

## Lecture 13: Reinforcement Learning with Verifiable Rewards

Prof. Noah Golowich

This lecture applies policy gradients to language models trained with verifiable rewards. We study four algorithms: RLOO, Dr. GRPO, GRPO, and MaxRL.

## 1 RLVR, REINFORCE, and PPO

For some tasks, such as solving a math problem or writing a program that must pass a set of tests, a verifier can check whether a completed response is correct. *Reinforcement learning with verifiable rewards* (RLVR) trains a language model on this feedback, so we do not need a target reasoning trace for each prompt [4].

### 1.1 Generation and the objective

Let  $x \sim \mathcal{D}$  be a prompt drawn from a fixed distribution of training questions. A language model with parameters  $\theta \in \mathbb{R}^d$  generates a response  $y = (y_1, \dots, y_T)$  one token at a time:

$$y_t \sim \pi_\theta(\cdot \mid x, y_{1:t-1}), \quad t = 1, \dots, T.$$

Here  $y_{1:t} = (y_1, \dots, y_t)$ , and  $y_{1:0}$  is the empty prefix. Tokens come from a finite vocabulary  $\mathcal{V}$  that contains an end-of-sequence token. Generation stops when this token is generated (it is included in  $y$ ) or after a fixed budget of  $T_{\max}$  tokens, so the length  $T = T(y) \leq T_{\max}$  varies across responses. The probability of a complete response is

$$\pi_\theta(y \mid x) = \prod_{t=1}^T \pi_\theta(y_t \mid x, y_{1:t-1}). \quad (1)$$

This is a sequential decision problem (i.e., can be viewed as a token-level MDP): the state is the prompt together with the tokens generated so far, the action is the next token, and the transition appends that token to the state. Once  $x$  is drawn, all of the randomness comes from the policy.

Throughout, the verifier gives an *outcome-level* reward  $R(x, y) \in \{0, 1\}$  after the response ends, where  $R(x, y) = 1$  means that  $y$  passes the verifier. There are no intermediate rewards (though the framework may be readily adapted to include them). The success probability on a prompt and the expected-reward objective are

$$p_\theta(x) := \mathbb{E}_{y \sim \pi_\theta(\cdot \mid x)}[R(x, y)], \quad J(\theta) := \mathbb{E}_{x \sim \mathcal{D}}[p_\theta(x)]. \quad (2)$$

We want to maximize  $J$ . We leave out KL regularization until we reach GRPO.

### 1.2 The REINFORCE estimator

We first recall some basic policy gradient methods. By (1), the gradient of the log-probability of a response (its *score*) is a sum of token scores:

$$\nabla_\theta \log \pi_\theta(y \mid x) = \sum_{t=1}^T \nabla_\theta \log \pi_\theta(y_t \mid x, y_{1:t-1}). \quad (3)$$

Applying  $\nabla p = p \nabla \log p$  gives

$$\nabla_{\theta} p_{\theta}(x) = \sum_y R(x, y) \nabla_{\theta} \pi_{\theta}(y | x) = \mathbb{E}_{y \sim \pi_{\theta}(\cdot | x)} [R(x, y) \nabla_{\theta} \log \pi_{\theta}(y | x)], \quad (4)$$

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{x \sim \mathcal{D}, y \sim \pi_{\theta}(\cdot | x)} \left[ \sum_{t=1}^T R(x, y) \nabla_{\theta} \log \pi_{\theta}(y_t | x, y_{1:t-1}) \right]. \quad (5)$$

So for a sampled prompt and response,  $R(x, y) \nabla_{\theta} \log \pi_{\theta}(y | x)$  is an unbiased estimate of  $\nabla_{\theta} J(\theta)$ . This is the *REINFORCE* estimator [1], and averaging independent samples reduces its variance. In particular, a successful response contributes the sum of its token scores and a failed response contributes nothing.

### 1.3 Values, advantages, and variance reduction

To evaluate a prefix, we ask how likely it is to pass the verifier once the current policy completes it. For any policy  $\pi$ , define

$$V^{\pi}(x, y_{1:t-1}) := \mathbb{E}_{\text{continuation from } \pi} [R(x, y) | x, y_{1:t-1}], \quad (6)$$

$$A^{\pi}(x, y_{1:t-1}, a) := \mathbb{E}_{\text{continuation from } \pi} [R(x, y) | x, y_{1:t-1}, y_t = a] - V^{\pi}(x, y_{1:t-1}). \quad (7)$$

In these conditional expectations the displayed prefix is fixed and only the remaining tokens are sampled. For a completed response the value is just  $R(x, y)$ , and  $V^{\pi_{\theta}}(x) = p_{\theta}(x)$ . The *advantage* measures how much choosing the token  $a$  changes the expected outcome, compared with sampling the next token from  $\pi$ ; its average over  $a \sim \pi(\cdot | x, y_{1:t-1})$  is zero.

