SPPD: Self-training with Process Preference Learning Using Dynamic Value Margin

Anonymous ACL submission

Abstract

Recently, enhancing the numerical and logical reasoning capability of Large Language Models (LLMs) has emerged as a research hotspot. Existing methods face several limitations: inference-phase techniques (e.g., Chain of Thoughts) rely on prompt selection and the pretrained knowledge; sentence-level Supervised Fine-Tuning (SFT) and Direct Preference Optimization (DPO) struggle with step-wise mathematical correctness and depend on stronger models distillation or human annotations; while Reinforcement Learning (RL) approaches incur high GPU memory costs and unstable training. To address these, we propose **S**elf-training framework integrating **P**rocess **P**reference learning using **D**ynamic value margin (SPPD). SPPD leverages a process-based Markov Decision Process (MDP) and Bellman optimality equation to derive **dynamic value margin** on step-level preference optimization, which employs tree-based self-sampling on model responses **without any distillation** from other models. Furthermore, we theoretically prove that SPPD is **equivalent to on-policy policy gradient methods** under reward constraints. Experiments on 7B-scale models demonstrate superior performance across in-domain and out-domain mathematical benchmarks. We open-source our code at https://anonymous.4open.science/r/SPPD-DCDD.

1 Introduction

Recently, the O-series models (OpenAI, 2024) have achieved a significant leap in the mathematical reasoning capabilities of LLMs. Consequently, enhancing the numerical and logical reasoning capability of LLMs has emerged as a research hotspot (Chen et al., 2023; Yu et al., 2023; Jimenez et al., 2023; Shao et al.; Liao et al., 2024b; Lai et al., 2024; Guo et al., 2025).

From now on, there are lots of methods to promote the model reasoning capability. During the **inference** phase, the most common and effective approach is to employ Chain of Thoughts (CoT) prompts, which can stimulate the model's inherent reasoning and thinking abilities (Wei et al., 2022). Similar methods include Tree of Thoughts (ToT) (Yao et al., 2024), Best of N (BoN) (Zheng et al., 2024; Yuan et al., 2024), Monte Carlo Tree Search (MCTS) (Feng et al., 2023; Zhang et al., 2024a), and so on. However, these methods do not involve training policy models but rely on increasing computational volume during the inference phase, heavily depending on prompt selection and the pretrained knowledge embedded within the model. Moreover, **SFT** (Zhang et al., 2024a; Feng et al., 2023) or **DPO** (Rafailov et al., 2024b,a) based on human annotations or feedback from more advanced AI also serves as an effective way to enhance the model's reasoning capabilities. These methods leverage human-curated selections or stronger open-source and close-source models to inject good reasoning paradigms, such as long-thought processes and reflection, into the model being trained. However, all these methods are at the sentence level, which does not align well with the requirement for correctness at every step in mathematical reasoning scenarios. Meanwhile, such methods are either constrained by time-consuming manual selection processes or require support from more powerful models, like STILL-2 (Min et al., 2024) and Skywork-o1-open (Skywork, 2024b). When the model to be trained is already the strongest reasoning model available, how can we further improve the model's reasoning performance without any distillation? While **RL-based** methods like Proximal Policy Optimization (PPO) (Schulman et al., 2017), Group Relative Policy Optimization (GRPO) (Shao et al.; Guo et al., 2025), Reinforcement Fine-Tuning (RFT) (Luong et al., 2024), etc., can address the aforementioned issues. However, these methods are online approaches involving numerous time-consuming inference operations during training, requiring loading and training multiple models, imposing high demands on GPU memory and leading to highly unstable training processes.

To solve above issues, we propose **S**elf-training with **P**rocess **P**reference learning using **D**ynamic value margin (SPPD). Unlike sentence-level SFT and DPO, we completely abandon the data distillation approach and propose optimizing at the step level by integrating dynamic value margin. Specifically, SPPD utilizes a process-based MDP and a process-based Bradley-Terry (BT) Model (Bradley and Terry, 1952). By leveraging the Bellman optimality equation (Barron and Ishii, 1989) and the online RL objective modeled with MDP (Rafailov et al., 2024a), SPPD derives step-wise direct preference optimization using **dynamic value margin**. Additionally, SPPD **does not rely on any stronger models for data distillation**. Instead, it employs a tree search approach, which utilizes step-level trajectory sampling solely on the model's own response and logits score. To ensure smoother and more effective training of SPPD, we introduce an SFT and DPO strategy based on PRM rejection sampling, progressively enhancing the model's reasoning capabilities from coarse-grained sentence-level optimization to fine-grained step-level refinement. Finally, we theoretically prove that under specific reward constraints, our method **is equivalent to on-policy policy gradient method**.

The experimental results demonstrate that SPPD achieves widespread and significant improvements across different model architectures of 7B size and various in-domain and out-domain mathematical test datasets. It surpasses most existing open-source models of the same size and some closed-source models, demonstrating the effectiveness and robustness of SPPD. Our contribution are summarized as follows: 1) We utilize the Bellman optimality equation and the online RL objective modeled with MDP to achieve SPPD and iteratively improve the reasoning capability. 2) We design a step-level tree self-sampling scheme without any distillation from stronger model. 3) We theoretically prove that our method is equivalent to on-policy policy gradient optimization.

2 Related Work

Enhance Reasoning Capability of LLMs. Recently, a substantial body of research focuses on enhancing the reasoning capabilities of LLMs. These methodologies are primarily divided into two categories: the inference phase and the Post-Training phase. During the inference phase, early studies concentrate on stimulating the model's inherent reasoning abilities by modifying prompts (Wei et al., 2022; Yao et al., 2024). Subsequent research leverages the consistency of multiple inferences by the model (Yuan et al., 2024; Wang et al., 2022) or integrates tree search strategies (Feng et al., 2023; Zhang et al., 2024a) to guide the model towards more accurate decoding processes. However, these approaches do not involve training and heavily rely on the model's intrinsic reasoning capabilities. In the Post-Training phase, SFT (Feng et al., 2023) and DPO (Rafailov et al., 2024b,a) emerge as primary enhancement techniques. These methods depend on human-curated selection of high-quality reasoning trajectories or distillation of responses from stronger models (Min et al., 2024) to improve the reasoning performance of smaller or weaker models. Nevertheless, these approaches are time-consuming and unsustainable. RL paradigms, exemplified by PPO (Schulman et al., 2017), GRPO (Guo et al., 2025; Shao et al.), and ReFT (Luong et al., 2024), effectively address the aforementioned issues but introduce significant GPU memory consumption and training instability challenges.

