LLM Alignment Through Successive Policy Re-weighting (SPR)

Xinnan Zhang[*]
University of Minnesota
Minneapolis, MN, USA
zhan9359@umn.edu

Siliang Zeng[*]
University of Minnesota
Minneapolis, MN, USA
zeng0176@umn.edu

Jiaxiang Li[*]
University of Minnesota
Minneapolis, MN, USA
li003755@umn.edu

Kaixiang Lin
Amazon
kaixianglin.cs@gmail.com

Mingyi Hong
University of Minnesota
Minneapolis, MN, USA
mhong@umn.edu

Abstract

Reinforcement learning (RL) is serving as the cornerstone of aligning large language models (LLMs) to human behavior, by providing an appealing formulation and a suite of effective algorithms for learning behavior strategies through interacting with the underlying environment. Current paradigm of RL-based methods for LLM alignment, such as reinforcement learning with human feedback (RLHF) involves utilizing a reward function learned from extensive offline datasets to expediate the online training of reinforcement learning. The reward function learned is then used for policy optimization to obtain an improved policy (i.e. the LLM). Despite the success of RL approaches in aligning LLM with offline datasets, there are significant computational/limit of resources concern on applying RL-based methods for LLMs. For example, standard RLHF requires simultaneously loading four models to the computing unit. In this paper, we develop a novel policy optimization algorithm named Successive Policy Re-weighting (SPR), matching the peak memory consumption of standard supervised fine-tune (SFT). Further, SPR can leverage both offline and online datasets to expediate online training and improve the sample efficiency. Specifically, SPR leverages a supervised learning subroutine to achieve policy improvement through re-weighting the policy according to the importance/performance of executed actions. Such simple and effective method is computationally inexpensive, requiring loading only one model at each update step, matching the computational cost of standard supervised fine-tuning procedure. Experimental results show that the proposed method can significantly outperform benchmark algorithms and accelerate the online training with available offline dataset.

1 Introduction

Aligning LLM to follow human instructions is of the central focus for artificial general intelligence (AGI). Recently, RL-based algorithms for large language model alignment, such as reinforcement learning with human feedback (RLHF, see Ouyang et al. (2022); Christiano et al. (2017)) are serving as the backbone of the recent success on the emergence of intelligence for LLMs such as GPT-4 (Achiam et al., 2023), Claude-3 (Anthropic, 2024) and Gemini-1.0 (Team et al., 2023).

[*]Equal contribution; more junior authors listed earlier.

Among these methods, the most iconic RLHF utilizes online preference datasets which consists of input prompt and two continuations, where one is preferred over the other, to train a reward model and use this reward model for policy optimization to iteratively improve LLMs. Therefore online policy optimization methods such as proximal policy optimization (PPO, see Schulman et al. (2017)) are the most prevalent in industry and the SOTA alignment approaches. On the other hand, Direct Preference Optimization (DPO, see Rafailov et al. (2024)) simplifies RLHF by training the policy/LLM directly while implicitly learns the reward model via log of the ratio of likelihood between the learned model and a reference model. Both RLHF and DPO are successful in terms of improving the instruction-following and reasoning ability of LLMs (see, for example Ouyang et al. (2022); Tunstall et al. (2023); Dubey et al. (2024)) over the most straightforward supervised fine-tune (SFT) approach, which is analogous to the plain behavior cloning approach in RL literature (Pomerleau, 1988; Osa et al., 2018). LLMs aligned using RLHF are thus believed to open the current AI boom toward AGI (Bubeck et al., 2023).

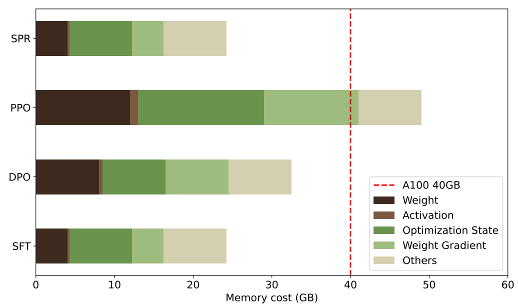


Figure 1: Estimated memory consumption of running the `Pythia-1B` model on `TL;DR` dataset with batch-size 2 on a single device, without checkpointing, memory offloading and distributed training such as Deepspeed. Here "SPR" is the implementation of our Algorithm 2.

There are several important distinctions between RL-based methods (such as RLHF) and DPO: First, RL-based methods require an explicit reward model for policy improvement, while DPO learns the policy directly; Second, RL-based method in general can effectively utilize an online pipeline using the explicit reward model, while the standard DPO is designed for offline data. RLHF utilizes standard policy optimization algorithm such as PPO at the policy optimization step, whose naive implementation requires loading four models at the same time, namely the reward, the policy, and two value function models; while DPO only requires loading two models (a policy and a reference model) simultaneously, but is arguably more sensitive to the distribution shift between the base model outputs and preference data, and it is not quite effective on challenging tasks (Xu et al., 2024; Lin et al., 2024; Ivison et al., 2024). Witnessing the success of RLHF, the memory efficiency issue of both RL-based method and DPO, as well as the limitation of offline nature of algorithms such as DPO, we pose the following question:

Can we effectively reduce the compute and memory requirement of policy optimization to SFT level, while still maintaining an explicit reward function for efficient online training?

In this paper, we provide an affirmative answer to above question by proposing a novel algorithm named **Successive Policy Re-weighting (SPR)**, which is a RL-based method, whose (peak) computational cost matches that of supervised fine-tuning (SFT). Specifically, our contributions are summarized as follows:

• We revisit the constrained policy optimization problem proposed in TRPO and a partial Lagrangian reformulation of such problem inspired by the advantage-weighted regression (AWR, see Peng et al. (2019)) method. We provide closed-form policy solution of the partial Lagrangian (see Theorem 1), and such a solution is exact and takes the normalization factor of the optimal policy into account (which is a departure from Peng et al. (2019), see Remark 2);

• We propose a novel iterative Successive Policy Re-weighting method (SPR, Algorithm 1). SPR utilizes an explicit reward to learn the Q-value function and the log-sum-exp value function, then learns the policy through re-weighting the policy probability based on the Q-value function. SPR only loads one (value or policy) model at every step, achieving policy optimization with the **same memory consumption as standard supervised fine-tuning (SFT)** (see Figure 1). We thus believe that the proposed SPR can serve as a powerful alternative to the memory-costly PPO for the RLHF pipeline when aligning LLMs.

• We conduct extensive numerical experiments to verify the effectiveness of the proposed method. Specifically, we compare the SPR with standard PPO and DPO, as well as a Best-of-N algorithm (Dong et al., 2023) on both 1b and 8b models. We compare both the reward value and win-rate of 1b model trained by different methods on a text-summarization dataset, and evaluate the 8b model trained by different methods on OpenLLM leaderboard. All our results show the superior performance of the proposed method over standard baselines.

2 Methodology

We model the LLM as a policy π in a Markov decision process (MDP). A MDP is defined by the tuple $(\mathcal{S}, \mathcal{A}, P, \mu, r, \gamma)$, which consists of the state space $\mathcal{S}$, the action space $\mathcal{A}$, the transition dynamics $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$, the initial state distribution $\mu(\cdot)$, the reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ and the discounted factor $\gamma \in (0, 1)$. Under a transition dynamics model P and a policy $\pi : \mathcal{S} \to \Delta_{\mathcal{A}}$ where $\Delta_{\mathcal{A}}$ is the probability simplex on the action space, further define the corresponding state visitation measure as $d_\pi(s) := (1 - \gamma) \sum_{t=0}^{T} \gamma^t P^\pi(s_t = s | s_0 \sim \eta)$ for any state $s \in \mathcal{S}$. Here T is horizon size, which can be any positive integer or ∞.

In the LLM context, $\mathcal{S}$ and $\mathcal{A}$ correspond to the space of input prompts and the space of output continuations, respectively. If we consider the entire input sentences as state space and the entire output continuations as action space, then there is no transition dynamics, i.e. horizon = 1. On the other hand, if we consider the token-level state/action space[2], i.e. the action is modeled as the next token in the sentence, the transition dynamics is deterministic and $P(s'|s, a)$ is always 1 if $s' = (s, a)$ and 0 otherwise. The reward model r is usually another LLM trained specifically to evaluate the score of given prompt/continuation tuples, and the policy model π is the LLM to be optimized.

