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

2.2.2 Side note on per-sequence vs per-token

I'll briefly make a comment here that we are computing the per-sequence entropy. We should choose some convention for how we measure the entropy of sequences with EOS tokens. We will count the entropy of the first EOS token, and the rest will have zero entropy.

We could also try measuring the per token entropy. This would involve introducing importance weights to account for the varying lengths of sequences. Perhaps I'll try this later.

2.2.3 Two-Batch estimators

With the previous approach we run into issues if we want to make comparisons to the true entropy change. Let's say we have a batch U which we use for gradient updates. We want to estimate the entropy of the model before and after the step. If we use the same batch U to estimate the entropy, we get a bias and we can't eliminate it like we can with the per-order-in-η estimators. So we should really introduce a second batch E which we use to estimate the entropy.

$$\widehat{\delta\mathcal{H}} = \frac{1}{B_E}\sum_{p_n \in B_E} \widehat{\mathcal{H}^{(G)}_{\theta+\delta_U\theta}(p_n)} - \frac{1}{B_E}\sum_{p_n \in B_E} \widehat{\mathcal{H}^{(G)}_{\theta}(p_n)}$$

where $\widehat{\mathcal{H}^{(G)}_{\theta}(p_n)}$ is the estimate of the entropy using prompt p_n and G generated responses. We've been treating each prompt group as an independent sample so far and will continue to do so, since the effects of G on the overall variance are subleading.

So then to make a proper comparison we should also use these same U and E batches to estimate $\delta\mathcal{H}_1$ (and also if we go to higher order in lr). This simplifies things a little bit as we don't need to worry about this U statistic thing. It's unfortunate that we have to use two batches instead of one though.

Let's continue to use this X and Y notation, with X_n^α or just X_n being the gradient of the entropy (with the loo baseline subtracted), similarly with Y. Then with

$$\bar{X} = \frac{1}{B_E}\sum_{n\in E} X_n, \qquad \bar{Y} = \frac{1}{B_U}\sum_{p\in U} Y_p$$

we have

$$\widehat{\delta\mathcal{H}_1} = \eta \bar{X}\cdot\bar{Y}$$

We assume that within each batch the X_n, Y_p for different prompts are uncorrelated. Then we define the vectors/matrices in the parameter space as

$$\mathbb{E}_E[\bar{X}] = \mu_X, \qquad \mathrm{Cov}_E(\bar{X}) = \frac{\Sigma_X}{B_E}, \qquad \mathbb{E}_U[\bar{Y}] = \mu_Y, \qquad \mathrm{Cov}_U(\bar{Y}) = \frac{\Sigma_Y}{B_U}$$

where $\Sigma_{X,Y}$ are matrices and $\mu_{X,Y}$ are vectors. The variance of $\widehat{\delta\mathcal{H}_1}$ is then

$$\mathrm{Var}(\widehat{\delta\mathcal{H}_1}) = \eta^2\left(\frac{\mu_X\Sigma_Y\mu_X}{B_U} + \frac{\mu_Y\Sigma_X\mu_Y}{B_E} + \frac{\mathrm{Tr}[\Sigma_X\Sigma_Y]}{B_E B_U}\right)$$

Now let's try and make an estimator of the variance. We've already introduced $\bar{X}, \bar{Y}$ as sample means of X and Y, now let's introduce the sample variance estimators

$$\hat{\Sigma}_X^{\alpha\beta} = \frac{1}{B_E - 1}\sum_{n\in E}(X_n - \bar{X})^\alpha (X_n - \bar{X})^\beta$$

Similarly for Y

Then we compute

$$\hat{V}_X = \frac{\hat{\mu}_Y\hat{\Sigma}_X\hat{\mu}_Y}{B_E} = \frac{1}{B_U(B_E-1)}\sum_{n\in E}\Big(\bar{Y}\cdot(X_n - \bar{X})\Big)^2$$

$$\hat{V}_Y = \frac{\hat{\mu}_X\hat{\Sigma}_Y\hat{\mu}_X}{B_U} = \frac{1}{B_E(B_U-1)}\sum_{n\in U}\Big(\bar{X}\cdot(Y_n - \bar{Y})\Big)^2$$

We could also try an estimate the $O(1/B_E B_U)$ term in the variance but for now i'm happy to assume its negligible. Then we can estimate the variance as just

$$\widehat{\mathrm{Var}(\widehat{\delta\mathcal{H}_1})} \approx \hat{V}_X + \hat{V}_Y$$

If we want to compute $\widehat{\delta\mathcal{H}_1}$ (we do), we will need to compute $\bar{X}, \bar{Y}$. So if we keep these around (only two parameter-sized vectors), then its pretty easy to also compute $\hat{V}_{X,Y}$ as they are simply just sums over prompts. We'd have to do a backward for every prompt but thats not so bad, and its trivial to spread the computation over multiple devices.

