REINFORCE++: A SIMPLE AND EFFICIENT APPROACH FOR ALIGNING LARGE LANGUAGE MODELS

Jian Hu
janhu9527@gmail.com

ABSTRACT

Reinforcement Learning from Human Feedback (RLHF) has emerged as a critical approach for aligning large language models with human preferences, witnessing rapid algorithmic evolution through methods such as Proximal Policy Optimization (PPO), Direct Preference Optimization (DPO), REINFORCE Leave One-Out (RLOO), ReMax, and Group Relative Policy Optimization (GRPO). We present REINFORCE++, an enhanced variant of the classical REINFORCE algorithm that incorporates key optimization techniques from PPO while eliminating the need for a critic network. Our approach achieves three primary objectives: (1) simplicity (2) enhanced training stability, and (3) reduced computational overhead. Through extensive empirical evaluation, we demonstrate that REINFORCE++ exhibits superior stability compared to GRPO and achieves greater computational efficiency than PPO while maintaining comparable performance. The implementation is available at https://github.com/OpenRLHF/OpenRLHF.

1 Introduction

The alignment of large language models with human preferences has become increasingly crucial in artificial intelligence research, with Reinforcement Learning from Human Feedback (RLHF) emerging as a leading methodology. The field has witnessed significant algorithmic innovations, from the foundational Proximal Policy Optimization (PPO) [4] to more recent approaches including Direct Preference Optimization (DPO) [3], REINFORCE Leave One-Out (RLOO) [6], ReMax [1], and Group Relative Policy Optimization (GRPO) [5].

PPO, while effective, requires a critic network that introduces additional computational overhead and potential training instabilities. Meanwhile, newer methods, like GRPO, address specific aspects of the optimization challenge but may introduce complexities and instability.

In this paper, we introduce REINFORCE++, which addresses these limitations through a thoughtful integration of PPO's optimization techniques into the classical REINFORCE framework. Our approach maintains the simplicity of REINFORCE notably eliminating the need for a critic network while preserving robust performance characteristics.

2 Background

2.1 REINFORCE Algorithm

The REINFORCE algorithm represents a fundamental policy gradient method in reinforcement learning, designed to optimize policies directly through gradient ascent on expected returns. Its operation encompasses several key components:

- **Policy Sampling:** The algorithm begins by sampling trajectories from the current policy, where the agent interacts with the environment to generate sequences of states, actions, and corresponding rewards.
- **Return Calculation:** For each trajectory, the algorithm computes discounted cumulative rewards:

$$G_t = \sum_{k=t+1}^{T} \gamma^{k-t} r_k \tag{1}$$

where γ represents the discount factor and r_k denotes the immediate reward at time step k.

- **Gradient Estimation:** The policy gradient is estimated using Monte Carlo methods:

$$\nabla_\theta J(\theta) = \mathbb{E}_\pi \left[G_t \nabla_\theta \log \pi_\theta(A_t|S_t) \right] \tag{2}$$

This formulation captures the fundamental relationship between policy parameters and expected returns.

- **Policy Update:** The policy parameters are updated through gradient ascent:

$$\theta_{t+1} = \theta_t + \alpha \nabla_\theta J(\theta) \tag{3}$$

where α represents the learning rate governing the magnitude of updates.

2.2 Challenges in RLHF

Contemporary RLHF implementations, particularly those based on PPO, often encounter several critical challenges:

- **Computational Overhead:** The requirement for a critic network in PPO introduces significant computational costs and memory requirements.
- **Training Instability:** The interplay between policy and value networks can lead to training instabilities, particularly in the context of language models [2].

3 REINFORCE++ Enhancements

REINFORCE++ incorporates several key optimizations to enhance training stability and efficiency:

3.1 Token-Level KL Penalty

We implement a token-level Kullback-Leibler (KL) divergence penalty between the RL model and the supervised fine-tuning (SFT) model distributions. This penalty is incorporated into the reward function as follows:

$$r(s_t, a_t) = \mathbf{I}(s_t = [\text{EOS}]) r(x, y) - \beta \, \text{KL}(t) \tag{4}$$

$$\text{KL}(t) = \log \left(\frac{\pi^{\text{RL}}_{\theta_{\text{old}}}(a_t|s_t)}{\pi^{\text{SFT}}(a_t|s_t)} \right) \tag{5}$$

where:

- x represents the input prompt
- y denotes the generated response
- $\mathbf{I}(s_t = [\text{EOS}])$ indicates whether t is the final token
- β is the KL penalty coefficient

This approach facilitates better credit assignment and seamless integration with Process Reward Models (PRM).

3.2 PPO-Clip Integration

We incorporate PPO's clipping mechanism to constrain policy updates:

$$L^{CLIP}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right] \tag{6}$$

where:

- $r_t(\theta)$ represents the policy ratio
- $\hat{A}_t$ denotes the advantage estimate
- ϵ controls the clipping threshold

3.3 Mini-Batch Updates

To enhance training efficiency, we implement mini-batch updates with the following characteristics:

- **Batch Processing:** Data is processed in smaller, manageable chunks rather than full-batch updates
- **Multiple Updates:** Each mini-batch allows for multiple parameter updates
- **Stochastic Optimization:** Introduces beneficial randomness for improved generalization

3.4 Reward Normalization and Clipping

We implement comprehensive reward processing:

- **Normalization:** Standardizes rewards using z-score normalization
- **Clipping:** Constrains reward values within predetermined bounds
- **Scaling:** Applies appropriate scaling factors for numerical stability

3.5 Advantage Normalization

The advantage function in REINFORCE++ is defined as:

$$A_t(s_t, a_t) = r(x, y) - \beta \cdot \sum_{i=t}^{T} \text{KL}(i) \tag{7}$$

We normalize these advantages using z-score normalization:

$$A_{\text{normalized}} = \frac{A - \mu_A}{\sigma_A} \tag{8}$$

where μ_A and σ_A represent the batch mean and standard deviation respectively.