Let us consider the most general case where the horizon T is ∞. We inspect the original policy optimization problem by dropping the KL-divergence constraint in (21) and obtain:

$$J(\pi) := \mathbb{E}_{\tau \sim \pi} \Big[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \Big], \tag{1}$$

where $\tau := (s_0, a_0, s_1, a_1, \cdots)$ denotes one trajectory, corresponding to one data point with prompt(s) and continuation(s). Under a policy/LLM π, we can define the corresponding value function V^π and the Q-function Q^π as below:

$$V^\pi(s) := \mathbb{E}_{\tau \sim \pi} \Big[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid s_0 = s \Big], \tag{2a}$$

$$Q^\pi(s, a) := r(s, a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} \big[V^\pi(s') \big]. \tag{2b}$$

We can further define the advantage function for each state action pair (s, a) as follows:

$$A^\pi(s, a) := Q^\pi(s, a) - V^\pi(s). \tag{3}$$

The fundamental idea of policy improvement is that, suppose there is a reference policy π', we can maximize the performance gap over the reference policy to achieve policy improvement:

$$\eta_{\pi'}(\pi) := J(\pi) - J(\pi'). \tag{4}$$

It turns out that the performance improvement of the policy π over the reference policy π' can be expressed by the advantage function $A^{\pi'}(s, a)$.

Lemma 1 (Lemma 1.16 in (Agarwal et al., 2019))**.** *For any policy π and π', the performance difference can be expressed as below:*

$$\eta_{\pi'}(\pi) = \frac{1}{1 - \gamma} \mathbb{E}_{s \sim d_\pi(\cdot), a \sim \pi(\cdot|s)} \big[A^{\pi'}(s, a) \big] \tag{5}$$

where $d_\pi(s) := (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t P^\pi(s_t = s | s_0 \sim \eta)$ denotes the state visitation measure under the policy π and $\mu(\cdot)$ denotes the initial state distribution.

In the LLM context, this performance difference lemma indicates that to align the LLM with the reward model, i.e. to maximize the policy improvement objective in (4), one just need to seek for a policy π which induces positive expected advantage $\mathbb{E}_{s \sim d_\pi(\cdot), a \sim \pi(\cdot|s)} \big[A^{\pi'}(s, a) \big] > 0$ over the reference policy π'. Therefore, we focus on maximizing (5). However, from a practical point of view, the dependency on sampling data from the visitation measure $d_\pi(\cdot)$ makes it difficult to optimize the

[2]The similar modeling can be considered if we model a dialogue as a sequence of state (questions) and actions (answers).

performance difference defined in (5). Following the trust region policy optimization (TRPO, see Schulman et al. (2015)) framework, we instead consider an approximation to $\eta_{\pi'}(\pi)$ by $\tilde{\eta}_{\pi'}(\pi)$:

$$\tilde{\eta}_{\pi'}(\pi) = \frac{1}{1-\gamma}\mathbb{E}_{s\sim d_{\pi'}(\cdot),a\sim\pi(\cdot|s)}\left[A^{\pi'}(s,a)\right] \tag{6}$$

where $d_{\pi'}(\cdot)$ denotes the state visitation measure under the reference policy π'. According to Theorem 1 in (Schulman et al., 2015), $\tilde{\eta}_{\pi'}(\pi)$ serves as a good approximation to $\eta_{\pi'}(\pi)$ when the two policies π and π' are close in terms of the KL-divergence. Thus, when maximizing the surrogate objective $\tilde{\eta}_{\pi'}(\pi)$ defined in (6) while penalizing the KL divergence between π and π', we are able to guarantee monotonic performance improvement at each policy iteration step.

Above discussions lead to the following optimization objective, which is also used in the TRPO: at each policy iteration step and given the previous policy π_{old}, one solves for the constrained policy optimization problem:

$$\max_{\pi} \quad \tilde{\eta}_{\pi_{\text{old}}}(\pi) \tag{7a}$$

$$s.t. \quad \mathbb{E}_{s\sim d_{\pi_{\text{old}}}(\cdot)}\left[D_{\text{KL}}\big(\pi(\cdot|s)||\pi_{\text{old}}(\cdot|s)\big)\right] \leq \epsilon, \tag{7b}$$

$$\sum_{a\in\mathcal{A}} \pi(a|s) = 1, \forall s \in \mathcal{S}, \tag{7c}$$

$$\pi(a|s) \geq 0, \forall s \in \mathcal{S}, a \in \mathcal{A}. \tag{7d}$$

In the following theorem, whose proof is provided in the Appendix C, we show the closed-form expression of the optimal policy.

Theorem 1. *The optimal policy π^* from (7a)-(7d) is*

$$\pi^*(a|s) = \pi_{\text{old}}(a|s)\exp\left(\frac{1}{\beta}\big(Q^{\pi_{\text{old}}}(s,a) - W^{\pi_{\text{old}}}(s)\big)\right) \tag{8}$$

where $\beta := \frac{1}{(1-\gamma)\alpha}$ and $W^{\pi_{\text{old}}}(s) := \beta\log\left(\mathbb{E}_{a\sim\pi_{\text{old}}(\cdot|s)}\left[\exp\left(\frac{1}{\beta}Q^{\pi_{\text{old}}}(s,a)\right)\right]\right)$ is defined as a reference function dependent on the state s, also known as the log-sum-exp value function.

Remark 1. Note that if we replace $W^{\pi_{\text{old}}}(s)$ by the value function $V^{\pi_{\text{old}}}(s) := \mathbb{E}_{a\sim\pi_{\text{old}}(\cdot|s)}Q^{\pi_{\text{old}}}(s,a)$, we arrive at Peng et al. (2019, equation (8)): $\pi^*(a|s) = \frac{1}{Z(s)}\pi_{\text{old}}(a|s)\exp\left(\frac{1}{\beta}\big(Q^{\pi_{\text{old}}}(s,a) - V^{\pi_{\text{old}}}(s)\big)\right)$ where $Z(s)$ is the partition function to normalize the policy. $Z(s)$ is not negligible when estimating the optimal policy, which motivates us to use the log-sum-exp function $W^{\pi_{\text{old}}}(s)$ instead. Similar trick can be observed in maximum entropy RL literature such as Garg et al. (2022). Theorem 1 frees us from estimating the impractical partition function $Z(s)$ and opens the gate for efficient algorithm for solving for the optimal policy, by solving the Q and W functions in (8) respectively.

3 Algorithm Design

In this section, we design a memory-efficient algorithm to approximate the optimal policy π^* in (8).

Algorithm Derivation. Now suppose we parameterize our policy by parameter θ in practice. Since the optimal policy for the constrained optimization problem defined in (7a)-(7d) enjoys a closed-form solution in (8), we directly approximate the optimal policy π^* corresponding to a reference policy π_{old} through solving the following KL divergence minimization problem:

$$\min_{\theta} \mathbb{E}_{s\sim d_{\pi_{\text{old}}}(\cdot)}\left[D_{\text{KL}}\big(\pi^*(\cdot|s)||\pi_\theta(\cdot|s)\big)\right]. \tag{9}$$

Based on the definition of the KL divergence $D_{\text{KL}}\big(\pi^*(\cdot|s)||\pi_\theta(\cdot|s)\big) = \mathbb{E}_{a\sim\pi^*(\cdot|s)}\left[\log\frac{\pi^*(a|s)}{\pi_\theta(a|s)}\right]$, we can obtain the following relations:

$$\mathbb{E}_{s\sim d_{\pi_{\text{old}}}(\cdot)}\left[D_{\text{KL}}\big(\pi^*(\cdot|s)||\pi_\theta(\cdot|s)\big)\right] = \mathbb{E}_{s\sim d_{\pi_{\text{old}}}(\cdot),a\sim\pi^*(\cdot|s)}\left[\log\pi^*(a|s) - \log\pi_\theta(a|s)\right]. \tag{10}$$

Eq. (10) implies that minimizing the KL divergence between π^* and π_θ is equivalent to solving the following maximum likelihood estimation (MLE) problem:

$$\max_\theta \; L_{\mathrm{MLE}}(\theta) := \mathbb{E}_{s\sim d_{\pi_{\mathrm{old}}}(\cdot), a\sim\pi^*(\cdot|s)}\big[\log \pi_\theta(a|s)\big]. \tag{11}$$

This implies that solving (11) is equivalent to solving (9), and the resulting policy π_θ approximates π^*. However, one critical issue in solving the MLE problem is that we are not able to sample actions from the optimal policy π^* even though the closed-form expression of π^* is available in (8).

