RL-Finetuning LLMs From On- and Off-Policy Data With a Single Algorithm

Yunhao Tang* **Taco Cohen** Meta FAIR **David W. Zhang** Meta FAIR **Michal Valko** INRIA **Rémi Munos*** Meta FAIR

Abstract

We introduce a novel reinforcement learning algorithm (AGRO, for Any-Generation Reward Optimization) for fine-tuning large-language models. AGRO leverages the concept of generation consistency, which states that the optimal policy satisfies the notion of consistency across any possible generation of the model. We derive algorithms that find optimal solutions via the sample-based policy gradient and provide theoretical guarantees on their convergence. Our experiments demonstrate the effectiveness of AGRO in both on-policy and off-policy settings, showing improved performance on the mathematical reasoning dataset over baseline algorithms.

1 INTRODUCTION

Large language models have displayed powerful capabilities, and upon fine-tuning, such can become helpful assistants to human users. Although supervised learning has been a dominant fine-tuning paradigm due to its efficacy and stability, reinforcement learning from human feedback (RLHF) has become an increasingly important complement [Christiano et al., 2017, Ziegler et al., 2019, Nakano et al., 2021, Ouyang et al., 2022, Bai et al., 2022]. Although previously it was commonly believed that reinforcement learning (RL) provides relatively lightweight fine-tuning on top of an existing model, it has also been proven to be highly effective in entailing new capabilities [Jaech et al., 2024].

The existing RL algorithms for language model fine-tuning can be largely categorized into two groups: online and offline. Online RL algorithms consist of iterative sampling and updating of the latest policy model, assuming access to an external reward model that provides the learning signal [Ziegler et al., 2019, Ouyang et al., 2022, Munos et al., 2023, Guo et al., 2024, Calandriello et al., 2024]. Meanwhile, offline RL algorithms directly extract information from a static offline dataset, bypassing the need for iterative sampling and potentially reward modeling [Zhao et al., 2023, Rafailov et al., 2024, Gheshlaghi Azar et al., 2024, Tang et al., 2024b, Richemond et al., 2024]. There is a clear trade-off between the two classes of algorithms: while offline algorithms are much cheaper to execute, they generally deliver less improvement compared to online algorithms [Xu et al., 2024, Tang et al., 2024a, Tajwar et al., 2024].

The dichotomy between online and offline algorithms also makes it difficult to directly adapt algorithms from one case to another. Most importantly, we will see that naively adapting the on-policy KL regularized policy gradient algorithm - one of the most prominent on-policy RL algorithms - to the off-policy case, will not lead to the optimal target policy (Figure 1). The natural motive is to build an algorithm that works seamlessly for both on-policy and off-policy cases.

In this work, we propose a novel RL algorithm called Any-Generation Reward Optimization (AGRO) for fine-tuning language models. Central to AGRO is the insight that the optimal policy for the RLHF problem satisfies a notion of *generation consistency*, which we will detail later. This consistency is rooted in the regularized policy optimization that underlies the RLHF formulation. This consistency allows us to derive algorithms that leverage both on-policy and off-policy data, and find optimal policy via sample-based learning with theoretical guarantees. Our technical contributions are detailed as follows.

- We identify generation consistency (Section 3), a notion of optimality condition inherent to the regularized policy optimization problem, from which we propose losses to search for the optimal policy and their sample-based gradient estimates (Sec-

Proceedings of the 29th International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300.

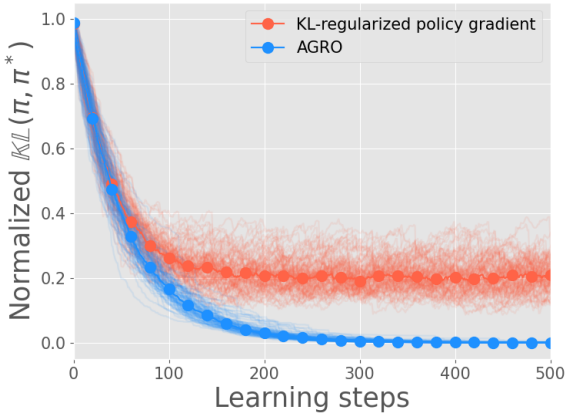


Figure 1: KL-regularized policy gradient vs. AGRO in the tabular case with off-policy data (data generated from π_{ref} rather than π). We measure the normalized KL-divergence $\mathbb{KL}(\pi, \pi^*)$ between π and the optimal policy π^* during training. We see that under off-policy data, regularized KL-regularized policy gradient does not converge to the optimal policy π^* while AGRO does.

tion 4).

- We propose Any-Generation Reward Optimization (AGRO, Section 5), which is derived from the consistency condition and can leverage both on-policy and off-policy data for policy optimization.
- We conclude with experimental ablations that showcase the competitive performance of AGRO in both on-policy and off-policy learning settings, with a focus on the mathematical reasoning data (Section 7). We also ablate on important hyper-parameters.

2 REINFORCEMENT LEARNING FOR LANGUAGE MODEL

A language model can be understood as a policy π in the context of RL. Given a prompt $x \in X$, the policy produces a response (or generation) $y \in Y$, which then gets evaluated by a certain reward function $r(x, y)$, which usually reflects human preference. The objective is to optimize π such that that the expected reward is maximized: depending on the application, the reward function can be a parametric network [Christiano et al., 2017, Ouyang et al., 2022] or programmatic [Lightman et al., 2023]. Formally, consider the regularized policy optimization problem to maximize the following regularized objective

$$\mathcal{G}(\pi) \stackrel{\text{def}}{=} \mathbb{E}_{x\sim\rho}\left[\mathbb{E}_{y\sim\pi(\cdot|x)}\left[r(x,y)\right] - \beta\mathbb{KL}(\pi(x), \pi_{\text{ref}}(x))\right] \tag{1}$$

which seeks to maximize the average reward subject to the KL regularization $\mathbb{KL}\left(\pi(x), \pi_{\text{ref}}(x)\right)$ that encourages π to stay close to some reference policy π_{ref} during optimization. Here, ρ denotes the sampling distribution over the space of prompt X; the coefficient $\beta \geq 0$ determines the trade-off between the policy optimization and the regularization. The reference policy π_{ref} is usually the supervised finetuned policy and the starting point of the optimization.

3 GENERATION CONSISTENCY FOR REGULARIZED RL

We start with the key observation that a notion of consistency condition is inherent to the optimal solution of the regularized RL (RLHF) problem. Formally, this result can be stated as follows.

Theorem 1. ***(Generation consistency at optimality)*** *Let $\pi^* \stackrel{def}{=} \arg\max_\pi \mathcal{G}(\pi)$ be the optimal policy to the RLHF problem defined by Eq. (1). Then for any generation y, the quantity*

$$r(x, y) - \beta \log \frac{\pi^*(y|x)}{\pi_{ref}(y|x)}$$

does not depend on y, it is a function of the prompt x only.

Proof. It is well-known that the optimal policy has an analytic form [Rafailov et al., 2024, Calandriello et al., 2024, Richemond et al., 2024]: for all x, y,

$$\pi^*(y|x) = \frac{\pi_{\text{ref}}(y|x)e^{\frac{1}{\beta}r(x,y)}}{e^{\frac{1}{\beta}\widetilde{V}^{\pi^*}(x)}}, \tag{2}$$

where the normalizing constant is

$$\widetilde{V}^{\pi^*}(x) \stackrel{\text{def}}{=} \beta \log\Big(\sum_{y\in Y} \pi_{\text{ref}}(y|x)e^{\frac{1}{\beta}r(x,y)}\Big).$$

We can verify that for any y, $r(x, y) - \beta \log \frac{\pi^*(y|x)}{\pi_{\text{ref}}(y|x)} = \widetilde{V}^{\pi^*}(x)$, which is a function of the prompt x only. □

As a technical remark, note we can also introduce the traditional value function which is defined as the expected reward $V^\pi(x) \stackrel{\text{def}}{=} \sum_{y\in Y} \pi(y|x)r(x, y)$, and notice that by plugging π^* back into Equation (1) we have the property that:

$$\widetilde{V}^{\pi^*}(x) \quad = \quad V^{\pi^*}(x) - \beta\mathbb{KL}(\pi^*(x), \pi_{\text{ref}}(x)).$$

That is, the normalizing constant $\widetilde{V}^{\pi^*}(x)$ is indeed a KL-regularized value function [Ziebart et al., 2008, Nachum et al., 2017, Levine, 2018].