Step-Level Direct Preference Optimization. In order to optimize and improve the model's reasoning capability from the step level, CPO (Zhang et al., 2024b) aligns each step of the CoT reasoning paths with those of ToT using the inherent preference information in the tree-search process, but it control LLMs to generate the thought data by prompt, which may influent the model generation quality. Step-DPO (Lai et al., 2024) treats individual reasoning steps as units for preference optimization. However, it utilizes the GPT4 to evaluate the correctness of step, which could bring introduced bias and is expensive. TPO (Liao et al., 2024b) claims that the policy can potentially learn more effectively from a ranked preference list of responses given the prompt and utilizes adaptive step reward to adjust the reward values of each step in the trajectory. However, it introduce a stronger form of "catastrophic forgetting" and imbalanced distribution of the preference tree reward values.

3 Preliminaries

In this section, we first define the step-level MDP in natural language process. Subsequently, based on

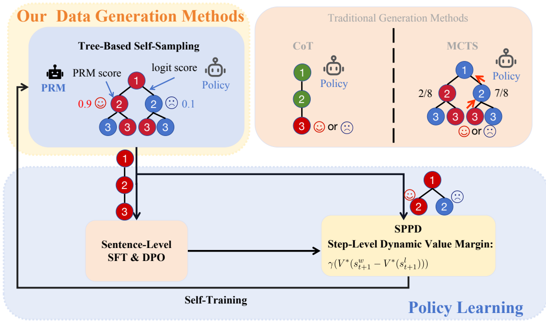

Figure 1: The framework of SPPD: unlike CoT and MCTS, Tree-Based Self-Sampling generates step trajectories with common prefixes and significantly preserves the output distribution of the policy. The former provides step preference signals for SPPD, while the latter theoretically ensures consistency with on-policy gradient methods, thereby enabling self-enhancement of the model's reasoning capabilities.

the step-level MDP, we modify the original RLHF objective and provide the optimal (fixed-point) solution to maximum casual entropy problem.

Step-Level MDP in LLMs. We describe the step-level MDP in natural language process. The step-level MDP is defined as the following quintuple: $\mathcal{M} = (\mathcal{A}, \mathcal{S}, f, r, \rho_0)$, where $\mathcal{A}$ represents the set of action spaces, consisting of a reasoning step a_t; $\mathcal{S}$ represents the set of states, which in natural language denotes the sequence of the problem and the current reasoning step $s_t = s_0|a_1|a_2|...|a_t$, where | denotes the string concatenation operation and s_0 is the problem. It is noteworthy that the selection of a_t depends on the current state. $f : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ represents the state transition function, indicating the transition from the current state to the next state after performing a certain action. Specifically, $f(s, a) = s|a$. $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, representing the immediate reward obtained after performing a certain action in the current state. ρ_0 represents the distribution of the problems.

RLHF objective with the Step-Level MDP. In the original RLHF objective (Ouyang et al., 2022), the rewards obtained from trajectories are modeled as a bandit problem (Zhao et al., 2024). However, such sparse rewards are not suitable for policy learning in models, especially in mathematical reasoning tasks (Riedmiller et al., 2018; Wilcox et al., 2022). Based on the step-level MDP, we modify the RLHF objective as follows (Rafailov et al., 2024a):

$$\max_{\pi_\theta} \mathbb{E}_{a_t \sim \pi_\theta(\cdot|\mathbf{s}_t)} [\sum_{t=0}^{T} (r(\mathbf{s}_t, \mathbf{a}_t) + \underbrace{\beta \log \pi_{\text{ref}}(\mathbf{a}_t \mid \mathbf{s}_t))}_{\text{KL penalty}}$$
$$+ \beta \mathcal{H}(\pi_\theta) \mid \mathbf{s}_0 \sim \rho(\mathbf{s}_0)], \quad (1)$$

where π_θ represents the large language policy model with learnable parameters, π_{ref} represents reference model and β is used to control the policy model not to deviate too far from the reference model, $\mathcal{H}(\pi_\theta)$ is the entropy of π_θ. This optimization problem is known as the **Maximum Causal Entropy**. Ziebart (2010) have proven that Equation (1) has a fixed-point solution π^*, defined as follows:

$$\pi^*(a_t \mid s_t) = \pi_{\text{ref}}(a_t|s_t) e^{(Q^*(s_t,a_t) - V^*(s_t))/\beta}, \quad (2)$$

where $V^*(s_t)$ represents the partition function of the π^* distribution, used to normalize the probability distribution, and $Q^*(s_t, a_t)$ denotes the expected sum of future immediate rewards starting from the state-action pair (s_t, a_t) under the policy π^*.

4 Method

In this section, we first propose a process preference learning scheme using dynamic value margins based on the step MDP and BT-model, and then refine this preference learning scheme using the reward equivalence. Additionally, we introduce a tree-based self-sampling method designed to generate step trajectories with common prefix. Finally, we introduce sentence-level SFT and DPO using PRM, aiming to make the model training smoother and more effective.

4.1 Process Preference Learning with Dynamic Value Margin

First, we derive the process preference learning with dynamic value margin starting from the optimal Bellman equation and revisit the traditional step DPO (Lai et al., 2024) from a different perspective.

Lemma 4.1 (Optimal Step Reward Function)**.** *Under the step MDP definition in Section 3 and fix solution for the maximum casual entropy problem (Equation (2)), the optimal step reward function can be calculate as follow:*

$$r(s_t, a_t) = \underbrace{\beta \log \frac{\pi^*(a_t|s_t)}{\pi_{ref}(a_t|s_t)}}_{\textit{Implicit Reward}} + \underbrace{V^*(s_{t+1}) - V^*(s_t)}_{\textit{Value Gain}}. \quad (3)$$

Prove for Lemma (4.1) is shown in Appendix D.1. Equation (3) demonstrates that the immediate reward in the MDP consists of the model's **implicit reward** and the **value gain** of the optimal value function. Assuming we have the following step-level preference pairs $(s_t, a_{t+1}^w, a_{t+1}^l)$, based on the step-level BT-model, we have the optimal preference distribution:

$$p^*(a_{t+1}^w \succ a_{t+1}^l) = \sigma\left(r(s_t, a_{t+1}^w) - r(s_t, a_{t+1}^l)\right).$$

Here, $\sigma(x) = 1/(1+e^{-x})$ is the sigmoid function. Finally, we give the step DPO loss using dynamic value margin.