In order to resolve this issue, we leverage the technique from importance sampling where we sample actions from the existing policy π_{old} and then weigh each sample (s, a) by its importance weight $\frac{\pi^*(a|s)}{\pi_{\mathrm{old}}(a|s)}$. Towards this end, we write:

$$L_{\mathrm{MLE}}(\theta) = \mathbb{E}_{s\sim d_{\pi_{\mathrm{old}}}(\cdot), a\sim\pi^*(\cdot|s)}\big[\log \pi_\theta(a|s)\big] = \mathbb{E}_{s\sim d_{\pi_{\mathrm{old}}}(\cdot), a\sim\pi_{\mathrm{old}}(\cdot|s)}\Big[\frac{\pi^*(a|s)}{\pi_{\mathrm{old}}(a|s)}\log \pi_\theta(a|s)\Big]. \tag{12}$$

Leveraging the relation between π^* and π_{old} as shown in (8), $L_{\mathrm{MLE}}(\cdot)$ can be rewritten as:

$$\begin{aligned} L_{\mathrm{MLE}}(\theta) &= \mathbb{E}_{s\sim d_{\pi_{\mathrm{old}}}(\cdot), a\sim\pi_{\mathrm{old}}(\cdot|s)}\Big[\frac{\pi^*(a|s)}{\pi_{\mathrm{old}}(a|s)}\log \pi_\theta(a|s)\Big] \\ &= \mathbb{E}_{s\sim d_{\pi_{\mathrm{old}}}(\cdot), a\sim\pi_{\mathrm{old}}(\cdot|s)}\Big[\exp\Big(\frac{1}{\beta}\big(Q^{\pi_{\mathrm{old}}}(s,a) - W^{\pi_{\mathrm{old}}}(s)\big)\Big)\log \pi_\theta(a|s)\Big]. \end{aligned} \tag{13}$$

With the state-action pairs (s, a) sampled from the policy π_{old}, we approximate the corresponding optimal policy through optimizing the supervised learning objective in (13). In the LLM context, (13) can also be viewed as a *re-weighting* process, where we increase or decrease the log likelihood of the previous policy π_{old} according to the approximate advantage function $Q^{\pi_{\mathrm{old}}} - W^{\pi_{\mathrm{old}}}$, i.e. we increase the likelihood of actions/continuations with higher rewards than the baseline and decrease the ones with lower reward values.

It remains to estimate the Q-function $Q^{\pi_{\mathrm{old}}}(s,a)$ and the reference function $W^{\pi_{\mathrm{old}}}(s)$. To estimate $Q^{\pi_{\mathrm{old}}}(s,a)$ under a policy π_{old}, we solve the fixed point of the Bellman operator $\mathcal{B}$:

$$\mathcal{B}^{\pi_{\mathrm{old}}}Q(s,a) = r(s,a) + \gamma\mathbb{E}_{s'\sim P(\cdot|s,a), a'\sim\pi_{\mathrm{old}}(\cdot|s')}\big[Q(s',a')\big]. \tag{14}$$

Following Fujimoto et al. (2018); Haarnoja et al. (2018), we use the temporal difference learning to minimize Bellman error to approximate $Q^{\pi_{\mathrm{old}}}(s,a)$ by a parameterized Q-function $Q_\phi(s,a)$.

Next, let us estimate the reference function $W^{\pi_{\mathrm{old}}}(s)$ under the policy π_{old}. Based on the definition of $W^{\pi_{\mathrm{old}}}(s)$ in (8), a naive way is to construct empirical estimations to approximate the reference function. For each state s, one can sample a small batch of N actions from policy $\pi_{\mathrm{old}}(\cdot|s)$ and then construct the empirical estimate $\widehat{W}^{\pi_{\mathrm{old}}}(s)$ defined as below:

$$\widehat{W}^{\pi_{\mathrm{old}}}(s) := \beta\log\Big(\sum_{i=1}^{N}\exp\big(\frac{1}{\beta}Q^{\pi_{\mathrm{old}}}(s,a_i)\big)\Big). \tag{15}$$

To obtain an accurate estimate of the reference function $W^{\pi_{\mathrm{old}}}(s)$, it is necessary to make the batch-size N large enough, which is computationally expensive.

The drawback of the empirical estimator in (15) motivates us to estimate the reference function $W^{\pi_{\mathrm{old}}}(s)$ through parameterized approximations. We observe that the reference function $W^{\pi_{\mathrm{old}}}(s)$ shares the same expression to the fitted parameter of Gumbel distribution by maximum likelihood estimation. According to Coles et al. (2001); Forbes et al. (2011); Garg et al. (2022), when we model a random variable x by Gumbel distribution $x = h - \epsilon$ where $\epsilon \sim \mathcal{G}(0, \beta)$ is a Gumbel noise, the log-likelihood has the following expression:

$$\mathbb{E}_x\Big[\log p(x; h, \beta)\Big] = \mathbb{E}_x\Big[\log\Big(\frac{1}{\beta}\exp\big(\frac{x-h}{\beta} - \exp(\frac{x-h}{\beta})\big)\Big)\Big]. \tag{16}$$

To fit the location parameter h through maximum likelihood estimation, we can minimize the following loss function:

$$\min_h \; f(h) := \mathbb{E}_x\Big[\exp\big(\frac{x-h}{\beta}\big) - \frac{x-h}{\beta} - 1\Big] \tag{17}$$

Algorithm 1 *Successive Policy Re-weighting (SPR)*

```
1: Input: Collect an demonstration dataset $\mathcal{D} = \{(s, a, s')\}$ and a given reward function $r(\cdot, \cdot)$.
2: Initialize the parameterized policy (LLM), Q-function and reference function as $\pi_\theta$, $Q_\phi$ and $W_\varphi$
3: for $k = 0, 1, \ldots, K-1$ do
4:     $\phi_{k+1} = \arg\min_\phi \ \mathbb{E}_{(s,a,s') \sim \mathcal{D}, a' \sim \pi_{\theta_k}(\cdot|s')} \Big[\big(Q_\phi(s,a) - r(s,a) - \gamma Q_{\bar{\phi}_k}(s', a')\big)^2\Big]$
5:     $\varphi_{k+1} = \arg\min_\varphi \ \mathbb{E}_{s \sim \mathcal{D}, a \sim \pi_{\theta_k}(\cdot|s)} \Big[\exp\Big(\frac{Q_{\phi_{k+1}}(s,a) - W_\varphi(s)}{\beta}\Big) - \frac{Q_{\phi_{k+1}}(s,a) - W_\varphi(s)}{\beta} - 1\Big]$
6:     $\theta_{k+1} = \arg\min_\theta \ \mathbb{E}_{s \sim \mathcal{D}, a \sim \pi_{\theta_k}(\cdot|s)} \Big[\exp\Big(\frac{1}{\beta}\big(Q_{\phi_{k+1}}(s,a) - W_{\varphi_{k+1}}(s)\big)\Big) \log \pi_\theta(a|s)\Big]$
7:     $\bar{\phi}_{k+1} = \alpha \phi_{k+1} + (1-\alpha)\bar{\phi}_k$
8:     if $k >$ sampling threshold then
9:         sample trajectories $\tau_1, \cdots, \tau_N$ from the current policy $\pi_{\theta_{k+1}}$
10:        add trajectories to the data buffer: $\mathcal{D} \leftarrow \mathcal{D} \cup \{\tau_1, \cdots, \tau_N\}$
11:    end if
12: end for
```

where the loss function $f(h)$ shares similar form to the Linex loss in the literature of econometrics (Parsian & Kirmani, 2002; Chang & Hung, 2007). Through solving the loss function $f(h)$ defined in (17), we obtain the maximum likelihood estimator of the location parameter as $\hat{h} = \beta \log \mathbb{E}_x\big[\exp\big(\frac{x}{\beta}\big)\big]$. Here, we observe that the estimate $\hat{h}$ shares the same expression to the reference function $W^{\pi_{\text{old}}}(s)$ defined in (8). This shows that for a fixed state s, the value of $W^{\pi_{\text{old}}}(s)$ solves the following optimization problem:

$$W^{\pi_{\text{old}}}(s) = \arg\min_w \ \mathbb{E}_{a \sim \pi_{\text{old}}(\cdot|s)}\Big[\exp\Big(\frac{Q^{\pi_{\text{old}}}(s,a) - w}{\beta}\Big) - \frac{Q^{\pi_{\text{old}}}(s,a) - w}{\beta} - 1\Big]. \quad (18)$$

Therefore, given a parameterized Q-function $Q_\phi(s,a)$ which approximates the exact Q-function $Q^{\pi_{\text{old}}}(s,a)$, we can train a parameterized reference function $W_\varphi(s)$ to approximate $W^{\pi_{\text{old}}}(s)$ for all states $s \in \mathcal{S}$ through optimizing the following loss function:

$$\min_\varphi \ L(\varphi) := \mathbb{E}_{a \sim \pi_{\text{old}}(\cdot|s)}\Big[\exp\Big(\frac{Q_\phi(s,a) - W_\varphi(s)}{\beta}\Big) - \frac{Q_\phi(s,a) - W_\varphi(s)}{\beta} - 1\Big]. \quad (19)$$

This completes our derivation of the algorithm. Now, we are ready to summarize our proposed Successive Policy Re-weighting (SPR) algorithm in Alg. 1. We solve the value function Q (parameterized by ϕ) and reference function W (parameterized by φ) by solving (14) and (19), respectively. Then we can solve for the policy/LLM π_θ by minimizing the re-weighting loss (13).

