RL studies

phil saad

July 2025

1 Introduction

...

2 Entropy in RL

Let's consider the entropy for the response of a policy π given a prompt p

$$\mathcal{H} = -\mathbb{E}_{t\sim\pi(\cdot|p)}[\log \pi(t|p)]$$

Here t is a sequence of tokens (let's fix the length L)

We want to study the change in the entropy over a gradient step, in which the parameters θ_α change as

$$\theta_\alpha \to \theta_\alpha + \delta\theta_\alpha = \theta_\alpha + \eta\partial_\alpha J$$

with

$$J = \mathbb{E}_{t\sim\pi}[A(t)]$$

where A is the advantage (we imagine probably using GRPO type advantage).

Note to self: J isn't $\mathbb{E}[SA]$, when we take the gradient we get $\partial_\alpha J = \partial_\alpha \sum_t \pi(t)A(t) = \sum_t \pi(t)\frac{\partial_\alpha \pi(t)}{\pi(t)}A(t) = \mathbb{E}_{t\sim\pi}[\partial_\alpha S(t)A(t)]$.

First let's just stick with linear order in $\delta\theta_\alpha$ or η

$$\delta\mathcal{H} \approx \sum_\alpha \partial_\alpha \mathcal{H}\delta\theta_\alpha$$

Let's address these two terms in order

2.1 Linear order

Let's look at the linear order change in the entropy for a fixed prompt, for simplicity - we just average over prompts in the end

$$\delta\mathcal{H} \approx \sum_\alpha \partial_\alpha \mathcal{H} \delta\theta_\alpha \tag{1}$$

$$= -\sum_\alpha \delta\theta_\alpha \times \partial_\alpha \sum_{\mathbf{t}} \pi_\theta(\mathbf{t}|p) \log \pi_\theta(\mathbf{t}|p) \tag{2}$$

$$= -\sum_\alpha \delta\theta_\alpha \times \sum_{\mathbf{t}} \partial_\alpha \pi_\theta \log \pi_\theta \tag{3}$$

$$= -\sum_\alpha \delta\theta_\alpha \times \mathbb{E}_{\mathbf{t}\sim\pi(\cdot|p)}[\log \pi \ \partial_\alpha \log \pi] \tag{4}$$

$$\tag{5}$$

where in the third line we used $\sum_{\mathbf{t}} \partial_\alpha \pi = 0$

The gradient of the objective J is familiar

$$\partial_\alpha J = \partial_\alpha \sum_t \pi(t) A(t) = \sum_t \pi(t) \frac{\partial_\alpha \pi(t)}{\pi(t)} A(t) = \mathbb{E}_{t\sim\pi}[\partial_\alpha \log \pi(t) A(t)]$$

So altogether

$$\delta H = -\eta \sum_\alpha \mathbb{E}[\log \pi \partial_\alpha \log \pi] \times \mathbb{E}[\partial_\alpha \log \pi \ A] + \mathcal{O}(\eta^2)$$

This is only true on expectation though - in practice we use a noisy estimate of the expectation value of the policy gradient, so that the linear order variation is

$$\delta H_1 \to \eta \sum_\alpha \mathbb{E}_t[g^{\mathcal{H}}_\alpha(t)] \times \frac{1}{G} \sum_{i=1}^{G} g^J_\alpha(t_i)$$

where $g^{\mathcal{H}}_\alpha(t) = -\log \pi(t) \partial_\alpha \log \pi(t)$ and $g^J_\alpha = A(t) \partial_\alpha \log \pi(t)$. On expectation this is the same as the earlier line, but that won't be true to second order since we will have two noisy gradients.

Okay let's explore this formula a bit more. We estimate these expectation values via sampling sequences of response tokens. Let $G(p)$ denote the set of sampled responses for a given prompt. For simplicity let's imagine there is just one prompt for now (trivial to include more). Let's also refer to $\log \pi_\theta(t|p)$ as $S_\theta(t|p)$, dropping any arguments and subscripts when they are clear from context. Later on we might also want to refer to individual tokens, let's call them $\tau_a \in t$.

It will be useful to subtract a baseline from $g^{\mathcal{H}}_\alpha(t)$:

$$-\mathbb{E}_t[g^{\mathcal{H}}_\alpha(t)] = \mathbb{E}_t[S(t)\partial_\alpha S(t)] = \mathbb{E}_t[(S(t) - \mathbb{E}[S])\partial_\alpha S(t)]$$

we can do this because $\mathbb{E}[\partial_\alpha S] = 0$. The advantage is already baseline subtracted so we don't need to do this for g^J.