Theorem 4.2 (Step DPO Loss Using Dynamic Value Margin.)**.** *If we aim to minimize the Kullback-Leibler(KL) divergence between the step-level preference distribution p_{data} in $\mathcal{D}_{step}$ and the model's current preference distribution p_θ under the sampling of π_{ref}, we can obtain the following loss function:*

$$\begin{aligned}\mathcal{L}_{step\text{-}dpo} = &-\mathbb{E}_{a_{t+1}^w, a_{t+1}^l \sim \pi_{ref}(\cdot|s_t)}[\\ &\log\sigma(\beta h_\theta(a_{t+1}^w, a_{t+1}^l)\\ &-(V^*(s_{t+1}^w) - V^*(s_{t+1}^l)))],\end{aligned} \quad (4)$$

where $h_\theta(a_{t+1}^w, a_{t+1}^l) = \log\frac{\pi_\theta(a_t^w|s_t)}{\pi_{ref}(a_t^w|s_t)} - \log\frac{\pi_\theta(a_t^l|s_t)}{\pi_{ref}(a_t^l|s_t)}$.

The prove is shown in Appendix D.2. In traditional step DPO (Lai et al., 2024), the value function prediction at each step is defined as 0. However, we argue that the **value gain** in the immediate reward (Equation (3)), or equivalently, the term $V^*(s_{t+1}^w) - V^*(s_{t+1}^l)$ in Equation (4), considers the difference in the optimal value function predictions for the preferred states. This manifests in the step DPO loss as a dynamic value margin that varies depending on the preferred states s_{t+1}^w and s_{t+1}^l, rather than treating all states uniformly. In practice, we use a PRM score to approximate the optimal value function. In Section 5, we will provide more profound theoretical insights and conclusions.

Reward Equivalence. To make the optimization process more controllable, we revise Equation (3) by introducing the concept of reward equivalence.

Lemma 4.3. *Reward Equivalence (Rafailov et al., 2024a)] Two reward functions r and r' are equivalent if and only if there exists a potential function $\Phi: \mathcal{S} \to \mathbb{R}$ that satisfies the following equation:*

$$r(s_t, a_t) = r'(s_t, a_t) + \Phi(f(s_t, a_t)) - \Phi(s_t).$$

In Equation (3), the potential function is our optimal value function, i.e., $\Phi(s) = V^*(s)$. At the same time, it is easy to see that when we scale this potential function, $\Phi'(s) = \gamma\Phi(s)$, Φ' still satisfies the definition of potential function. Therefore, we can modify Equation (3) to obtain an equivalent reward expression:

$$r^\gamma(s_t, a_t) = r(s_t, a_t) + \gamma\Phi(f(s_t, a_t)) - \gamma\Phi(s_t).$$

Repeating the derivation in Section 4.1 , we modify the final loss as follows:

$$\begin{aligned}\mathcal{L}_{\text{step-dpo}}^\gamma = &-\mathbb{E}_{a_{t+1}^w, a_{t+1}^l \sim \pi_{\text{ref}}(\cdot|s_t)}[\\ &\log\sigma(\beta h_\theta(a_{t+1}^w, a_{t+1}^l)\\ &-\gamma(V^*(s_{t+1}^w) - V^*(s_{t+1}^l)))].\end{aligned} \quad (5)$$

Remark. *Although the concept of reward equivalence in Rafailov et al. (2024a) implies that the optimal preference model belongs to the same equivalence class, including the original step-DPO when $\gamma = 0$, the introduction of γ makes the optimization process more controllable due to its influence on optimization. This has been verified in Section 6.3.*

4.2 Tree-Based Self-Sampling on LLMs

Traditional reasoning algorithms (token-level decoding) is almost impossible to guarantee the generation of reasoning trajectories with identical prefixes. To address this issue, this paper adopts a tree-structured reasoning approach, as illustrated in Figure 1. Specifically, the process is divided into four steps: "Selection, Expansion, Collection and Scoring". During the selection process, at the current state s_t, we record the average log probability score for each child node a_t, defined as:

$$s(a_t|s_t) = \frac{1}{|a_t|}\sum_{i=0}^{|a_t|}\log\pi_{\text{infer}}(a_{t,i}|s_t|a_{t,<i}),$$

where $|a_t|$ represents the token length of the current step, $a_{t,<i}$ denotes the first $i-1$ tokens of a_t, and π_{infer} represents the probability distribution output of the inference model (policy in RL). In practice, we set $\pi_{\text{infer}} = \pi_{\text{ref}}$. Furthermore, we normalize the score distribution of all child nodes and perform sampling to select child nodes. Each selection starts from the root node and proceeds until reaching a leaf node that contains the final answer. If a node is not a terminal node and has no child nodes, we expand the node to obtain C possible reasoning steps. After performing the above steps

K times, we traverse the expanded prefix tree and collect all answers that contain complete reasoning paths. Finally, we invoke the PRM to score each step of the reasoning trajectory, resulting in the final step-level dataset:

$$\mathcal{D}_{\text{step}} = \{(s_0^{(i)}, s_t^{(i,j)}, v_t^{(i,j)}) \mid i \in [N], j \in [K], t \in |\tau^{(i,j)}|\},$$

where N is the number of the problems, $v_t^{(i,j)}$ represents the PRM score of the state $s_t^{(i,j)}$ in the j-th prefix sequence of the problem $s_0^{(i)}$.

4.3 PRM-Enhanced SFT & DPO

To make the model's learning process smoother, we introduce the concept of curriculum learning, initially allowing the model to learn strategies at the sentence-level. This step leverages the signal responses from the PRM on sampled trajectories to perform rejection sampling, and employs both supervised learning and preference learning to continuously improve the model's reasoning capabilities. Specifically, we define the following positive and negative sample trajectories:

$$\tau_+^{(i)} = \max_{j\in[K]} \min_{v_t^{(i,j)}} \mathcal{D}_{\text{step}}^+,$$

$$\tau_-^{(i)} = \min_{j\in[K]} \min_{v_t^{(i,j)}} \mathcal{D}_{\text{step}}^-.$$

Here, $\mathcal{D}_{\text{step}}^+$ and $\mathcal{D}_{\text{step}}^-$ represent complete trajectories with correct and incorrect final answers, respectively. During the SFT phase, we minimize the next token prediction loss on $\tau_+^{(i)}$. In the DPO phase, we select positive samples from $\{\tau_+^{(i)}\}_{i=1}^N$ and negative samples from $\{\tau_-^{(i)}\}_{i=1}^N$, thereby constructing preference samples for sentence-level DPO. We emphasize that SFT and DPO optimize the model's reasoning capabilities at a coarse-grained level, aiming to warm up the model's reasoning abilities and lay the foundation for subsequent step-level preference learning.