In all this we can imagine that we also sample G_E responses per prompt for the E batch and a different number G_U for the U batch. For the purposes of

estimating the entropy, we get more mileage out of increasing B_E and keeping $G_E = 1$, while for the U batch we're interested in what we actually do during trainig which is to have G_U be 8 in my case, and with G_U being whatever we use during training as well. We aren't trying to get a good estimate of the true loss gradient, but we are trying to get a good estimate of the entropy!

2.2.4 Lessons about variance reduction and getting better estimates

A problem I'm encountering is that the distribution of token or sequence entropies has a heavy right tail. I'm using the GSM8k dataset, which has several thousand prompts in the train set and over a thousand in the test set. But i'm seeing that to get a reliable estimate of the entropy i need well over a thousand prompts. To be clear, what I did was to sample B prompts and then sample G=8 gens per prompt to keep consistent with what we are actually measuring in this project. Then I compute the mean entropy per prompt over the G responses. Then I look at how the distribution of observed entropies behaves as I increase B. I find that even at B = 1024 the distribution is significantly different than at B = 4096 (as big as I went). I simply quantified this by comparing the 50th, 80th, 95th percentile values and noticing an order 10 percent difference. Probably there are more sophisticated and useful measurements of how different by observed distribution is as I increase the batch size (as another naive one I could have computed the KL between the two by fitting to the distributions P,Q and then sampling from P to compute the expectation of log P/Q (not sure how else to do it)). But this naive approach was enough to convince me that I will need to do some serious improvement of my approach.

I've now learned about two approaches to variance reduction: control variates and Rao-Blackwellization. My impression here is that while it can't hurt, control variates are not going to solve my problem, which is due to the heavy tail not being sampled much at all until I have prohibitively large batch sizes (easy to see on a histogram that the tail is sparsely sampled until batch sizes of order 512- way too big for me!).

As a side note, I'm not really sure how to apporach finding control variates... do I just measure the covariance between my quantity of interest and anything else I can think of and then see what has a big covariance, or just pick quantities that intuitively have a covariance but are also easier to estimate? Is there a more systematic approach?

Perhaps in my case, Rao-Blackwellization will be helpful. I asked ChatGPT to teach me about Rao-Blackwellization and asked about how I might apply it in this context. It explained to me that we can think of using the estimator (here for simplicity we start by considering a single prompt)

$$\hat{H}_{RB} = -\frac{1}{G} \sum_{t_n \in G} \sum_{k=1}^{L} \left(\sum_{t_k \in V} \pi(t_k | t_{<k}^{(n)}, p_n) \log \pi(t_k | t_{<k}^{(n)}, p_n) \right)$$

as using Rao-Blackwellization. I'm not sure how to think precisely about the "sufficient statistic" in this case, but we are exploiting the fact that when we can

condition on the prefix to the kth token, we know the full next token entropy, so I can see how this is at least morally Rao-Blackwellization...

Let's study this estimator more. Clearly it is unbiased, since the expectation value of a given term is just the true entropy for the k'th token, averaged over the prefix with the prompt fixed. Actually let's more systematically show that this is unbiased:

Consider the expectation value for the term for a given token

$$\mathbb{E}_{t^{(n)}\sim\pi(\cdot|p_n)}\left[\sum_{t'_k\in V}\pi(t'_k|t^{(n)}_{<k},p_n)\log\pi(t'_k|t^{(n)}_{<k},p_n)\right]$$

We can see that the thing in the expectation value actually doesn't depend on t_k (note, different from t'_k which is being summed over), as well as the $t_{>k}$. So the expectation value can simply be written as $\sum_{t_{<k}}\pi(t_{<k}|p_n)$...

