

## A Detailed Theoretical Analysis

### A.1 Theoretical understanding on solving the minimax problem in Eq 1 with PET

Here, we explain in detail how the PET algorithm in Alg 1 solves the theoretical problem in Eq. 1. Recall that Eq. 1 is a minimax problem based on rejection sampling as follows:

$$\min_{r \in \mathcal{R}} \max_{\pi \in \Pi_{\text{RS}}^{n, \pi_0}} (V_r^\mu(\pi) - V_r^\mu(\pi_{\text{ref}})) + \beta \cdot \mathcal{L}_{\mathcal{D}}(r).$$

By Proposition 2.1, we have  $\pi_{\text{RS}}(\pi_0, n, r^t) \in \arg \max_{\pi \in \Pi_{\text{RS}}^{n, \pi_0}} V_r^\mu(\pi)$ . Therefore, solving the minimax problem in Eq 1 is equivalent to solving the minimization problem as follows:

$$\min_{r \in \mathcal{R}} (V_r^\mu(\pi_{\text{RS}}(\pi_0, n, r)) - V_r^\mu(\pi_{\text{ref}})) + \beta \cdot \mathcal{L}_{\mathcal{D}}(r).$$

Denote

$$f(r, \pi) := (V_r^\mu(\pi) - V_r^\mu(\pi_{\text{ref}})) + \beta \cdot \mathcal{L}_{\mathcal{D}}(r)$$

and

$$h(r) := (V_r^\mu(\pi_{\text{RS}}(\pi_0, n, r^t)) - V_r^\mu(\pi_{\text{ref}})) + \beta \cdot \mathcal{L}_{\mathcal{D}}(r).$$

Then, our goal is to solve the problem  $\min_{r \in \mathcal{R}} h(r)$ . For any reward model  $r_0 \in \mathcal{R}$ , we have

$$\begin{aligned} \nabla h(r)|_{r=r_0} &= \nabla_r f(r, \pi)|_{r=r_0, \pi=\pi_{\text{RS}}(\pi_0, n, r_0)} + \nabla_\pi f(r, \pi)|_{r=r_0, \pi=\pi_{\text{RS}}(\pi_0, n, r_0)} \cdot \nabla_r \pi_{\text{RS}}(\pi_0, n, r_0)|_{r=r_0} \\ &= \nabla_r f(r, \pi)|_{r=r_0, \pi=\pi_{\text{RS}}(\pi_0, n, r_0)} \end{aligned}$$

The second equality holds because  $\pi = \pi_{\text{RS}}(\pi_0, n, r)$  is the optimizer for  $\max_{\pi \in \Pi_{\text{RS}}^{n, \pi_0}} f(r, \pi)$ , so we have  $\nabla_\pi f(r, \pi)|_{r=r_0, \pi=\pi_{\text{RS}}(\pi_0, n, r_0)} \equiv 0, \forall r_0 \in \mathcal{R}$ . This is an important result as it shows that to compute  $\nabla h(r)|_{r=r_0}$ , we don't need to compute  $\nabla_r \pi_{\text{RS}}(\pi_0, n, r)|_{r=r_0}$ . The latter represents how the rejection sampling would change if the reward function changes, which is hard to approximate in practice.

A standard approach to find the minimizer of  $\min_r h(r)$  is to perform stochastic gradient descent on  $h(r)$ . By the previous results, we can implement this process by approximating the value of  $\nabla h(r)|_{r=r_0} = \nabla_r f(r, \pi)|_{\pi=\pi_{\text{RS}}(\pi_0, n, r^t)}$ .

Next, we show that PET in Alg 1 is essentially the process of stochastic gradient descent to solve the problem  $\min_{r \in \mathcal{R}} h(r)$ . Recall  $V_r^\mu(\pi) = \mathbb{E}_{x \sim \mu, a \sim \pi(\cdot|x)}[r(x, a)]$ . At line 5, the prompts  $x_i$  and the data minibatch  $\mathcal{D}_t$  are sampled from the dataset  $\mathcal{D}$ . At line 6, the responses  $a_i, a_{\text{ref}, i}$  are stochastically sampled from the current policy  $\pi^t$  and the reference policy  $\pi_{\text{ref}}$ . Therefore, the expectation of the average loss  $l^t/M$  equals to the value of  $h(r^t)$ :

$$\mathbb{E}[\sum_{i \in [M]} [r^t(x_i, \pi^t(x_i, a_i)) - r^t(x_i, \pi^t(x_i, a_{\text{ref}, i})) + \beta \cdot \mathcal{L}_{\mathcal{D}_t}(r^t)]]/M = (V_r^\mu(\pi^t) - V_r^\mu(\pi_{\text{ref}})) + \beta \cdot \mathcal{L}_{\mathcal{D}}(r)$$