**Baselines do not change the gradient.** Fix a nonterminal prefix  $(x, y_{1:t-1})$ . The next-token score has conditional mean zero:

$$\begin{aligned} & \mathbb{E}_{y_t \sim \pi_{\theta}(\cdot | x, y_{1:t-1})} [\nabla_{\theta} \log \pi_{\theta}(y_t | x, y_{1:t-1})] \\ &= \sum_{a \in \mathcal{V}} \pi_{\theta}(a | x, y_{1:t-1}) \nabla_{\theta} \log \pi_{\theta}(a | x, y_{1:t-1}) = \nabla_{\theta} \sum_{a \in \mathcal{V}} \pi_{\theta}(a | x, y_{1:t-1}) = 0. \end{aligned} \quad (8)$$

So for any baseline  $b(x, y_{1:t-1})$  that depends only on the prefix,

$$\mathbb{E}[b(x, y_{1:t-1}) \nabla_{\theta} \log \pi_{\theta}(y_t | x, y_{1:t-1}) | x, y_{1:t-1}] = 0,$$

and subtracting the baseline from  $R(x, y)$  in each term of (5) leaves the gradient unchanged:

$$\nabla_{\theta} J(\theta) = \mathbb{E} \left[ \sum_{t=1}^T (R(x, y) - b(x, y_{1:t-1})) \nabla_{\theta} \log \pi_{\theta}(y_t | x, y_{1:t-1}) \right]. \quad (9)$$

Because responses have different lengths, we apply (8) only at prefixes reached before the response ends; equivalently, we set the score terms after termination to zero.

With  $b = V^{\pi_{\theta}}$ , the weight  $R(x, y) - V^{\pi_{\theta}}(x, y_{1:t-1})$  is a Monte Carlo estimate of the advantage: conditional on  $(x, y_{1:t})$ , its expectation is  $A^{\pi_{\theta}}(x, y_{1:t-1}, y_t)$ . Conditioning each term of (9) on  $(x, y_{1:t})$  therefore gives

$$\nabla_{\theta} J(\theta) = \mathbb{E} \left[ \sum_{t=1}^T A^{\pi_{\theta}}(x, y_{1:t-1}, y_t) \nabla_{\theta} \log \pi_{\theta}(y_t | x, y_{1:t-1}) \right]. \quad (10)$$

In practice, a learned value model  $V_{\phi}(x, y_{1:t-1})$  supplies an approximate baseline. Its accuracy affects the variance of the Monte Carlo estimator but not its expectation, as long as the baseline does not depend on the sampled continuation. (Fitting  $V_{\phi}$  on the same responses can create such a dependence, which the argument above does not cover.) When we update the policy, we treat the baseline values as constants and differentiate only the policy log-probabilities.

**Baselines can reduce variance.** Fix a prefix  $(x, y_{1:t-1})$ ; throughout this paragraph, we condition on it. Since  $\mathbb{E}[(R(x, y) - b)\nabla_{\theta} \log \pi_{\theta}(y_t | x, y_{1:t-1})]$  does not depend on the scalar  $b$ , the variance of the token- $t$  term, summed over coordinates, is

$$\begin{aligned} \text{tr Cov}\left((R(x, y) - b)\nabla_{\theta} \log \pi_{\theta}(y_t | x, y_{1:t-1})\right) \\ = \mathbb{E}\left[(R(x, y) - b)^2 \|\nabla_{\theta} \log \pi_{\theta}(y_t | x, y_{1:t-1})\|_2^2\right] - \|\mathbb{E}[R(x, y)\nabla_{\theta} \log \pi_{\theta}(y_t | x, y_{1:t-1})]\|_2^2. \end{aligned}$$

Only the first term depends on  $b$ . It is a quadratic in  $b$ , and it is minimized at

$$b = \frac{\mathbb{E}[R(x, y) \|\nabla_{\theta} \log \pi_{\theta}(y_t | x, y_{1:t-1})\|_2^2]}{\mathbb{E}[\|\nabla_{\theta} \log \pi_{\theta}(y_t | x, y_{1:t-1})\|_2^2]}. \quad (11)$$

If the score norm  $\|\nabla_{\theta} \log \pi_{\theta}(a | x, y_{1:t-1})\|_2$  is the same for every token  $a$ , this minimizer is  $\mathbb{E}[R(x, y)] = V^{\pi_{\theta}}(x, y_{1:t-1})$ . Since  $R(x, y) \in \{0, 1\}$ , the expected squared reward factor then drops from  $\mathbb{E}[R(x, y)^2] = V^{\pi_{\theta}}(x, y_{1:t-1})$  to  $\text{Var}(R(x, y)) = V^{\pi_{\theta}}(x, y_{1:t-1})(1 - V^{\pi_{\theta}}(x, y_{1:t-1}))$ . A nearly certain success then produces a small update, instead of reinforcing every token just because the prompt was easy.