5 Theoretical Analysis

In this Section, we prove that the equivalence between offline step DPO and online policy gradient under the specific reward definition.

Definition 5.1 (Preference decoding model π_θ^p induced by π_θ)**.** *Assume that when $s = s_t$, the possible action space $\mathcal{A}_t = \{a_{t+1}^w, a_{t+1}^l\}$. We define π_θ^p as the following parameterized distribution:*

$$\pi_\theta^p(a_{t+1}^w|s_t) = \sigma(r_{\theta,t}^w - r_{\theta,t}^l),$$

where,

$$r_{\theta,t}^w = \beta \log \frac{\pi_\theta(a_{t+1}^w|s_t)}{\pi_{ref}(a_{t+1}^w|s_t)} - V^*(s_{t+1}^w) + V^*(s_t),$$

$$r_{\theta,t}^l = \beta \log \frac{\pi_\theta(a_{t+1}^l|s_t)}{\pi_{ref}(a_{t+1}^l|s_t)} - V^*(s_{t+1}^l) + V^*(s_t).$$

Remark. *The preference decoding model π_θ^p can be viewed as performing sampling on a binary prefix tree based on preference probabilities. This model relies on the probability outputs of the standard language model π_θ.*

Lemma 5.1 (Online Policy Gradient on π_θ^p (Lin and Zhou, 2019))**.** *For any MDP, the expected long-term reward on π_θ^p is given by $J(\theta) = \sum_\tau \pi_\theta^p(\tau) r(\tau)$, where $r(\tau)$ represents the long-term reward of trajectory τ. The policy gradient of this expected long-term reward on π_θ^p is:*

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta^p}\left[r(\tau)\sum_{t=0}^{T-1}\nabla_\theta \log \pi_\theta^p(a_{t+1}^w|s_t)\right]. \tag{6}$$

Theorem 5.2 (Equivalence Between Offline Step DPO and Online Policy Gradient)**.** *If we define the reward in Equation (6) as $r(\tau) = \prod_{i=1}^T \frac{\pi_{ref}(a_t|s_t)}{\pi_\theta^p(a_t|s_t)}$, and define the **Offline every-step preference loss** as:*

$$\mathcal{L}_{every\text{-}step} = \mathbb{E}_{\tau\sim\pi_{ref}^p}\left[-\sum_{t=0}^{T-1}\log \pi_\theta^p(a_{t+1}^w|s_t)\right], \tag{7}$$

then the following equivalence holds:

$$\nabla_\theta J(\theta) = -\nabla_\theta \mathcal{L}_{every\text{-}step}.$$

The prove is shown in Appendix D.3.

Remark. *It is easy to see that $\mathcal{L}_{every\text{-}step}$ (Equation (7)) can be considered as the equivalent expression of $\mathcal{L}_{step\text{-}dpo}$ (Equation (4)) when the sampling tree branches at $C = 2$ and preference sampling is performed for every action at each step. Theorem 5.2 demonstrates that, **under the specific definition of the reward, optimizing the gradient of the offline***

preference loss is equivalent to the policy gradient of the preference decoding model in the online setting. *Additionally, for the definition of the reward $r(\tau)$, when the reward is large, it indicates that the trajectory probability output $\pi_\theta^p(\tau)$ of the preference decoding model is relatively small. To reduce the overall loss, the optimization process will focus more on the loss of this particular trajectory at this step.*

6 Experiments

6.1 Setup

Datasets. For the training prompt data, we sample a total of 10k prompts from the training datasets of GSM8k (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021), with GSM8K and MATH accounting for 40% and 60% respectively. We use Qwen2.5-7B-Base (Yang et al., 2024) and Llama3.1-8B-Instruct (Meta@AI, 2024) as the base models, and employ Skywork-o1-Open-PRM-Qwen-2.5-7B (Skywork, 2024a) as PRM to generate $\mathcal{D}_{\text{step}}$ using the step data generation method mentioned in Section 4.2. For more information regarding the data format and PRM, please refer to the Appendix A & B.

Evaluation. The maximum generation length for inference is set at 2048. The test set includes in-domain subsets such as GSM8k and MATH500, as well as out-domain subsets like Gaokao2023 (Liao et al., 2024a), OCW Course (OCW) (Lewkowycz et al., 2022), and the OlympiadBench (He et al., 2024) test subset OE-TO-MATH-COMP. The testing methods comprise: 1) **Greedy-CoT**: Test results based on greedy decoding and CoT prompt pass@1. 2) **MAJ@N**: Repeat inference N times based on the CoT prompt, and select the most frequently occurring answer as the final answer. 3) **ORM_VOTE@N**: Repeat inference N times based on the CoT prompt, use Skywork-o1-Open-PRM-Qwen-2.5-7B as the ORM for scoring, aggregate scores for identical answers, and choose the answer with the highest score. 4) **ORM_MAX@N**: Omit the step of aggregating scores for identical answers in **ORM_VOTE@N** and directly select the answer with the highest score. More evaluation methods refer to Appendix C.

Implementation. During the data generation phase, we perform tree sampling for each question with a count of $K = 64$, and each node branches into $C = 2$. When selecting step-level preference pairs, to mitigate the impact of PRM scoring noise, we only use action preference pairs with a scoring difference exceeding 0.5 for training (PRM scores range between 0 and 1). In the SFT phase, we use the Adam optimizer with a learning rate of 5e-6, while in the DPO and step-DPO phases, we employ the SGD optimizer with a learning rate of 1e-5, both utilizing the cosine method for learning rate decay. The β for both DPO and step DPO is set to 0.1. The γ for step DPO is chosen from $\{0.1,0.5,1.0,2.0,5.0\}$. All experiments are conducted on 8 Nvidia 80GB H800 GPUs.