At line 8, the algorithm computes the gradient of  $l^t$ , so Alg 1 is essentially a standard process of stochastic gradient descent to solve the minimization problem  $\min_{r \in \mathcal{R}} h(r)$ . Therefore, PET is a convenient implementation to solve the minimax problem based on rejection sampling in Eq. 1.

*Remark A.1.* Here we argue that PET approximates the most pessimistic low-prediction-loss reward model against the rejection sampling process. Denote  $\hat{r}$  the solution to the minimization problem  $\hat{r} \in \min_{r \in \mathcal{R}} (V_r^\mu(\pi) - V_r^\mu(\pi_{\text{ref}})) + \beta \cdot \mathcal{L}_{\mathcal{D}}(r)$ . For any reward model  $r \in \mathcal{R}$  that has the same or lower prediction loss on the dataset  $\mathcal{L}_{\mathcal{D}}(r) \leq \mathcal{L}_{\mathcal{D}}(\hat{r})$ , we have  $(V_{\hat{r}}^\mu(\pi) - V_{\hat{r}}^\mu(\pi_{\text{ref}})) \leq (V_r^\mu(\pi) - V_r^\mu(\pi_{\text{ref}}))$ . This is because

$$\begin{aligned} &(V_{\hat{r}}^\mu(\pi) - V_{\hat{r}}^\mu(\pi_{\text{ref}})) + \beta \cdot \mathcal{L}_{\mathcal{D}}(r) \leq (V_r^\mu(\pi) - V_r^\mu(\pi_{\text{ref}})) + \beta \cdot \mathcal{L}_{\mathcal{D}}(r) \\ \Rightarrow &(V_{\hat{r}}^\mu(\pi) - V_{\hat{r}}^\mu(\pi_{\text{ref}})) - (V_r^\mu(\pi) - V_r^\mu(\pi_{\text{ref}})) \leq \beta \cdot (\mathcal{L}_{\mathcal{D}}(r) - \mathcal{L}_{\mathcal{D}}(\hat{r})) \\ \Rightarrow &(V_{\hat{r}}^\mu(\pi) - V_{\hat{r}}^\mu(\pi_{\text{ref}})) - (V_r^\mu(\pi) - V_r^\mu(\pi_{\text{ref}})) \leq 0 \\ \Rightarrow &(V_{\hat{r}}^\mu(\pi) - V_{\hat{r}}^\mu(\pi_{\text{ref}})) \leq (V_r^\mu(\pi) - V_r^\mu(\pi_{\text{ref}})) \end{aligned}$$

In practice, we use the PET algorithm to approximate the solution  $\pi_0$  and find that the prediction loss of the learned reward model is as low as that of the proxy reward, which is specially trained to minimize the prediction loss. This implies that the reward model  $\hat{r}$  is also the most pessimistic reward model that gives minimal relative score to the rejection sampling process among the reward models with low values of prediction loss. This observation also supports our argument that the reward model learned from PET is pessimistic.

## A.2 Proof for Proposition 2.1

Proposition 2.1 describes an intuitive property of rejection sampling. It is equivalent to the statement that when optimizing a policy on a reward model  $r_1$  through the rejection sampling process, the RS process that sets its reward model as  $r = r_1$  achieves the highest reward on  $r_1$ . For any rejection sampling process with the same base policy and number of samples, the distribution of the sampled responses during the process is always the same. Therefore, for any outcome of sampled responses, the RS process that sets  $r = r_1$  can always output the response with the highest reward on  $r_1$ . Formally, the proof for Proposition 2.1 is as follows.

