# SPO: Self Preference Optimization with Self Regularization

**Yuhao Sun, Yifan Zhang, Quandong Wang, QinZhuo Wu, Wei Liu, Jian Luan**
MiLM PLUS, Xiaomi Inc
{sunyuhao1, zhangyifan27, wuqinzhuo, liuwei40, luanjian}@xiaomi.com
quandwang@hotmail.com

## Abstract

Direct Preference Optimization (DPO) is a widely used offline preference optimization algorithm that enhances the simplicity and training stability of reinforcement learning through reward function reparameterization from PPO. Recently, SimPO (Simple Preference Optimization) and CPO (Contrastive Preference Optimization) have proposed reference-free preference optimization methods to simplify DPO's training process. We observe that these reference-free methods exhibit higher training efficiency but are prone to overoptimization, leading to performance degradation. To address these issues, we propose Self Preference Optimization (SPO). SPO employs the SiLU function to replace the conventional logsigmoid loss function. The SiLU function attains its minimum at a finite value, preventing the model from excessively amplifying the chosen-rejected sample probability ratio and thereby mitigating overoptimization problem. We theoretically demonstrate that the SPO loss is an upper bound of the DPO loss, implying that optimizing the SPO objective implicitly optimizes the DPO objective. We evaluate SPO's effectiveness across multiple benchmarks including AlpacaEval 2 and MT-Bench. Experimental results show that SPO achieves a 7% improvement over SimPO in length-controlled win rate on AlpacaEval 2, while demonstrating superior performance on MT-Bench.

## 1 Introduction

Benefiting from large-scale pre-training (Radford et al., 2019; Mann et al., 2020) and instruction fine-tuning on high-quality data (Wei et al., 2021), large language models (LLMs) have demonstrated exceptional capabilities in generating human-like responses. One of the key steps in building state-of-the-art LLMs is preference optimization, which aligns pre-trained LLMs with human preferences using human assessment data, making the model more helpful, truthful, and harmless.

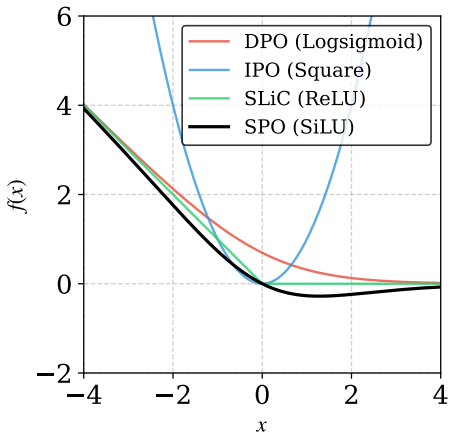


Figure 1: The primary difference among most offline preference optimization methods lies in the choice of the loss function, with logsigmoid (softplus) being a popular option. However, logsigmoid approaches its minimum at negative infinity, which may lead to overoptimization. To mitigate this issue, SPO replaces logsigmoid with SiLU, which has a finite minimum value and possesses regularization properties.

Recently, researchers have been exploring simpler offline algorithms, with Direct Preference Optimization (DPO) (Rafailov et al., 2024) being a representative approach. DPO reparameterizes the reward function in PPO (Schulman et al., 2017), enabling direct policy learning from preference data without the need for an explicit reward model. Due to its simplicity and stability, DPO has been widely adopted in practical applications.

DPO requires a reference policy to ensure that the aligned model does not deviate excessively from the reference model. In contrast, approaches like SimPO (Meng et al., 2024), CPO (Xu et al., 2024a) and ORPO (Hong et al., 2024) propose methods that do not rely on a reference model. Current methods typically implement SFT and preference optimization sequentially. However, this often leads to catastrophic forgetting and complicates the training process. ORPO integrates both



*Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 5601–5614

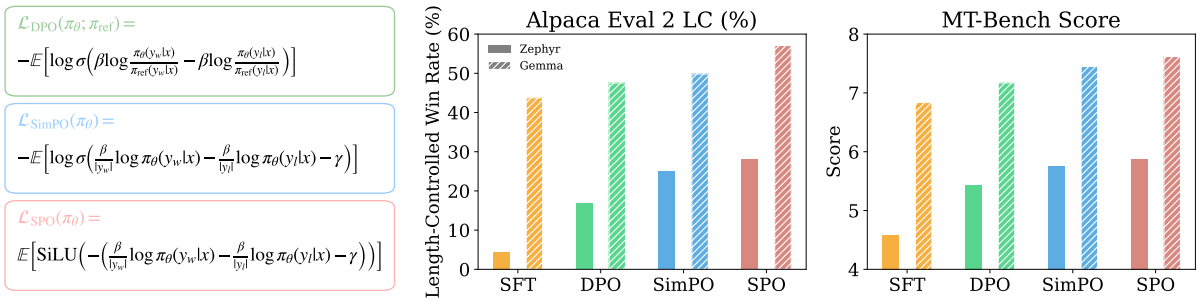


Figure 2: Comparison of loss functions. SPO mainly differ from DPO and SimPO in their loss functions, as indicated in the shaded box. SPO replace logsigmoid by sigmoid linear unit(SiLU), which act as a self regularize activation function. SPO outperforms DPO and SimPO across AlpacaEval 2 and Arena-Hard when applied to different base models.

processes simultaneously, but this results in performance degradation (Wang et al., 2024). SimPO observes that longer sequences tend to have lower log probabilities, which introduces a length bias in the model. To mitigate this effect, SimPO incorporates length normalization into the loss function. However, these methods still suffer from the problem of over-optimization.

In preference learning, especially in DPO and related methods, over-optimization refers to the phenomenon where the policy drifts too far from the reference model. It happens when maximizing the proxy reward (such as the likelihood gap or reward model score) leads to decreased overall performance, consistency, or diversity. After writing out the DPO loss in its standard form, we arrive at the equivalent expression:

$$L(\pi_\theta; , \pi_{\text{ref}}) = \log\left(1 + \left(\frac{\pi_\theta(y_l \mid x)/\pi_{\text{ref}}(y_l \mid x)}{\pi_\theta(y_w \mid x)/\pi_{\text{ref}}(y_w \mid x)}\right)^\beta\right) \quad (1)$$

It is immediate that this term is minimized when:

$$\pi_\theta(y_l \mid x) = 0 \quad (2)$$

If the probability of the rejected sample is pushed too close to zero, the ratio becomes infinitely large regardless of the actual probability of the preferred sample. In this case, the model may exploit weaknesses in the reward signal by focusing solely on minimizing the probability of the negative sample while ignoring the probability of the positive sample, leading it to deviate from the preferred output and the reference policy.