3.1 Deriving loss functions from generation consistency

Henceforth for simplicity, we introduce the regularized reward which depends on the policy π,

$$R_\beta^\pi(x,y) \stackrel{\text{def}}{=} r(x,y) - \beta \log \frac{\pi(y|x)}{\pi_{\text{ref}}(y|x)}.$$

Theorem 1 implies the following property on the regularized reward $R_\beta^{\pi^*}$ of the optimal policy: for any generation y,

$$R_\beta^{\pi^*}(x,y) - \mathbb{E}_{y' \sim \pi(\cdot|x)} \left[R_\beta^{\pi^*}(x,y') \right] = 0. \tag{3}$$

We can hence define a squared loss that enforces the above equality in order to find the optimal policy, i.e., we minimize with respect to π, from every x,

$$\mathbb{E}_{y \sim \pi(\cdot|x)} \left[\left(R_\beta^\pi(x,y) - \mathbb{E}_{y' \sim \pi(\cdot|x)} \left[R_\beta^\pi(x,y') \right] \right)^2 \right].$$

Note that if we think of $R_\beta^\pi(x,y)$ as a random variable, the above can be understood as its variance under the distribution $y \sim \pi(\cdot|x)$. In general, we can define the loss as the expectation (over x) of the variance (over y) of $R_\beta^\pi(x,y)$, which, when minimized, will recover π^*. We now consider both the on-policy and off-policy settings.

When y is sampled under the current policy being optimized π, we define the **on-policy loss**

$$\mathcal{L}(\pi) \stackrel{\text{def}}{=} \frac{1}{2} \mathbb{E}_{x \sim \rho} \left[\mathbb{V}_{y \sim \pi(\cdot|x)} \left(R_\beta^\pi(x,y) \right) \right].$$

where we notice that both the regularized reward R_β^π and the sampling depend on π. Similarly, in the off-policy case (e.g., responses y have been generated by some behavior policy μ), e.g. using a fixed dataset or a replay buffer, we define the **off-policy loss**:

$$\mathcal{L}_\mu(\pi) \stackrel{\text{def}}{=} \frac{1}{2} \mathbb{E}_{x \sim \rho} \left[\mathbb{V}_{y \sim \mu(\cdot|x)} \left(R_\beta^\pi(x,y) \right) \right].$$

Importantly, we show that globally minimizing both losses will lead to the optimal policy.

Theorem 2. ***(The unique global minimizer is the optimal policy)*** *Consider the space Π of policies that have the same support as $\pi_{ref}(\cdot|x)$ on all states $x \in$ support(ρ). Assume $\mu \in \Pi$. Then π^* is the unique global minimum in Π of both $\mathcal{L}(\pi)$ and $\mathcal{L}_\mu(\pi)$ on support(ρ).*

Proof. We provide the proof here as the argument is quite constructive. First notice that $\mathcal{L}(\pi) \geq 0$ and $\mathcal{L}_\mu(\pi) \geq 0$, and from (3) we have that $\mathcal{L}(\pi^*) = 0$ and $\mathcal{L}_\mu(\pi^*) = 0$. Thus π^* is a global minimum of both $\mathcal{L}(\pi)$ and $\mathcal{L}_\mu(\pi)$.

Now, let's prove that π^* is the unique global minimum of both $\mathcal{L}(\pi)$ and $\mathcal{L}_\mu(\pi)$. The losses $\mathcal{L}(\pi)$ and $\mathcal{L}_\mu(\pi)$ are the expectation (under $x \sim \rho$) of the variance of $R_\beta^\pi(x,y)$ when $y \sim \pi(\cdot|x)$ (respectively when $y \sim \mu(\cdot|x)$).

For any policy $\pi \in \Pi$, for any $x \in$ support(ρ) we have that $\pi(\cdot|x)$ has the same support as $\pi_{\text{ref}}(\cdot|x)$, which, from Eq. 2, is the same as the support of $\pi^*(\cdot|x)$. Thus, for the on-policy loss, for any $x \in$ support(ρ), this variance is zero if and only if for any $y \in$ support$(\pi(\cdot|x))$ (respectively $y \in$ support$(\mu(\cdot|x))$ for the off-policy loss), the term $R_\beta^\pi(x,y)$ is independent of y (but can be a function of x), which is equivalent to say that

$$\pi(y|x) \propto \pi_{\text{ref}}(y|x) e^{\frac{1}{\beta} r(x,y)}.$$

From (2) we deduce that $\pi(y|x) = \pi^*(y|x)$. Thus π^* is the unique global minimum in Π of both $\mathcal{L}(\pi)$ and $\mathcal{L}_\mu(\pi)$ on the support of ρ. □

4 GRADIENT OF THE CONSISTENCY LOSSES

We now analyze the gradient of the loss functions above that shed light on their intuitions.

4.1 Gradient of the off-policy loss

Throughout, we consider a parametric policy π and let ∇ denote the gradient with respect to its parameters. We start with the gradient of the off-policy loss $\mathcal{L}_\mu(\pi)$.

Lemma 1. ***(Off-policy gradient)*** *The gradient of the off-policy loss $\mathcal{L}_\mu(\pi)$ with respect to the policy parameter is:*

$$\nabla \mathcal{L}_\mu(\pi) = -\beta \mathbb{E}_{\substack{x \sim \rho \\ y \sim \mu(\cdot|x)}} \left[R_\beta^\pi(x,y) \nabla \log \pi(y|x) \right].$$

Proof. The proof follows from taking the derivative of a squared loss using the fact that $\nabla R_\beta^\pi(x,y) = -\beta \nabla \log \pi(y|x)$, and that $\mathbb{E}_{y \sim \mu(\cdot|x)} \left[\left(R_\beta^\pi(x,y) - R_\beta^\pi(x,\mu) \right) \nabla R_\beta^\pi(x,\mu) \right] = 0$. This concludes the proof. □

Notice that we can also subtract the baseline $R_\beta^\pi(x,\mu) \stackrel{\text{def}}{=} \mathbb{E}_{y' \sim \mu(\cdot|x)} \left[R_\beta^\pi(x,y') \right]$ from the regularized reward $R_\beta^\pi(x,y)$ above for the purpose of variance reduction without introducing any bias, since

$$\mathbb{E}_{y \sim \mu(\cdot|x)} \left[R_\beta^\pi(x,\mu) \nabla \log \pi(y|x) \right] = 0.$$

Equivalence to RLHF gradient when on-policy. We can interpret the above gradient as updating with respect to the regularized reward $R^\pi_\beta(x,y)$. In fact, we can further break down the gradient for prompt x as

$$-\beta\mathbb{E}_{y\sim\mu(\cdot|x)}\left[r(x,y)\nabla\log\pi(y|x)-\frac{\beta}{2}\nabla\Big(\log\frac{\pi(y|x)}{\pi_{\text{ref}}(y|x)}\Big)^2\right]$$

where the first term maximizes reward while the second term minimizes the squared loss divergence between π and π_{ref}. When the sampling is on-policy $\mu=\pi$, the gradient of the squared loss is aligned with the gradient of the KL regularization [Calandriello et al., 2024, Tang et al., 2024b]. In this case, the off-policy AGRO gradient is equivalent to the RLHF gradient from Eqn (1).