Algorithm Simplification. Note that if we model the entire input prompt and output continuation as the state and action respectively, the horizon will be automatically reduced to one. In this case, the Q value function reduces to the reward function r and line 4 of Algorithm 1 is discarded immediately, also W in (15) becomes the log-sum-exp of the reward function r. The resulting *simplified* algorithm is presented in Algorithm 2, which can be implemented for LLM alignment. Notably, both Algorithm 1 and 2 only require a single model to be loaded in the memory for each of it update step.

Discussion. SPR significantly decreases the memory consumption of PPO by reducing the number of models loaded in the memory from four to one, matching the memory requirement of standard SFT. The proposed method saves more memory comparing to the DPO since DPO requires loading a froze reference model during the training step, while SPR only loads one model throughout its algorithm process; On the other hand, SPR still allows online data and explicit reward feedback through iterative sampling-and-fitting process, greatly increase its potential in alignment comparing to offline-data-only method such as SFT or DPO. The memory and time consumption of SPR comparing with related methods are summarized in Figure 1.

4 Experimental Results

4.1 Experiment Setup

Model and Datasets. We conduct our experiment in two settings: First, we test with `EleutherAI/pythia-1b-deduped` model over `TL;DR` dataset; Second, we train with `LLaMA3-SFT` model over a mixture of various prompt datasets, including UltraFeedback Cui et al. (2023), HelpSteer

Algorithm 2 *Successive Policy Re-weighting (SPR) for LLM alignment*

```
1: Input: Collect an demonstration dataset $\mathcal{D} = \{(s, a)\}$ and a given reward function $r(\cdot, \cdot)$.
2: Initialize the parameterized policy (LLM), Q-function and reference function as $\pi_\theta$, $Q_\phi$ and $W_\varphi$
3: for $k = 0, 1, \ldots, K-1$ do
4:     $\varphi_{k+1} = \arg\min_\varphi \ \mathbb{E}_{s\sim\mathcal{D}, a\sim\pi_{\theta_k}(\cdot|s)} \Big[\exp\Big(\frac{r(s,a)-W_\varphi(s)}{\beta}\Big) - \frac{r(s,a)-W_\varphi(s)}{\beta} - 1\Big]$
5:     $\theta_{k+1} = \arg\min_\theta \ \mathbb{E}_{s\sim\mathcal{D}, a\sim\pi_{\theta_k}(\cdot|s)} \Big[\exp\Big(\frac{1}{\beta}\big(r(s,a) - W_{\varphi_{k+1}}(s)\big)\Big)\log\pi_\theta(a|s)\Big]$
6:     if $k >$ sampling threshold then
7:         sample prompt-continuation pairs $(s_1, a_1), \cdots, (s_N, a_N)$ from the current policy $\pi_{\theta_{k+1}}$
8:         add prompt-continuation pairs to the data buffer: $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_1, a_1), \cdots, (s_N, a_N)\}$
9:     end if
10: end for
```

Wang et al. (2023) and so on (see Dong et al. (2024) for the details of the training data). We test the performance of the model over various downstream tasks.

Evaluation. For the first experiment setting, we use a model trained from `Pythia-6.9b` with `TL;DR` dataset[3] as the reward evaluator. Note that the experiment is conducted on 1b model and we believe that this 6.9b model is capable of serving as the ground truth for this text summarizing task. We also record the win rate of the model generated summary against the ground truth summary, evaluated by GPT-3.5-turbo-0125.

For the second experiment setting, following Chen et al. (2024); Li et al. (2024), we evaluated the aligned model trained from `LLaMA3-SFT` over the Open LLM Leaderboard (Beeching et al., 2023). Open LLM Leaderboard involves various downstream tasks to test the performance of LLM through different dimensions, where we specifically evaluate an LLM based on six tasks: commonsense reasoning (Arc Clark et al. (2018), HellaSwag Zellers et al. (2019), Winogrande Sakaguchi et al. (2021)), multi-task language understanding (MMLU Hendrycks et al. (2020)), human falsehood mimic (TruthfulQA Lin et al. (2021)) and math problem solving (GSM8K, Cobbe et al. (2021)).

Implementation detail. Instead of training a reference function as in step 4 of Algorithm 2, we use the empirical estimate $\widehat{W}^{\pi_{\text{old}}}(s)$ in (15) by generating N responses based on the current policy, due to its efficiency in practice. In our experiment, considering the time efficiency and performance limitations due to the scaling law Kaplan et al. (2020), we split the whole training `TL;DR` dataset of evenly into 10 pieces, so 11.7K prompts are used to generate N independent responses for training in each iteration, and iterate the training dataset for 3 epochs on each data pieces. For the 8b model, we used 20K prompts per iteration and trained for 2 epochs on each data pieces. When we exhaust all the data, we restart from the first split of the dataset[4]. We refer to Appendix D for more details.

4.2 Results on 1b models

We present the performance of `EleutherAI/pythia-1b-deduped` model on four different algorithms, including SPR (Algorithm 2), Best-of-N (Dong et al. (2023)), DPO[5] and PPO[6]. We use the SFT pythia-1b model[7] as the initial model. Here Best-of-N refers to selecting the sample with highest score, as evaluated by the 6.9b reward model, to train the policy model, which serves as a strong baseline.

In Fig. 2, it can be seen that both SPR and Best-of-N algorithm outperform DPO in terms of reward score. However, while the Best-of-N algorithm struggles to consistently surpass DPO in win rate, SPR still demonstrates strong performance in win rate against reference summaries. Furthermore, as N increases, both Best-of-N and SPR achieve higher performance, attributed to a more accurate estimation of the reference function and improved sample quality. Although the reward score of all four difference algorithms continue to increase, the win rate essentially converges to a stationary level.

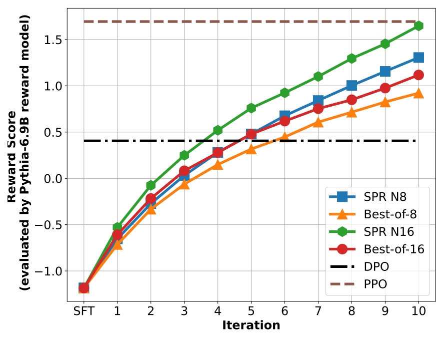
[3]see this link for the detail of how to obtain this reward model from the base model of `EleutherAI/pythia-6.9b-deduped`.

[4]For example, in Fig. 2, we exhaust all the training data when the "Iteration" on x-axis equals to 10.

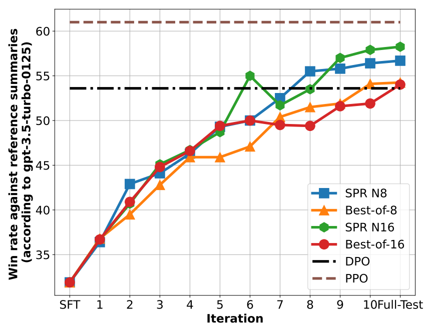
[5]We use this pythia-1b DPO checkpoint.

[6]We use this pythia-1b PPO checkpoint.

[7]We use this pyhia-1b SFT checkpoint



(a) Reward score



(b) Win rate against reference summaries

Figure 2: Left: The reward scores evaluated by Pythia-6.9b reward model trained on the TL;DR dataset on win rate of SPR algorithm on Pythia-1b model through the entire train split of the TL;DR dataset; Right: The win rate of our models' summaries over the human-generated reference summaries on the first 1000 test split of the TL;DR dataset, judged by GPT 3.5, "Full-Test" means evaluated on the whole test dataset. The x-axis represents the iteration number, each iteration uses 1/10 training data. To ensure fairness in testing, responses are generated using greedy search.

4.3 Results on 8b models

In Figure 4 and Table 1, we compare the performance of our fine-tuned model by SPR algorithm with the base model LLaMA3-SFT on each task included in the leaderboard. Our SPR algorithm demonstrates superior performance compared to the Best-of-N approach after five iterations. In both cases, SPR effectively improves the performance of SFT-ed model and the average performance, surpassing the performance of the DPO easily. Notably, our algorithm shows strong potential on the improving the LLM's ability of solving mathematical problem, with an almost 6% improvement on the GSM8K task. In Figure 3, we show the average score of SPR algorithm with $N = 32$ compared to the DPO baseline over a total of 7 iterations. The results demonstrate a progressive increase in the average score, ranging from 68.66 to 70.65, indicating the effectiveness of our algorithm.