In general the value only approximates the minimizer (11), and a poorly chosen baseline can even increase the variance. The calculation also concerns a single token's term; the variance of the sum over  $t$  includes covariances between tokens.

## 1.4 PPO and reuse of sampled responses

REINFORCE uses *on-policy* responses, sampled from the same policy  $\pi_{\theta}$  at which we evaluate the gradient. PPO [2] instead samples a batch from a frozen policy  $\pi_{\text{old}} := \pi_{\theta_{\text{old}}}$  and takes several optimization steps on that batch. As soon as  $\theta$  changes, the responses are *off-policy*. For the  $t$ th token of a sampled response  $y$ , define the ratio

$$\rho_t(\theta; x, y) := \frac{\pi_{\theta}(y_t | x, y_{1:t-1})}{\pi_{\text{old}}(y_t | x, y_{1:t-1})}. \quad (12)$$

The exact importance weight of the whole response is the product  $\pi_{\theta}(y | x)/\pi_{\text{old}}(y | x) = \prod_{t=1}^T \rho_t(\theta; x, y)$ , which can have high variance for long responses. PPO instead uses a local surrogate built from the individual token ratios.

For a clipping parameter  $\epsilon \in (0, 1)$ , define

$$\ell_{\epsilon}(\rho, A) := \min\{\rho A, \text{clip}(\rho, 1 - \epsilon, 1 + \epsilon)A\}, \quad (13)$$

where  $\text{clip}(u, a, b) = \min\{\max\{u, a\}, b\}$ . Using a value model  $V_{\phi}$  that is frozen during the policy update, the PPO policy objective is

$$L_{\text{PPO}}(\theta) := \mathbb{E}_{x \sim \mathcal{D}, y \sim \pi_{\text{old}}(\cdot | x)} \left[ \sum_{t=1}^T \ell_{\epsilon}(\rho_t(\theta; x, y), R(x, y) - V_{\phi}(x, y_{1:t-1})) \right]. \quad (14)$$

We maximize  $L_{\text{PPO}}$  over  $\theta$  while holding the sampled responses,  $\pi_{\text{old}}$ , and  $V_{\phi}$  fixed. If  $V_{\phi} = V^{\pi_{\text{old}}}$ , then  $R(x, y) - V_{\phi}(x, y_{1:t-1})$  is a conditionally unbiased estimate of  $A^{\pi_{\text{old}}}(x, y_{1:t-1}, y_t)$ . For other value models it is still a valid baseline at  $\theta = \theta_{\text{old}}$ , under the independence condition above. PPO is often run with alternative advantage estimates (e.g., as obtained by generalized advantage estimation) instead, and these add bias when the value model is inaccurate.

At  $\theta = \theta_{\text{old}}$ , every ratio equals 1 and clipping is inactive. Since  $\nabla_{\theta} \rho_t(\theta; x, y)$  equals  $\nabla_{\theta} \log \pi_{\theta}(y_t | x, y_{1:t-1})$  at  $\theta = \theta_{\text{old}}$  (where gradients don't flow through  $\theta_{\text{old}}$ ), the gradient of  $L_{\text{PPO}}$  at  $\theta_{\text{old}}$  is the right-hand side of (9) with  $b = V_{\phi}$ , which is  $\nabla_{\theta} J(\theta_{\text{old}})$ . Away from  $\theta_{\text{old}}$ , the gradient of the surrogate is generally a biased estimate of  $\nabla_{\theta} J$ , since the prefixes still come from  $\pi_{\text{old}}$  and clipping modifies the update. After several steps, PPO samples a fresh batch and resets  $\pi_{\text{old}}$  to the current policy.

## 2 Group-based estimators: RLOO, Dr. GRPO, and GRPO

A value model has to estimate the success probability at every prefix of a response. Instead, we can sample several complete responses to the same prompt and build a baseline from their rewards. Then there is no value model to train.

### 2.1 REINFORCE leave-one-out (RLOO)

Fix a prompt  $x$  and draw  $G \geq 2$  independent responses  $y^{(1)}, \dots, y^{(G)} \sim \pi_\theta(\cdot | x)$ , with rewards  $R_i := R(x, y^{(i)})$  and group mean  $\bar{R} := \frac{1}{G} \sum_{j=1}^G R_j$ . RLOO uses the average reward of the other  $G - 1$  responses as the baseline for response  $i$  [3]:

$$\hat{g}_{\text{RLOO}}(x) := \frac{1}{G} \sum_{i=1}^G \left( R_i - \frac{1}{G-1} \sum_{j \neq i} R_j \right) \nabla_\theta \log \pi_\theta(y^{(i)} | x). \quad (15)$$

By (3), the same scalar multiplies every token score in response  $i$ . The baseline depends on the prompt and the other responses but not on the prefix of  $y^{(i)}$ , so it does not estimate the value at each prefix.

