-)} \right). \tag{79}$$

Since $s^{(0)} = 0$, we have

$$s^{(1)} = s^{(0)} + c = c, \tag{80}$$

$$s^{(2)} = s^{(1)} + c = 2c, \tag{81}$$

$$\vdots \tag{82}$$

$$s^{(t)} = tc. \tag{83}$$

Substituting c back into the expression, we obtain

$$s^{(t)} = t\eta \left(\frac{N^-w^-}{N^+(N^+w^+ + N^-w^-)} \right). \tag{84}$$

□

Corollary 5. In the special case where there is one positive example ($N^+ = 1$) and one negative example ($N^- = 1$), and the weights are $w^+ = w^- = 1$ (as in Direct Preference Optimization), the score $s^{(t)}$ at iteration t is:

$$s^{(t)} = \frac{\eta t}{2}. \tag{85}$$

Proof. Substituting $N^+ = N^- = 1$ and $w^+ = w^- = 1$ into the expression for $s^{(t)}$:

$$s^{(t)} = t\eta \left(\frac{1 \times 1}{1 \times (1 \times 1 + 1 \times 1)} \right) \tag{86}$$

$$= t\eta \left(\frac{1}{1 \times (1 + 1)} \right) \tag{87}$$

$$= t\eta \left(\frac{1}{2} \right) \tag{88}$$

$$= \frac{\eta t}{2}. \tag{89}$$

□

Lemma 5. Consider the general case where positive examples may have different weights w_i^+, and each positive example i has its own score $s_i^{(t)}$ at iteration t. Assuming initial scores $s_i^{(0)} = 0$ for all positive examples, the score $s_i^{(t)}$ of positive example i at iteration t, up to a linear approximation, is given by

$$s_i^{(t)} = t\eta w_i^+ \left(\frac{B_0}{A_0 Z_0} \right), \tag{90}$$

where $A_0 = \sum_{k \in Y^+} w_k^+$, $B_0 = \sum_{j \in Y^-} w_j^-$, $Z_0 = A_0 + B_0$, and η is the learning rate.

Proof. At iteration $t = 0$, the initial scores are $s_i^{(0)} = 0$ for all $i \in Y^+$. The sums are:

$$A_0 = \sum_{k \in Y^+} w_k^+ e^{s_k^{(0)}} = \sum_{k \in Y^+} w_k^+ = W^+, \tag{91}$$

$$B_0 = \sum_{j \in Y^-} w_j^- e^{s_j^{(0)}} = \sum_{j \in Y^-} w_j^- = W^-. \tag{92}$$

The total sum is $Z_0 = A_0 + B_0 = W^+ + W^-$.

The gradient for each positive example i at $t = 0$ is:

$$\frac{\partial L_{\text{weighted}}}{\partial s_i^{(0)}} = -w_i^+ e^{s_i^{(0)}} \left(\frac{B_0}{A_0 Z_0} \right) = -w_i^+ \left(\frac{B_0}{A_0 Z_0} \right). \tag{93}$$

The update rule is:

$$s_i^{(1)} = s_i^{(0)} - \eta \frac{\partial L_{\text{weighted}}}{\partial s_i^{(0)}} = \eta w_i^+ \left(\frac{B_0}{A_0 Z_0} \right). \tag{94}$$

Assuming that the term $\frac{B_0}{A_0 Z_0}$ remains approximately constant over iterations (which holds when η is small and changes in $s_i^{(t)}$ are small), the score at iteration t is:

$$s_i^{(t)} = t\eta w_i^+ \left(\frac{B_0}{A_0 Z_0} \right). \tag{95}$$

□

Remark 2. The approximation assumes that A_t, B_t, and Z_t remain close to their initial values A_0, B_0, and Z_0 over the iterations considered, and the score values remain small. This is reasonable for small learning rates η and a limited number of iterations t.

D.3 Stationary Points of the Weighted Contrastive Loss

We now analyze the stationary points of our weighted contrastive loss function.

Lemma 6. For the weighted contrastive loss function, the stationary point occurs when the probabilities of the negative samples approach zero, i.e.,

$$P_\theta(y_i \mid x) \to 0 \quad \text{for all } y_i \in Y^-. \tag{96}$$

Proof. From Lemma **??**, the gradient of the weighted contrastive loss with respect to the model logits $s_\theta(y_i \mid x)$ is:

$$\frac{\partial L_{\text{weighted}}}{\partial s_\theta(y_i \mid x)} = \begin{cases} p_i^{\text{weighted}} - p_i^{\text{pos}}, & \text{if } y_i \in Y^+, \\ p_i^{\text{weighted}}, & \text{if } y_i \in Y^-. \end{cases} \tag{97}$$

At a stationary point, the gradient must be zero for all y_i. Consider the negative samples $y_i \in Y^-$. Setting the gradient to zero yields:

$$\frac{\partial L_{\text{weighted}}}{\partial s_\theta(y_i \mid x)} = p_i^{\text{weighted}} = 0. \tag{98}$$

Since p_i^{weighted} is the normalized weighted probability of y_i, given by:

$$p_i^{\text{weighted}} = \frac{w_i e^{s_\theta(y_i|x)}}{\sum_{j=1}^{K} w_j e^{s_\theta(y_j|x)}}, \tag{99}$$

and $w_i > 0$, the only way for p_i^{weighted} to be zero is if $e^{s_\theta(y_i|x)} = 0$, which implies:

$$s_\theta(y_i \mid x) \to -\infty \quad \Longrightarrow \quad P_\theta(y_i \mid x) \to 0 \quad \text{for } y_i \in Y^-. \tag{100}$$