6.2 Main Result

Compared to the base model: Our approach achieves significant improvements without utilizing any stronger model's responses for distillation shown in Table 1. Specifically, using SFT-PRM, we observe enhancements of 4.4% and 5.8% on the in-domain evaluation datasets MATH and GSM8k, respectively. With DPO-PRM, the improvements are 3.8% and 1.2%, respectively, on these same datasets. Building on this foundation, we further enhances the model's reasoning capabilities using SPPD, achieving additional improvements of 2.8% and 0.5% on the two evaluation datasets. The gains from SPPD stem from leveraging PRM signals, transitioning from coarse-grained optimization at the sentence level to fine-grained dynamic optimization at the step level. Additionally, during the inference phase, increasing computational load and employing the **ORM_VOTE** aggregation strategy further demonstrates the model's peak reasoning capabilities, achieving accuracies of 79% and 94.7% on MATH and GSM8k, respectively, outperforming current models of similar size.

Continued gains in the second stage: In the first stage, the training data generated by the base model has been fully utilized. Following the principles of offline RL, we update the policy model's sampling trajectories, using the best model trained in the first stage as our new policy model to repeat our training process. This resulted in the SPPD-Stage2 model. Compared to SPPD, SPPD-Stage2 achieves further improvements of 1.2% and 0.5% on MATH and GSM8k, respectively. These results highlight the effectiveness of updating the policy model and demonstrate the robustness of the SPPD.

6.3 Ablation Study

Different Base Model. We evaluate the effectiveness of the SPPD method on different base models, specifically Llama3.1-8B-Instruct and Qwen2.5-

Model	Size	Open	General	MATH500	GSM8k
Claude-3-Opus*	-	✗	✔	60.1	95.0
GPT4-1106 (Achiam et al., 2023)*	-	✗	✔	64.3	91.4
GPT4o-0513*	-	✗	✔	76.6	95.8
o1 (OpenAI, 2024)*	-	✗	✔	94.8	-
Qwen2-7B-Instruct-Step-DPO (Lai et al., 2024)	7B	✔	✗	55.0	85.4
DeepSeek-MATH-7B-Instruct (Shao et al.)	7B	✔	✗	44.4	80.9
OpenMath2-Llama3.1-8B (Toshniwal et al., 2024)	8B	✔	✗	65.4	90.1
Llama3.1-8B-Instruct (Meta@AI, 2024)	8B	✔	✔	47.0	82.6
Qwen2.5-7B-Instruct (Yang et al., 2024)	7B	✔	✔	72.8	89.3
Qwen2.5-7B-Base	7B	✔	✔	60.0	82.3
+SFT-PRM	7B	✔	✗	64.4	88.1
+SFT-PRM & DPO-PRM	7B	✔	✗	68.2	89.3
+SPPD	7B	✔	✗	**71.0** +2.8%	**89.8** +0.5%
+SPPD+MAJ@64	7B	✔	✗	76.4	93.2
+SPPD+ORM_MAX@64	7B	✔	✗	74.0	**94.9**
+SPPD+ORM_VOTE@64	7B	✔	✗	**79.0**	94.7
+SPPD-Stage2	7B	✔	✗	**72.2** +4.0%	**90.3** +1.0%
+SPPD-Stage2+MAJ@64	7B	✔	✗	78.6	93.6
+SPPD-Stage2+ORM_MAX@64	7B	✔	✗	78.0	**95.0**
+SPPD-Stage2+ORM_VOTE@64	7B	✔	✗	**80.4** +12.2%	94.6 +5.3%

Table 1: Main Results. * denotes we use officially reported results.

7B-Instruct. Given that Instruct models undergo sufficient optimization at the sentence level, we do not perform PRM-SFT and PRM-DPO training on these models. Instead, we directly utilize the trajectories from the Instruct models for dynamic value margin step DPO training. The results appear in Table 2. The findings indicate that on the Llama3.1-8B-Instruct model, the SPPD method achieves improvements of 4.6% and 3.6% on the MATH and GSM8k evaluation datasets, respectively. On the Qwen2.5-7B-Instruct model, the SPPD method improves performance by 2.2% and 0.8%, respectively. These experimental results demonstrate that the SPPD method performs well across different base models, showcasing its robustness with respect to the choice of base model.

Generalization on Out-Domain Distributions. To evaluate the generalization capabilities of SPPD on out-domain distributions, we select three out-domain evaluation datasets: GaoKao2023, OCW and OlympaidBench (using only the OlympaidBench-OE-TO-MATH-COMP portion). The results are presented in Table 3. The experiments show that using Qwen2.5-7B-Base as the base model, after applying SPPD, there are steady improvements across all three out-of-domain evaluation datasets. Specifically, improvements over the base model stand at 8.8%, 13.7%, and 5.6%, respectively. Over PRM-DPO, the improvements reach 1.8%, 4.8%, and 2.4%, respectively. Furthermore, the reasoning capabilities see further enhancement through the ORM_VOTE aggregation strategy.

Effectiveness of Dynamic Value Margin. In Section 4.1, we model the dynamic value margin variation using MDP approach, deriving a step DPO method with dynamically changing margins from a mathematical perspective. To validate the effectiveness of this dynamic value margin approach, we use Qwen2.5-7B-Base and Llama3.1-8B-Instruct as base models, followed by PRM-SFT and PRM-DPO training. We then compare SPPD with both no-margin step DPO ($\gamma = 0$) and fixed-margin step DPO. The results are summarized in Table 4. The findings reveal that fixed-margin step DPO outper-

Model	MATH500	GSM8K
Llama3.1-8B-Instruct	46.6	81.2
+SPPD	51.2	84.8
	+4.6%	+3.6%
+SPPD+MAJ@64	58.2	88.5
+SPPD+ORM_MAX@64	**67.0**	**92.0**
+SPPD+ORM_VOTE@64	66.4	90.7
	+19.8%	+9.5%
Qwen2.5-7B-Instruct	72.8	89.3
+SPPD	75.0	91.1
	+2.2%	+0.8%
+SPPD+MAJ@64	80.6	93.4
+SPPD+ORM_MAX@64	77.0	**95.2**
+SPPD+ORM_VOTE@64	**82.2**	94.6
	+9.4%	+5.3%

Table 2: Result on Llama3.1-8B-Instruct and Qwen2.5-7B-Instruct.

forms no-margin step DPO, indicating that adjusting the margin benefits the learning process of step DPO. Meanwhile, Compared to fixed-margin step DPO, SPPD demonstrates superior performance. On the Qwen model, improvements on MATH and GSM8K are 0.9% and 0.31%, respectively, while on the Llama model, the improvements are 2.0% and 1.3%, respectively. This improvement stems from our consideration of the value model score differences between preference pairs during modeling, which dynamically adjusts the margin for preference learning based on signals from the value model. SPPD makes the step-level preference training more reliable and reduces the risk of overfitting.