Figure 3 illustrates an over-optimization issue where both positive and negative sample probabilities decrease simultaneously from SimPO's training curves. Both DPO and SimPO optimize the probability difference between positive and negative samples but their objectives allow a scenario where both positive and negative samples probabilities decrease, with the negative sample probability decreasing more significantly. In an extreme case, if the negative sample probability reaches zero, the optimization objective becomes entirely unrelated to the positive samples. This creates a reward hacking issue (Xu et al., 2024b), where the model can minimize the loss simply by only reducing the probability of negative samples and completely ignore the positive samples. Ideally, only the negative sample probability should decrease while the positive sample probability remains stable. Practically, this may be difficult to achieve because there is a certain degree of correlation between positive and negative samples. Therefore, how to design preference optimization algorithms to achieve both high performance and high efficiency remains an open challenge.

We believe that the choice of the loss function is key to addressing this issue. As shown in the figure 1, a recent study (Tang et al., 2024) has pointed out that different methods mainly differ in their loss functions, and proposed GPO (Generalized Preference Optimization), which parameterizes preference optimization losses through a family of convex functions $f$, and views DPO, IPO (Azar et al., 2024), and SLiC (Zhao et al., 2023) as special cases. We found that methods like DPO typically use logsigmoid as the loss function. However, due to the properties of the logsigmoid function, it reaches its minimum when the probability ratio between positive and negative samples approaches infinity. This causes the model to excessively amplify the probability gap between positive and negative samples. On the other hand, if ReLU is used

as the loss function, the gradient becomes zero when the positive-to-negative sample probability ratio exceeds a certain threshold, which may lead to underfitting.

In this paper, we investigate the role and impact of loss functions in pairwise preference datasets for preference optimization models and propose a simple and effective preference optimization method—Self-Preference Optimization (SPO), as shown in figure 2, which utilize SiLU as loss function in the training process. Compared to DPO, our method does not require a reference model, making it resource-efficient. Additionally, unlike other reference-free methods like ORPO and SimPO, we avoid the performance degradation caused by simplifying the preference optimization process through loss function regularization.

After training the SPO model and comparison models with Ultrafeedback (Cui et al., 2024), we evaluate them on instruction-following benchmarks, including AlpacaEval 2 (Dubois et al., 2024) and MT-bench (Zheng et al., 2023). Under the Gemma (Team et al., 2024) model setup, SPO outperforms DPO by up to 9.6% in length-controlled win rate on AlpacaEval 2 compared to GPT-4-turbo (et al., 2023), and surpasses SimPO by 7.3%. We perform ablation studies on the key design aspects of SPO. Since preference optimization methods are sensitive to hyperparameters, we conducted a hyperparameter search based on the suggestions in the SimPO paper to achieve optimal performance. The ablation experiments show that the SPO loss function outperforms other loss functions. Furthermore, referring to methods like CPO and RPO that use SFT loss for regularization, we compare the effects of different SFT weights. The results indicate that a small SFT weight slightly improves the performance on MT-bench while decreasing performance on AlpacaEval 2, suggesting that the regularization effect of the SPO loss function itself is strong, and therefore using only the SPO loss is also viable.

Our contributions can be summarized as follows:

- We propose SPO, a novel reference-free offline preference optimization method. SPO mitigates overoptimization issues and enhances preference optimization effectiveness by regularization of the loss function.

- We theoretically prove that the SPO loss is an upper bound of the DPO loss, meaning that optimizing the SPO objective implicitly optimizes the DPO objective.

- We conduct extensive experiments on various instruction-following benchmarks. Our results show that SPO outperforms comparison methods such as DPO and SimPO across multiple models and benchmarks, demonstrating the effectiveness of SPO.

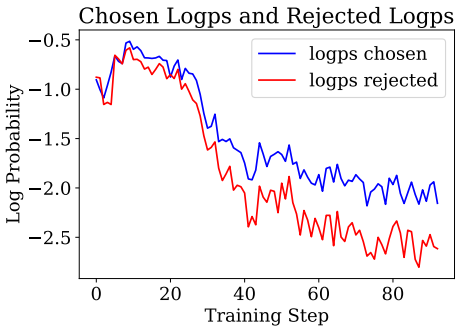


Figure 3: Log probabilities for chosen and rejected responses when using SimPO for preference optimization of the Zephyr model on the Ultrafeedback dataset.

## 2 Related Work

**Offline preference optimization.** Reinforcement Learning from Human Feedback (RLHF) is a method designed to align large language models (LLMs) with human preferences and values. The RLHF process typically consists of three key stages: supervised fine-tuning, reward model training, and policy optimization, with Proximal Policy Optimization (PPO) being a widely adopted algorithm in the policy optimization phase. To address the complexity and inefficiency of online preference optimization, researchers have explored alternative approaches in offline preference optimization, with Direct Preference Optimization (DPO) being a notable example. In this study, we focus exclusively on offline settings to avoid iterative training processes, aiming to leverage the advantages of offline preference optimization while addressing its inherent limitations.

**Preference optimization objectives.** In addition to DPO (Rafailov et al., 2024), many other methods have been proposed. Some approach explores simplified preference optimization objectives that do not rely on a reference model, such as ORPO (Hong et al., 2024) and SimPO (Meng et al., 2024). ORPO merges the SFT and preference optimization stages and optimizes the odds ratio of positive and negative samples. SimPO

uses length normalization techniques to improve the performance of the model. RPO (Liu et al., 2024) regularizes the optimization process using SFT loss. Although this method has been applied in previous research, RPO theoretically demonstrates its effectiveness. GPO (Tang et al., 2024) points out that the main differences between preference optimization methods lie in the design of the loss function and proposes a unified perspective to generalize them. In this study, we compare SPO with a series of offline optimization algorithms, including DPO, IPO (Azar et al., 2024), SLiC (Zhao et al., 2023), and SimPO, and find that SPO outperforms these methods in various experimental settings.

**Overoptimization in DPO.** In the context of direct preference optimization for large language models, overoptimization (Michaud et al., 2020; Gao et al., 2023) can lead to discrepancies between model behavior and human expectations, ultimately reducing performance. When the model relies on an imperfect, overfitted, and misgeneralized proxy reward, it may perform well on limited data but lose effectiveness in real-world applications (Xu et al., 2024b). To address this issue, researchers have explored both theoretical and practical approaches to better understand and manage the uncertainty in learning human preferences from finite data. Our approach tackles this problem by providing a solution that not only avoids overoptimization but is also easily implementable in practice, thereby improving the model's generalization capability and its adaptability to real-world tasks.