4.2 Gradient of the on-policy loss

When differentiating the on-policy loss $\mathcal{L}(\pi)$ with respect to the policy parameter we need to take into account the fact that the sampling distribution depends on the parameter as well. Writing $f(x,y,\pi)=\frac{1}{2}\left(R^\pi_\beta(x,y)-R^\pi_\beta(x,\pi)\right)^2$, the gradient thus contains two terms: the pathwise-derivative (PD) estimate and the likelihood ratio (LR) estimate [Glasserman, 2004]:

$$\nabla\mathcal{L}(\pi)=\underbrace{\mathbb{E}\left[\nabla f(x,y,\pi)\right]}_{\nabla_{\text{PD}}\mathcal{L}(\pi)}+\underbrace{\mathbb{E}\left[f(x,y,\pi)\nabla\log\pi(y|x)\right]}_{\nabla_{\text{LR}}\mathcal{L}(\pi)},$$

where the expectation is under $x\sim\rho, y\sim\pi(\cdot|x)$. The pathwise-derivative $\nabla_{\text{PD}}\mathcal{L}(\pi)$ is the gradient of the off-policy loss $\nabla\mathcal{L}_\mu(\pi)$ when setting $\mu=\pi$, i.e. $\nabla\mathcal{L}_\mu(\pi)|_{\mu=\pi}$,

$$\begin{aligned}&=-\beta\mathbb{E}_{\substack{x\sim\rho\\ y\sim\pi(\cdot|x)}}\left[R^\pi_\beta(x,y)\nabla\log\pi(y|x)\right].\\&=-\beta\mathbb{E}_{\substack{x\sim\rho\\ y\sim\pi(\cdot|x)}}\left[\left(R^\pi_\beta(x,y)-R^\pi_\beta(x,\pi)\right)\nabla\log\pi(y|x)\right].\end{aligned}$$

Now, the likelihood ratio estimate $\nabla_{\text{LR}}\mathcal{L}(\pi)$ is

$$\frac{1}{2}\mathbb{E}_{\substack{x\sim\rho\\ y\sim\pi(\cdot|x)}}\left[\left(R^\pi_\beta(x,y)-R^\pi_\beta(x,\pi)\right)^2\nabla\log\pi(y|x)\right].$$

Notice again that we can substract as baseline the variance of $R^\pi_\beta(x,y)$ as a possible variance reduction technique without introducing bias, leading to this estimate of $\nabla_{\text{LR}}\mathcal{L}(\pi)$:

$$\frac{1}{2}\mathbb{E}_{\substack{x\sim\rho\\ y\sim\pi(\cdot|x)}}\left[\left(\left(R^\pi_\beta(x,y)-R^\pi_\beta(x,\pi)\right)^2-b(x)\right)\nabla\log\pi(y|x)\right],$$

where the baseline is $b(x)\overset{\text{def}}{=}\mathbb{V}_{y'\sim\pi(\cdot|x)}\left(R^\pi_\beta(x,y')\right)$ which we can approximate with samples. Finally, the full gradient of the on-policy loss is

$$\nabla\mathcal{L}(\pi)=\nabla_{\text{PD}}\mathcal{L}(\pi)+\nabla_{\text{LR}}\mathcal{L}(\pi)$$

We provide a few remarks as follows.

Alignment of $\nabla_{\text{PD}}\mathcal{L}(\pi)$ with the gradient of the original objective. Notice that the pathwise-derivative estimate is aligned with the gradient of the original policy objective $\mathcal{G}$:

$$\nabla_{\text{PD}}\mathcal{L}(\pi)=\nabla\mathcal{L}_\mu(\pi)|_{\mu=\pi}=-\beta\nabla\mathcal{G}(\pi).$$

Different learning dynamics, same optimum. Notice that since both $\pi\mapsto\mathcal{G}(\pi)$ and $\pi\mapsto\mathcal{L}(\pi)$ have π^* as optimum, by following the full gradient $\nabla\mathcal{L}(\pi)$ or the pathwise-derivative $\nabla_{\text{PD}}\mathcal{L}(\pi)$ only will both lead to the optimal policy π^*. However the learning dynamics generated by these two gradients will be different.

Gradient magnitude. It may seem that $\nabla_{\text{PD}}\mathcal{L}(\pi)$ is negligible compared to $\nabla_{\text{LR}}\mathcal{L}(\pi)$, specially when β is small. However, rewriting $R^\pi_\beta(x,y)=r(x,y)-\beta\log\frac{\pi(y|x)}{\pi_{\text{ref}}(y|x)}=\beta\log\frac{\pi^*(y|x)}{\pi(y|x)}+\widetilde{V}^{\pi^*}(x)$, we see that $R^\pi_\beta(x,y)-R^\pi_\beta(x,\pi)=\beta\left(\log\frac{\pi^*(y|x)}{\pi(y|x)}+\mathbb{KL}(\pi(x),\pi^*(x))\right)$ is of order $O(\beta)$, thus both $\nabla_{\text{PD}}\mathcal{L}(\pi)$ and $\nabla_{\text{LR}}\mathcal{L}(\pi)$ are of order $\mathcal{O}(\beta^2)$.

5 ANY-GENERATION REWARD OPTIMIZATION

We introduce the Any-Generation Reward Optimization (AGRO) algorithm, which finds the optimal policy by leveraging the any generation consistency. We introduce a few variants of the algorithm. Throughout, we assume that we can collect samples $\{x,(y_i,r_i)_{1\le i\le n}\}$ composed of prompts $x\sim\rho$, and for each x, one or several responses $y_1,\dots,y_n$ generated by some behavior policy μ (when off-policy) of by the current policy π (when on-policy). We write the corresponding rewards $\{r_i=r(x,y_i)\}_{1\le i\le n}$.

5.1 The off-policy AGRO algorithm

Single generation per prompt $n=1$. When $n=1$ we only have access to a single response per prompt. In that case one way to minimize the loss $\pi\mapsto\mathcal{L}_\mu(\pi)$ is to introduce a prompt-dependent function $V(x)$ and jointly minimize over π and V the loss $\mathcal{L}_\mu(\pi,V)$ defined as follows

$$\mathbb{E}_{\substack{x\sim\rho\\ y\sim\mu(\cdot|x)}}\left[\left(r(x,y)-V(x)-\beta\log\frac{\pi(y|x)}{\pi_{\text{ref}}(y|x)}\right)^2\right].$$

This loss has been introduced in [Richemond et al., 2024] where it was proven that the global optimum (both over π and V) leads to the optimal policy π^*. The globally minimizing value function is also the normalizing constant $\widetilde{V}^{\pi^*}(x)$. In this paper we focus on the

case $n \geq 2$ for which we avoid having to learn an auxiliary value function. In practice, this might be more desirable due to relative implementation simplicity.

Several generations per prompt. Assume we now have $n > 1$ responses $y_1, \ldots, y_n$ issued from the same prompt $x \sim \rho$. We do not have to learn an auxiliary value function of the prompt like in the previous paragraph. Instead we can build an unbiased estimate of the loss $\pi \mapsto \mathcal{L}_\mu(\pi)$ by approximating the expectation by an empirical average.

Off-policy AGRO algorithm consists in following the gradient estimate $\nabla\widehat{\mathcal{L}}_\mu(\pi) \overset{\text{def}}{=}$

$$-\frac{\beta}{n}\sum_{i=1}^n \Big[R_\beta^\pi(x, y_i) - \underbrace{\widehat{R}_{\beta,-i}^\pi(x,\mu)}_{\text{baseline}}\Big]\nabla \log \pi(y_i|x), \quad (4)$$

where the baseline is defined as the leave-one-out average [Kool et al., 2019]:

$$\widehat{R}_{\beta,-i}^\pi(x,\mu) \overset{\text{def}}{=} \frac{1}{n-1}\sum_{j\neq i}\left(r_j - \beta \log \frac{\pi(y_j|x)}{\pi_{\text{ref}}(y_j|x)}\right).$$

It does not introduce any bias but usually helps reduce the variance of the estimate. When the sampling is on-policy $\mu = \pi$, the above gradient estimate is akin to REINFORCE with leave-one-out baseline [Ahmadian et al., 2024a, Shao et al., 2024].