Impact of γ. To investigate the impact of the

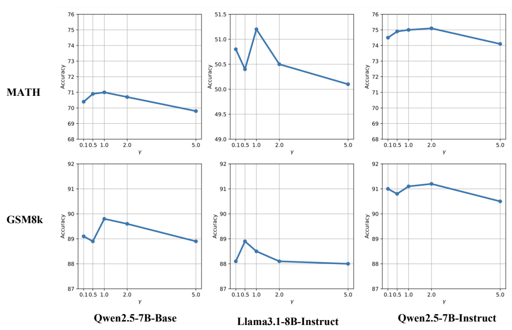

Figure 2: Impact of γ in dynamic value margin.

hyperparameter γ on the SPPD method as described in Formula 5, we selecte three base models: Qwen2.5-7B-Base, Llama3.1-8B-Instruct, and Qwen2.5-7B-Instruct. We adjust γ within the set $\{0.1, 0.5, 1.0, 2.0, 5.0\}$ and evaluated the performance of these models on the MATH and GSM8k datasets. The results are presented in Figure 2. Our experimental findings indicate that selecting an appropriate γ is beneficial for the training of SPPD. It is observed that both excessively large and small values of γ are detrimental to the training of dynamic value margins in SPPD, thereby affecting the generalization to some extent. However, overall, the performance remains relatively stable, particularly on the GSM8k dataset. This suggests that a balanced choice of γ is crucial for optimizing the effectiveness of the SPPD approach across different models.

Model	GaoKao2023	OCW	OlympaidBench*
Qwen2.5-7B-Base	48.0	6.3	20.5
+SFT-PRM	52.2	19.1	22.8
+SFT-PRM & DPO-PRM	55.0	16.1	23.7
+SPPD	**56.8**	**20.0**	**26.1**
	+1.8%	+4.8%	+2.4%
+SPPD+MAJ@64	62.6	29.4	43.3
+SPPD+ORM_MAX@64	63.4	28.3	41.4
+SPPD+ORM_VOTE@64	**64.4**	**30.9**	**45.4**
	+9.4%	+14.8%	+21.7%

Table 3: Result on out-domain test datasets. OlympaidBench* denotes we only use OlympaidBench-OE-TO-Math-COMP test dataset.

Model	Method	Margin	MATH500	GSM8K
Qwen2.5-7B	SPPD	Dynamic	**71.00**	**89.80**
	Step-dpo-	0	69.60	89.40
	fix-margin	γ^*	70.10	89.49
Llama3.1-8B	SPPD	Dynamic	**51.2**	**84.8**
	Step-dpo-	0	48.8	83.2
	fix-margin	γ^*	49.2	83.5

Table 4: SPPD vs fixed margin step DPO on Qwen2.5-7B-Base and Llama3.1-8B-Instruct. γ^* represents $\gamma(V^*(s_{t+1}^w) - V^*(s_{t+1}^l)) = \gamma^*$ in Formula 5.

7 Conclusion

In this work, we propose SPPD, a self-training with process preference learning using dynamic value margin. SPPD utilizes the Bellman optimality equation and the online RL objective modeled with MDP and designs a step-level tree self-sampling scheme without any distillation. Moreover, we propose a SFT and DPO scheme using PRM for rejection sampling, making the training of SPPD smothor and more effective. Finally, we theoretically demonstrate that under specific reward constraints, our method is equivalent to on-policy policy gradient optimization.

Limitations

Several limitations remain in our current work. Firstly, our work relies on the effectiveness of PRM, and studies have shown that PRM's performance varies across different policy models and task scenarios; some PRMs may fail under specific tasks (Zheng et al., 2024). This work neglects the updates of PRM. As policy is continuously iterated, PRM faces the risk of becoming ineffective. Additionally, both PPO and GRPO are modeled based on bandit, and how to integrate MDP modeling with on-policy methods remains an important subject for future research.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

EN Barron and H Ishii. 1989. The bellman equation for minimizing the maximum cost. *NONLINEAR ANAL. THEORY METHODS APPLIC.*, 13(9):1067–1090.

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

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. 2023. Teaching large language models to self-debug. *arXiv preprint arXiv:2304.05128*.

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

Xidong Feng, Ziyu Wan, Muning Wen, Stephen Marcus McAleer, Ying Wen, Weinan Zhang, and Jun Wang. 2023. Alphazero-like tree-search can guide large language model decoding and training. *arXiv preprint arXiv:2309.17179*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, et al. 2024. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. *arXiv preprint arXiv:2402.14008*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2023. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770*.

Xin Lai, Zhuotao Tian, Yukang Chen, Senqiao Yang, Xiangru Peng, and Jiaya Jia. 2024. Step-dpo: Step-wise preference optimization for long-chain reasoning of llms. *arXiv preprint arXiv:2406.18629*.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. 2022. Solving quantitative reasoning problems with language models, 2022. *URL https://arxiv. org/abs/2206.14858*.

Minpeng Liao, Wei Luo, Chengxi Li, Jing Wu, and Kai Fan. 2024a. Mario: Math reasoning with code interpreter output–a reproducible pipeline. *arXiv preprint arXiv:2401.08190*.

Weibin Liao, Xu Chu, and Yasha Wang. 2024b. Tpo: Aligning large language models with multi-branch & multi-step preference trees. *arXiv preprint arXiv:2410.12854*.

Kaixiang Lin and Jiayu Zhou. 2019. Ranking policy gradient. *arXiv preprint arXiv:1906.09674*.

Trung Quoc Luong, Xinbo Zhang, Zhanming Jie, Peng Sun, Xiaoran Jin, and Hang Li. 2024. Reft: Reasoning with reinforced fine-tuning. *arXiv preprint arXiv:2401.08967*.

Meta@AI. 2024. Introducing llama 3.1: Our most capable models to date.

Yingqian Min, Zhipeng Chen, Jinhao Jiang, Jie Chen, Jia Deng, Yiwen Hu, Yiru Tang, Jiapeng Wang, Xiaoxue Cheng, Huatong Song, et al. 2024. Imitate, explore, and self-improve: A reproduction report on slow-thinking reasoning systems. *arXiv preprint arXiv:2412.09413*.

OpenAI. 2024. Openai o1-mini.

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.

Rafael Rafailov, Joey Hejna, Ryan Park, and Chelsea Finn. 2024a. From r to q^*: Your language model is secretly a q-function. *arXiv e-prints*, pages arXiv–2404.