To check unbiasedness, condition on  $x$ . For  $j \neq i$ , the responses  $y^{(i)}$  and  $y^{(j)}$  are independent, and the score of a response has mean zero, since  $\sum_y \pi_\theta(y | x) \nabla_\theta \log \pi_\theta(y | x) = \nabla_\theta \sum_y \pi_\theta(y | x) = 0$ . Hence

$$\mathbb{E}[R_j \nabla_\theta \log \pi_\theta(y^{(i)} | x) | x] = \mathbb{E}[R_j | x] \mathbb{E}[\nabla_\theta \log \pi_\theta(y^{(i)} | x) | x] = p_\theta(x) \cdot 0 = 0.$$

Combining this with (4) shows

$$\mathbb{E}[\hat{g}_{\text{RLOO}}(x) | x] = \nabla_\theta p_\theta(x). \quad (16)$$

The argument needs the baseline for response  $i$  to leave out  $R_i$ , because  $R_i$  and  $\nabla_\theta \log \pi_\theta(y^{(i)} | x)$  are dependent. The leave-one-out baseline is an average of  $G - 1$  independent Bernoulli( $p_\theta(x)$ ) rewards, so it has mean  $p_\theta(x)$  and variance  $p_\theta(x)(1 - p_\theta(x))/(G - 1)$ ; larger groups estimate the prompt's success probability more accurately. Each response contributes its own gradient term and also enters the baselines of the others. The  $G$  terms in (15) are therefore dependent, and the variance of  $\hat{g}_{\text{RLOO}}(x)$  is not just the variance of one term divided by  $G$ .

### 2.2 Dr. GRPO: including the response in its own baseline

Dr. GRPO replaces the leave-one-out average with the mean of the whole group, including response  $i$  itself [4, Appendix A]:

$$\hat{g}_{\text{Dr}}(x) := \frac{1}{G} \sum_{i=1}^G (R_i - \bar{R}) \nabla_\theta \log \pi_\theta(y^{(i)} | x). \quad (17)$$

The group mean puts weight  $1/G$  on  $R_i$  and weight  $(G - 1)/G$  on the average of the other rewards, so for every sample,

$$R_i - \bar{R} = \frac{G-1}{G} \left( R_i - \frac{1}{G-1} \sum_{j \neq i} R_j \right).$$

Comparing with (15) and (16),

$$\hat{g}_{\text{Dr}}(x) = \frac{G-1}{G} \hat{g}_{\text{RLOO}}(x), \quad \mathbb{E}[\hat{g}_{\text{Dr}}(x) | x] = \frac{G-1}{G} \nabla_\theta p_\theta(x). \quad (18)$$

So Dr. GRPO is RLOO with the gradient scaled by  $1 - 1/G$ . The bias does not change the direction of the gradient, and multiplying by  $G/(G - 1)$  removes it. For a fixed  $G$ , ordinary gradient ascent can absorb this factor into the step size.

**The off-policy clipped objective.** In practice, we draw the group from  $\pi_{\text{old}}$ , compute its rewards once, and reuse it for several policy updates. The Dr. GRPO surrogate is

$$L_{\text{Dr}}(\theta) := \mathbb{E}_{x \sim \mathcal{D}, y^{(1:G)} \sim \pi_{\text{old}}(\cdot|x)} \left[ \frac{1}{G} \sum_{i=1}^G \sum_{t=1}^{T(y^{(i)})} \ell_{\epsilon}(\rho_t(\theta; x, y^{(i)}), R_i - \bar{R}) \right], \quad (19)$$

where  $y^{(1:G)}$  denotes the group of  $G$  independent responses and the coefficients  $R_i - \bar{R}$  are held fixed during differentiation. At  $\theta = \theta_{\text{old}}$ , differentiating the term inside the expectation gives exactly (17); later updates have the surrogate and clipping biases from Section 1.

### 2.3 GRPO and KL regularization

GRPO (introduced before Dr. GRPO) was originally proposed to avoid the memory and compute cost of a separate value model [5]. Its authors also point out that comparing each response with others for the same prompt matches how learned reward models are usually trained, on comparisons between responses to the same question. Here consider the special case of binary verifier rewards.