$$\sum_{t'_k,t_{<k}}\pi(t_{<k}|p)\pi(t'_k|t_{<k},p_n)\log\pi(t'_k|t_{<k},p_n)$$

Let's rename t'_k to t_k and use $\pi(t_{<k}|p)\pi(t_k|t_{<k},p_n)=\pi(t_{<=k}|p_n)=\sum_{t_{>k}}\pi(t|p_n)$ to get

$$\sum_t\pi(t|p_n)\log\pi(t_k|t_{<k},p_n)$$

Now if we sum this over k, the only k dependent terms are the logs which combine to give the full sequence logprob

$$\sum_t\pi(t|p_n)\log\pi(t|p_n)=-\mathbb{E}[H]$$

To generalize to the case where we also sample from prompts is trivial, since the probability for an output factorizes as $\pi(t)=\pi(t|p)D(p)$ where D is the (trivial) prompt distribution, and we only are interested in the entropy of the tokens conditioned upon a prompt, so

$$\mathbb{E}_{t,p\sim\pi(t|p)D(p)}[H]=\sum_{t,p}\pi(t|p)D(p)\log\pi(t|p)\xrightarrow[\text{sample p}]{}\frac{1}{B}\sum_{p_n\in B}\pi(t|p_n)\log\pi(t|p_n)$$

I had chatgpt prove that the variance is reduced - Thanks!

Proposition (Rao–Blackwell variance reduction for per-sequence entropy). Fix a prompt p. Let the random sequence $t=(t_1,\dots,t_L)\sim\pi(\cdot\mid p)$ be generated autoregressively. For each step k, define the prefix σ-algebra

$$\mathcal{F}_k\;=\;\sigma\big(p,\,t_{<k}\big),\qquad k=1,\dots,L,$$

the step surprisal

$$Z_k\;\equiv\;-\log\pi\big(t_k\,\big|\,t_{<k},p\big),$$

and its conditional mean (next-token entropy at the prefix)

$$H_k \;\equiv\; \mathbb{E}\big[Z_k \,\big|\, \mathcal{F}_k\big] \;=\; -\sum_{u\in V} \pi\big(u \,\big|\, t_{<k}, p\big)\, \log \pi\big(u \,\big|\, t_{<k}, p\big).$$

Consider the two per-sequence sums

$$S \;\equiv\; \sum_{k=1}^{L} Z_k \quad \text{and} \quad S_{\mathrm{RB}} \;\equiv\; \sum_{k=1}^{L} H_k.$$

Then $\mathbb{E}[S] = \mathbb{E}[S_{\mathrm{RB}}]$ and

$$\mathrm{Var}(S_{\mathrm{RB}}) \;\le\; \mathrm{Var}(S),$$

with equality if and only if $\mathrm{Var}(Z_k \mid \mathcal{F}_k) = 0$ for all k (i.e., each Z_k already equals H_k almost surely).

Proof. Unbiasedness follows from the tower property:

$$\mathbb{E}[S] = \sum_{k=1}^{L} \mathbb{E}[Z_k] = \sum_{k=1}^{L} \mathbb{E}\big[\mathbb{E}[Z_k \mid \mathcal{F}_k]\big] = \sum_{k=1}^{L} \mathbb{E}[H_k] = \mathbb{E}[S_{\mathrm{RB}}].$$

For variance, set the martingale differences $D_k \equiv Z_k - H_k$. Then $\mathbb{E}[D_k \mid \mathcal{F}_k] = 0$ by construction, and

$$\mathbb{E}[D_k \mid \mathcal{F}_{k-1}] = \mathbb{E}\big[\,\mathbb{E}[D_k \mid \mathcal{F}_k] \,\big|\, \mathcal{F}_{k-1}\big] = 0,$$

so D_k has mean zero given either $\mathcal{F}_{k-1}$ or $\mathcal{F}_k$. Moreover, for $i \neq j$ with $i < j$,

$$\mathbb{E}[D_i D_j] = \mathbb{E}\big[D_i\, \mathbb{E}[D_j \mid \mathcal{F}_{j-1}]\big] = 0,$$

so the $\{D_k\}$ are pairwise uncorrelated.

Write $S = S_{\mathrm{RB}} + \sum_{k=1}^{L} D_k$. Then

$$\mathrm{Var}(S) = \mathrm{Var}(S_{\mathrm{RB}}) + \mathrm{Var}\Big(\sum_{k=1}^{L} D_k\Big) + 2\,\mathrm{Cov}\Big(S_{\mathrm{RB}}, \sum_{k=1}^{L} D_k\Big).$$

The cross term vanishes: for each j,

$$\mathbb{E}\Big[\Big(\sum_{k<j} H_k\Big) D_j\Big] = 0 \quad \big(\textstyle\sum_{k<j} H_k \text{ is } \mathcal{F}_{j-1}\text{-measurable}\big), \qquad \mathbb{E}\Big[\Big(\sum_{k\ge j} H_k\Big) D_j\Big] = 0 \quad \big(\textstyle\sum_{k\ge j} H_k \text{ is } \mathcal{F}_{j}\text{-measurable}\big),$$

hence $\mathrm{Cov}(S_{\mathrm{RB}}, \sum_k D_k) = 0$.