## 3 SPO: Self Preference Optimization

In this section, we first introduce the background of DPO. Then, we derive the reference-free SPO objective by replacing the actual reference model with an ideal reference model and regularizing it using the SiLU function. Finally, we incorporate additional length normalization and SFT regularization to obtain the final SPO objective. The pseudocode of our method is as follows (Algorithm 1).

### 3.1 Preliminaries

DPO (Rafailov et al., 2024) (Direct Preference Optimization) has emerged as a widely adopted offline preference optimization method. Unlike traditional approaches that rely on explicitly training a reward model, DPO introduces a reparameterization of the reward function $r$ using a closed-form expression,

**Algorithm 1 : Self Preference Optimization**

**Input:** Dataset ($\mathcal{D}$) with prompts and responses, policy LM $\pi_\theta$, total number of iterations $T$, learning rate $\alpha_t$
**for** $t = 0$ to $T$ **do**
  Sample a mini-batch of tuples $(x, y_w, y_l)$ from $\mathcal{D}$,
  Compute $\mathcal{L}_{\text{SPO}}$ via Eq. (17),
  Update policy parameters $\theta$ using adamw optimizer with learning rate $\alpha_t$.
**end for**

which directly incorporates the optimal policy:

$$r(x, y) = \beta \log \frac{\pi_\theta(y \mid x)}{\pi_{\text{ref}}(y \mid x)} + \beta \log Z(x), \quad (3)$$

where $\pi_\theta$ represents the learned policy, $\pi_{\text{ref}}$ is a reference policy—typically derived from a supervised fine-tuned (SFT) model—and $Z(x)$ is a partition function ensuring normalization.

Preferences are denoted as $y_w \succ y_l \mid x$, where $y_w$ and $y_l$ are the preferred and dispreferred answers, respectively. By leveraging this reward formulation within the Bradley-Terry (BT) ranking model, DPO expresses the probability of preferring one response over another as:

$$p(y_w \succ y_l \mid x) = \sigma(r(x, y_w) - r(x, y_l)), \quad (4)$$

where $\sigma(\cdot)$ denotes the sigmoid function. This formulation allows DPO to model preference data directly via the policy, bypassing the need for a separate reward model. The corresponding optimization objective for DPO is then given by:

$$L(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x,y_w,y_l)\sim D} \log \sigma\left(\beta \log \frac{\pi_\theta(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)} - \beta \log \frac{\pi_\theta(y_l \mid x)}{\pi_{\text{ref}}(y_l \mid x)}\right) \quad (5)$$

Similarly, SimPO also directly models preferences via the policy but introduces a length normalization term to reduce length bias. The objective function of SimPO is defined as:

$$\mathcal{L}_{\text{SimPO}}(\pi_\theta) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}} \log \sigma\left(\frac{\beta}{|y_w|} \log \pi_\theta(y_w|x) - \frac{\beta}{|y_l|} \log \pi_\theta(y_l|x) - \gamma\right). \quad (6)$$

Here, $|y_w|$ and $|y_l|$ denote the lengths of the preferred and less preferred responses, respectively.

The hyperparameter $\gamma$ serves as a margin to encourage sufficient separation between the two terms, in SPO, we derive this margin parameter from a different perspective.

### 3.2 Deriving of SPO Objective

Compared to SFT, DPO has some drawbacks. First, DPO has lower memory efficiency: it requires twice the memory capacity to store both the parameterized policy and the reference policy simultaneously. Second, DPO has lower speed efficiency: the model needs to execute two policies sequentially, doubling the processing time.

Recently, some methods such as ORPO and SimPO have proposed reference-free approaches. However, these reference-free methods are prone to overoptimization, which may lead to distribution shifts and performance degradation. To address these issues, we propose SPO.

The basic form for SPO is as follows:

$$\begin{aligned} \mathcal{L}_{\text{SPO}}(\pi_\theta) = \mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}} \\ SiLU\Big(-(\beta\log\pi_\theta(y_w|x) - \\ \beta\log\pi_\theta(y_l|x) - \gamma)\Big). \end{aligned} \tag{7}$$

Next, we theoretically prove that the SPO objective is an upper bound of the DPO objective.

**Theorem 1.** *Assume that for all $(x, y_w, y_l) \sim D$ the reference model satisfies*

$$\frac{\pi_{\text{ref}}(y_w \mid x)}{\pi_{\text{ref}}(y_l \mid x)} \geq k, \tag{8}$$

*for some constant $k > 0$. Define the DPO loss as*

$$L(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x,y_w,y_l)\sim D} \log\sigma\left(\beta\log\frac{\pi_\theta(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)} - \beta\log\frac{\pi_\theta(y_l \mid x)}{\pi_{\text{ref}}(y_l \mid x)}\right) \tag{9}$$

*Let the idealized reference model $\tilde{\pi}$ satisfy*

$$\frac{\tilde{\pi}(y_w \mid x)}{\tilde{\pi}(y_l \mid x)} = k. \tag{10}$$

*Then,*

$$L(\pi_\theta; \pi_{\text{ref}}) \leq L(\pi_\theta; \tilde{\pi}) + C, \tag{11}$$

*with*

$$C = \mathbb{E}_{(x,y_w,y_l)\sim D}\,\beta\left(\log\frac{\pi_{\text{ref}}(y_w \mid x)}{\pi_{\text{ref}}(y_l \mid x)} - \log k\right) \tag{12}$$

Based on this formula, we further replace the logsigmoid function with SiLU. SiLU (Ramachandran et al., 2017) is a smooth variant of ReLU, first proposed in the GELU (Hendrycks and Gimpel, 2016) paper. GELU was originally motivated by combining ReLU (Nair and Hinton, 2010) and the regularization properties of Dropout (Srivastava et al., 2014). While the Swish (SiLU) activation was introduced later, its functional form was already presented in the GELU paper as an approximation. Swish's main contribution lies in rediscovering "GELU" through an extensive search of activation functions and demonstrating its effectiveness over ReLU. We use SiLU primarily due to its simpler form and its adoption in modern llm architectures like LLaMA's SwiGLU (Grattafiori et al., 2024; Shazeer, 2020). Additionally, as the function graphs of SiLU and GELU are fundamentally similar, SiLU inherits GELU's regularization properties. We found that in the preference optimization scenario, SiLU exhibits a similar regularization effect, since SiLU's minimum value is finite rather than approaching infinity, it prevents the model from overly optimizing the probability difference between positive and negative samples, thus providing a regularization effect.