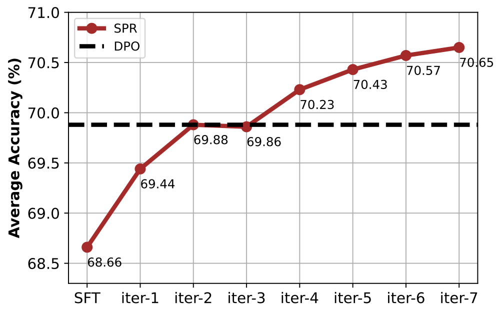


Figure 3: The average score of SPR with $N = 32$ and DPO baseline at different iterations on the Open LLM leaderboard datasets.

Tasks Metrics	Arc Challenge acc_norm	TruthfulQA MC2 acc	Winogrande acc	GSM8k strict-match	HellaSwag acc_norm	MMLU	Average
LLaMA3-SFT	62.29%	53.49%	**78.14%**	72.55%	81.03%	64.49%	68.66%
LLaMA3-DPO	**65.36%**	**60.02%**	77.43%	70.96%	81.56%	63.95%	69.88%
Best-of-16	63.32%	56.03%	78.13%	75.97%	81.36%	65.22%	70.01%
SPR-16	63.57%	56.14%	77.74%	**76.12%**	**81.68%**	**65.34%**	**70.10%**

Table 1: Performance of Policy in Open LLm Leaderboard for four different algorithms. SPR and Best-of-N algorithm with $N = 16$ are run over five iterations.

5 Conclusions and Limitations

In this paper we proposed a memory-efficient policy optimization approach named successive policy re-weighting (SPR), matching the peak memory consumption of standard supervised fine-tune (SFT). SPR is flexible for offline and online policy training and achieves state-of-the-art performance on our experiment for aligning 1b and 8b models. The computational time of the proposed method is in

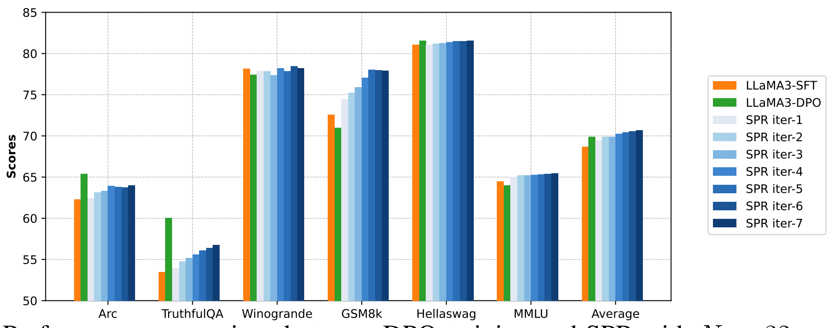


Figure 4: Performance comparison between DPO training and SPR with $N = 32$ across the six benchmark datasets.

general higher than SFT or DPO, and the performance is largely dependent on the number of samples N we use to estimate the reference function in (15). Future works include tuning SPR pipeline to make it work for very large LLMs (>50b) and exploring alignment methods that are both time- and memory-efficient.

References

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

Alekh Agarwal, Nan Jiang, Sham M Kakade, and Wen Sun. Reinforcement learning: Theory and algorithms. *CS Dept., UW Seattle, Seattle, WA, USA, Tech. Rep*, 32, 2019.

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms. *arXiv preprint arXiv:2402.14740*, 2024.

Anthropic. Claude 3 haiku: our fastest model yet. `https://www.anthropic.com/news/claude-3-haiku`, 2024.

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

Philip J Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. *arXiv preprint arXiv:2302.02948*, 2023.

Edward Beeching, Clémentine Fourrier, Nathan Habib, Sheon Han, Nathan Lambert, Nazneen Rajani, Omar Sanseviero, Lewis Tunstall, and Thomas Wolf. Open llm leaderboard. `https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard`, 2023.

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

Ronen I Brafman and Moshe Tennenholtz. R-max - a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3(Oct):213–231, 2002.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.

Yen-Chang Chang and Wen-Liang Hung. Linex loss functions with applications to determining the optimum process parameters. *Quality & Quantity*, 41:291–301, 2007.

Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. Self-play fine-tuning converts weak language models to strong language models. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 6621–6642. PMLR, 21–27 Jul 2024. URL `https://proceedings.mlr.press/v235/chen24j.html`.

Ching-An Cheng, Tengyang Xie, Nan Jiang, and Alekh Agarwal. Adversarially trained actor critic for offline reinforcement learning. In *International Conference on Machine Learning*, pp. 3852–3878. PMLR, 2022.

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

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

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

Stuart Coles, Joanna Bawa, Lesley Trenner, and Pat Dorazio. *An introduction to statistical modeling of extreme values*, volume 208. Springer, 2001.

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

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022.

Hanze Dong, Wei Xiong, Deepanshu Goyal, Yihan Zhang, Winnie Chow, Rui Pan, Shizhe Diao, Jipeng Zhang, KaShun SHUM, and Tong Zhang. RAFT: Reward rAnked FineTuning for Generative Foundation Model Alignment. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL `https://openreview.net/forum?id=m7p5O7zblY`.

Hanze Dong, Wei Xiong, Bo Pang, Haoxiang Wang, Han Zhao, Yingbo Zhou, Nan Jiang, Doyen Sahoo, Caiming Xiong, and Tong Zhang. RLHF workflow: From reward modeling to online RLHF. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL `https://openreview.net/forum?id=a13aYUU9eU`.

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

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. Model alignment as prospect theoretic optimization. In *Forty-first International Conference on Machine Learning*, 2024.

Catherine Forbes, Merran Evans, Nicholas Hastings, and Brian Peacock. *Statistical distributions*. John Wiley & Sons, 2011.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pp. 1587–1596. PMLR, 2018.

Divyansh Garg, Joey Hejna, Matthieu Geist, and Stefano Ermon. Extreme q-learning: Maxent rl without entropy. In *The Eleventh International Conference on Learning Representations*, 2022.

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, et al. Reinforced Self-training (ReST) for Language Modeling. *arXiv preprint arXiv:2308.08998*, 2023.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Shengyi Huang, Michael Noukhovitch, Arian Hosseini, Kashif Rasul, Weixun Wang, and Lewis Tunstall. The n+ implementation details of rlhf with ppo: A case study on tl; dr summarization. *arXiv preprint arXiv:2403.17031*, 2024.

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

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-based offline reinforcement learning. *Advances in neural information processing systems*, 33:21810–21823, 2020.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2021.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pp. 611–626, 2023.

Jiaxiang Li, Siliang Zeng, Hoi-To Wai, Chenliang Li, Alfredo Garcia, and Mingyi Hong. Getting more juice out of the sft data: Reward learning from human demonstration improves sft for llm alignment. *arXiv preprint arXiv:2405.17888*, 2024.

Ziniu Li, Tian Xu, and Yang Yu. Policy optimization in RLHF: The impact of out-of-preference data. *arXiv preprint arXiv:2312.10584*, 2023a.

Ziniu Li, Tian Xu, Yushun Zhang, Zhihang Lin, Yang Yu, Ruoyu Sun, and Zhi-Quan Luo. Remax: A simple, effective, and efficient reinforcement learning method for aligning large language models. In *Forty-first International Conference on Machine Learning*, 2023b.

Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods. *arXiv preprint arXiv:2109.07958*, 2021.

Yong Lin, Skyler Seto, Maartje ter Hoeve, Katherine Metcalf, Barry-John Theobald, Xuan Wang, Yizhe Zhang, Chen Huang, and Tong Zhang. On the limited generalization capability of the implicit reward model induced by direct preference optimization. *arXiv preprint arXiv:2409.03650*, 2024.

Fei Liu et al. Learning to summarize from human feedback. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

Yao Liu, Adith Swaminathan, Alekh Agarwal, and Emma Brunskill. Off-policy policy gradient with state distribution correction. *arXiv preprint arXiv:1904.08473*, 2019.

Cong Lu, Philip Ball, Jack Parker-Holder, Michael Osborne, and Stephen J Roberts. Revisiting design choices in offline model based reinforcement learning. In *International Conference on Learning Representations*, 2021.

Ofir Nachum, Bo Dai, Ilya Kostrikov, Yinlam Chow, Lihong Li, and Dale Schuurmans. Algaedice: Policy gradient from arbitrary experience. *arXiv preprint arXiv:1912.02074*, 2019.

Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

Takayuki Osa, Joni Pajarinen, Gerhard Neumann, J Andrew Bagnell, Pieter Abbeel, Jan Peters, et al. An algorithmic perspective on imitation learning. *Foundations and Trends® in Robotics*, 7(1-2): 1–179, 2018.

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

Ahmad Parsian and SNUA Kirmani. Estimation under linex loss function. In *Handbook of applied econometrics and statistical inference*, pp. 75–98. CRC Press, 2002.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1, 1988.

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

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–16, 2020. doi: 10.1109/SC41405.2020.00024.