Besides subtracting the group mean, GRPO divides by the group standard deviation of the rewards,  $(\frac{1}{G} \sum_{j=1}^G (R_j - \bar{R})^2)^{1/2}$ . For binary rewards this equals  $\sqrt{\bar{R}(1 - \bar{R})}$ , so response  $i$  gets the advantage

$$\frac{R_i - \bar{R}}{\sqrt{\bar{R}(1 - \bar{R})}}, \quad (20)$$

and every token of the response is assigned this value. When all rewards in the group agree, the standard deviation is zero, and we set every advantage to zero.

GRPO also uses a reference policy  $\pi_{\text{ref}}$  with positive next-token probabilities, held fixed during the policy updates. Penalizing divergence from  $\pi_{\text{ref}}$  keeps the policy from drifting too far from this reference model. Do not confuse it with  $\pi_{\text{old}}$ , the policy that sampled the current batch. For  $\beta \geq 0$ , the original GRPO objective has the form [5, Eq. (3)]

$$L_{\text{GRPO}}(\theta) := \mathbb{E}_{x \sim \mathcal{D}, y^{(1:G)} \sim \pi_{\text{old}}(\cdot|x)} \left[ \frac{1}{G} \sum_{i=1}^G \frac{1}{T(y^{(i)})} \sum_{t=1}^{T(y^{(i)})} \left\{ \ell_{\epsilon} \left( \rho_t(\theta; x, y^{(i)}), \frac{R_i - \bar{R}}{\sqrt{\bar{R}(1 - \bar{R})}} \right) - \beta \left( \frac{\pi_{\text{ref}}(y_t^{(i)} | x, y_{1:t-1}^{(i)})}{\pi_{\theta}(y_t^{(i)} | x, y_{1:t-1}^{(i)})} - \log \frac{\pi_{\text{ref}}(y_t^{(i)} | x, y_{1:t-1}^{(i)})}{\pi_{\theta}(y_t^{(i)} | x, y_{1:t-1}^{(i)})} - 1 \right) \right\} \right]. \quad (21)$$

The term multiplying  $\beta$  is a sampled estimate of a KL divergence. It is nonnegative, since  $u - \log u - 1 \geq 0$  for  $u > 0$ . At a fixed prefix, if the next token is sampled from  $\pi_{\theta}$ , its expectation is

$$\begin{aligned} \sum_{a \in \mathcal{V}} \pi_{\theta}(a | x, y_{1:t-1}) \left[ \frac{\pi_{\text{ref}}(a | x, y_{1:t-1})}{\pi_{\theta}(a | x, y_{1:t-1})} - 1 + \log \frac{\pi_{\theta}(a | x, y_{1:t-1})}{\pi_{\text{ref}}(a | x, y_{1:t-1})} \right] \\ = D_{\text{KL}}(\pi_{\theta}(\cdot | x, y_{1:t-1}) \| \pi_{\text{ref}}(\cdot | x, y_{1:t-1})), \end{aligned}$$

because the first two terms in the brackets sum to  $1 - 1 = 0$ . This identity concerns the value of the penalty when the next token is sampled on-policy. With samples reused from  $\pi_{\text{old}}$ , the unweighted penalty in (21) is only a surrogate.

A group whose rewards all agree contributes no reward update, but its KL penalty can still contribute.

## 2.4 GRPO's two normalizations

Compared with the expected-reward objective, GRPO reweights the gradient in two ways, and Dr. GRPO removes both [4]. To isolate them, set  $\beta = 0$  and look at the gradient at  $\theta_{\text{old}}$ , where clipping is inactive. Each token score in response  $i$  is multiplied by

$$\frac{R_i - \bar{R}}{GT(y^{(i)})\sqrt{\bar{R}(1 - \bar{R})}} \quad \text{for GRPO}, \quad \frac{R_i - \bar{R}}{G} \quad \text{for Dr. GRPO}.$$

The factor  $1/T(y^{(i)})$  reweights responses by length. Among successful responses, it gives each token of a short response a larger positive coefficient than a token of a long one. Among failed responses, it shrinks the per-token penalty of long responses, which can encourage long incorrect responses. Removing  $1/T(y^{(i)})$  restores the gradient of the complete-response log-probability, which is a sum of token scores. A normalization shared by all responses, such as  $1/T_{\text{max}}$ , would not cause this problem.

The factor  $1/\sqrt{\bar{R}(1 - \bar{R})}$  reweights prompts by difficulty. For large groups,  $\bar{R}$  is close to  $p_\theta(x)$ , so this factor is large on very easy and very hard prompts. GRPO therefore amplifies the centered updates from these prompts relative to prompts of intermediate difficulty, and this changes how prompts are weighted in  $\nabla_\theta J(\theta) = \mathbb{E}_x[\nabla_\theta p_\theta(x)]$ . If every response in a finite group succeeds, or every one fails, the update is still zero, because  $R_i - \bar{R} = 0$  for every  $i$ .