Finally, using pairwise uncorrelatedness and the definition of conditional variance,

$$\mathrm{Var}\Big(\sum_{k=1}^{L} D_k\Big) = \sum_{k=1}^{L} \mathbb{E}[D_k^2] = \sum_{k=1}^{L} \mathbb{E}\big[\,\mathrm{Var}(Z_k \mid \mathcal{F}_k)\big].$$

Therefore,

$$\boxed{\mathrm{Var}(S) = \mathrm{Var}(S_{\mathrm{RB}}) + \sum_{k=1}^{L} \mathbb{E}\big[\,\mathrm{Var}(Z_k \mid \mathcal{F}_k)\big]}$$

and the inequality follows, with equality exactly when each $\mathrm{Var}(Z_k \mid \mathcal{F}_k) = 0$. □

Remark (random prompts). If p is random with distribution D, simply include P in the filtration: $\mathcal{F}_k = \sigma(P, t_{<k})$. All equalities above hold after an outer expectation over P. For per-token reporting, replace S and S_{RB} by S/L and S_{RB}/L; the same argument applies (condition on L first if random).

2.2.5 Brief remark on computing $\hat{H}_{RB}$

Let's briefly discuss how we should compute the RB estimate for the entropy in practice, starting from next token (temperature scaled) logits z_l^a , in the case that we use the full vocabulary $a \in V$ or some subset like using top k or top p ($a \in S$)

With the full vocab we compute (dropping indices) $Z = \mathrm{logsumexp}(z)$ and $p = \mathrm{softmax}(z)$, and use

$$H(p) = Z - \sum_a p_a z^a$$

Relatively convenient!

If we only sample from a subset of tokens, define $Z = \mathrm{logsumexp}(z_{all})$, $Z_S = \mathrm{logsumexp}(z_S)$, $s = \exp(Z_S - Z)$, $\epsilon = 1 - s$. Then the full entropy decomposes as

$$H(full) = sH(p_S/s) - s\log s - \epsilon \log \epsilon + \text{"Tail error"}$$

where the tail error is some really small thing, and the p_S are renormalized probabilties.

2.3 Revisiting entropy variation estimate

After comparing various estimates of the entropy, comparing convergence and variance, I'm conviced I should use a superior estimator for the step change in entropy $\delta\mathcal{H}_1$. Let's start by not thinking about control variates, and start with (using our earlier notation)

$$\mathcal{H} = \sum_{k=1}^{L} \mathbb{E}[H_k]$$

where again $H_k = -\sum_{t_k \in V} \pi(t_k|t_{<k}, p) \log \pi(t_k|t_{<k}, p)$, and the expectation is with respect to the prefix $t_{<k}$ (and prompt). If we take the gradient of this we get the usual sum of two terms

$$\partial_\alpha \mathcal{H} = \mathbb{E}\left[\sum_{k=1}^{L} \partial_\alpha H_k\right] + \mathbb{E}\left[\left(\sum_{k=1}^{L} H_k\right) \partial_\alpha \log \pi(t|p)\right]$$

Then we estimate both expectation values by sampling B_E prompts and completions from $\pi(t|p)D(p)$.

The first term doesn't need any further processing, but we can use the score identity to add a baseline to the second term.

$$\mathbb{E}\left[\left(\sum_{k=1}^{L} H_k\right)\partial_\alpha \log \pi(t|p)\right] = \mathbb{E}\left[\sum_{k=1}^{L}\Big(G_k - b_k\Big)\partial_\alpha \log \pi(t_k|t_{<k},p)\right]$$

Where $G_k = \sum_{k'\geq k} H_{k'}$ is the "entropy to go" and the baseline b_k is any function of $t_{<k}, p$. So basically we are treating the sequence like an episode in RL, and using the REINFORCE tricks to derive the above formula.