Similarly, for positive samples $y_i \in Y^+$, the gradient is:

$$\frac{\partial L_{\text{weighted}}}{\partial s_\theta(y_i \mid x)} = p_i^{\text{weighted}} - p_i^{\text{pos}} = 0. \tag{101}$$

This implies:

$$p_i^{\text{weighted}} = p_i^{\text{pos}}. \tag{102}$$

Since the probabilities of the negative samples approach zero, the denominator in p_i^{weighted} becomes:

$$\sum_{j=1}^{K} w_j e^{s_\theta(y_j|x)} \approx \sum_{k \in Y^+} w_k e^{s_\theta(y_k|x)}. \tag{103}$$

Therefore, $p_i^{\text{weighted}} \approx p_i^{\text{pos}}$, satisfying the condition for the gradient to be zero for positive samples.

Thus, at the stationary point, the probabilities of the negative samples approach zero. □

Remark 3. When the probabilities of the negative samples approach zero, the scores $s_\theta(y_i \mid x)$ for $y_i \in Y^-$ tend to $-\infty$. Since:

$$e^{s_\theta(y_i|x)} = \frac{P_\theta(y_i \mid x)}{P_{\text{ref}}(y_i \mid x)} \to 0, \tag{104}$$

the weighted contributions of the negative samples to the numerator and denominator of L_{weighted} become negligible.

Consequently, the numerator and denominator of L_{weighted} become equal:

$$\sum_{i \in Y^+} w_i e^{s_\theta(y_i|x)} \approx \sum_{j=1}^{K} w_j e^{s_\theta(y_j|x)}. \tag{105}$$

Therefore:

$$L_{\text{weighted}} = -\log \left(\frac{\sum_{i \in Y^+} w_i e^{s_\theta(y_i|x)}}{\sum_{j=1}^{K} w_j e^{s_\theta(y_j|x)}} \right) \approx -\log 1 = 0. \tag{106}$$

This implies that the loss vanishes when the probabilities of the negative samples approach zero, indicating that the model has successfully minimized the loss by focusing entirely on the positive responses.

E Baselines Used for Comparison

When dealing with reward datasets where each instruction has more than two $K > 2$ responses, one common approach is to convert the data into pairwise preferences and then apply preference optimization techniques such as Direct Preference Optimization (DPO). Several strategies can be adopted for this purpose, each offering distinct trade-offs in terms of dataset richness and computational overhead.One straightforward method, as implemented by Zephyr Tunstall et al. (2023), involves selecting the response with the highest reward and pairing it with a randomly chosen response from the remaining responses for each instruction. Another variant involves pairing the highest-rewarded

response with the lowest-rewarded response for each instruction, ensuring a clear distinction between preferences.Additionally, alternative baselines can be explored to enhance performance by incorporating more suboptimal responses during training. By applying DPO to combinations of responses, we can significantly expand the preference dataset and potentially achieve improved optimization. Two notable baselines in this context are:

DPOx$\binom{K}{2}$: In this approach, all possible pairwise combinations of $\binom{K}{2}$ are generated, and DPO is applied to the entire combinatorial dataset. This method ensures the model is exposed to a comprehensive range of preference relationships, including those involving suboptimal responses.

DPOx$(K-1)$: Here, the response with the highest reward is paired individually with each of the remaining $(K-1)$ responses. This strategy emphasizes the contrast between the top response and all others, potentially reinforcing the model's understanding of optimal preferences.

Other baselines, such as InfoNCA and NCA, suggest that naively applying DPO to combinations of responses may lead to suboptimal performance. They leveraged Noise Contrastive Estimation (NCE) to bridge the gap in handling reward datasets explicitly annotated with scalar evaluations. According to their findings, the theoretical guarantees they provide ensure convergence, which is not guaranteed when applying DPO in this manner.

F Reward Loss Computation

In this section we provide the actual code used to compute the reward losses.

```python
import torch

def swepo_loss(pi_logps, ref_logps, rewards, beta, alpha, weight_type):
    """
    pi_logps: policy logprobs for K responses, shape (Batch_Size, K)
    ref_logps: reference logprobs for K responses, shape (Batch_Size, K)
    rewards: reward labels for K responses, shape (Batch_Size, K)
    beta: Temperature parameter for the SWEPO loss
    alpha: rating weight
    norm: weighting scheme for the reward score (0 or 1 or 2)
    """

    logits = pi_logps - ref_logps  # Compute logits
    rewards = rewards / alpha # Normalizing the reward value to logits
        scale

    mean_rewards = torch.mean(rewards, dim=-1)
    if self.norm > 0:
        weights = torch.abs(rewards - mean_rewards.reshape(-1, 1))
        weights = torch.pow(weights, norm) * beta
    else:
        deviation_reward = 0

    pos_mask = (rewards > mean_rewards.reshape(-1, 1)) * 1
    neg_mask = torch.logical_not(pos_mask) * 1

    eps = 1e-10
    logits = (logits + weight) * beta
    logits = logits - logits.max(dim=-1, keepdim=True)[0]  # Stabilize
        logits
    softmax_val = torch.softmax(logits + eps, dim=-1)
    pos_sum = torch.clamp(torch.sum(softmax_val * pos_mask, dim=-1), min=
        eps)
    neg_sum = torch.clamp(torch.sum(softmax_val * neg_mask, dim=-1), min=
        eps)

    losses = -1 * torch.log(pos_sum / (pos_sum + neg_sum + eps * 2))

    return losses.mean()
```