These arguments were used to motivate Dr. GRPO when the goal is to maximize the unregularized expected reward  $J$ .

## 3 Maximum likelihood reinforcement learning (MaxRL)

MaxRL [6] maximizes the log-probability of success instead of the probability of success. As before, the response  $y$  contains every generated token, including any reasoning and the final answer, and  $R(x, y)$  checks only the completed response. In this section, all estimators use independent on-policy responses, with no KL penalty or clipping.

### 3.1 From expected reward to likelihood of success

Suppose  $p_\theta(x) > 0$  for every training prompt, and define

$$J_{\text{ML}}(\theta) := \mathbb{E}_{x \sim \mathcal{D}}[\log p_\theta(x)]. \quad (22)$$

The probability inside the logarithm sums over *all* accepted responses,  $p_\theta(x) = \sum_{y: R(x, y)=1} \pi_\theta(y | x)$ . If the verifier checks a final answer extracted deterministically from  $y$ , then  $p_\theta(x)$  is the marginal likelihood of the correct answer, summed over reasoning traces. In general, it is the probability that the verifier accepts.

For a single prompt,  $p_\theta(x)$  and  $\log p_\theta(x)$  both increase with the success probability. Across prompts that share parameters, however, the two gradients weight prompts differently:

$$\nabla_\theta J(\theta) = \mathbb{E}_x[\nabla_\theta p_\theta(x)], \quad \nabla_\theta J_{\text{ML}}(\theta) = \mathbb{E}_x \left[ \frac{\nabla_\theta p_\theta(x)}{p_\theta(x)} \right].$$

For example, with equally weighted prompts, raising one prompt's success probability from 0.01 to 0.02 increases  $J$  by the same amount as raising another's from 0.50 to 0.51. The log-likelihood increases by  $\log 2$  in the first case but only by  $\log(1.02)$  in the second. The logarithm rewards relative improvement, which puts more weight on hard prompts.

### 3.2 A finite-order approximation and $\text{pass@}k$

Fix a prompt  $x$ . For  $k \geq 1$ , let  $\text{pass@}k_\theta(x)$  be the probability that at least one of  $k$  independent responses from  $\pi_\theta(\cdot | x)$  passes the verifier. By independence,

$$\text{pass@}k_\theta(x) = 1 - (1 - p_\theta(x))^k. \quad (23)$$

Writing  $\log p_\theta(x) = \log(1 - (1 - p_\theta(x)))$  and expanding in powers of  $1 - p_\theta(x) \in [0, 1)$  gives

$$\log p_\theta(x) = - \sum_{k=1}^{\infty} \frac{(1 - p_\theta(x))^k}{k} = - \sum_{k=1}^{\infty} \frac{1 - \text{pass@}k_\theta(x)}{k}. \quad (24)$$

Truncating at order  $m \geq 1$  gives the per-prompt MaxRL objective and its average over prompts:

$$F_m(\theta; x) := - \sum_{k=1}^m \frac{(1 - p_\theta(x))^k}{k}, \quad J_{\text{MaxRL}}^{(m)}(\theta) := \mathbb{E}_x[F_m(\theta; x)]. \quad (25)$$

Equivalently,  $F_m(\theta; x) = \sum_{k=1}^m \text{pass@}k_\theta(x)/k - \sum_{k=1}^m 1/k$ , and the second sum does not depend on  $\theta$ . In particular,  $J_{\text{MaxRL}}^{(1)} = J - 1$ , so order 1 gives the usual expected-reward gradient. As  $m \rightarrow \infty$ ,  $F_m(\theta; x) \rightarrow \log p_\theta(x)$ .

Differentiating the finite sum and summing a geometric series yields

$$\nabla_\theta F_m(\theta; x) = \sum_{k=1}^m \frac{1}{k} \nabla_\theta \text{pass@}k_\theta(x) = \sum_{k=1}^m (1 - p_\theta(x))^{k-1} \nabla_\theta p_\theta(x) = \frac{1 - (1 - p_\theta(x))^m}{p_\theta(x)} \nabla_\theta p_\theta(x). \quad (26)$$

Letting  $m \rightarrow \infty$  in this explicit formula shows that

$$\nabla_\theta \log p_\theta(x) = \sum_{k=1}^{\infty} \frac{1}{k} \nabla_\theta \text{pass@}k_\theta(x),$$

so we never have to justify exchanging the derivative with the infinite sum. The coefficient of  $\nabla_\theta p_\theta(x)$  in (26) is at most  $\min\{m, 1/p_\theta(x)\}$ . It is about  $m$  when  $p_\theta(x)$  is much smaller than  $1/m$ , and it approaches the maximum-likelihood weight  $1/p_\theta(x)$  as  $m$  grows.