The logsigmoid function and the softplus function are closely related. Specifically, the logsigmoid function can be expressed in terms of the softplus function as follows:

$$\begin{aligned} \text{logsigmoid}(x) &= -\log(1 + e^{-x}) \\ &= -\text{softplus}(-x). \end{aligned} \tag{13}$$

As shown in the figure 4, SiLU plus a constant is an upper bound of softplus.

**Proposition 1.** *For any $x \in \mathbb{R}$,*

$$\textit{SiLU}(x) + \ln(2) \geq \textit{softplus}(x), \tag{14}$$

*where*

$$\textit{SiLU}(x) = \frac{x}{1 + e^{-x}} \tag{15}$$

*and*

$$\textit{softplus}(x) = \ln(1 + e^x) \tag{16}$$

According to Theorem 1 and Proposition 1, the SPO loss is an upper bound of the DPO loss. Therefore, optimizing the SPO objective will minimize the DPO objective, theoretically proving the effectiveness of the SPO objective.

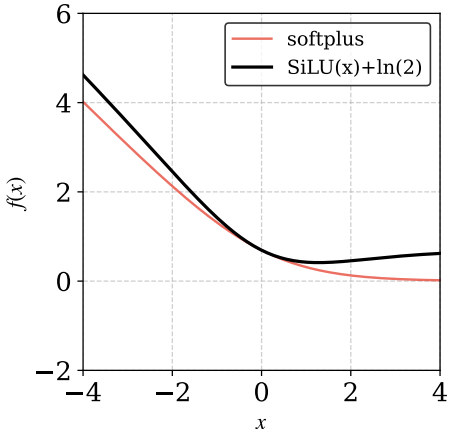


Figure 4: SiLU is smaller than softplus, but SiLU plus an appropriate constant becomes larger than softplus.

### 3.3 Length Normalization

Furthermore, we incorporate length normalization to enhance the model's robustness. The complete SPO loss function is as follows:

$$\mathcal{L}_{\text{SPO}}(\pi_\theta) = \mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\left[SiLU\left(-\left(\frac{\beta}{|y_w|}\log\pi_\theta(y_w|x) - \frac{\beta}{|y_l|}\log\pi_\theta(y_l|x) - \gamma\right)\right)\right]. \tag{17}$$

## 4 Experiments

In this section, our goal is to understand the performance of SPO compared to other preference optimization methods under different experimental settings. We outline our experimental settings (Section 4.1) and present and analyze the main results (Section 4.2). Then, we show the results of ablation studies to compare the impact of several key designs on SPO (Section 4.3).

### 4.1 Experimental Settings

**Evaluation bechmarks.** We primarily use the automatic evaluation frameworks, MT-Bench and AlpacaEval 2, to evaluate our models. These benchmarks assess the models' conversational abilities across various types of questions. Automatic evaluation is highly similar to human evaluation and is more cost-effective, making it widely adopted by the community. As shown in Table 1, AlpacaEval 2 includes 805 questions from 5 datasets, and MT-Bench covers 8 categories, consisting of 80 two-turn dialogues. In addition to these two benchmarks, Arena-Hard (Li et al., 2024) is also a common choice. However, we found that most of the questions in the Arena-Hard benchmark are coding questions and thus lack diversity, which could introduce bias into the evaluation results. Therefore, we decided not to use this benchmark. We report scores following the evaluation protocols of each benchmark. For AlpacaEval 2, we report both the win rate (WR) and the length-controlled win rate (LC). The LC metric is specifically designed to be robust against model verbosity. For MT-Bench, we report the average MT-Bench score with GPT-4o (Hurst et al., 2024) as the evaluation models.

**Base models.** We use two models, Zephyr-7B (Tunstall et al., 2023) and Gemma-2-9B (Team et al., 2024), for preference optimization, the Gemma model is larger and more powerful. The main purpose of using different models as the base model is to verify the generality of SPO. The training process for Zephyr starts by training a base model on the UltraChat-200k (Ding et al., 2023) dataset to obtain an SFT model. We directly use the open-source versions of these models because they have undergone extensive testing, making them more powerful and robust. Then, we use the SFT model as a starting point and perform preference optimization on the UltraFeedback dataset. For the Gemma model, since it has already undergone SFT, we directly perform preference optimization without an additional SFT process. We also use the UltraFeedback dataset for preference optimization to ensure a fair comparison.

**Comparison methods.** As shown in Table 2, we compare SPO with other offline preference optimization methods. SFT can be regarded as a preference optimization method that uses only positive samples. Similar to DPO, CPO uses sequence log-likelihood as a reward and trains alongside an SFT objective. IPO is a theoretically grounded method that avoids DPO's assumption that pairwise preferences can be replaced with pointwise rewards by using a squared loss function. SLiC uses hinge loss for optimization and includes a margin parameter. SimPO employs length normalization, and CPO uses SFT regularization. We found these techniques to be effective in our experiments and thus incorporated them into the SPO loss. We thoroughly tuned the hyperparameters for each baseline method and reported the best performance. Most preference optimization methods, except for SLiC and IPO, use the logsigmoid function as the loss function. In our main experiments, we compared SFT, DPO, CPO, and SimPO. In ablation studies, we compared the effects of different loss functions, such as ReLU, GELU, and Square.

Table 1: Evaluation details for AlpacaEval 2 and MT-Bench

| Benchmark | Questions | Judge Model | Metric |
|---|---|---|---|
| AlpacaEval 2 (Dubois et al., 2024) | 805 | GPT-4 Turbo | LC & win rate |
| MT-Bench (Zheng et al., 2023) | 80 | GPT-4o | Rating of 1-10 |

Table 2: Comparison of Methods and Their Objective Functions

| Method | Objective |
|---|---|
| SFT (Wei et al., 2021) | $-\lambda \log \pi_\theta(y_w\|x)$ |
| DPO (Rafailov et al., 2024) | $-\log \sigma \left(\beta \log \frac{\pi_\theta(y_w\|x)}{\pi_{\text{ref}}(y_w\|x)} - \beta \log \frac{\pi_\theta(y_l\|x)}{\pi_{\text{ref}}(y_l\|x)}\right)$ |
| SLiC (Zhao et al., 2023) | $\max(0, \delta - \log \pi_\theta(y_w\|x) + \log \pi_\theta(y_l\|x)) - \lambda \log \pi_\theta(y_w\|x)$ |
| IPO (Azar et al., 2024) | $\left(\log \frac{\pi_\theta(y_w\|x)}{\pi_{\text{ref}}(y_w\|x)} - \log \frac{\pi_\theta(y_l\|x)}{\pi_{\text{ref}}(y_l\|x)}\right)^2$ |
| CPO (Xu et al., 2024a) | $-\log \sigma \left(\beta \log \pi_\theta(y_w\|x) - \beta \log \pi_\theta(y_l\|x)\right) - \lambda \log \pi_\theta(y_w\|x)$ |
| SimPO (Meng et al., 2024) | $-\log \sigma \left(\frac{\beta}{\|y_w\|} \log \pi_\theta(y_w\|x) - \frac{\beta}{\|y_l\|} \log \pi_\theta(y_l\|x) - \gamma\right)$ |
| **SPO (ours)** | $\text{SiLU}\left(-\left(\frac{\beta}{\|y_w\|} \log \pi_\theta(y_w\|x) - \frac{\beta}{\|y_l\|} \log \pi_\theta(y_l\|x) - \gamma\right)\right)$ |