Choosing a baseline seems tricky. The optimal is $\mathbb{E}[G_k|t_{<k}]$. I asked chatgpt about some good choices: A fancy choice would be to use a learned approximation of that, but that seems unnecessarily complicated. Perhaps we'll try that later though. A zeroeth order choice would be $b_k = H_k$. I'm still trying to get better intuition for this stuff so here I'll note for myself that we can think about it this way:

Consider the per-step term

$$(G_k - b_k)\partial_\alpha \log \pi(t_k|t_{<k},p)$$

we would like to choose b_k to subtract the "predictable from the current state $(t_{<k}, p)$) part of G_k. Doing this optimally would leave only the truly random part of $(G_k - b_k)$. So the intiution is that we should try and remove as much of the predictable part of G_k (given current state) as possible. H_k is completely predictable given the state.

We can do better by choosing

$$b_k = H_k + \mu_k^{resid}$$

where μ_k^{resid} is an EMA of $G_k - H_k$ as a crude approximation of the return to go, where we should keep the explicitly computable H_k term separate.

I think what I will do is start with this EMA approximation, and then maybe later try using regression on various logit statistics to have a fancier approximation.

2.4 Validity of linear order approx

So my first goal is to test the linear order approximation, not really caring much about whether the learning rates at which this approximation is valid are good for training or not (probably not?). Hopefully the second order correction (studied next) will be enough to have a good approximation in reasonable ranges of the learning rate.

Let's take a moment to discuss the ranges of learning rate where this approximation would be good. The thing we need to study is the Hessian of the entropy, combined with the optimizer's preconditioning factors.

First let's study the simpler case where we have just a simple diagonal preconditioner and so

$$\delta\theta_\alpha = -\eta P_\alpha \partial_\alpha L$$

(here let's call $L = -J$ so that our sign conventioins match ordinary gradient descent)

Which we write more simply as $\delta\theta = \eta P g$

Then the linear order approx is good when the second order change is smaller than the first, which we can write as

$$\eta \ll \eta^* = \frac{2|(\nabla H)^T (Pg)|}{|(Pg)^T (\nabla^2 H)(Pg)|}$$

And then apparently there are some nice tricks to compute HVPs (Hessian-Vector Products), so it won't be impossible to compute

$$(\nabla^2 H)(Pg)$$

Now let's consider the adam optimizer. The update direction is now

$$\delta\theta_t \approx -\eta(\alpha_t g_t + \beta_t m_t) = -\eta s_t$$

where we keep track of the optimizer states. So we can still form s_t and then compute

$$\frac{2|(\nabla H)^T s_t|}{|s_t^T (\nabla^2 H) s_t|}$$

and use tricks to compute the HVP $(\nabla^2) s_t$.

We already compute s_t and ∇H for the leading order term.

2.5 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)

2.6 Fisher Kernel with adamw

Using Adamw makes things a bit more complicated (or anything with momentum) because now the updates aren't 'local' in the sense that they depend on past gradients and past batches. For step t

$$\delta\theta_\alpha^t = \eta\left(\frac{m_\alpha^t}{\sqrt{v_\alpha^t} + \epsilon} - \lambda\theta_\alpha^t\right)$$

with

$$m_\alpha = \frac{1-\beta_1}{1-\beta_1^t}\sum_{t'=1}^{t}\beta_1^{t-t'}g_\alpha^{t'}$$

Similarly for v, but we're going to pretend v is constant.

let's focus on the non-weight-decay contribution to the change in entropy. with some hopefully obvious notation (I should not have used t for step here..)

$$\delta H_1^t = -\eta\sum_\alpha \frac{1}{\sqrt{v_\alpha^t}+\epsilon}\frac{1-\beta_1}{1-\beta_1^t}\mathbb{E}_{\pi_t}[S_t\partial_\alpha S_t]\sum_{t'=1}^{t}\beta_1^{t-t'}g_\alpha^{t'}$$

whre $g_\alpha^{t'} = \mathbb{E}_{\pi_{t'}}[A\partial_\alpha S_{t'}]$

So we have this more complicated thing

$$K^{nl}(X,P,t|X',P',t') = \sum_\alpha \frac{1}{\sqrt{v_\alpha^t}+\epsilon}\partial_\alpha S_t\partial_\alpha S_{t'}$$

nl stands for nonlocal. here t, t' denote which version of the model are used to compute the scores.

Conceptually, this thing is a simple generalization of the SGD version. But this seems a lot more annoying to study. But I suspect any qualitative lessons will not be very different if we ignore momentum. So first, the best thing to study is the simplified no-momentum or weight decay version of this, which i guess is RMSPROP? Basically, just keep the preconditioning factor, and ignore momentum.