The specific case of two samples. Assume we now have $n = 2$ generations y_1 and y_2 from the same prompt $x \sim \rho$. The empirical gradient $\nabla\widehat{\mathcal{L}}_\mu(\pi)$ is

$$-\beta\left[r_1 - r_2 - \beta \log \frac{\pi(y_1|x)\pi_{\text{ref}}(y_2|x)}{\pi(y_2|x)\pi_{\text{ref}}(y_1|x)}\right]\nabla \log \frac{\pi(y_1|x)}{\pi(y_2|x)}.$$

This algorithm resembles the contrastive policy gradient algorithm of Flet-Berliac et al. [2024] and the IPO algorithm of Gheshlaghi Azar et al. [2024] where the preference $p(y_1 \succ y_2)$ is replaced by the reward difference $r_1 - r_2$.

Comparison of off-policy AGRO vs. KL-regularized policy gradient. To illustrate the key property of off-policy AGRO, we compare with an adaptation of KL-regularized policy gradient algorithm to the off-policy learning case. Concretely, the n-sample KL-regularized policy gradient is implemented as

$$\frac{1}{n}\sum_{i=1}^n (r_i - \bar{r}_{-i})\nabla \log \pi(y_i|x) - \beta \nabla \mathbb{KL}\left(\pi(\cdot|x), \pi_{\text{ref}}(\cdot|x)\right),$$

where $\bar{r}_{-i}$ is the leave-one-out reward baseline. We note that this gradient differs from off-policy AGRO gradient in Eqn (4) on the regularization term as hinted earlier.

Figure 1 compares the KL divergence between the optimized policy and optimal policy $\mathbb{KL}(\pi, \pi^*)$ over updates. We see that the regularized policy gradient algorithm generally fails to converge to π^*: the KL-regularization is not the *correct* regularization in the off-policy case. On the other hand, off-policy AGRO gradient converges to π^* as expected.

Note also that the decrease in the KL divergence $\mathbb{KL}(\pi, \pi^*)$ is in fact equivalent to increase in the regularized reward $\mathcal{G}(\pi) = \mathbb{E}_{y\sim\pi}\left[R_\beta^\pi(x,y)\right]$ since $\mathbb{E}_{y\sim\pi}\left[R_\beta^\pi(x,y)\right] = \beta\mathbb{KL}(\pi,\pi^*) + \widetilde{V}^{\pi^*}(x)$. We make this result formal in the appendix B.

5.2 The on-policy AGRO algorithm

The on-policy AGRO algorithm consists in following the gradient estimate consisting of two parts $\nabla\widehat{\mathcal{L}}(\pi) = \nabla_{\text{PD}}\widehat{\mathcal{L}}(\pi) + \nabla_{\text{LR}}\widehat{\mathcal{L}}(\pi)$ where $\nabla_{\text{PD}}\widehat{\mathcal{L}}(\pi)$ is the pathwise derivative estimate (the *off-policy AGRO* gradient applied to $\mu = \pi$)

$$-\frac{\beta}{n}\sum_{i=1}^n \Big[R_\beta^\pi(x,y_i) - \widehat{R}_{\beta,-i}^\pi(x,\pi)\Big]\nabla \log \pi(y_i|x) \quad (5)$$

and $\nabla_{\text{LR}}\widehat{\mathcal{L}}(\pi)$ is the likelihood ratio gradient estimate: $\frac{1}{2n}\sum_{i=1}^n \Big(R_\beta^\pi(x,y_i) - \widehat{R}_{\beta,-i}^\pi(x,\pi)\Big)^2 \nabla \log \pi(y_i|x)$. Thus the full on-policy AGRO gradient $\nabla\widehat{\mathcal{L}}(\pi)$ can be written as

$$\frac{1}{2n}\sum_{i=1}^n \Big(R_\beta^\pi(x,y_i) - \widehat{R}_{\beta,-i}^\pi(x,\pi) - \beta\Big)^2 \nabla \log \pi(y_i|x). \quad (6)$$

Proposition 1. ***(Unbiased on-policy AGRO gradient)*** *$\nabla\widehat{\mathcal{L}}(\pi)$ is an unbiased estimate of $\nabla\mathcal{L}(\pi)$.*

Proof. From the off-policy AGRO gradient we have $\mathbb{E}_{x\sim\rho, y_1,\ldots,y_n\sim\pi(\cdot|x)}\left[\nabla_{\text{PD}}\widehat{\mathcal{L}}(\pi)\right] = \nabla_{\text{PD}}\mathcal{L}(\pi)$. Now again, using that $y_1, \ldots, y_n$ are i.i.d. and that $\widehat{R}_{\beta,-i}^\pi(x,\pi)$ is independent of y_i (and in expectation equals $R_\beta^\pi(x,\pi)$), we have that $\mathbb{E}_{x\sim\rho, y_1,\ldots,y_n\sim\pi(\cdot|x)}\left[\nabla_{\text{LR}}\widehat{\mathcal{L}}(\pi)\right] = \nabla_{\text{LR}}\mathcal{L}(\pi)$. Thus $\mathbb{E}\left[\nabla\widehat{\mathcal{L}}(\pi)\right] = \nabla\mathcal{L}(\pi)$ and the proof is concluded. □

We discuss a few properties of the gradient estimate.

Variance reduction. Notice that we can build the estimate

$$\nabla_{\text{LR}}^{\text{biased}}\widehat{\mathcal{L}}(\pi) \overset{\text{def}}{=} \frac{1}{2n}\sum_{i=1}^n \big(A_i - \overline{A}\big)\nabla \log \pi(y_i|x),$$

where $A_i \overset{\text{def}}{=} \left(R^\pi_\beta(x, y_i) - \widehat{R}^\pi_{\beta,-i}(x, \pi)\right)^2$, and $\overline{A} \overset{\text{def}}{=} \frac{1}{n-1}\sum_{j\neq i} A_j$ is a leave-one-out baseline. This estimates may have a lower variance than $\nabla_{\text{LR}}\widehat{\mathcal{L}}(\pi)$ but at the price of introducing some bias. The bias stems from the fact that the baseline $\frac{1}{n-1}\sum_{j\neq i} A_j$ now depends on the sample y_i, as opposed to typical regular leave-one-out applications [Kool et al., 2019].

Relation to on-policy RL. The usual approach in on-policy RL consists in maximizing the original objective $\pi \mapsto \mathcal{G}(\pi)$ by following the regularized policy gradient:

$$\nabla\widehat{\mathcal{G}}(\pi) \overset{\text{def}}{=} \frac{1}{n}\sum_{i=1}^{n}\left[r_i - \beta \log \frac{\pi(y_i|x)}{\pi_{\text{ref}}(y_i|x)}\right]\nabla \log \pi(y_i|x),$$

Notice that, similarly to Eq. (5), the leave-one-out baseline $\widehat{R}^\pi_{\beta,-i}(x, \pi)$ can be subtracted from the regularized reward in the expression above without introducing any bias (this is called the RLOO algorithm in Ahmadian et al. [2024b]).