4 Empirical Evaluation

4.1 Experimental Setup

4.1.1 Hyper-parameters

Our experimental configuration employed the following parameters:

- **KL Penalty Coefficient (β)**
  - General Scenario: 0.01
  - Mathematical Test: 0.001
- **Training Parameters**
  - Maximum Samples: $25{,}000$
  - Samples per Prompt: 4
  - Rollout Batch Size: 256
  - Training Batch Size: 128
  - Actor Learning Rate: 5×10^{-7}
  - Critic Learning Rate: 9×10^{-6}
  - Discount Factor (γ): 1.0
  - Clip ϵ: 0.2

4.1.2 Datasets

We utilized the following datasets:

- **General Domain**
  - Prompt Dataset: prompt-collection-v0.1
  - Reward Model Training: preference_700K
- **Mathematical Domain**
  - Training Data: MetaMathQA
  - A closed-source mathematical reward model

4.1.3 Base Models

Our experiments utilized:

- Llama3.1-8B-SFT
- Qwen2.5-7B-SFT

4.2 Results and Analysis

Our experimental results demonstrate several key findings:

- **General scenarios with Bradley-Terry Reward Models:** REINFORCE++ exhibits superior stability compared to GRPO, particularly in preventing reward hacking and managing output length (Figure 1).
- **Rule-Based Reward Model:** Under rule-based reward scenarios, REINFORCE++ achieves comparable performance to GRPO with group normalization (Figure 2).
- **Mathematical Reward Model:** In mathematical problem-solving scenarios, REINFORCE++ demonstrates a better reward increase per unit KL divergence compared to GRPO (Figure 3).

Figure 1: Comparative analysis shows that PPO/REINFORCE++ have smaller length hacking issues compared to GRPO in general scenarios with Bradley-Terry Reward Models.

5 Conclusion

REINFORCE++ successfully combines the simplicity of the REINFORCE algorithm with crucial optimization techniques from PPO, achieving improved stability and performance without the computational overhead of a critic network.

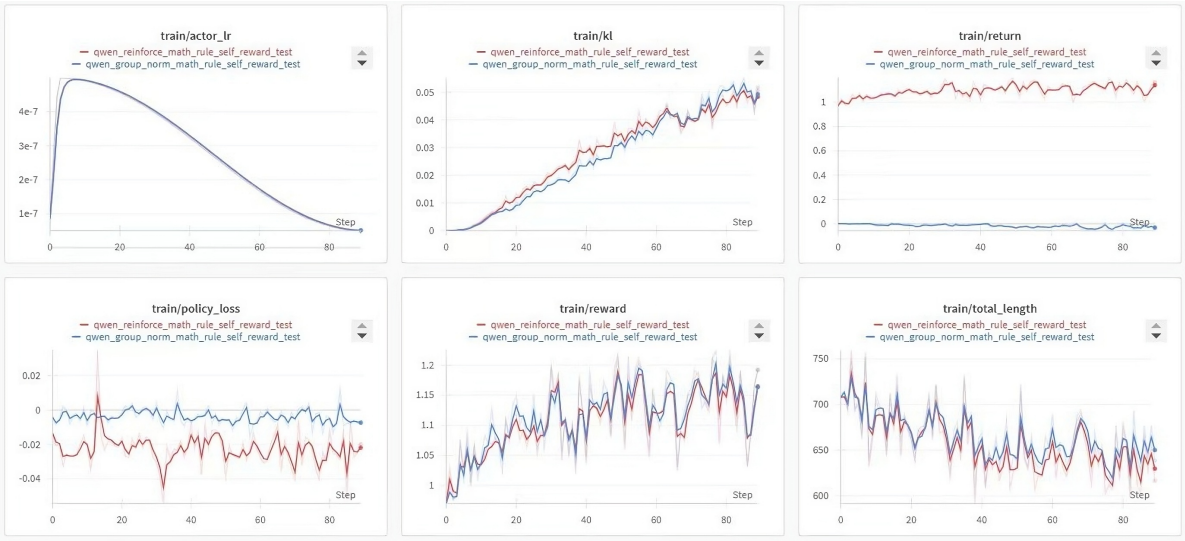

Figure 2: Performance comparison in mathematical scenario 1 showing comparable results between REINFORCE++ and GRPO(Group Norm) under rule-based rewards.

Figure 3: Mathematical scenario 2 results show that, under the same unit KL consumption, REINFORCE++/RLOO achieves a greater reward increase compared to GRPO (Group Norm).

References

[1] Ziniu Li, Tian Xu, Yushun Zhang, Yang Yu, Ruoyu Sun, and Zhi-Quan Luo. Remax: A simple, effective, and efficient method for aligning large language models. *arXiv preprint arXiv:2310.10505*, 2023.

[2] 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. *arXiv preprint arXiv:2203.02155*, 2022.

[3] 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. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[4] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 1132–1145. PMLR, 2017.

[5] 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. *arXiv preprint arXiv:2402.03300*, 2024.

[6] Yuan Wu et al. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms. *arXiv preprint arXiv:2402.14740*, 2024.