Now it's also useful to rewrite our equation for $\delta\mathcal{H}_1$ as

$$\delta\mathcal{H}_1 = \frac{-\eta}{G}\sum_{i=1}^{G}\mathbb{E}_t\Big[(S(t)-\overline{S})\;K_1(t,t_i')\;A(t_i')\Big]$$

where $\overline{S} = \mathbb{E}_t[S(t)]$ and the (first order in learning rate) kernel K_1 is

$$K_1(t,t') = \sum_\alpha \partial_\alpha S(t)\partial_\alpha S(t')$$

I think this might be called the "fisher kernel". Clearly it is related to the Fisher information - If $w_{\alpha t} = \partial_\alpha S(t)$, then $K_1(t,t') = w^T w$ and $F_{\alpha\beta} = ww^T$.

The fisher kernel has a pretty simple interpretation. Consider the first order step change in the logprob for sequence t. Using the previous derivations we see

$$\delta\log\pi(t) = \eta\frac{1}{B}\sum_{i=1}^{B}K(t,t_i')A(t_i')$$

So it is a measure of the effect a sequence t_i' used in the update has on the logprob of getting sequence t (the advantage just tells us the strength and direction of that effect)

- We've assumed that the objective function is the on-policy objective, with the normalization $1/G$ per prompt. In my current implementation of my RL trainer, I am using the dr. grpo normalization, where I also normalize by the maximum response length per prompt.
- We should also average over prompts

Together, these simply modify

$$\delta\mathcal{H}_1 \to -\eta\frac{1}{BG}\sum_{p\in B}\frac{1}{L_{max}(p)}\sum_{i=1}^{G}\mathbb{E}_{t\sim\pi(\cdot|p)}\Big[(S(t)-\overline{S})\;K_1(t,t_i')\;A(t_i')\Big]$$

In this form we can see a more clear relationship with the result of Cui et al. Let's imagine K was diagonal, then

$$\mathbb{E}[\delta\mathcal{H}_1] \to \mathrm{Cov}(S(t),A(t))$$

The difference with Cui et al's result is this is per-sequence, not per-token.

But K is likely not diagonal (this is something we should study experimentally!). Let's try and make some predictions for the behavior of K_1. The matrix elements $w_{\alpha t}$ hopefully scale like $1/\sqrt{N_{params}}$ since apparently initializing the weights to be of order $1/\sqrt{N_{params}}$ is typical, and it is apparently standard practice to have training keep the weights of order this size, for stability. Then the diagonal elements of K_1 would be of order one (in terms of the scaling with the number of model parameters). So this is a "nice" quantity to study.

Perhaps as a very crude model, we might model $w_{\alpha t}$ as a Wishart random matrix.

However, a totally random matrix is probably a poor approximation. In reality, this matrix probably has a very blocky structure. Let's incorporate the fact that we would have different prompts as well as many responses per prompt, so that we should really think of the sequence index as like $(t|p)$, where p ranges over some fixed number D of prompts and t ranges over all V^L possible response sequences. The matrix elements of w will probably be correlated if the prompts are the same, and even more so if the responses t share subsequences. So e.g. for two sequences $(t_1|p), (t_2|p)$ that differ by one token, the matrix elements of w will be probably highly correlated.

If w was a true Wishart random matrix, we would have some interesting crossover at $V^L \sim N_{params}$, where K would go from nearly the identity for $V^L < N_{params}$ to very much not. But perhaps we can still get some intuition from this. Let's first note that the sequence length L does not have to be very big before K will necessarily have some zero eigenvalues. With $V \sim 10^6$ and $N_{params} \sim 10^9 - 10^{11}$ (I wonder how LoRA affects all of this??) then L need only be order one. But that doesn't necessarily mean that these zero eigenvalues are relevant. Most sequences are junk, so most of the matrix is irrelevant. (In other words, the typical sequences dominate the expectation values, so matrix elements for atypical sequences can basically be ignored - if we projected onto the typical subspace that would be fine)

2.1.1 LET'S WRITE DOWN SOME GOALS

- Estimate $\delta\mathcal{H}_1$ during training, compare to real step change in entropy. Compute matrix elements of $K_1(t, t')$, look at diagonal, between sequences in same prompt, and sequences from different prompts to get a sense of how big contributions from different pairs of sequences are. Use some matrix sketching technques as well. We should take into account conditioning factors from adam vs sgd $\delta\theta_\alpha \to \eta P_\alpha \partial_\alpha J$, though maybe it doesn't make much of a difference in the structure of fisher kernel. In the end we want to understand 1) if linear order is a good enough estimation of the entropy change per step and 2) beyond the sequences identified in cui et al (sequences with negative advantage and a token with outlier very negative logprob), are there other types of sequences which may contribute to entropy collapse? In situations with long sequences, off-diagonal sequences may become more important, try and study this in theory, though unlikely we can do experiments...