Rajkumar Ramamurthy, Prithviraj Ammanabrolu, Kianté Brantley, Jack Hessel, Rafet Sifa, Christian Bauckhage, Hannaneh Hajishirzi, and Yejin Choi. Is reinforcement learning (not) for natural language processing: Benchmarks, baselines, and building blocks for natural language policy optimization. *arXiv preprint arXiv:2210.01241*, 2022.

Marc Rigter, Bruno Lacerda, and Nick Hawes. Rambo-rl: Robust adversarial model-based offline reinforcement learning. *arXiv preprint arXiv:2204.12581*, 2022.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897. PMLR, 2015.

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, YK Li, Yu Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Trieu H Trinh, Yuhuai Wu, Quoc V Le, He He, and Thang Luong. Solving olympiad geometry without human demonstrations. *Nature*, 625(7995):476–482, 2024.

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

Masatoshi Uehara and Wen Sun. Pessimistic model-based offline reinforcement learning under partial coverage. In *International Conference on Learning Representations*, 2021.

Zhilin Wang, Yi Dong, Jiaqi Zeng, Virginia Adams, Makesh Narsimhan Sreedhar, Daniel Egert, Olivier Delalleau, Jane Polak Scowcroft, Neel Kant, Aidan Swope, et al. Helpsteer: Multi-attribute helpfulness dataset for steerlm. *arXiv preprint arXiv:2311.09528*, 2023.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992.

Shusheng Xu, Wei Fu, Jiaxuan Gao, Wenjie Ye, Weilin Liu, Zhiyu Mei, Guangju Wang, Chao Yu, and Yi Wu. Is DPO superior to PPO for LLM alignment? a comprehensive study. In *Forty-first International Conference on Machine Learning*, 2024. URL `https://openreview.net/forum?id=6XH8R7YrSk`.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *Advances in Neural Information Processing Systems*, 33:14129–14142, 2020.

Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. Combo: Conservative offline model-based policy optimization. *Advances in neural information processing systems*, 34:28954–28967, 2021.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

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

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

Appendix

A Related Work

In this section, we discuss related work on both the LLMs and reinforcement learning with offline datasets.

RL for LLMs alignment. RL-based methods have show great success in improving the abilities of LLMs, including text-summarizing (Liu et al., 2020; Ziegler et al., 2019), story-telling (Ziegler et al., 2019), instruction-following (Ouyang et al., 2022; Ramamurthy et al., 2022), problem-solving (Trinh et al., 2024), etc. The most popular RL pipeline for LLM alignment is RLHF (Ouyang et al., 2022; Christiano et al., 2017), which models the reward using the popular Bradley-Terry model (Bradley & Terry, 1952) and optimize the policy by policy optimization methods, such as REINFORCE (Williams, 1992; Ahmadian et al., 2024; Li et al., 2023b), PPO (Schulman et al., 2017), GRPO (Shao et al., 2024) and R-max (Brafman & Tennenholtz, 2002). On the other hand, the extra computational cost introduced by RL-based method is not negligible. DPO (Rafailov et al., 2024) serves as a powerful substitute for RL-based method by directly optimizing for the LLM/policy (and the reward is implicitly represented by the policy), making the memory consumption as low as the standard SFT. However the absence of reward model in DPO pipeline is drawing discussions on its limited generalization ability in the LLM alignment community (Li et al., 2023a; Xu et al., 2024). Standard DPO utilizes an offline pre-collected preference dataset. Several other methods, such as KTO (Ethayarajh et al., 2024), sequence likelihood calibration (SLiC, see Zhao et al. (2023)) and identity preference optimization (IPO, see Azar et al. (2024)), also lie in the scope of offline preference data alignment. On the other hand, methods like online DPO (Dong et al., 2024) and Reinforced Self-Training (ReST, see Gulcehre et al. (2023)) utilize new samples with higher rewards from current model for further policy improvements.

Offline RL. Offline RL considers the problem of learning a policy from a fixed datasets where the reward value is provided for each collected transition samples. In (Liu et al., 2019; Nachum et al., 2019), model-free offline RL algorithms are proposed to solve the importance sampling problem. In (Kumar et al., 2020; Cheng et al., 2022), conservatism is incorporated into the value function to avoid overestimation in the offline RL setting. In (Kostrikov et al., 2021), a general algorithm for offline RL is proposed which can avoid any queries to values of out-of-sample actions during training while still enabling multi-step dynamic programming. For the model-based offline RL algorithms, Kidambi et al. (2020) first constructs the estimated world model by utilizing diverse and large transition dataset and then sets hard threshold on the model uncertainty for constructing terminating states to avoid dangerous explorations. In (Yu et al., 2020), the authors proposes a model-based offline policy optimization algorithm (MOPO) which utilizes uncertainty estimation techniques to construct a penalty function to regularize the reward function. Therefore, MOPO can learn a conservative policy which stays in the low-uncertainty region to avoid the distribution shift issue. As a follow-up work, Lu et al. (2021) revisits the design choices of several key hyperparameters in MOPO and fine-tune the corresponding hyperparameters in MOPO to guarantee strong performance. In Yu et al. (2021), the authors propose a model-based offline RL algorithm called COMBO which does not rely on explicit uncertainty estimation. By regularizing the value function on out-of-distribution state-action pairs generated in the estimated world model, COMBO can benefit from the conservatism without requiring explicit uncertainty estimation techniques. As a remark, the algorithms proposed in (Yu et al., 2020; Kidambi et al., 2020; Lu et al., 2021; Yu et al., 2021) all perform conservative policy optimization in a well-constructed dynamics model and the estimated dynamics model keeps fixed during the training of the RL agent. Different from those algorithms mentioned above, (Uehara & Sun, 2021; Rigter et al., 2022) incorporate conservatism into the constructed dynamics model. By adversarially modifying the estimated dynamics model to minimize the value function under the current policy, the proposed methods can learn a robust policy with respect to the environment dynamics and can obtain probably approximately correct (PAC) performance guarantee.

Hybrid RL. In Nair et al. (2020), the authors provide a simple framework which can first leverage large offline dataset for pre-training and then quickly perform online fine-tuning of RL policies. In the proposed algorithm, sample-efficient dynamic programming method is used for policy evaluation and a maximum likelihood estimation problem is further designed for solving the optimal policy. In Garg et al. (2022), the authors develop a new framework to solve maximum entropy reinforcement learning, which directly estimates the optimal Bellman operator without replying on explicit access to the underlying policy. The proposed framework can be used to develop simple but effective reinforcement learning algorithms, which have the flexibility to well in both fully offline or hybrid RL settings. In Ball et al. (2023), the authors first revisit the design choices in existing off-policy methods. Moreover, with a set of minimal but important changes to the existing off-policy RL algorithms, the authors further show that the existing off-policy RL algorithms can achieve reliable performance by leveraging offline data when learning online.

B Preliminaries on LLM alignments

We model the LLM as a policy π in a Markov decision process (MDP). A MDP is defined by the tuple $(\mathcal{S}, \mathcal{A}, P, \mu, r, \gamma)$, which consists of the state space $\mathcal{S}$, the action space $\mathcal{A}$, the transition dynamics $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$, the initial state distribution $\mu(\cdot)$, the reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ and the discounted factor $\gamma \in (0, 1)$. Under a transition dynamics model P and a policy $\pi : \mathcal{S} \to \Delta_{\mathcal{A}}$ where $\Delta_{\mathcal{A}}$ is the probability simplex on the action space, further define the corresponding state visitation measure as $d_\pi(s) := (1-\gamma)\sum_{t=0}^{T} \gamma^t P^\pi(s_t = s | s_0 \sim \eta)$ for any state $s \in \mathcal{S}$. Here T is horizon size, which can be any positive integer or ∞.

In the LLM context, $\mathcal{S}$ and $\mathcal{A}$ correspond to the space of input prompts and the space of output continuations, respectively. If we consider the entire input sentences as state space and the entire output continuations as action space, then there is no transition dynamics, i.e. horizon = 1. On the other hand, if we consider the token-level state/action space[8], i.e. the action is modeled as the next token in the sentence, the transition dynamics is deterministic and $P(s'|s, a)$ is always 1 if $s' = (s, a)$ and 0 otherwise. The reward model r is usually another LLM trained specifically to evaluate the score of given prompt/continuation tuples, and the policy model π is the LLM to be optimized.

Now consider an LLM parameterized by θ. Denote $\pi_\theta(a|s)$ as the probability of outputting a given input prompt s, and we assume the horizon $T = 1$ for the rest of this section. The following discussions review three common procedures for fine-tuning LLM: (1) supervised fine-tuning (SFT) over demonstration dataset, (2) reinforcement learning with human feedback (RLHF) over preference dataset, and (3) direct preference optimization (DPO).