We see that, although the losses $\pi \mapsto \mathcal{G}(\pi)$ and $\pi \mapsto \mathcal{L}(\pi)$ have the same global optimum π^*, the on-policy AGRO gradient $\nabla\widehat{\mathcal{L}}(\pi)$ is not aligned with the usual on-policy RL gradient $\nabla\widehat{\mathcal{G}}(\pi)$. Actually, $\nabla\widehat{\mathcal{L}}(\pi)$ is the sum of two terms, $\nabla_{\text{PD}}\widehat{\mathcal{L}}(\pi)$, which is aligned with $\nabla\widehat{\mathcal{G}}(\pi)$, and $\nabla_{\text{LR}}\widehat{\mathcal{L}}(\pi)$, which intends to reduce the variance $\pi \mapsto \mathbb{V}_{y\sim\pi(\cdot|x)}\left(R^{\pi'}_\beta(x, y)\right)|_{\pi'=\pi}$.

6 DISCUSSIONS ON RELATED WORK AND EXTENSIONS

We further discuss how the AGRO formulation can be extended and how it relates to prior work.

Pairwise preference. A dominant paradigm in RLHF is where the reward $r(x, y)$ is derived from a pairwise preference dataset, usually modeled using the Bradley-Terry (BT) assumption [Bradley and Terry, 1952]: $p(y \succ y'|x) = \exp\left(r(x, y) - r(x, y')\right)$. A direct implication of generation consistency from Theorem 1 is that for any $y, y' \in Y$,

$$r(x, y) - r(x, y') = \beta \log \frac{\pi(y|x)}{\pi_{\text{ref}}(y|x)} - \beta \log \frac{\pi(y'|x)}{\pi_{\text{ref}}(y'|x)},$$

which when combined the BT loss produces the direct preference optimization loss [Rafailov et al., 2024]. Similar arguments can be extended to other generic loss functions for pairwise preference data [Zhao et al., 2023, Gheshlaghi Azar et al., 2024, Tang et al., 2024b].

Connections to regularized value learning. The generation consistency (Theorem 1) can be understood as a special case of the general consistency for regularized value based learning in the RL literature [Schulman et al., 2017, Nachum et al., 2017]. Indeed, if we interpret the log ratio $\log \frac{\pi(y|x)}{\pi_{\text{ref}}(y|x)}$ as an advantage function [Tang et al., 2023], then the policy optimization algorithm is understood as a value learning algorithm. At the token level, the consistency condition can be expressed as

$$\widetilde{V}^{\pi^*}(y_{1:t}) = \sum_{s=t}^{t'-1}\left(r_s - \beta \log \frac{\pi^*(y_s)}{\pi_{\text{ref}}(y_s)}\right) + \widetilde{V}^{\pi^*}(y_{1:t'}),$$

where $\widetilde{V}^{\pi^*}(y_{1:t})$ is a value function that depends on the whole partial sequence. Here, we have omitted the dependency on the prompt x and r_s is the per-token reward. Such a formulation might be useful in cases where dense rewards are used [Uesato et al., 2022, Lightman et al., 2023].

Previously, we have established that the gradient of the off-policy AGRO with on-policy sampling was aligned with the gradient as the RLHF problem. Indeed, viewing AGRO as a value-based learning algorithm, this equivalence can be seen as a special case of the equivalence between soft Q-learning and regularized policy optimization [Schulman et al., 2017, Nachum et al., 2017, O'Donoghue, 2022].

7 EXPERIMENTS

Throughout, we focus on the mathematical reasoning dataset MATH [Hendrycks et al., 2021] where the prompt x consists of asking the model a mathematical question with a short-form ground truth answer a^* available. Given the model generation $y = (c, a)$ which typically consists of a step-by-step chain-of-thought reasoning c and a proposed answer a, the reward is computed as a match between a and a^*. We adopt Sympy [Meurer et al., 2017] to automatically match the answers and assign a reward of $r = 1$ if there is a match and $r = 0$ otherwise. We train our models on the MATH training set, which defines the prompt distribution ρ, with various objective alternatives introduced above. Throughout, we provide the model with a system prompt that asks for a step-by-step solution, followed by a final answer. Our experiments are based on the 8B model from the Llama 3 model family [Dubey et al., 2024]. All algorithmic variants apply identical hyper-parameters such as learning rate, and that they all apply $n = 4$ samples for gradient estimations. All experiments are conducted with an entropy regularization coefficient $\beta > 0$ which we have ablated in the main paper.

Hyper-parameters. All experiments are conducted with identical hyper-parameter settings: we always

apply a batch size of $B = 64$ prompts per update, and sample $n = 4$ distinct generations per prompt. All training and evaluation sampling are conducted at a temperature of $\tau = 1$ and with top-p $= 1$.

Data. We train on the MATH training set with 7500 examples and evaluate on the test set with 2500 examples. A supervised fine-tuning on the training set is conducted to warm up the RL training, hence the gap between training and test set accuracy.

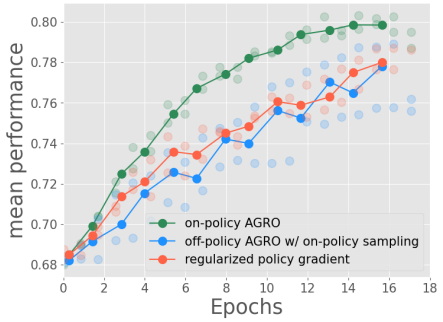
For both training and evaluation, we provide system instructions that ask the model to generate a response with step-by-step solution, followed by a final conclusion phrased as *the final answer is* followed by the answer predicted by the model. This is consistent with the prompt structure discussed for Llama models [Dubey et al., 2024].

7.1 On-policy learning

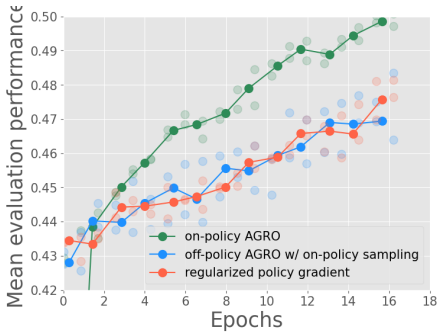
We start with the on-policy learning where we compare three algorithmic variants: regularized policy gradient algorithm (with leave-one-out baseline, aka RLOO), off-policy AGRO algorithm with on-policy sampling, and on-policy AGRO algorithm. For the base experiment, we apply a regularization coefficient of $\beta = 0.001$.

Figure 2(a) shows the training performance as the training progresses. We ran each algorithmic variant with 2 seeds and show the average. A few observations are in order: (1) Note that in the case of perfect on-policy sampling, the off-policy AGRO algorithm (for $\mu = \pi$) should behave identically as the regularized policy gradient algorithm. We see that this is indeed the case within statistical errors across independent runs; (2) The on-policy AGRO algorithm seems more data-efficient, producing faster progress on the reward given same number of epochs on the training set. All algorithms can be implemented with similar computational cost per iteration, and as a result, the number of epochs is an approximation to the total computational cost. This means that the on-policy AGRO algorithm is also more data-efficient compared to alternatives - it requires less compute to achieve the same level of training performance.

Figure 2(b) shows the evaluation performance during the course of training. The results are fairly correlated with the training rewards - on-policy ARGO achieves the best performance overall, with a +7% lift in performance compared to π_{ref}, while other baselines has a lift of +4%.



(a) Training performance



(b) Evaluation performance

Figure 2: Figure (a): Training performance for on-policy learning. We compare three algorithmic alternatives: regularized policy gradient algorithm (red), off-policy AGRO algorithm with on-policy sampling (blue) and on-policy AGRO algorithm (green), all with regularization $\beta = 0.001$. The performance is evaluated against the learning iterations, with each iteration being 100 gradient updates. We observe a similar performance from the regularized policy gradient algorithm and off-policy AGRO, which is expected; on-policy AGRO seems to derive better performance over other baselines, given the same number of learning steps. Figure (b): Evaluation performance. We plot the evaluation performance as a function of the learning iterations. We see that the training and evaluation performance are quite correlated, and that on-policy AGRO seems to achieve the best performance across different algorithmic variants.