### 4.2 Main Results

As shown in Table 3 and 4, the simple SFT method can also achieve some improvements over the baseline(The baseline model is the off-the-shelf model we use as a starting point), but the preference optimization algorithms that use positive and negative sample pairs show greater improvements. We believe this is because the positive sample data is not perfect, and the contrast between positive and negative samples provides a direction for improvement of the positive samples. Compared to other preference optimization algorithms, SPO demonstrates consistent improvements on these evaluation datasets. Moreover, the comparison of results between the Zephyr model and the Gemma model shows that SPO achieves greater improvements on a stronger baseline, validating its robustness and effectiveness. Without a reference model, SPO has smaller memory usage and computational requirements compared to DPO, making it simple and efficient to implement.

### 4.3 Ablation Studies

We conducted all ablation experiments on the Zephyr model, including the choice of loss function, the effect of SFT regularization, and the impact of the hyperparameters beta and gamma. The effect of length normalization has been validated in previous papers, so we do not discuss it further.

**Comparison of different loss functions.** As shown in Table 5, we trained the model using common activation functions as loss functions, where ReLU corresponds to SLiC and square corresponds to IPO. The identity function directly optimizes the log probability ratio, but this leads to a large amount of repetitive content in the model output, resulting in the lowest benchmark scores. We believe this is a result of overoptimization, highlighting the necessity of loss function regularization. GELU and SiLU achieved the best results among these functions. Notably, the performance of the GELU function is similar to that of SiLU, because GELU can be approximated by SiLU, making it equivalent to using SiLU with different beta values. Since SiLU is simpler in form compared to GELU, we implemented SiLU in SPO.

**Analysis of training dynamics.** As shown in Figure 5, we compare the training logs, chosen log probabilities (logps), and accuracies of the Zephyr model when trained on UltraFeedback using the logsigmoid(SimPO) and SiLU(SPO) loss functions. It can be observed that the positive sample probabilities decrease to some extent in both methods, but the decrease is less pronounced in SPO compared to SimPO, indicating that the SPO loss function helps prevent overoptimization to some extent. On the other hand, the accuracies of both methods continue to rise, suggesting that the preference optimization algorithm enhances the model's ability to distinguish between positive and negative samples. Ultimately, the accuracy of SPO is slightly higher than that of SimPO, validating the effectiveness of SPO.

**The impact of SFT regularization.** As shown in Table 6, we trained the Zephyr model with different SFT weights. When the SFT weight is relatively

Table 3: Performance of Different Methods on Zephyr-7B and Gemma-2-9B

| Method | Zephyr-7B | | | Gemma-2-9B | | |
|---|---|---|---|---|---|---|
| | AlpacaEval 2 | | MT-Bench | AlpacaEval 2 | | MT-Bench |
| | LC | WR | Score | LC | WR | Score |
| Baseline | 4.46 | 2.55 | 4.58 | 43.75 | 39.01 | 6.84 |
| SFT | 6.55 | 5.37 | 4.81 | 44.26 | 40.14 | 6.92 |
| DPO | 16.83 | 14.16 | 5.44 | 47.54 | 44.39 | 7.17 |
| CPO | 22.31 | 23.14 | 5.64 | 48.50 | 46.62 | 7.31 |
| SimPO | 24.92 | 25.47 | 5.72 | 49.84 | 50.56 | 7.45 |
| **SPO (ours)** | **28.04** | **29.35** | **5.87** | **57.1** | **53.05** | **7.62** |

Table 4: Performance of Different Methods on Mistral-Instruct-7B and LLaMA-Instruct-7B

| Method | Mistral-Instruct-7B | | | LLaMA-Instruct-7B | | |
|---|---|---|---|---|---|---|
| | AlpacaEval 2 | | MT-Bench | AlpacaEval 2 | | MT-Bench |
| | LC | WR | Score | LC | WR | Score |
| Baseline | 12.1 | 11.86 | 5.96 | 22.45 | 23.06 | 6.32 |
| SFT | 16.34 | 15.13 | 6.02 | 26.77 | 27.52 | 6.40 |
| DPO | 20.47 | 19.67 | 6.15 | 30.21 | 31.44 | 6.42 |
| CPO | 24.05 | 24.72 | 6.12 | 33.80 | 32.92 | 6.55 |
| SimPO | 28.53 | 28.20 | 6.22 | 37.57 | 37.02 | 6.59 |
| **SPO (ours)** | **33.48** | **34.16** | **6.34** | **42.55** | **40.69** | **6.75** |

Table 5: Comparison of loss function

| act | AlpacaEval 2 | | MT-Bench |
|---|---|---|---|
| | LC | WR | Score |
| Identity | 0.00 | 0.00 | 1.17 |
| ReLU | 23.89 | 26.23 | 5.56 |
| Square | 21.8 | 22.32 | 5.41 |
| GELU | 26.34 | **29.86** | 5.76 |
| SiLU (SPO) | **28.04** | 29.35 | **5.87** |

Table 6: The Impact of SFT Regularization

| $\lambda$ | AlpacaEval 2 | | MT-Bench |
|---|---|---|---|
| | LC | WR | Score |
| 0 | **28.04** | **29.35** | 5.87 |
| 0.1 | 24.47 | 24.01 | **5.92** |
| 0.5 | 15.9 | 13.18 | 5.58 |
| 1 | 14.25 | 10.95 | 5.31 |

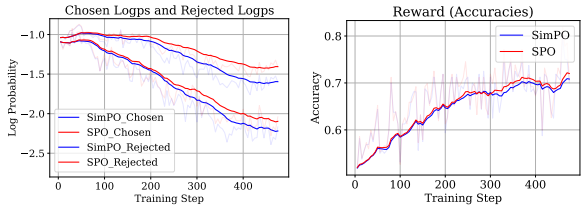


Figure 5: The training curves of the Zephyr model on the UltraFeedback dataset, with chosen log probabilities on the left and accuracies on the right. The red line represents SPO, and the blue line represents SimPO.