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

Martin Riedmiller, Roland Hafner, Thomas Lampe, Michael Neunert, Jonas Degrave, Tom Wiele, Vlad Mnih, Nicolas Heess, and Jost Tobias Springenberg. 2018. Learning by playing solving sparse reward tasks from scratch. In *International conference on machine learning*, pages 4344–4353. PMLR.

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. *URL https://arxiv.org/abs/2402.03300*.

Skywork. 2024a. Skywork/skywork-o1-open-prm-qwen-2.5-7b.

Skywork. 2024b. Sskywork-o1-open.

Shubham Toshniwal, Wei Du, Ivan Moshkov, Branislav Kisacanin, Alexan Ayrapetyan, and Igor Gitman. 2024. Openmathinstruct-2: Accelerating ai for math with massive open-source instruction data. *arXiv preprint arXiv:2410.01560*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Albert Wilcox, Ashwin Balakrishna, Jules Dedieu, Wyame Benslimane, Daniel Brown, and Ken Goldberg. 2022. Monte carlo augmented actor-critic for sparse reward deep reinforcement learning from suboptimal demonstrations. *Advances in neural information processing systems*, 35:2254–2267.

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

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2024. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2023. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. 2024. Self-rewarding language models. *arXiv preprint arXiv:2401.10020*.

Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. 2024a. Rest-mcts*: Llm self-training via process reward guided tree search. *arXiv preprint arXiv:2406.03816*.

Xuan Zhang, Chao Du, Tianyu Pang, Qian Liu, Wei Gao, and Min Lin. 2024b. Chain of preference optimization: Improving chain-of-thought reasoning in llms. *arXiv preprint arXiv:2406.09136*.

Heyang Zhao, Chenlu Ye, Quanquan Gu, and Tong Zhang. 2024. Sharp analysis for kl-regularized contextual bandits and rlhf. *arXiv preprint arXiv:2411.04625*.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2024. Processbench: Identifying process errors in mathematical reasoning. *arXiv preprint arXiv:2412.06559*.

Brian D Ziebart. 2010. *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. Carnegie Mellon University.

A Data Example

We demonstrate the preference trajectories sampled from the tree using SPPD as shown below. As previously mentioned, the chosen step and the rejected step are selected based on their PRM score difference exceeding 0.5.

Training data format.

[**Question**]
Evaluate: $64^2 - 36^2$.
[**Common Prefix**]
Step 1: Recognize that the given expression is of the form $a^2 - b^2$, which can be factorized using the difference of squares method.
Step 2: Apply the difference of squares formula, which is $a^2 - b^2 = (a+b)(a-b)$.
Step 3: Identify a and b in the given expression. Here, $a = 64$ and $b = 36$.
Step 4: Evaluate the sum and difference of a and b.
a + b = 64 + 36

a - b = 64 - 36
Step 5: Multiply the result from step 4 back together.
[**Chosen Step**]
$(64 + 36)(64 - 36)$
Step 6: Perform the addition and subtraction operations.
The sum of 64 and 36 is 100. The difference of 64 and 36 is 28.
[**Chosen PRM Score**]
0.934
[**Rejected Step**]
Step 6: $(64 + 36) * (64 - 36) = (100) * (28) = 1400$
[**Rejected PRM Score**]
0.258

B PRM score distribution

We use the Skywork-o1-Open-PRM-Qwen-2.5-7B (Skywork, 2024a)model to score 545,990 reasoning trajectories obtained from Qwen2.5-7B-Base (Yang et al., 2024) through Tree-Based Self-Sampling. The score for the t-th step of the i-th trajectory is denoted as $v_t^{(i)}$.

First, we calculate three metrics (ORM score, Mean PRM score, and Minium PRM score) on trajectories that produce correct answers and those that result in incorrect answers. If a metric exceeds 0.5, the PRM considers the sample to be a correct trajectory; otherwise, it is deemed an incorrect trajectory. We then compute the PRM accuracy rates under these three metrics, see Table 5. The experimental results demonstrate that Skywork-o1-Open-PRM-Qwen-2.5-7B exhibits strong discriminative ability for both correct and incorrect trajectories under sampled trajectories. Specifically, the ORM metric shows superior performance in identifying correct trajectories, achieving over 90% accuracy. In contrast, the minimum PRM score excels in distinguishing incorrect trajectories, reaching an accuracy of 92.5%. However, using the mean PRM score, the discriminative ability for correct trajectories is notably higher than for incorrect trajectories. This is because Skywork-o1-Open-PRM-Qwen-2.5-7B can effectively identify erroneous steps, resulting in high scores (close to 1) before these steps occur, which renders the mean PRM score ineffective for judging incorrect trajectories. Conversely, the minimum PRM score identifies the lower bound of trajectory scoring, making it the most suitable metric for evaluating incorrect trajectories.

Metric	#	ORM	Mean PRM	Minium PRM
Correct	281,983	0.908	0.920	0.705
Incorrect	264,007	0.870	0.696	0.925

Table 5: Skywork-o1-Open-PRM-Qwen-2.5-7B accuracy.

Meanwhile, we divide each trajectory into five equal segments, calculate the average score for each segment, and plot the score distribution in box plots categorized by correct and wrong trajectories, as shown in the Figure 3. The figure indicates that for correct trajectories, PRM assigns relatively high scores to all steps with smaller variance; for wrong trajectories, the segment scores given by PRM tend to decrease on average as they get closer to the answer, with the variance also decreasing, suggesting that PRM's confidence in the wrong trajectory leading to an incorrect answer increases.

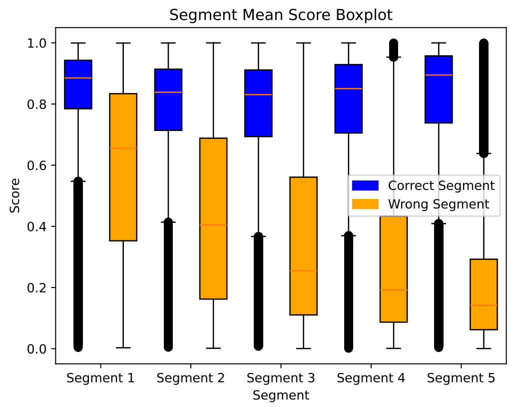

Figure 3: Skywork-o1-Open-PRM-Qwen-2.5-7B distribution.

C Evaluation

C.1 Evaluation Prompts

For a fair evaluation, the same prompt and format is applied to our trained models as well as other open-source models:

Prompt used for evaluation.

[SYSTEM]
Please reason step by step and put your answer in \\boxed{}.