7.2 Off-policy and offline learning

We emulate the extreme case of off-policy learning where the sampling distribution is from the reference policy π_{ref} and fixed throughout training. We consider a few algorithmic baselines: AGRO and regularized policy gradient. To adapt the regularized policy gradient to the offline case, we consider the per-token KL regularization: given a trajectory y, the regularization is computed as $\sum_{t=1}^{T} \mathbb{KL}(\pi_t, \pi_{\text{ref},t})$ which generally differs from the sequence-level KL regularization in the off-policy case [Calandriello et al., 2024, Tang et al., 2024b].

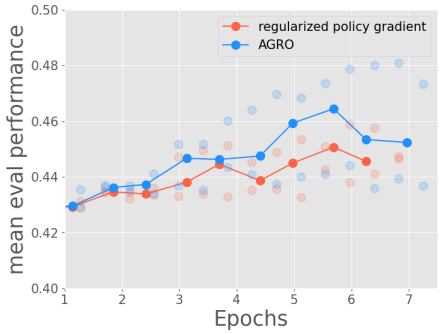
Figure 3(a) shows the evaluation performance of off-policy AGRO when compared against the baseline of

a vanilla regularized policy gradient. AGRO and regularized policy gradient both apply a regularization of $\beta = 0.01$. We make a few observations: (1) When regularization is weak, as with the case of unregularized policy gradient, the performance crashes as we go through more epochs on the training set. Such drop in performance does not happen for the on-policy case, which illustrates the potential instability of off-policy learning; (2) Off-policy AGRO seems to obtain better performance overall compared to KL-regularized policy gradient algorithm, obtaining both better peak performance and better performance at a fixed epochs.

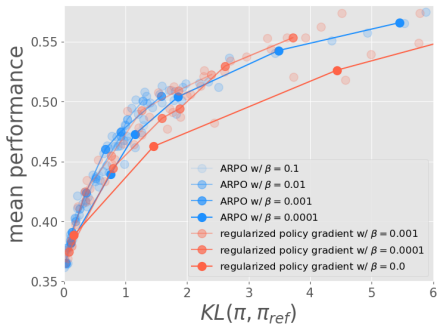
Impact of the level of off-policyness. We have found that the difference between off-policy AGRO and KL-regularized policy gradient seems to enlarge as the level of off-policyness increases. Recall that the two algorithms are effectively equivalent when the sampling is fully on-policy, and their difference in regularization increases in the off-policy case. In Figure 4 of Appendix A we show further results on comparing the on-policy and off-policy performance: note that though off-policy learning provides overall less improvement on performance, it is as KL-efficient as on-policy.

Ablation on regularization coefficient β. In the on-policy experiments, we have varied the coefficient for both the regularized policy gradient algorithm $\beta \in \{0, 0.0001, 0.001\}$ and the off-policy AGRO algorithm $\beta \in \{0.0001, 0.001, 0.01, 0.1\}$. The ablation results are presented in Figure 3(b). In an expected way, strong regularization tends to slow down the policy optimization progress, i.e., with the same number of learning steps, the performance improvement is much slower at a large value of β. However, strong regularization makes the algorithm more KL-efficient. Intuitively, this is because a strong regularization makes the update much more conservative and this prevents the policy from deviating too much from the reference policy, while making progress on the training reward. For completeness, we also ablate the case where the optimization is unregularized ($\beta = 0$), there is no incentive in preventing the policy from staying close to the reference policy. We find that no regularization makes the algorithm generally much more KL-inefficient and causes severe instability during off-policy learning.

We finally comment on the choice of a good default value for β that achieves a good balance between regularization and the default reward. For the squared regularization, note that its gradient scales as $\mathcal{O}(T)$. As a result, in order to have an effective regularization at a constant scale, we require $\beta = \mathcal{O}(1/T)$. We also note that if the updates are per token rather than per sequence [Grinsztajn et al., 2024], the scale of β will be roughly $\mathcal{O}(1)$. We also discuss the subtle technical details of AGRO implementation at the token level in Appendix C.



(a) Evaluation performance for off-policy experiments



(b) Ablation on the effect of regularization

Figure 3: Figure (a): Evaluation performance for off-policy (offline) learning. We compare two algorithms: KL-regularized policy gradient (green) and off-policy AGRO (blue). We see that as training progresses, AGRO seems to obtain better overall performance than KL-regularized policy gradient. Figure (b): Ablation on the regularization coefficient β. We see that as β increases, all algorithms generally become more KL-efficient as they are more conservative with the updates. However, they also tend to deviate less from the origin reference policy π_{ref} with a fixed number of updates.

8 DISCUSSION AND LIMITATION

We have proposed AGRO, a fine-tuning algorithm leveraging both on-policy and off-policy data for regularized policy optimization. AGRO enjoys theoretical guarantees and yields compelling performance improvements over baseline methods in a few data generating settings.

We note that the off-policy learning enabled by AGRO is not omnipotent and that learning can still be unstable due to off-policy data generation. As a result, future study includes a more careful investigation into the source of off-policy instability and how one might leverage orthogonal techniques such as importance sampling.

References

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

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms, 2024b. URL https://arxiv.org/abs/2402.14740.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

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.

Daniele Calandriello, Daniel Guo, Remi Munos, Mark Rowland, Yunhao Tang, Bernardo Avila Pires, Pierre Harvey Richemond, Charline Le Lan, Michal Valko, Tianqi Liu, et al. Human alignment of large language models through online preference optimisation. *arXiv preprint arXiv:2403.08635*, 2024.

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.

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.

Yannis Flet-Berliac, Nathan Grinsztajn, Florian Strub, Eugene Choi, Chris Cremer, Arash Ahmadian, Yash Chandak, Mohammad Gheshlaghi Azar, Olivier Pietquin, and Matthieu Geist. Contrastive policy gradient: Aligning llms on sequence-level scores in a supervised-friendly fashion. *arXiv preprint arXiv:2406.19185*, 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 Sanjoy Dasgupta, Stephan Mandt, and Yingzhen Li, editors, *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, volume 238 of *Proceedings of Machine Learning Research*, pages 4447–4455. PMLR, 02–04 May 2024. URL https://proceedings.mlr.press/v238/gheshlaghi-azar24a.html.

P Glasserman. Monte carlo methods in financial engineering, 2004.

Nathan Grinsztajn, Yannis Flet-Berliac, Mohammad Gheshlaghi Azar, Florian Strub, Bill Wu, Eugene Choi, Chris Cremer, Arash Ahmadian, Yash Chandak, Olivier Pietquin, et al. Averaging log-likelihoods in direct alignment. *arXiv preprint arXiv:2406.19188*, 2024.

Shangmin Guo, Biao Zhang, Tianlin Liu, Tianqi Liu, Misha Khalman, Felipe Llinares, Alexandre Rame, Thomas Mesnard, Yao Zhao, Bilal Piot, et al. Direct language model alignment from online ai feedback. *arXiv preprint arXiv:2402.04792*, 2024.

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

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Wouter Kool, Herke van Hoof, and Max Welling. Buy 4 reinforce samples, get a baseline for free! 2019.

Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, Amit Kumar, Sergiu Ivanov, Jason K. Moore, Sartaj Singh, Thilina Rathnayake, Sean Vig, Brian E. Granger, Richard P. Muller, Francesco Bonazzi, Harsh Gupta, Shivam Vats, Fredrik Johansson, Fabian Pedregosa, Matthew J. Curry, Andy R. Terrel, Štěpán Roučka, Ashutosh Saboo, Isuru Fernando, Sumith Kulal, Robert Cimrman, and Anthony Scopatz. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103, January 2017. ISSN 2376-5992. doi: 10.7717/peerj-cs.103. URL https://doi.org/10.7717/peerj-cs.103.