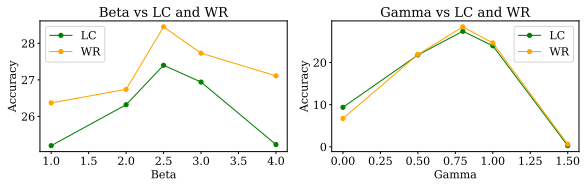


Figure 6: The impact of different hyperparameters beta and gamma under the settings of the Zephyr model.

small, MT-Bench can achieve some improvement, but the performance on AlpacaEval 2 decreases. A larger SFT weight results in a decline in performance on both evaluation sets. We believe that SFT regularization has some effect, but the SiLU function in the SPO loss function itself has a strong regularization effect, so not using SFT regularization is also feasible.

**The impact of hyperparameters.** As shown in Figure 6, the preference optimization algorithms are relatively sensitive to the hyperparameters beta and gamma, so parameter tuning is required to achieve optimal performance. We found that for the Zephyr model, the best model performance is achieved when beta and gamma are set to 2.5 and 0.8, respectively. Therefore, we used this setting in our main experiments. Through calculations,

we found that the minimum value of the loss function is achieved when the log probability ratio of positive and negative samples is 1.31, which corresponds to a positive to negative sample probability ratio of 3.7 for the ideal reference model.

## 5 Conclusion

In this paper, we propose Self-Preference Optimization (SPO), which is a new reference-free offline preference optimization method. SPO mitigates the overoptimization issues in DPO and other reference-free methods through loss function regularization. The key design of SPO lies in replacing the logsigmoid loss function with SiLU. We theoretically prove that the SPO loss function is an upper bound of the DPO loss function, meaning that optimizing the SPO objective implicitly optimizes the DPO objective.

Experimental results show that, across models of various sizes, SPO is preferred over other preference optimization methods in evaluations on AlpacaEval 2 and MT-Bench. Furthermore, as the base model size increases, SPO's win rate against DPO also improves, further demonstrating its stability and generality. Extensive ablation studies indicate that loss function regularization of SPO is crucial and validate its effectiveness.

## Limitations

We only study off-policy methods such as DPO and SimPO, whose behavior may differ from on-policy preference optimization algorithms like PPO. While conducting a comprehensive analysis of various preference optimization methods, we did not cover a broader range of preference optimization algorithms. Additionally, we evaluated our method on only two model size and two public human feedback datasets, whereas some recent studies suggest that automated evaluations may exhibit certain biases compared to human evaluations. We leave a more extensive comparative study for future work.

## Ethics Statement

This study focuses on simplifying the DPO training process and mitigating overoptimization issues during training. Experiments are conducted using publicly available data and pre-trained models, no new models will be released for public use. Therefore, there are no ethical concerns.

## References


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.

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

Tri Dao. 2023. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*.

Runji Wang R.J. Chen R.L. Jin Ruyi Chen Shanghao Lu Shangyan Zhou Shanhuang Chen Shengfeng Ye Shiyu Wang Shuiping Yu Shunfeng Zhou Shuting Pan S.S. Li et al. DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z.F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J.L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

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

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

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin,

Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Leo Gao, John Schulman, and Jacob Hilton. 2023. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning*, pages 10835–10866. PMLR.

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

Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.

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.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Tianhao Wu, Banghua Zhu, Joseph E Gonzalez, and Ion Stoica. 2024. From crowdsourced data to high-quality benchmarks: Arena-hard and benchbuilder pipeline. *arXiv preprint arXiv:2406.11939*.

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*.

Ben Mann, N Ryder, M Subbiah, J Kaplan, P Dhariwal, A Neelakantan, P Shyam, G Sastry, A Askell, S Agarwal, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 1.

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

Eric J Michaud, Adam Gleave, and Stuart Russell. 2020. Understanding learned reward functions. *arXiv preprint arXiv:2012.05862*.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.

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

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

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

Prajit Ramachandran, Barret Zoph, and Quoc V Le. 2017. Searching for activation functions. *arXiv preprint arXiv:1710.05941*.

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.

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

Noam Shazeer. 2020. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

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

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Shengyi Huang, Kashif Rasul, Alvaro Bartolome, Alexander M. Rush, and Thomas Wolf. The Alignment Handbook.

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

Zhichao Wang, Bin Bi, Shiva Kumar Pentyala, Kiran Ramnath, Sougata Chaudhuri, Shubham Mehrotra, Xiang-Bo Mao, Sitaram Asur, et al. 2024. A comprehensive survey of llm alignment techniques: Rlhf, rlaif, ppo, dpo and more. *arXiv preprint arXiv:2407.16216*.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

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

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

Yao Zhao, Rishabh Joshi, Tianqi Liu, Misha Khalman, Mohammad Saleh, and Peter J Liu. 2023. Slic-hf: Sequence likelihood calibration with human feedback. *arXiv preprint arXiv:2305.10425*.

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

## A Implementation Details

We use the PyTorch framework and modify the code based on the alignment handbook (Tunstall et al.) For all models in our experiments, we use DeepSpeed Zero 3 (Rasley et al., 2020) for parallel training and Flash Attention 2 (Dao, 2023) for acceleration. The Zephyr model is trained using 4 x 80GB Nvidia H100 GPUs, while the Gemma 9B model, which requires more memory, is trained using 8 x 80GB Nvidia H100 GPUs. As suggested by previous research, all models are trained for one epoch on the Ultra Feedback dataset. Training the Zephyr model takes approximately 2 hours, while training the Gemma model takes about 4 hours. For optimization, we use the AdamW (Loshchilov, 2017) optimizer with a cosine decay learning rate schedule. To maintain a fixed batch size of 128 across experiments with different numbers of GPUs, we adjust the gradient accumulation. For input length, each instance is truncated and padded to 1,024 tokens for Zephyr and 2,048 tokens for Gemma.

During the inference and evaluation phase, for AlpacaEval 2, we use a sampling decoding strategy to generate responses and set the temperature to 0.7, following SimPO. For MT-Bench, we adhere to the official decoding configuration, which defines different sampling temperatures for different categories. When analyzing the evaluation results, we found that accessing the gpt4 series models has rate limitations, and exceeding the rate limit can result in network errors, leading to abnormal scores. To ensure accurate evaluation, we set the concurrency to 2 and check the completeness of the results after evaluation. For each model, the MT-Bench evaluation takes about 20 minutes, and the AlpacaEval 2 evaluation takes about 1 hour.