### 3.3 The maximum-likelihood gradient as a conditional expectation

Successful responses alone are enough to estimate the gradient of the log success probability. The following is Theorem 1 of Tajwar et al. [6], in our notation.

**Theorem 1** (Conditional maximum-likelihood gradient). *Fix a prompt  $x$  with  $p_\theta(x) > 0$ . Under the differentiability and finite-support assumptions of Section 1,*

$$\nabla_\theta \log p_\theta(x) = \mathbb{E}_{y \sim \pi_\theta(\cdot | x)}[\nabla_\theta \log \pi_\theta(y | x) | R(x, y) = 1]. \quad (27)$$

*Proof.* By (4) and  $\nabla \log p = \nabla p/p$ ,

$$\nabla_\theta \log p_\theta(x) = \frac{1}{p_\theta(x)} \sum_{y: R(x, y)=1} \pi_\theta(y | x) \nabla_\theta \log \pi_\theta(y | x).$$

Conditioned on success, an accepted response  $y$  has probability  $\pi_\theta(y | x)/p_\theta(x)$ , and every other response has probability zero. So the right-hand side is exactly the expectation in (27).  $\square$

One way to use this is to sample until some response succeeds and take its score. That estimate is unbiased for (27), but it needs  $1/p_\theta(x)$  responses on average, which is expensive on hard prompts. We would rather use a fixed number of responses per prompt.

### 3.4 A fixed sampling budget gives exactly a truncated objective

Draw  $G \geq 1$  independent responses to  $x$ , with rewards  $R_i$  and group mean  $\bar{R}$  as in Section 2. Consider the estimator

$$\hat{g}_G(x) := \begin{cases} \frac{1}{G\bar{R}} \sum_{i=1}^G R_i \nabla_{\theta} \log \pi_{\theta}(y^{(i)} | x), & \bar{R} > 0, \\ 0, & \bar{R} = 0, \end{cases} \quad (28)$$

which averages the scores of the successful responses. REINFORCE would divide the same sum by  $G$  instead of by the number of successes  $G\bar{R}$ . Because of the all-failure case,  $\hat{g}_G(x)$  is a biased estimate of  $\nabla_{\theta} \log p_{\theta}(x)$ . Its expectation is nonetheless exactly the gradient of a truncated objective.

**Theorem 2** (Estimator–objective equivalence; [6, Theorem 2]). *Fix a prompt  $x$  with  $p_{\theta}(x) > 0$  and an integer  $G \geq 1$ . Under the assumptions of Theorem 1, the estimator in (28) satisfies*

$$\mathbb{E}[\hat{g}_G(x) | x] = \nabla_{\theta} F_G(\theta; x) = \sum_{k=1}^G \frac{1}{k} \nabla_{\theta} \text{pass@} k_{\theta}(x). \quad (29)$$

Averaging it over sampled prompts therefore gives an unbiased estimator of  $\nabla_{\theta} J_{\text{MaxRL}}^{(G)}(\theta)$ .

*Proof.* Keep  $x$  fixed. First condition on the vector of success indicators  $(R_1, \dots, R_G)$ , for any pattern with positive probability and at least one success. Since the responses are independent, each response with  $R_i = 1$  is then distributed as a draw from  $\pi_{\theta}(\cdot | x)$  conditioned on success. Theorem 1 therefore gives

$$\begin{aligned} \mathbb{E} \left[ \frac{1}{G\bar{R}} \sum_{i=1}^G R_i \nabla_{\theta} \log \pi_{\theta}(y^{(i)} | x) \middle| x, R_1, \dots, R_G \right] &= \frac{1}{G\bar{R}} \sum_{i: R_i=1} \nabla_{\theta} \log p_{\theta}(x) \\ &= \nabla_{\theta} \log p_{\theta}(x), \end{aligned}$$

since exactly  $G\bar{R}$  of the indicators equal 1. On the event that every response fails, which has probability  $(1 - p_{\theta}(x))^G$ , the estimator is zero. Averaging over the indicators,

$$\mathbb{E}[\hat{g}_G(x) | x] = (1 - (1 - p_{\theta}(x))^G) \nabla_{\theta} \log p_{\theta}(x) = \frac{1 - (1 - p_{\theta}(x))^G}{p_{\theta}(x)} \nabla_{\theta} p_{\theta}(x),$$

which gives (29) by (26) with  $m = G$ . Taking an expectation over  $x \sim \mathcal{D}$  proves the last statement.  $\square$

If  $p_{\theta}(x) = 1$ , the same proof applies, and both sides are zero under our positive-probability model. If no response can pass the verifier, then  $p_{\theta}(x) = 0$  for every  $\theta$ , and both (28) and the finite-order gradient are zero. The log-likelihood and the conditioning on success are then undefined, so we exclude such prompts from the maximum-likelihood statements.