2.2 Estimating $\delta\mathcal{H}_1$

So in practice we want to measure $\delta\mathcal{H}_1$ and compare it to the true step change in entropy. In order to do that we need to estimate $\mathbb{E}_t[g_\alpha^{\mathcal{H}}]$ (or after rearranging, the $\mathbb{E}_t$ in $\frac{1}{G}\sum_i \mathbb{E}_t[(S - \overline{S})K_1 A]$. If we use the same set of samples for the expectation value in $g_\alpha^{\mathcal{H}}$ as were used for the updates, we're not necessarily

going to get a great estimate. But let's think about how we can get an unbiased estimator using just the rollouts used for the updates.

If we only had one prompt, we would simply use the U statistic

$$\delta\hat{\mathcal{H}}_1 = \frac{1}{G(G-1)}\frac{1}{L_{max}(p)}\sum_{i\neq j}(S(t_i) - \overline{S}_{loo})K(t_i, t_j)A(t_j)$$

Here we are starting to keep track of the per-prompt normalization $1/L_{max}(p)$ used in dr grpo (which i am using because it apparently helps manage responses length growth). Also, we will be more careful and use the LOO baseline.

When we have multiple prompts, samples within the same prompt group don't count as independent samples: we sample by sampling a batch B of prompts from the dataset D and then for each prompt sample G responses. But then responses in each group are of course only indpendently sampled from the conditional distribution $\pi(\cdot|p)$. So we shouldn't do a naive U statistic across all samples. Instead, the best we can do is a sort of U statistic across prompt groups, to get

$$\delta\hat{\mathcal{H}}_1 = \frac{\eta}{B(B-1)}\sum_{p_n\neq p_m}\frac{1}{L_{max}(p_m)}\frac{1}{G(G-1)}\times \tag{6}$$

$$\sum_{t_h\in G_n, t_g\in G_m}(S(t_h|p_n) - \overline{S}_n)K((t_h|p_n),(t_g,p_m))A(t_g|p_m) \tag{7}$$

Where here the $\overline{S}_n$ is the mean logprob for the prompt group G_n, and the $1/(G(G-1))$ is because of the LOO, not because of some $h \neq g$ thing, since h and g are from different prompts, so we sum over all pairs.

To compute the variance of this estimator we follow a derivation from chat gpt, which says this is a standard derivation from Hoeffding...

We first define the kernel

$$h_0(p_n, p_m) = \frac{G}{L_{max}(p_m)(G-1)}\mathbb{E}_{t_h\sim\pi(\cdot|p_n), t_g\sim\pi(\cdot|p_m)}\Big[(S(t_h|p_n) - \overline{S}_n)K((t_h|p_n),(t_g,p_m))A(t_g|p_m)\Big]$$

It's convenient to symmetrize it over p_n, p_m to get just h. Then the expected value of $\delta\mathcal{H}_1$ (averaging over the samples used for updates) is

$$T = \mathbb{E}[\delta\mathcal{H}_1] = \mathbb{E}[h(p, p')]$$

where we refer to this as our target T. We are interested in variance of the U statistic

$$U = \frac{1}{B(B-1)}\sum_{n\neq m}h(p_n, p_m)$$

We then define

$$\phi_1(p) = \mathbb{E}_{p'\sim D}[h(p, p')] - T$$

where clearly $\mathbb{E}_p[\phi(p)] = 0$. Then we define the "canonical/degenerate remainder"

$$\phi_2(p, p') = h(p, p') - T - \phi_1(p) - \phi_1(p')$$

for which the expectation over either or both arguments is zero. So now with $h(p, p') = T + \phi_2(p, p') + \phi_1(p) + \phi_2(p')$ we have this formula for the difference between our estimator and the mean

$$U - T = \frac{1}{B(B-1)} \sum_{n \neq m} \phi_1(p_n) + \phi_1(p_m) + \phi_2(p_n, p_m)$$

Across all pairs, $\phi_1(p)$ appears $2(B-1)$ times

$$\sum_{n \neq m} \phi_1(p_n) + \phi_1(p_m) = 2(B-1) \sum_n \phi_1(p_n)$$

so that

$$U - T = \frac{2}{B} \sum_n \phi_1(p_n) + \frac{1}{B(B-1)} \sum_{n \neq m} \phi_2(p_n, p_m)$$

Now we want to show that terms with ϕ_1 and ϕ_2 are uncorrelated so their variances add:

$$\mathrm{Cov}\Big(\sum_n \phi_1(p_n), \sum_{m \neq l} \phi_2(p_m, p_l) \Big) = \sum_{n, m \neq l} \mathbb{E}_{p_n, p_m, p_l}[\phi_1(p_n) \phi_2(p_m, p_l)]$$

In the case $n \neq m \neq l$, this is zero because the expectations of ϕ_1 and ϕ_2 are separately zero. In the case of $n = m$ or $n = l$ then we still get zero because the conditional expectation over the other argument of ϕ_2 gives zero.