## B Theorem Proof

Proof of Theorem 1

*Proof.* We begin with the loss using the reference model:

$$
\begin{aligned}
L(\pi_\theta; \pi_{\text{ref}}) &= -\mathbb{E}_{(x,y_w,y_l)\sim D} \\
&\log \sigma\left(\beta \log \frac{\pi_\theta(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)} - \beta \log \frac{\pi_\theta(y_l \mid x)}{\pi_{\text{ref}}(y_l \mid x)}\right) \\
&= -\mathbb{E}_{(x,y_w,y_l)\sim D} \\
&\log \sigma\left(\beta \log \frac{\pi_\theta(y_w \mid x)}{\pi_\theta(y_l \mid x)} - \beta \log \frac{\pi_{\text{ref}}(y_w \mid x)}{\pi_{\text{ref}}(y_l \mid x)}\right).
\end{aligned} \tag{18}
$$

Since

$$\frac{\pi_{\text{ref}}(y_w \mid x)}{\pi_{\text{ref}}(y_l \mid x)} \geq k \Longrightarrow \log \frac{\pi_{\text{ref}}(y_w \mid x)}{\pi_{\text{ref}}(y_l \mid x)} \geq \log k, \tag{19}$$

we have

$$\beta \log \frac{\pi_\theta(y_w \mid x)}{\pi_\theta(y_l \mid x)} - \beta \log \frac{\pi_{\text{ref}}(y_w \mid x)}{\pi_{\text{ref}}(y_l \mid x)} \leq \beta \log \frac{\pi_\theta(y_w \mid x)}{\pi_\theta(y_l \mid x)} - \beta \log k. \tag{20}$$

Define the loss with the idealized reference model:

$$L(\pi_\theta; \tilde{\pi}) = -\mathbb{E}_{(x,y_w,y_l)\sim D} \log \sigma\left(\beta \log \frac{\pi_\theta(y_w \mid x)}{\pi_\theta(y_l \mid x)} - \beta \log k\right). \tag{21}$$

Because the sigmoid $\sigma(\cdot)$ is strictly increasing, we obtain

$$\sigma\left(\beta \log \frac{\pi_\theta(y_w \mid x)}{\pi_\theta(y_l \mid x)} - \beta \log \frac{\pi_{\text{ref}}(y_w \mid x)}{\pi_{\text{ref}}(y_l \mid x)}\right) \leq \sigma\left(\beta \log \frac{\pi_\theta(y_w \mid x)}{\pi_\theta(y_l \mid x)} - \beta \log k\right). \tag{22}$$

Taking negative logarithms (a decreasing operation) gives

$$-\log \sigma\left(\beta \log \frac{\pi_\theta(y_w \mid x)}{\pi_\theta(y_l \mid x)} - \beta \log \frac{\pi_{\text{ref}}(y_w \mid x)}{\pi_{\text{ref}}(y_l \mid x)}\right) \geq -\log \sigma\left(\beta \log \frac{\pi_\theta(y_w \mid x)}{\pi_\theta(y_l \mid x)} - \beta \log k\right). \tag{23}$$

Thus, from (18) and (21) we immediately have

$$L(\pi_\theta; \pi_{\text{ref}}) \geq L(\pi_\theta; \tilde{\pi}). \tag{24}$$

For the reverse bound, note that $\log \sigma(z)$ is Lipschitz with constant 1 because

$$\frac{d}{dz} \log \sigma(z) = 1 - \sigma(z) \leq 1. \tag{25}$$

Thus, for any $\Delta \geq 0$ and any $z$,

$$\log \sigma(z - \Delta) \geq \log \sigma(z) - \Delta. \tag{26}$$

Set

$$z = \beta \log \frac{\pi_\theta(y_w \mid x)}{\pi_\theta(y_l \mid x)} - \beta \log k \tag{27}$$

and

$$\Delta(x) = \beta\Big(\log \frac{\pi_{\text{ref}}(y_w \mid x)}{\pi_{\text{ref}}(y_l \mid x)} - \log k\Big) \geq 0. \tag{28}$$

Then,

$$-\log \sigma\Big(z - \Delta(x)\Big) \leq -\log \sigma(z) + \Delta(x). \tag{29}$$

Taking expectations over $D$ gives

$$L(\pi_\theta; \pi_{\text{ref}}) \leq L(\pi_\theta; \tilde{\pi}) + \mathbb{E}_{(x,y_w,y_l)\sim D}\, \Delta(x). \tag{30}$$

Defining

$$C = \mathbb{E}_{(x,y_w,y_l)\sim D}\, \Delta(x) \tag{31}$$

we conclude that

$$L(\pi_\theta; \pi_{\text{ref}}) \leq L(\pi_\theta; \tilde{\pi}) + C. \tag{32}$$

This completes the proof. □

Proof of Proposition 1

*Proof.* Define the difference function

$$f(x) = \text{SiLU}(x) + \ln(2) - \text{softplus}(x). \tag{33}$$

Simplifying, we obtain

$$f(x) = -\frac{x}{1 + e^x} + \ln(2) - \ln(1 + e^{-x}). \tag{34}$$

The derivative of $f(x)$ is

$$f'(x) = \frac{xe^{-x}}{(1 + e^{-x})^2}, \tag{35}$$

whose sign is determined by $x$: $f'(x) > 0$ for $x > 0$ and $f'(x) < 0$ for $x < 0$. Thus, $f(x)$ attains its minimum at $x = 0$, where $f(0) = 0$. Furthermore, as $x \to \pm\infty$, $f(x) \to \ln(2) > 0$. Therefore, $f(x) \geq 0$ for all $x \in \mathbb{R}$, and the proposition holds. □

## C Gradient Analysis

The loss function of Self Preference Optimization (SPO) in its basic form is defined as:

$$L_{\text{SPO}}(\pi_\theta) = \mathbb{E}_{(x,y_w,y_l)\sim D}\,[\text{SiLU}\,(-\,(\beta \log \pi_\theta(y_w|x) - \beta \log \pi_\theta(y_l|x) - \gamma\ ))] \tag{36}$$

Here, $\pi_\theta(y|x)$ is the policy function, representing the probability of selecting action $y$ given input

$x$. $y_w$ and $y_l$ are positive and negative samples, respectively. $\beta$ and $\gamma$ are hyperparameters. The SiLU function is defined as:

$$\text{SiLU}(z) = z \cdot \sigma(z) \tag{37}$$

where $\sigma(z) = \frac{1}{1+e^{-z}}$ is the Sigmoid function. Let $z = -\left(\beta \log \pi_\theta(y_w|x) - \beta \log \pi_\theta(y_l|x) - \gamma\right)$.