SFT. Given a *demonstration dataset* $\mathcal{D} := \{(s, a)\}$ collected from an expert policy π^E (i.e. $a \sim \pi^E(\cdot|s)$), the SFT optimizes the following problem:

$$\max_\theta \ \ell_{\text{SFT}}(\theta) := \mathbb{E}_{(s,a)\sim\mathcal{D}} \left[\log \pi_\theta\left(a|s\right)\right]. \tag{20}$$

The above problem shares the same optimal solutions with $\min_\theta \ \mathbb{E}_{s\sim\mu}[D_{\text{KL}}(\pi^{\text{E}}\left(\cdot|s\right) \| \pi_\theta\left(\cdot|s\right))]$ (see (10)), and the latter shows that SFT aims at imitating the demonstration dataset via minimizing the KL divergence. It is worth noting that the SFT stage described here is closely related to the imitation learning approach used in the RL literature for learning from demonstration (Osa et al., 2018), whose goal is to mimic the policy of an expert.

RLHF. Let $r_\phi(s, a)$ denote a reward model parameterized by ϕ, which evaluates a given input and output pair (s, a). Then the LLM can be fine-tuned by the following RL problem:

$$\max_\theta \ \ell_{\text{RL}}(\theta) := \mathbb{E}_{s\sim\mu, a\sim\pi_\theta(\cdot|s)}\left[r_\phi(s, a)\right] - \beta\mathbb{E}_{s\sim\mu}[D_{\text{KL}}(\pi_\theta\left(\cdot|s\right) \| \pi_{\text{ref}}\left(\cdot|s\right))], \tag{21}$$

where π_{ref} is a fixed reference model and D_{KL} is the KL-divergence to regulate the policy around the reference policy π_{ref}. Note that the KL regularization term in (21) is not computable given the sheer amount of possible output y (which could be *corpus_size*$^{\textit{max\_sequence\_length}}$ in most language model tasks), therefore (21) is usually solved by standard policy optimization techniques such as PPO (Schulman et al., 2017).

To find an appropriate reward model $r_\phi(s, a)$, RLHF (see e.g., Christiano et al. (2017)) leverages a set of *preference dataset* $\mathcal{P} := \{(s, a_w, a_l)\}$, where each data contains a pair of output a_w, a_l, and a_w is preferred over a_l by human labeler (denoted as $a_w \succ a_l$). The Bradley-Terry model (Bradley & Terry, 1952) assumes that the probability of choosing a_w over a_l is

$$\mathbb{P}\left(a_w \succ a_l \mid s\right) = \frac{\exp(r(s, a_w))}{\exp(r(s, a_w)) + \exp\left(r\left(s, a_l\right)\right)} = \sigma\left(r(s, a_w) - r\left(s, a_l\right)\right)$$

[8]The similar modeling can be considered if we model a dialogue as a sequence of state (questions) and actions (answers).

where σ is the sigmoid function. One can formulate the following problem to find the reward model:

$$\max_{\phi} \; \ell_{\rm RM}(\phi) := \mathbb{E}_{s\sim\mu,(a_l \prec a_w)\sim\pi^P(\cdot|s)}\Big[\log\Big(\sigma\big(r_\phi(s,a_w) - r_\phi(s,a_l)\big)\Big)\Big]. \tag{22}$$

It is widely observed in the literature that, models trained via episodically learning the policy (21) and learning the reward (22) typically outperform those that are only trained using SFT (Ouyang et al., 2022). The reward model guides the performance of the LLM and allows a better generalization ability via the consistent input of the preference data from human labeler.

DPO. DPO (Rafailov et al., 2024) proposes to incorporate reward learning implicitly by utilizing the structure of the optimal solution of the RL problem (21). Specifically, (21) implies that the optimal policy should satisfy:

$$r(s,a) = \beta\log\left(\frac{\pi_\theta(a|s)}{\pi_{\rm ref}(a|s)}\right) + \beta\log Z_\theta(s), \tag{23}$$

where $Z_\theta(s)$ is the partition function. Plugging this equation back to (22) we get the DPO loss:

$$\max_{\theta} \; \mathbb{E}_{s\sim\mu,(a_l \prec a_w)\sim\pi^P(\cdot|s)}\Big[\log\Big(\sigma\big(\beta\log\left(\frac{\pi_\theta(a_w|s)}{\pi_{\rm ref}(a_w|s)}\right) - \beta\log\left(\frac{\pi_\theta(a_l|s)}{\pi_{\rm ref}(a_l|s)}\right)\big)\Big)\Big]. \tag{24}$$

The difference between DPO and RLHF. Let us have a brief discussion about the above DPO reformulation. First, it is important to note that the fact that optimal policy of (21) takes the form of (23) *implicitly* says that there *must exist* a reward model r such that

$$\pi(a|s) = \frac{\pi_{\rm ref}(a|s)\exp\left(\frac{1}{\beta}r_\phi(s,a)\right)}{\sum_{\tilde a\in\mathcal{A}}\pi_{\rm ref}(\tilde a|s)\exp\left(\frac{1}{\beta}r_\phi(s,\tilde a)\right)}. \tag{25}$$

However, in the DPO formulation, after directly plugging (23) into (22), problem (24) *directly* optimizes the policy parameter θ, so there is no way to ensure that the optimized policy still satisfies (25). Therefore, a correct way to recover the solution of (21) is to solve the following problem (by introducing the explicit optimal policy constraint):

$$\begin{aligned}&\max_{\phi} \; \mathbb{E}_{s\sim\mu,(a_l \prec a_w)\sim\pi^P(\cdot|s)}\Big[\log\Big(\sigma\big(\beta\log\left(\frac{\pi(a_w|s)}{\pi_{\rm ref}(a_w|s)}\right) - \beta\log\left(\frac{\pi(a_l|s)}{\pi_{\rm ref}(a_l|s)}\right)\big)\Big)\Big]\\ &\text{s.t. } \pi := \arg\max_{\pi}\mathbb{E}_{s\sim\mu,a\sim\pi(\cdot|s)}\left[r_\phi(s,a)\right] - \beta\mathbb{E}_{s\sim\mu}[D_{\rm KL}(\pi\left(\cdot|s\right)\|\pi_{\rm ref}\left(\cdot|s\right))].\end{aligned} \tag{26}$$

The difference between (26) and (24) indicates that DPO in (24) is not exactly an RLHF scheme, but a supervised preference data-fitting scheme. In other words, the constrained optimization problem of RLHF is simplified and the solution space of DPO is actually larger than that of RLHF. Related discussion about limitation of DPO from different angles can also be found in Xu et al. (2024); Lin et al. (2024); Ivison et al. (2024).

C Proof of Theorem 1

In this section, we show the solution to the constrained policy optimization problem defined in (7a)-(7d).

Proof of Theorem 1. In this proof, we will first write down the *partial* Lagrangian function, which only considers the constraints (7b)-(7c). After solving the partial Lagrangian function, we will show that the constraint (7d) is satisfied.

Let α and $\zeta := \{\zeta_s | s\in\mathcal{S}\}$ denote the dual variables of the constraints (7b) and (7c), respectively. Then the partial Lagrangian function can be expressed as below:

$$\begin{aligned}\mathcal{L}(\pi,\alpha,\zeta) := &\frac{1}{1-\gamma}\mathbb{E}_{s\sim d_{\pi_{\rm old}}(\cdot),a\sim\pi(\cdot|s)}\left[A^{\pi_{\rm old}}(s,a)\right] + \alpha\Big(\epsilon - \mathbb{E}_{s\sim d_{\pi_{\rm old}}(\cdot)}\left[D_{\rm KL}\left(\pi(\cdot|s)||\pi_{\rm old}(\cdot|s)\right)\right]\Big)\\ &+\sum_{s\in\mathcal{S}}\zeta_s\big(1-\sum_{a\in\mathcal{A}}\pi(a|s)\big).\end{aligned}$$

Through taking partial derivative of $\mathcal{L}(\pi, \alpha, \zeta)$ w.r.t. $\pi(a|s)$, we can obtain the following equation:

$$\frac{\partial}{\partial \pi(a|s)}\mathcal{L}(\pi, \alpha, \zeta) = \frac{1}{1-\gamma} d_{\pi_{\text{old}}}(s) A^{\pi_{\text{old}}}(s,a) - \alpha d_{\pi_{\text{old}}}(s)\Big(- \log \pi_{\text{old}}(a|s) + \log \pi(a|s) + 1\Big) - \zeta_s.$$

Through setting the partial derivative $\frac{\partial}{\partial \pi(a|s)}\mathcal{L}(\pi, \alpha, \zeta)$ to 0, we obtain

$$\frac{1}{1-\gamma} d_{\pi_{\text{old}}}(s) A^{\pi_{\text{old}}}(s,a) - \alpha d_{\pi_{\text{old}}}(s)\Big(- \log \pi_{\text{old}}(a|s) + \log \pi(a|s) + 1\Big) - \zeta_s = 0.$$