Okay now let's compute the variance of each term. Apparently we should call $\zeta_1 = \mathrm{Var}(\phi_1(p))$

$$\zeta_1 = \mathrm{Var}(\phi_1(p)) = \mathrm{Var}\big(\mathbb{E}[h(p, p')|p] - T \big) = \mathrm{Var}\big(\mathbb{E}[h(p, p')|p] \big)$$

and similarly $\zeta_2 = \mathrm{Var}(\phi_2(p, p'))$.

We have

$$\mathrm{Var}\Big(\frac{2}{B} \sum_n \phi_1(p_n) \Big) = \frac{4}{B^2} \times B \times \mathrm{Var}(\phi_1(p)) = \frac{4}{B} \zeta_1$$

since the ϕ_i are independent.

Let's call $S = \sum_{n \neq m} \phi_2(p_n, p_m)$. Then

$$\mathrm{Var}(S) = 2 \sum_{n \neq m} \mathrm{Var}(\phi_2(p_n, p_m)) = B(B-1) \zeta_2$$

where we used the fact that the $\phi_2(p_n, p_m)$ are uncorrelated unless both indices are the same, since the conditional expectation over any index of just one copy

is zero. The factor of two is because in a quadruple sum over n, m, p, q we can pair $n = p, m = q$ or $n = q, m = p$. Then

$$\mathrm{Var}\Big(\frac{S}{B(B-1)}\Big) = \frac{2}{B(B-1)}\zeta_2$$

And then altogether

$$\mathrm{Var}(U) = \frac{4}{B}\zeta_1 + \frac{2}{B(B-1)}\zeta_2$$

So for large B we are dominated by the first term and the variance goes as $1/B$.

2.2.1 Measuring ζ_1

So if we use the estimator U, we should probably try and measure ζ_1 in order ot know if our batch size is big enough to be giving a good estimate. Chat gpt helped me understand some ways to measure this. First some preliminaries:

It's helpful to define some quantites. For each prompt p_n we define

$$X_n = \frac{1}{G}\sum_{g\in G_n}(S_{n,g} - \overline{S}_{n,-g})g_{n,g}$$

where $\overline{S}_{n,-g}$ is the LOO average (we will use $-idx$ for LOO), and $g_{n,g}$ is the gradient of $S_{n,g}$, and

$$Y_n = \frac{1}{L_{max}(p_n)G}\sum_{g\in G_n} A_{n,g}g_{n,g}$$

so that $U = \frac{1}{B(B-1)}\sum_{n\neq m} X_n \cdot Y_m$. A good way to compute this in practice is

$$U = \frac{B}{B-1}\overline{X}\cdot\overline{Y} - \frac{1}{B(B-1)}\sum_n X_n \cdot Y_n$$

with $\overline{X} = \frac{1}{B}\sum_n X_n$, same for Y.

Let's also define the "row averages"

$$r_n = \frac{1}{B-1}\sum_{m\neq n} h(p_m, p_n) = \frac{1}{2}(X_n \cdot \overline{Y}_{-n} + Y_n \cdot \overline{X}_{-n})$$

where we use the minus subscripts to denote the LOO average. Then $\frac{1}{B}\sum_n r_n = U$. If we define

$$\phi_n = r_n - U$$

which has zero mean over n. ϕ_n is estimating $\phi(p_n)$ and then the variance of ϕ_n is an estimator of ζ_1

$$\boxed{\hat{\zeta}_1 = \frac{1}{B-1}\sum_n \phi_n^2}$$

(note the $B-1$, related to the constraint $\sum_n \phi_n = 0$).