*Proof.* Given a base policy  $\pi_0$ , a positive integer  $n$ , and a prompt  $x$ , consider a stochastic process where  $n$  responses are i.i.d drawn from the policy at the prompt  $a_i \stackrel{i.i.d}{\sim} \pi_0(\cdot|x), i \in [n]$ . Let  $E(x)$  be the space of all possible outcomes. For any outcome  $e \in E(x)$ , let  $\{a_1, \dots, a_n\}$  be the sampled responses. For any two different reward models  $r_1 \neq r_2$ , denote  $i_1^* \in \arg \max_{i \in [n]} r_1(x, a_i)$  and  $i_2^* \in \arg \max_{i \in [n]} r_2(x, a_i)$  as the optimal index on the two rewards. Denote  $v_1(e, r) := r(x, a_{i_1^*})$  and  $v_2(e, r) := r(x, a_{i_2^*})$  where  $r$  is any reward model, then by definition, we have  $v_1(e, r_1) \geq v_2(e, r_1)$ . Consider two rejection sampling policies  $\pi_1 = \pi_{RS}(\pi_0, r_1, n), \pi_2 = \pi_{RS}(\pi_0, r_2, n)$  defined on the base policy  $\pi_0$ , sampling number  $n$ , and the reward models  $r_1, r_2$ . Their performance on the reward model  $r_1$  for any prompt distribution  $\mu$  satisfies  $V_{r_1}^\mu(\pi_1) = \mathbb{E}_{x \sim \mu, e \sim E(x)}[v_1(e, r_1)], V_{r_1}^\mu(\pi_2) = \mathbb{E}_{x \sim \mu, e \sim E(x)}[v_2(e, r_1)]$ . By the previous result  $v_1(e, r_1) \geq v_2(e, r_1)$ , we have  $V_{r_1}^\mu(\pi_1) \geq V_{r_1}^\mu(\pi_2)$ , which conclude the proof.  $\square$

## A.3 Proof for Theorem 3.3

*Proof.* Our proof generally follows the proof technique in Liu et al. [2024b]. Let  $\hat{r}$  be the reward solution to  $\hat{r} \in \arg \min_{r \in \mathcal{R}}$

$$\begin{aligned}
& V_{r^*}^\mu(\pi) - V_{r^*}^\mu(\hat{\pi}) \\
&= (V_{r^*}^\mu(\pi) - V_{\hat{r}}^\mu(\pi)) + (V_{\hat{r}}^\mu(\pi) - V_{\hat{r}}^\mu(\hat{\pi})) + (V_{\hat{r}}^\mu(\hat{\pi}) - V_{r^*}^\mu(\hat{\pi})) \\
&\leq (V_{r^*}^\mu(\pi) - V_{\hat{r}}^\mu(\pi)) + (V_{\hat{r}}^\mu(\hat{\pi}) - V_{r^*}^\mu(\hat{\pi})) \\
&= (V_{r^*}^\mu(\pi) - V_{\hat{r}}^\mu(\pi)) + (V_{\hat{r}}^\mu(\hat{\pi}) - V_{\hat{r}}^\mu(\pi_{\text{ref}}) + \beta \cdot \mathcal{L}_{\mathcal{D}}(\hat{r}) - (V_{r^*}^\mu(\hat{\pi}) - V_{\hat{r}}^\mu(\pi_{\text{ref}}) + \beta \cdot \mathcal{L}_{\mathcal{D}}(r^*))) + \\
&\quad \beta \cdot (\mathcal{L}_{\mathcal{D}}(r^*) - \mathcal{L}_{\mathcal{D}}(\hat{r})) \\
&\leq (V_{r^*}^\mu(\pi) - V_{\hat{r}}^\mu(\pi)) + \beta \cdot (\mathcal{L}_{\mathcal{D}}(r^*) - \mathcal{L}_{\mathcal{D}}(\hat{r}))
\end{aligned}$$

The first equality utilizes the optimality of the policy solution  $\hat{\pi} \in \arg \max_{\pi \in \Pi_{RS}} V_{\hat{r}}^\mu(\pi)$ . The second equality utilizes the optimality of the reward solution  $\hat{r} \in \arg \min_{r \in \mathcal{R}} (V_{\hat{r}}^\mu(\pi) - V_{\hat{r}}^\mu(\pi_{\text{ref}}) + \beta \cdot \mathcal{L}_{\mathcal{D}}(r))$ . Intuitively, the formulation in the last line is bounded for any reward function  $\hat{r}$ . If the reward function is close to the true reward  $r^*$ , then both differences  $V_{r^*}^\mu(\pi) - V_{\hat{r}}^\mu(\pi)$  and  $\mathcal{L}_{\mathcal{D}}(r^*) - \mathcal{L}_{\mathcal{D}}(\hat{r})$  should be small. If  $\hat{r}$  is very different from  $r^*$ , then its prediction loss on the dataset  $\mathcal{L}_{\mathcal{D}}(\hat{r})$  would also be high, so that the performance gap can still be bounded. This indicates the importance of adding the prediction loss as a constraint on the reward model in Eq. 1. Formally, based on the results of Theorem 5.3 in Liu et al. [2024b], with probability at least  $1 - \delta$ , the last term can be bound by