Rémi Munos, Michal Valko, Daniele Calandriello, Mohammad Gheshlaghi Azar, Mark Rowland, Zhaohan Daniel Guo, Yunhao Tang, Matthieu Geist, Thomas Mesnard, Andrea Michi, et al. Nash

learning from human feedback. *arXiv preprint arXiv:2312.00886*, 2023.

Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. Bridging the gap between value and policy based reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 2775–2785, 2017.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

Brendan O'Donoghue. On the connection between bregman divergence and value in regularized markov decision processes. *arXiv preprint arXiv:2210.12160*, 2022.

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.

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.

Pierre Harvey Richemond, Yunhao Tang, Daniel Guo, Daniele Calandriello, Mohammad Gheshlaghi Azar, Rafael Rafailov, Bernardo Avila Pires, Eugene Tarassov, Lucas Spangher, Will Ellsworth, et al. Offline regularised reinforcement learning for large language models alignment. *arXiv preprint arXiv:2405.19107*, 2024.

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, 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.

Fahim Tajwar, Anikait Singh, Archit Sharma, Rafael Rafailov, Jeff Schneider, Tengyang Xie, Stefano Ermon, Chelsea Finn, and Aviral Kumar. Preference fine-tuning of llms should leverage suboptimal. *On-Policy Data*, 2024.

Yunhao Tang, Rémi Munos, Mark Rowland, and Michal Valko. Va-learning as a more efficient alternative to q-learning. In *International Conference on Machine Learning*, pages 33739–33757. PMLR, 2023.

Yunhao Tang, Daniel Zhaohan Guo, Zeyu Zheng, Daniele Calandriello, Yuan Cao, Eugene Tarassov, Rémi Munos, Bernardo Ávila Pires, Michal Valko, Yong Cheng, et al. Understanding the performance gap between online and offline alignment algorithms. *arXiv preprint arXiv:2405.08448*, 2024a.

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

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.

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. *arXiv preprint arXiv:2404.10719*, 2024.

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.

Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.

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.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See the Discussion and Limitation section.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] This is a foundational RL algorithm contribution for language model fine-tuning with no direct negative societal impact.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] See the appendix.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 7 and the appendix.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See the appendix.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

APPENDICES: RL-finetuning LLMs from on- and off-policy data with a single algorithm

A KL-efficiency comparison of on-policy and off-policy

We consider a synthetic off-policy learning setting where we keep a buffer of samples seen during on-policy sampling, and replay the the data at a probability $p = 0.5$ to randomly replace the on-policy samples. Notice that conceptually at the extreme when $p = 0$ where no on-policy sample is replayed, the off-policyness is made extreme and we recover the offline setting.

As before, Figure 4 shows the training performance as the training progresses. We plot against the KL divergence and compare the off-policy performance against the on-policy method. We make a few observations: (1) The

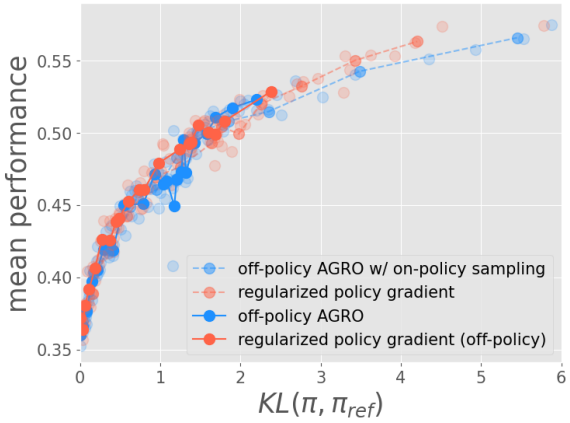


Figure 4: Comparison of on–policy vs off-policy algorithms. Within the same number of updates, on-policy algorithms generally deviate more from the reference policy π_{ref}, leading to larger increase in training performance. However, the KL-performance curve traced out by the off-policy algorithms aligns with the on-policy algorithms, indicating that we do not suffer any KL inefficiency despite being off-policy.

off-policy learning process is generally less data-efficient than the on-policy alternatives. This might be explained by the fact that the on-policy samples generally help drive progress faster than off-policy samples; (2) The off-policy learning is as KL-efficient as on-policy learning, though within the same amount of learning steps, on-policy learning drives up the KL divergence more, hence making faster progress to the training performance; (3) There is not significant difference between the off-policy AGRO algorithm and regularized policy gradient, despite subtle difference in the regularization loss.

B Additional theoretical results

B.1 Equivalence between KL divergence and performance under regularized reward

We have hinted at the connection between the KL divergence $\mathbb{KL}(\pi, \pi^*)$ and the performance under regularized reward. We make the statement formal below, which is a special case of [O'Donoghue, 2022]. Whenever the context is clear we drop the dependency on x.

Lemma 2. *The KL divergence* $\mathbb{KL}(\pi, \pi^*) \stackrel{def}{=} \mathbb{E}_{x\sim\rho}\left[\mathbb{KL}(\pi(\cdot|x), \pi^*(\cdot|x))\right]$ *is related to the performance* $\mathcal{G}(\pi) \stackrel{def}{=} \mathbb{E}_{x\sim\rho, y\sim\pi}\left[R^{\pi}_{\beta}(x, y)\right]$ *under the regularized policy as*

$$\beta\mathbb{KL}(\pi, \pi^*) = \mathcal{G}(\pi^*) - \mathcal{G}(\pi).$$

Proof. By the property of the optimal policy $\pi^*(y|x) = \pi_{\text{ref}}(y|x)\exp(\beta^{-1}r(x, y))/\exp(\beta^{-1}\widetilde{V}^{\pi^*}(x))$, we have

$$\beta\mathbb{KL}(\pi, \pi^*) = \mathbb{E}_{x\sim\rho, y\sim\pi(\cdot|x)}\left[-r(x, y) + \beta\log\frac{\pi(y|x)}{\pi_{\text{ref}}(y|x)} + \widetilde{V}^{\pi^*}(x)\right] = -\mathcal{G}(\pi) + \mathbb{E}_{x\sim\rho}\left[\widetilde{V}^{\pi^*}(x)\right].$$

We conclude the proof by noticing that $\mathbb{E}_{x\sim\rho}\left[\widetilde{V}^{\pi^*}(x)\right] = \mathbb{E}_{x\sim\rho}\left[V^{\pi^*}(x) - \beta\mathbb{KL}(\pi(\cdot|x), \pi^*(\cdot|x))\right] = \mathcal{G}(\pi^*)$. □

B.2 Further variance reduction for estimating $\nabla_{\text{LR}}\mathcal{L}(\pi)$

In Appendix C we introduced token-level gradient estimates to reduce the variance of estimating the gradients $\nabla_{\text{PD}}\mathcal{L}$ and $\nabla\mathcal{G}$. Here we focus on reducing the variance of $\nabla_{\text{LR}}\mathcal{L}$.