### 3.5 A baseline and the practical MaxRL update

The denominator  $G\bar{R}$  in (28) depends on the same responses as the numerator, so the usual baseline argument does not apply directly, and we have to recheck the expectation. One control variate with mean zero is the average score of all  $G$  responses:

$$\mathbb{E} \left[ \frac{1}{G} \sum_{i=1}^G \nabla_{\theta} \log \pi_{\theta}(y^{(i)} | x) \middle| x \right] = 0.$$

Subtracting it from  $\hat{g}_G(x)$  gives another unbiased estimator of  $\nabla_\theta F_G(\theta; x)$ . When every response fails, this estimator equals minus the average score rather than zero. The subtraction reduces variance only if the average score has a large enough positive covariance with  $\hat{g}_G(x)$ ; having mean zero is not enough.

The practical on-policy MaxRL algorithm [6] makes this subtraction only when at least one response succeeds, and skips groups in which every response fails. Since  $R_i/(G\bar{R}) - 1/G = (R_i - \bar{R})/(G\bar{R})$ , its update is

$$\hat{g}_{\text{practical}}(x) := \begin{cases} \frac{1}{G} \sum_{i=1}^G \frac{R_i - \bar{R}}{\bar{R}} \nabla_\theta \log \pi_\theta(y^{(i)} | x), & \bar{R} > 0, \\ 0, & \bar{R} = 0. \end{cases} \quad (30)$$

In words: sample a group and compute its mean reward. If the mean is positive, assign the coefficient  $(R_i - \bar{R})/\bar{R}$  to every token of response  $i$ . Then average the resulting score gradients over responses and prompts, holding the mean rewards and coefficients fixed during differentiation. Compared with Dr. GRPO, this update divides by the group mean. Compared with GRPO, it divides by the mean rather than the standard deviation, and it does not normalize by response length.

Skipping the all-failure groups changes the expected gradient. For  $G \geq 2$ ,

$$\mathbb{E}[\hat{g}_{\text{practical}}(x) | x] = \nabla_\theta F_{G-1}(\theta; x). \quad (31)$$

To verify this, write the indicator of the all-failure event as  $\prod_{j=1}^G (1 - R_j)$ . Then

$$\hat{g}_{\text{practical}}(x) = \hat{g}_G(x) - \left(1 - \prod_{j=1}^G (1 - R_j)\right) \frac{1}{G} \sum_{i=1}^G \nabla_\theta \log \pi_\theta(y^{(i)} | x).$$

The average score has mean zero. For the remaining term, the independence of the responses, (4), and the zero mean of the score give, for each  $i$ ,

$$\begin{aligned} \mathbb{E} \left[ \prod_{j=1}^G (1 - R_j) \nabla_\theta \log \pi_\theta(y^{(i)} | x) \middle| x \right] &= (1 - p_\theta(x))^{G-1} \mathbb{E}[(1 - R_i) \nabla_\theta \log \pi_\theta(y^{(i)} | x) | x] \\ &= -(1 - p_\theta(x))^{G-1} \nabla_\theta p_\theta(x). \end{aligned}$$

Combining these with Theorem 2 and (26),

$$\begin{aligned} \mathbb{E}[\hat{g}_{\text{practical}}(x) | x] &= \left( \sum_{k=1}^G (1 - p_\theta(x))^{k-1} - (1 - p_\theta(x))^{G-1} \right) \nabla_\theta p_\theta(x) \\ &= \sum_{k=1}^{G-1} (1 - p_\theta(x))^{k-1} \nabla_\theta p_\theta(x) = \nabla_\theta F_{G-1}(\theta; x). \end{aligned}$$

For  $G = 1$  the practical update is always zero, so it needs at least two responses. The estimator in Theorem 2 thus corresponds to order  $G$ , while the commonly implemented centered update corresponds to order  $G - 1$ .

## References

- [1] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992. <https://doi.org/10.1007/BF00992696>.
- [2] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. 2017. <https://arxiv.org/abs/1707.06347>.
- [3] 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. 2024. <https://arxiv.org/abs/2402.14740>.
- [4] Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding R1-Zero-Like Training: A Critical Perspective. *COLM*, 2025. <https://arxiv.org/abs/2503.20783>.
- [5] Zhihong Shao et al. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. 2024. <https://arxiv.org/abs/2402.03300>.
- [6] Fahim Tajwar, Guanning Zeng, Yueer Zhou, Yuda Song, Daman Arora, Yiding Jiang, Jeff Schneider, Ruslan Salakhutdinov, Haiwen Feng, and Andrea Zanette. Maximum Likelihood Reinforcement Learning. 2026. <https://arxiv.org/abs/2602.02710>.

**AI Use.** I prepared these lecture notes with the aid of AI assistance; I take full responsibility for all content in the notes.