Then we obtain the closed-form expression of the optimal policy π^* as below:

$$\log \pi^*(a|s) = \frac{A^{\pi_{\text{old}}}(s,a)}{(1-\gamma)\alpha} + \log \pi_{\text{old}}(a|s) - 1 - \frac{\zeta_s}{\alpha d_{\pi_{\text{old}}}(s)}, \tag{27a}$$

$$\pi^*(a|s) = \pi_{\text{old}}(a|s) \exp\Big(\frac{A^{\pi_{\text{old}}}(s,a)}{(1-\gamma)\alpha}\Big) \exp\Big(- 1 - \frac{\zeta_s}{\alpha d_{\pi_{\text{old}}}(s)}\Big). \tag{27b}$$

Here we can denote $\beta := (1-\gamma)\alpha$. Then according to the expression of $\pi(a|s)$ in (27b), we obtain the following relation:

$$\pi(a|s) \propto \pi_{\text{old}}(a|s) \exp\Big(\frac{1}{\beta} A^{\pi_{\text{old}}}(s,a)\Big). \tag{28}$$

Based on the constraint (7c), we know that $\pi(\cdot|s)$ is a distribution so that $\sum_{a\in\mathcal{A}} \pi(a|s) = 1$. Therefore, according to the expressions in (27b) and (28), we can obtain the following expression of the optimal policy π^* as below:

$$\pi^*(a|s) = \frac{\pi_{\text{old}}(a|s) \exp\Big(\frac{1}{\beta} A^{\pi_{\text{old}}}(s,a)\Big)}{\sum_{a'\in\mathcal{A}} \pi_{\text{old}}(a'|s) \exp\Big(\frac{1}{\beta} A^{\pi_{\text{old}}}(s,a')\Big)}.$$

Recall that $A^{\pi_{\text{old}}}(s,a) := Q^{\pi_{\text{old}}}(s,a) - V^{\pi_{\text{old}}}(s)$ has been defined in (3), then we can rewrite the expression of $\pi^*(a|s)$:

$$\begin{aligned}\pi^*(a|s) &= \frac{\pi_{\text{old}}(a|s) \exp\Big(\frac{1}{\beta}\big(Q^{\pi_{\text{old}}}(s,a) - V^{\pi_{\text{old}}}(s)\big)\Big)}{\sum_{a'\in\mathcal{A}} \pi_{\text{old}}(a'|s) \exp\Big(\frac{1}{\beta}\big(Q^{\pi_{\text{old}}}(s,a') - V^{\pi_{\text{old}}}(s)\big)\Big)} \\ &= \frac{\pi_{\text{old}}(a|s) \exp\Big(\frac{1}{\beta} Q^{\pi_{\text{old}}}(s,a)\Big)}{\sum_{a'\in\mathcal{A}} \pi_{\text{old}}(a'|s) \exp\Big(\frac{1}{\beta} Q^{\pi_{\text{old}}}(s,a')\Big)}.\end{aligned} \tag{29}$$

Then we can define a reference function $W^{\pi_{\text{old}}}(s)$ as below (a trick that was also employed in Garg et al. (2022)):

$$W^{\pi_{\text{old}}}(s) := \beta \log\Big(\mathbb{E}_{a\sim\pi_{\text{old}}(\cdot|s)}\Big[\exp\Big(\frac{1}{\beta} Q^{\pi_{\text{old}}}(s,a)\Big)\Big]\Big). \tag{30}$$

By plugging the definition of $W^{\pi_{\text{old}}}(s)$ into (2b), we can express the optimal policy $\pi^*(a|s)$ as below:

$$\pi^*(a|s) = \pi_{\text{old}}(a|s) \exp\Big(\frac{1}{\beta}\big(Q^{\pi_{\text{old}}}(s,a) - W^{\pi_{\text{old}}}(s)\big)\Big). \tag{31}$$

According to the closed-form expression of the optimal policy π^* in (31), we obtain that $\pi^*(a|s)$ is non-negative for any state-action pair (s,a) and thus the constraint (7d) is satisfied. $\square$

D Details of the experiment setting

We follow the 1b experiment as in (Huang et al. (2024)) and 8b experiment as in (Dong et al. (2024)), where we utilize DeepSpeed ZeRO-3 (Rajbhandari et al. (2020)) and FlashAttention-2 (Dao et al. (2022)) to reduce the memory cost. To accelerate data generation, we use VLLM (Kwon et al., 2023) for inference. We use eight NVIDIA A100-40G to do the training with per device batch size of 64 for 1b model and per device batch size of 16 for 8b model. We train all models with bfloat16 precision. We set the learning rate to be 3e-6 for 1b model and 5e-7 for 8b model with the cosine learning rate scheduler. We consider the max sequence length to be 565 for 1b models and 4096 for 8b models.

We also list the metric and number of shots used for LLM evaluation on each dataset.

Dataset	Arc Challenge	TruthfulQA MC2	Winogrande	GSM-8K	HellaSwag	MMLU
Metric	acc_norm	acc	acc	strict-match	acc_norm	acc
Num. of Shots	25	0	5	5	10	5

Table 2: A summarization of the benchmarks we use in this work. We list the metric and number of shots used for LLM evaluation on each dataset.

E More Experiment Results

In this section, we provide more experiment results. we include Tables 3 and 4, which correspond to Table 1 in the main body, as well as Table 5, which corresponds to Figure 4 in the main body.

Tasks Metrics	Arc Challenge acc_norm	TruthfulQA MC2 acc	Winogrande acc	GSM8k strict-match	HellaSwag acc_norm	MMLU	Average
`LLaMA3-SFT`	62.29%	53.49%	78.14%	72.55%	81.03%	64.49%	68.66%
`Bo16-round1`	62.54%	54.09%	77.51%	73.77%	81.14%	65.08%	69.02%
`Bo16-round2`	62.80%	54.51%	77.82%	75.13%	81.31%	65.17%	69.46%
`Bo16-round3`	63.14%	55.01%	78.06%	75.36%	81.34%	65.08%	69.66%
`Bo16-round4`	63.48%	55.54%	78.37%	75.81%	81.49%	65.25%	69.99%
`Bo16-round5`	63.32%	56.03%	78.14%	75.97%	81.36%	65.22%	70.01%

Table 3: Performance of Policy in Open LLm Leaderboard for for Best-of-16 algorithm.

Tasks Metrics	Arc Challenge acc_norm	TruthfulQA MC2 acc	Winogrande acc	GSM8k strict-match	HellaSwag acc_norm	MMLU	Average
`LLaMA3-SFT`	62.29%	53.49%	78.14%	72.55%	81.03%	64.49%	68.66%
`Bo16-round1`	62.46%	54.27%	77.66%	74.30%	81.21%	64.94%	69.14%
`Bo16-round2`	62.97%	54.81%	77.58%	75.06%	81.29%	65.13%	69.47%
`Bo16-round3`	63.57%	55.51%	78.06%	75.97%	81.43%	65.19%	69.96%
`Bo16-round4`	63.91%	55.86%	77.82%	76.04%	81.54%	65.35%	70.09%
`Bo16-round5`	63.57%	56.14%	77.74%	76.12%	81.68%	65.34%	70.10%

Table 4: Performance of Policy in Open LLm Leaderboard for Best-of-16 Re-weighting algorithm.

Tasks Metrics	Arc Challenge acc_norm	TruthfulQA MC2 acc	Winogrande acc	GSM8k strict-match	HellaSwag acc_norm	MMLU	Average
`LLaMA3-SFT`	62.29%	53.49%	78.14%	72.55%	81.03%	64.49%	68.66%
`SPR-round1`	62.37%	53.89%	77.90%	74.45%	81.03%	65.00%	69.44%
`SPR-round2`	63.14%	54.74%	77.82%	75.21%	81.20%	65.17%	69.88%
`SPR-round3`	63.31%	55.15%	77.35%	75.89%	81.27%	65.20%	69.86%
`SPR-round4`	63.91%	55.62%	78.22%	77.03%	81.36%	65.23%	70.23%
`SPR-round5`	63.82%	56.07%	77.82%	78.01%	81.51%	65.33%	70.43%
`SPR-round6`	63.74%	56.41%	78.45%	77.94%	81.50%	65.38%	70.57%
`SPR-round7`	63.99%	56.74%	78.22%	77.94%	81.57%	65.41%	70.65%
`RLHFlow/LLaMA3-DPO-iter1`	63.31%	57.19%	78.14%	74.30%	80.00%	64.65%	69.60%
`RLHFlow/LLaMA3-DPO-iter2`	65.36%	60.02%	77.43%	70.96%	81.56%	63.95%	69.88%

Table 5: Performance of Policy in Open LLm Leaderboard for Best-of-32 SPR algorithm.