$$V_{r^*}^\mu(\pi) - V_{r^*}^\mu(\hat{\pi}) \leq \frac{\mathcal{C}_{\mu_{\mathcal{D}}}(\mathcal{R}, \pi, \pi_{\text{ref}})^2}{8\kappa^2 \cdot \beta} + \frac{3\beta}{N} \log\left(\frac{N_{\epsilon}(\mathcal{R}, \|\cdot\|_{\infty})}{\delta}\right).$$

Here,  $\kappa = \frac{1}{(1+\exp(R))^2}$  is a constant, and  $N_\epsilon(\mathcal{R}, \|\cdot\|_\infty)$  is the  $\epsilon$ -covering number for the reward model class Cheng et al. [2022]. Setting

$$\beta = \frac{\sqrt{N}}{2\kappa \cdot \sqrt{6 \log\left(\frac{N_\epsilon(\mathcal{R}, \|\cdot\|_\infty)}{\delta}\right)}}$$

concludes the proof. Note that the bound here is tighter than the bound on Liu et al. [2024b], because there is no KL regularization in our formulation at Eq. 1. Adding a KL regularization multiplied by any positive coefficient to Eq. 1 will increase the bound on the performance gap analysis, which is undesired for sampling efficiency.  $\square$

## B Policy Optimization Methods

**KL regularized proximal policy optimization (KL-PPO):** For a prompt distribution  $\mu$ , the ‘KL divergence’ between two policies can be defined as  $\text{KL}_\mu(\pi_1, \pi_2) := \mathbb{E}_{x \sim \mu}[\text{KL}(\pi_1(\cdot|x) \parallel \pi_2(\cdot|x))]$ , where  $\text{KL}(\pi_1(\cdot|x) \parallel \pi_2(\cdot|x)) := \sum_{a \in \mathcal{A}} \pi_1(a|x) \cdot \log \frac{\pi_1(a|x)}{\pi_2(a|x)}$  is the standard KL divergence between two distributions. Consider a proxy policy  $\pi_{\text{ref}}$  of the dataset. The proxy policy is trained to generate a response distribution that is similar to the dataset response distribution. This is often achieved by a standard supervised fine-tuning process on the prompts and responses from the dataset  $\mathcal{D}$  [Touvron et al., 2023]. The KL divergence between a policy  $\pi$  and  $\pi_{\text{ref}}$  intuitively indicates how well it is covered by the dataset. Then, the learning goal of the agent becomes

$$\hat{\pi} \leftarrow \arg \max_{\pi \in \Pi} V_{\hat{r}}^\mu(\pi) + \eta \cdot \text{KL}_\mu(\pi, \pi_{\text{ref}}),$$

where  $\Pi$  is a model family,  $\eta > 0$  is the weight of the KL regularization in the optimization target. The most popular way to solve this optimization problem is by using the PPO algorithm [Ouyang et al., 2022]. Combining the reward modeling step and the policy optimization process gives the popular RLHF algorithm ‘PPO-KL’ as shown in Alg 3

**Rejection sampling process (RS):** Rejection sampling, also known as best-of-N sampling, is an inference-time policy optimization method. Given a base policy model  $\pi_0$ , a reward model  $\hat{r}$ , and a positive integer  $n$ , the process of rejection sampling is defined in Alg 4. In practice, the reward model  $\hat{r}$  is trained by minimizing the prediction loss, and the base policy  $\pi_0$  is usually set as a proxy policy for the dataset. The rejection sampling process is effectively a policy as it takes a prompt as input and stochastically outputs a response. An RS policy can be directly implemented on a policy model and a reward model without any training. It also achieves a higher reward on the reward model  $\hat{r}$  compared to the base policy  $\pi_0$ . The pessimism in rejection sampling implicitly relies on the notion of KL regularization. When the value of  $n$  is small, the RS policies will have a limited KL regularization compared to the base policy  $\pi_0$ , which intuitively reduces the risk of reward hacking Beirami et al. [2024]. One can achieve a higher performance on the reward model with rejection sampling by increasing the value of  $n$ , but in this case, the risk of reward hacking also increases Huang et al. [2025]. In practice, reward hacking has been observed in RS with a relatively high value of  $n$  Touvron et al. [2023], Gao et al. [2023]. Therefore, it is important to develop principled methods to free RS policies from reward hacking.