The loss function can be rewritten as:

$$L_{\text{SPO}}(\pi_\theta) = \mathbb{E}_{(x,y_w,y_l)\sim D}\left[\text{SiLU}(z)\right] \tag{38}$$

First, compute the derivative of the SiLU function with respect to $z$:

$$\begin{aligned}\frac{d\text{SiLU}(z)}{dz} &= \sigma(z) + z\sigma(z)(1-\sigma(z)) \\ &= \sigma(z)(1 + z(1-\sigma(z)))\end{aligned} \tag{39}$$

Next, compute the derivative of $z$ with respect to $\theta$:

$$\frac{\partial z}{\partial \theta} = -\beta\left(\frac{\partial \log \pi_\theta(y_w|x)}{\partial \theta} - \frac{\partial \log \pi_\theta(y_l|x)}{\partial \theta}\right) \tag{40}$$

Thus, the gradient of the SPO loss function with respect to $\theta$ is:

$$\frac{\partial L_{\text{SPO}}(\pi_\theta)}{\partial \theta} = \mathbb{E}_{(x,y_w,y_l)\sim D}\left[\frac{\partial \text{SiLU}(z)}{\partial z} \cdot \frac{\partial z}{\partial \theta}\right] \tag{41}$$

$$\begin{aligned}\frac{\partial L_{\text{SPO}}(\pi_\theta)}{\partial \theta} &= \mathbb{E}_{(x,y_w,y_l)\sim D}\left[\sigma(z)(1 + z(1-\sigma(z)))\right. \\ &\left.\cdot -\beta\left(\frac{\partial \log \pi_\theta(y_w|x)}{\partial \theta} - \frac{\partial \log \pi_\theta(y_l|x)}{\partial \theta}\right)\right.\end{aligned} \tag{42}$$

Another commonly used loss function is the logsigmoid form:

$$\begin{aligned}L_{\text{logsigmoid}}(\pi_\theta) = \mathbb{E}_{(x,y_w,y_l)\sim D} \\ \left[-\log \sigma\left(\beta \log \pi_\theta(y_w|x) - \beta \log \pi_\theta(y_l|x) - \gamma\right)\right]\end{aligned} \tag{43}$$

The gradient derivation for this form is as follows:

Let $z = \beta \log \pi_\theta(y_w|x) - \beta \log \pi_\theta(y_l|x) - \gamma$.

The loss function becomes:

$$L_{\text{logsigmoid}}(\pi_\theta) = \mathbb{E}_{(x,y_w,y_l)\sim D}\left[-\log \sigma(z)\right] \tag{44}$$

The derivative of the Sigmoid function is:

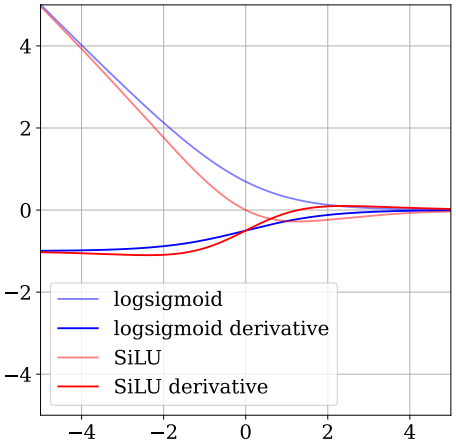


Figure 7: Comparison of SPO and DPO gradients, red: SPO, blue: DPO, solid lines represent the derivatives, transparent lines represent the corresponding original functions.

$$\frac{d\sigma(z)}{dz} = \sigma(z)(1-\sigma(z)) \tag{45}$$

Thus, the gradient of the logsigmoid loss function with respect to $z$ is:

$$\begin{aligned}\frac{\partial L_{\text{logsigmoid}}(\pi_\theta)}{\partial z} &= -\frac{1}{\sigma(z)} \cdot \sigma(z)(1-\sigma(z)) \\ &= -(1-\sigma(z))\end{aligned} \tag{46}$$

Next, compute the derivative of $z$ with respect to $\theta$:

$$\frac{\partial z}{\partial \theta} = \beta\left(\frac{\partial \log \pi_\theta(y_w|x)}{\partial \theta} - \frac{\partial \log \pi_\theta(y_l|x)}{\partial \theta}\right) \tag{47}$$

Thus, the gradient of the logsigmoid loss function with respect to $\theta$ is:

$$\begin{aligned}\frac{\partial L_{\text{logsigmoid}}(\pi_\theta)}{\partial \theta} &= \mathbb{E}_{(x,y_w,y_l)\sim D} \\ &\left[\frac{\partial L_{\text{logsigmoid}}(\pi_\theta)}{\partial z} \cdot \frac{\partial z}{\partial \theta}\right]\end{aligned} \tag{48}$$

$$\begin{aligned}\frac{\partial L_{\text{logsigmoid}}(\pi_\theta)}{\partial \theta} &= \mathbb{E}_{(x,y_w,y_l)\sim D}\left[-(1-\sigma(z))\right. \\ &\left.\cdot\beta\left(\frac{\partial \log \pi_\theta(y_w|x)}{\partial \theta} - \frac{\partial \log \pi_\theta(y_l|x)}{\partial \theta}\right)\right]\end{aligned} \tag{49}$$

**Advantages of SPO Gradients.** In SPO gradients, the term $\sigma(z)(1 + z(1 - \sigma(z)))$ dynamically adjusts the gradient direction based on $z$, which can be considered as the weight of the gradient, the other term is the same as in the logsigmoid form loss. As shown in figure 7, the weight term in the logsigmoid gradient is always negative, resulting in a fixed gradient direction. In contrast, the SPO gradient changes direction when the value exceeds a certain threshold, preventing overoptimization.

## D Other Details

**Scientific artifacts.** We utilized various scientific achievements in our paper, including preference datasets, foundational large language models (LLMs), and training and evaluation tools. All the achievements used are properly cited. Current large models and preference datasets may encompass a wide range of data types and leverage data from various domains and sources, so we do not provide detailed information in this paper. Readers can refer to the original sources for more information. In this paper, we primarily use these achievements for non-distribution and non-commercial purposes, in compliance with their licensing requirements.

**Use of AI assistants.** To reduce the cost of manual revisions, we used ChatGPT (Ouyang et al., 2022) and DeepSeek R1 (DeepSeek-AI, 2025) to revise the language of the paper. The revisions were made solely to enhance the clarity and readability of the text and not for any other purpose.