The cross product terms of the quadratic can be expressed as a function of $\mathbb{E}_{y\sim\pi}\left[\log\frac{\pi(y)}{\pi_{\text{ref}}(y)}\nabla\log\pi(y)\right]$ thus can be handled using the estimates discussed in Appendix C. Now we focus on reducing the variance of estimating

the squared term

$$g(\pi) \stackrel{\text{def}}{=} \mathbb{E}_{y\sim\pi}\left[\left(\log\frac{\pi(y)}{\pi_{\text{ref}}(y)}\right)^2\nabla\log\pi(y)\right].$$

We have

$$g(\pi) \quad = \quad \mathbb{E}_{y\sim\pi}\left[\sum_{t=1}^T\sum_{t'=1}^T\sum_{s=1}^T\log\frac{\pi(y_t)}{\pi_{\text{ref}}(y_t)}\log\frac{\pi(y_{t'})}{\pi_{\text{ref}}(y_{t'})}\nabla\log\pi(y_s)\right].$$

The triple sum $\sum_{t=1}^T\sum_{t'=1}^T\sum_{s=1}^T$ can be decomposed as

- the sum over terms $1\le t,t'<s\le T$ which are zero, since $\mathbb{E}_{y_s}\left[\nabla\log\pi(y_s|y_{1:s-1})\right]=0$,
- the sum over terms $1\le s\le t,t'\le T$, which can be regrouped as $\sum_{s=1}^T\nabla\log\pi(y_s)\left(\sum_{t=s}^T\log\frac{\pi(y_t)}{\pi_{\text{ref}}(y_t)}\right)^2$
- the sum over terms $1\le t<s<t'\le T$ and $1\le t'<s<t\le n$, which can benefit from computing explicitly one-step expectations (similarly as what has been done in Appendix C), which gives

$$2\sum_{t=1}^T\log\frac{\pi(y_t)}{\pi_{\text{ref}}(y_t)}\sum_{s=t+1}^T\nabla\log\pi(y_s)\sum_{t'=s+1}^T\mathbb{KL}(\pi_{t'},\pi_{\text{ref},t'}).$$

Putting everything together, the low-variance estimate of $g(\pi)$ is

$$\hat{g} \quad = \quad \sum_{s=1}^T\nabla\log\pi(y_s)\left[\left(\sum_{t=s}^T\log\frac{\pi(y_t)}{\pi_{\text{ref}}(y_t)}\right)^2+2\sum_{t=1}^{s-1}\log\frac{\pi(y_t)}{\pi_{\text{ref}}(y_t)}\sum_{t'=s+1}^T\mathbb{KL}(\pi_{t'},\pi_{\text{ref},t'})\right].$$

C Implementation of AGRO at the token level

In the previous sections, we have expressed the gradient of our losses using sequence-level generations $y\sim\pi(\cdot)$ (in this section we omit writing the dependence on the prompt x for simplicity). However, for LLM applications, the response y is usually produced as a sequence of individual decisions (tokens) $(y_t)_{1\le t\le T}$ generated in an auto-regressive way, i.e., $y_t\sim\pi(\cdot|y_{1:t-1})$, such that $\pi(y)=\pi(y_1)\pi(y_2|y_1)\dots\pi(y_T|y_{1:T-1})$ [Ouyang et al., 2022]. Using simplified notations, we write: $\pi(y)=\pi(y_1)\pi(y_2)\dots\pi(y_T)$ and define $\pi_t=\pi(\cdot|y_{1:t-1})$.

Now, we show that we can use this specific token-level form to derive lower-variance gradient estimates. In particular we focus on the term

$$g(\pi)\stackrel{\text{def}}{=}\mathbb{E}_{y\sim\pi}\left[\log\frac{\pi(y)}{\pi_{\text{ref}}(y)}\nabla\log\pi(y)\right],$$

which appears in the gradients $\nabla_{\text{PD}}\mathcal{L}(\pi)$ and $\nabla\mathcal{G}(\pi)$, as well as in the cross product of the quadratic term in $\nabla_{\text{LR}}\mathcal{L}(\pi)$.

We can build three estimate of $g(\pi)$:

$$\begin{aligned}
\hat{g}_1 &= \sum_{t=1}^T\sum_{t'=1}^T\nabla\log\pi(y_t)\log\frac{\pi(y_{t'})}{\pi_{\text{ref}}(y_{t'})}\\
\hat{g}_2 &= \sum_{t=1}^T\sum_{t'=t}^T\nabla\log\pi(y_t)\log\frac{\pi(y_{t'})}{\pi_{\text{ref}}(y_{t'})}\\
\hat{g}_3 &= \sum_{t=1}^T\nabla\log\pi(y_t)\Big(\log\frac{\pi(y_t)}{\pi_{\text{ref}}(y_t)}+\sum_{t'=t+1}^T\mathbb{KL}(\pi_{t'},\pi_{\text{ref},t'})\Big).
\end{aligned}$$

These three estimates are unbiased. In fact $\hat{g}_1$ is just the vanilla estimate of $g(\pi)$, thus $\mathbb{E}_{y\sim\pi}[\hat{g}_1]=g(\pi)$.

Now $\hat{g}_2$ ignores the lower-triangular terms of $\hat{g}_1$, which in expectation are zero, since, for $t' < t$, the term $\mathbb{E}_{y\sim\pi}\left[\nabla \log \pi(y_t) \log \frac{\pi(y_{t'})}{\pi_{\text{ref}}(y_{t'})}\right]$ equals

$$\mathbb{E}_{y_{1:t-1}\sim\pi}\Big[\log \frac{\pi(y_{t'})}{\pi_{\text{ref}}(y_{t'})} \underbrace{\mathbb{E}_{y_t\sim\pi}\left[\nabla \log \pi(y_t)|y_{1:t-1}\right]}_{=0}\Big] = 0.$$

Thus by ignoring these terms $\hat{g}_2$ reduces the variance of $\hat{g}_1$.

Finally, $\hat{g}_3$ further reduce the variance of $\hat{g}_2$ by explicitly computing the expectation over each single next token (information that is usually available in an LLM since the probability of all the immediate next tokens are output simultaneously by the network). Indeed, for $t' > t$, the term $\mathbb{E}_{y\sim\pi}\left[\nabla \log \pi(y_t) \log \frac{\pi(y_{t'})}{\pi_{\text{ref}}(y_{t'})}\right]$ equals

$$\begin{aligned}
&\mathbb{E}_{y_{1:t'-1}\sim\pi}\left[\nabla \log \pi(y_t) \mathbb{E}_{y_{t'}\sim\pi}\left[\log \frac{\pi(y_{t'})}{\pi_{\text{ref}}(y_{t'})}\Big| y_{1:t'-1}\right]\right] \\
&= \mathbb{E}_{y_{1:t'-1}\sim\pi}\left[\nabla \log \pi(y_t) \mathbb{KL}(\pi_{t'}, \pi_{\text{ref},t'})\right].
\end{aligned}$$

Thus, we can define several variants of the AGRO algorithm (as well as the regularized policy gradient algorithm minimizing $\mathcal{G}$) by implementing the gradient of the losses $\nabla_{\text{PD}}\mathcal{L}$, $\nabla\mathcal{G}$, $\nabla\mathcal{L}_\mu$ and $\nabla_{\text{LR}}\mathcal{L}$, using either the naive gradient estimate $\hat{g}_1$, the one that removes the lower-triangular terms $\hat{g}_2$, or $\hat{g}_3$ to further reduce the variance by computing explicitly the expectation over the immediate next tokens.

Notice that in the off-policy case, a similar variance reduction technique can be applied for $\nabla\mathcal{L}_\mu(\pi)$ by estimating the term $\mathbb{E}_{y\sim\mu}\left[\log \frac{\pi(y)}{\pi_{\text{ref}}(y)} \nabla \log \pi(y)\right]$ by $\sum_{t=1}^{T} \nabla \log \pi(y_t) \sum_{t'=t}^{T} \log \frac{\pi(y_t)}{\pi_{\text{ref}}(y_t)}$. Finally for interested readers, in Appendix B.2, we describe a further variance reduction technique to estimate the quadratic term $\mathbb{E}_{y\sim\pi}\left[\left(\log \frac{\pi(y)}{\pi_{\text{ref}}(y)}\right)^2 \nabla \log \pi(y)\right]$ that appears in $\nabla_{\text{LR}}\mathcal{L}$.

We note that much of the discussion here extends to fine-tuning algorithms in prior work [Rafailov et al., 2024, Gheshlaghi Azar et al., 2024, Calandriello et al., 2024, Richemond et al., 2024], where sequence level algorithm requires careful adaptation to the token level implementation.