Another method is the "jackknife" variant. This is a standard technique so I won't go through the derivation, just introduce relevant quantities. We will need our estimator U but with one sample removed, $U_{(-n)}$. We can compute it with

$$U_{(-n)} = \frac{BU - 2r_n}{B-2}$$

With the mean of these $\overline{U_{(-\cdot)}} = \frac{1}{B}\sum_n U_{(-n)}$, the jackknife estimate of the variance is

$$\boxed{\widehat{\mathrm{Var}(U)}_{jackknife} = \frac{B-1}{B}\sum_n \left(U_{(-n)} - \overline{U_{(-\cdot)}}\right)^2}$$

and so $\hat{\zeta}_{1,jackknife} = \frac{B}{4}\widehat{\mathrm{Var}(U)}_{jackknife}$

I think what I will do is compute both of these and compare them

2.3 Comparison to the gradient noise scale

I think it would be good to compare with the math behind the gradient noise scale for the loss from the paper "An Empirical Model of large batch training". Let's first go through a derivation of the relevant formulas so we can see if any of their techniques are useful.

We have the "True" loss

$$L_{true} = \mathbb{E}_{t\sim D}[-\log \pi_\theta(t)]$$

where t are sequences of tokens drawn from a "true" distribution D.

To first order in the learning rate we have (using ordinary SGD for now)

$$\delta L_1 = \sum_\alpha \partial_\alpha L_{true}\delta\theta_\alpha \tag{8}$$

$$= -\underbrace{\mathbb{E}_{t\sim D}[\partial_\alpha \log \pi(t)]}_{-G_\alpha}\ \underbrace{\frac{\eta}{B}\sum_{i=1}^{B}\partial_\alpha \log \pi(t_i')}_{-\eta G_\alpha^{est}} \tag{9}$$

$$= -\frac{\eta}{B}\mathbb{E}_{t\sim D}\Big[\sum_\alpha \partial_\alpha \log \pi(t)\partial_\alpha \log \pi(t_i')\Big] \tag{10}$$

$$= -\frac{\eta}{B}\sum_{i=1}^{B}\mathbb{E}_{t\sim D}\Big[K_1(t,t_i')\Big] \tag{11}$$

Okay so it involves exactly this kernel we ecountered, the Fisher Kernel. If we generalize from ordinary SGD in the perfectly conditioned case, we still get the Fisher Kernel, but the appropriately preconditioned one.

This illustrates again the interpretation of the Fisher kernel as a meausure of how much a seqeunce t' used in an update influences the log probability of getting sequence t

Let's go to second order. I'll drop the explicit sums over parameters since its always simple einstein convention

$$\delta L_2 = \frac{1}{2}\partial_\alpha \partial_\beta L_{true} \delta\theta_\alpha \delta\theta_\beta \tag{12}$$

$$= \frac{\eta^2}{2} \underbrace{\mathbb{E}_{t\sim D}\Big[\partial_\alpha \log \pi(t) \partial_\beta \log \pi(t) - \frac{\partial_\alpha \partial_\beta \pi(t)}{\pi(t)}\Big]}_{H_{\alpha\beta}} \underbrace{\frac{1}{B^2}\sum_{i,j=1}^{B} \partial_\alpha \log \pi(t'_i) \partial_\beta \log \pi(t''_j)}_{G^{est}_\alpha G^{est}_\beta} \tag{13}$$

$$= \frac{\eta^2}{2}\frac{1}{B^2}\sum_{i,j=1}^{B} \mathbb{E}_{t\sim D}\Big[K_1(t'_i, t) K_1(t, t''_j) - \tilde{K}_2(t'_i, t''_j)\Big] \tag{14}$$

where $\tilde{K}_2(t'_i, t''_j) = \mathbb{E}_{t\sim D}\Big[\partial_\alpha \log \pi(t'_i) \partial_\beta \log \pi(t''_j) \frac{\partial_\alpha \partial_\beta \pi(t)}{\pi(t)}\Big]$

From here on I'll define $S = \log \pi$ $S_\alpha = \partial_\alpha S$.

We're interested in the expected value of the change in loss. to understand this we consider the mean and covariance of the noisy gradients

$$G_\alpha = \mathbb{E}_{t\sim D}[G^{est}_\alpha] = \mathbb{E}_{t\sim D}[S_\alpha(t)]$$

$$\frac{1}{B}\Sigma_{\alpha\beta} = \mathbb{E}_{t,t'\sim D}[(G^{est}_\alpha - G_\alpha)(G^{est}_\beta - G_\beta)] = \mathbb{E} = \frac{1}{B} Cov(S_\alpha, S_\beta)$$

Then the expected value of the change in loss, to second order, is

2.4 Second order in learning rate

It is easy to extend this computation to second order in the learning rate.

$$\delta\mathcal{H} = \sum_\alpha \partial_\alpha \mathcal{H} \delta\theta_\alpha + \frac{1}{2}\sum_{\alpha,\beta} \partial_\alpha \partial_\beta \mathcal{H} \delta\theta_\alpha \delta\theta_\beta + \mathcal{O}(\eta^2)$$

(PENDING REWRITE)

2.4.1 Next steps