[Question]
{question}.

D Prove

D.1 Prove for Lemma (4.1)

Lemma D.1 (Optimal Step Reward Function)**.** *Under the step MDP definition3 and fix solution for the maximum casual entropy problem (Equation (2)), the optimal step reward function can be calculate as follow:*

$$r(s_t, a_t) = \underbrace{\beta \log \frac{\pi^*(a_t|s_t)}{\pi_{ref}(a_t|s_t)}}_{\textit{Implicit Reward}} + \underbrace{V^*(s_{t+1}) - V^*(s_t)}_{\textit{Value Gain}}. \tag{8}$$

Proof. According to the Bellman optimality equation (Barron and Ishii, 1989) in step MDP, we have:

$$Q^*(s_t, a_t) = r(s_t, a_t) + V^*(f(s_t, a_t)). \tag{9}$$

Here, if $s_{t+1} = f(s_t, a_t)$ is a terminal state, then $V^*(f(s_t, a_t)) = 0$. Meanwhile, if we log-linearize the Equation (2), we have:

$$Q^*(s_t, a_t) = \beta \log \frac{\pi^*(a_t|s_t)}{\pi_{\text{ref}}(a_t|s_t)} + V^*(s_t). \tag{10}$$

Therefore, combine the Equation (9) & (10), we have:

$$r(s_t, a_t) = \underbrace{\beta \log \frac{\pi^*(a_t|s_t)}{\pi_{\text{ref}}(a_t|s_t)}}_{\text{Implicit Reward}} + \underbrace{V^*(s_{t+1}) - V^*(s_t)}_{\text{Value Gain}}.$$

□

D.2 Prove for Theorem D.2

Theorem D.2 (Step DPO Loss Using Dynamic Value Margin.)**.** *If we aim to minimize the Kullback-Leibler(KL) divergence between the step-level preference distribution p_{data} in $\mathcal{D}_{step}$ and the model's current preference distribution p_θ under the sampling of π_{ref}, we can obtain the following loss function:*

$$\begin{aligned}\mathcal{L}_{\textit{step-dpo}} = -\mathbb{E}_{a^w_{t+1}, a^l_{t+1} \sim \pi_{\textit{ref}}(\cdot|s_t)}[\\ \log \sigma(\beta h_\theta(a^w_{t+1}, a^l_{t+1}) \\ - (V^*(s^w_{t+1}) - V^*(s^l_{t+1})))],\end{aligned}$$

where $h_\theta(a^w_{t+1}, a^l_{t+1}) = \log \frac{\pi_\theta(a^w_t|s_t)}{\pi_{\textit{ref}}(a^w_t|s_t)} - \log \frac{\pi_\theta(a^l_t|s_t)}{\pi_{\textit{ref}}(a^l_t|s_t)}$.

Proof. According to the Equation (3), we have:

$$\begin{aligned}&p_\theta(a^w_{t+1} \succ a^l_{t+1}|s_t) \\ &= \sigma(\beta h_\theta(a^w_{t+1}, a^l_{t+1}) - (V^*(s^w_{t+1}) - V^*(s^l_{t+1})))\end{aligned}$$

So the KL divergence between p_θ and p_{data} under the sampling of π_{ref} is:

$$\begin{aligned}&\mathbb{E}_{a^w_{t+1}, a^l_{t+1} \sim \pi_{ref}(\cdot|s_t)}[D_{KL}(p_{data}||p_\theta)] \\ &= \mathbb{E}_{a^w_{t+1}, a^l_{t+1} \sim \pi_{ref}(\cdot|s_t)}[\\ &p_{data}(a^w_{t+1} \succ a^l_{t+1}|s_t) \log \frac{p_{data}(a^w_{t+1} \succ a^l_{t+1}|s_t)}{p_\theta(a^w_{t+1} \succ a^l_{t+1}|s_t)} \\ &+ p_{data}(a^l_{t+1} \succ a^w_{t+1}|s_t) \log \frac{p_{data}(a^l_{t+1} \succ a^w_{t+1}|s_t)}{p_\theta(a^l_{t+1} \succ a^w_{t+1}|s_t)}] \\ &= -\mathbb{E}_{a^w_{t+1}, a^l_{t+1} \sim \pi_{\text{ref}}(\cdot|s_t)}[\log p_\theta(a^w_{t+1} \succ a^l_{t+1}|s_t)],\end{aligned}$$

which is the same as Equation (4).

□

D.3 Prove for Theorem 5.2

Theorem D.3 (Equivalence Between Offline Step DPO and Online Policy Gradient)**.** *If we define the reward in Equation (6) as $r(\tau) = \prod_{i=1}^{T} \frac{\pi_{ref}(a_t|s_t)}{\pi^p_\theta(a_t|s_t)}$, and define the **Offline every-step preference loss** as:*

$$\mathcal{L}_{\textit{every-step}} = \mathbb{E}_{\tau \sim \pi^p_{\textit{ref}}}\left[-\sum_{t=0}^{T-1} \log \pi^p_\theta(a^w_{t+1}|s_t)\right],$$

then the following equivalence holds:

$$\nabla_\theta J(\theta) = -\nabla_\theta \mathcal{L}_{\textit{every-step}}.$$

Proof.

$$\begin{aligned}&\nabla_\theta \mathcal{L}_{every-step} \\ &= \mathbb{E}_{\tau \sim \pi^p_{ref}}[-\sum_{t=0}^{T-1} \nabla_\theta \log \pi^p_\theta(a^w_{t+1}|s_t))] \\ &= \mathbb{E}_{\tau \sim \pi^p_\theta}[-\frac{\pi^p_{ref}(\tau)}{\pi^p_\theta(\tau)} \sum_{t=0}^{T-1} \nabla_\theta \log \pi^p_\theta(a^w_{t+1}|s_t))] \\ &= \mathbb{E}_{\tau \sim \pi^p_\theta}[\\ &- \prod_{i=0}^{T-1} \frac{\pi^p_{ref}(a_{t+1}|s_t)}{\pi^p_\theta(a_{t+1}|s_t)} \sum_{t=0}^{T-1} \nabla_\theta \log \pi^p_\theta(a^w_{t+1}|s_t))] \\ &= \mathbb{E}_{\tau \sim \pi^p_\theta}[-r(\tau) \sum_{t=0}^{T-1} \nabla_\theta \log \pi^p_\theta(a^w_{t+1}|s_t))] \\ &= -\nabla_\theta J(\theta).\end{aligned}$$

□