---

### Algorithm 3 KL-PPO

---

- 1: **Input:** Reference policy  $\pi_{\text{ref}}$ , Dataset  $\mathcal{D}$ , KL weight  $\eta$
  - 2: **Step 1:** Reward modeling  $\hat{r} \leftarrow \min_{r \in \mathcal{R}} \mathcal{L}_{\mathcal{D}}(r)$
  - 3: **Step 2:** Policy optimization with PPO  $\hat{\pi} \leftarrow \max_{\pi \in \Pi} V_{\hat{r}}^\mu(\pi) - \eta \cdot \text{KL}_\mu(\pi, \pi_{\text{ref}})$
  - 4: **Return:** Policy model  $\hat{\pi}$
- 

## C Additional Experiment Details

### Hyper-parameter setups:

**Algorithm 4** Rejection sampling process

- 
- 1: **Input:** Base policy  $\pi_0$ , reward model  $\hat{r}$ , number of samples  $n$ , prompt  $x$
  - 2: Draw responses  $a_i \stackrel{i.i.d.}{\sim} \pi_0(\cdot|x), \forall i \in [n]$
  - 3: Generate rewards  $r_i = \hat{r}(x, a_i), \forall i \in [n]$
  - 4: **Return:** Response  $a_{i^*}$ , where  $i^* \in \arg \max_{i \in [n]} r_i$
- 

Configuration	PET	PPO
learning rate	$3e-8$	$3e-6$
learning scheduler type	cosine	cosine
batch size	128	128
gradient accumulation steps	16	16
training epoch	1	1
pessimistic coefficient $\beta$	10	/
KL regularization weight $\eta$	/	0.05
optimizer	adamw torch	adamw torch
precision	bfloat16	bfloat16

Table 5: Training configurations for PET and PPO.

We list detailed training configurations for PET and PPO in Table 5. For standard SFT model and proxy reward model training, we follow the same setup as in [Huang et al., 2024b]. For all baseline methods, we follow the original authors’ implementations or descriptions. Specifically, for RPO, we remove the default chat template provided by its source code, as it is unsuitable for our summarization and IMDB tasks. This is confirmed by our observation that including the template degrades RPO’s performance on our tasks, so the template is not included in our RPO training.

**Statistical significance:** To verify that the methods we test with have stable output, we test our method PET-PPO and other baselines, including KL-PPO, DPO, RPO, and  $\chi$ PO, on both TL;DR summarization and IMDB datasets with a different random seed. In Table. 6 and Table. 7, we show the empirical mean and confidence intervals of the evaluation results on the RLHF methods from the two random seeds.

	KL-PPO	DPO	RPO	$\chi$ PO	<b>PET-PPO</b>
Summarization	$40.5 \pm 0.3$	$37.5 \pm 1.3$	$33.6 \pm 0.4$	$38.2 \pm 1.0$	<b><math>41.4 \pm 0.6</math></b>
IMDB	$92.0 \pm 0.2$	<b><math>95.0 \pm 0.0</math></b>	$93.1 \pm 0.4$	$93.0 \pm 0.2$	<b><math>96.1 \pm 2.2</math></b>

Table 6: Empirical mean and standard errors of the evaluation results for the RLHF methods on the summarization and IMDB dataset

	KL-PPO	DPO	RPO	$\chi$ PO
<b>PET-PPO</b>	<b><math>57.2 \pm 0.0</math></b>	$51.2 \pm 1.2$	<b><math>62.0 \pm 3.6</math></b>	<b><math>55.0 \pm 2.2</math></b>

Table 7: Empirical mean and standard errors for the win rate of the PET-PPO method against other RLHF baselines on the summarization dataset.

**Computational resource:** We use 2 NVIDIA A100-PCIE GPUs, each with 40GB VRAM and 2 48-core230 Intel Xeon Silver 4214R CPU.