A Note About KL Divergence

Arushi Somani

November 2, 2025

1 KL Divergence

Suppose you're in Vegas, and you've had the misfortune of encountering a crooked croupier (let's call him p). You suspect he is using loaded dice. You've been watching him for a while, and you've developed a theory for how the dice are loaded (call your theory q). The KL divergence between q and p is the measure of how surprised you and your wallet would be, on average, if you were betting according to your theory q but the dice were behaving according to p. This surprise is also a measure of the distance between your and the croupier's distribution— how far was your guess?

Imagine now that perhaps you rightfully called out this crooked croupier, which led to us getting kicked out of that casino. But of course we must get even. So we find some long-suffering friend of ours. We need a method of long-distance communication. We determine that it shall be that you will blink at him – something like morse code – from the buffet next door. So we prep him to read blinks, and send him in.

To make the code as efficient as possible, we use information theory and our distribution q. If a play x appears with probability $q(x)$ according to us, the optimal number of bits to encode it is $-\log_2 q(x)$. We'll note that this assigns fewer bits to more likely things, thus reducing the total amount we need to blink in the direction of our friend.

In the event that our distribution is wrong and the actual distribution is p, we *should've* assigned $-\log_2 p(x)$ bits to that play instead. So our waste encoded for play x would be $-\log_2 q(x) - (-\log_2 p(x))$ or $\log_2 \frac{p(x)}{q(x)}$. If you leave your friend playing for a while at this table, the estimated waste would be:

$$\mathbb{E}_{x\sim p}\left[\log \frac{p(x)}{q(x)}\right] = \sum_x p(x) \cdot \log_2 \frac{p(x)}{q(x)} \tag{1}$$

This is **forward KL divergence**— and can be thought of as the wasted bits you need to send over based on the difference between your distribution and the true distribution.

1.1 KL Divergence and Entropy

Playing with this equation, we can discover something else quite insightful. First we know that:

$$\mathrm{KL}[p||q] = \sum_x p(x) \log \frac{p(x)}{q(x)} \tag{2}$$

Let's expand that log ratio:

$$\mathrm{KL}[p||q] = \sum_x p(x)[\log p(x) - \log q(x)] \tag{3}$$

Separate the terms:

$$\mathrm{KL}[p||q] = \sum_x p(x) \log p(x) - \sum_x p(x) \log q(x) \tag{4}$$

Now recognize what these pieces are:

$$\mathrm{KL}[p||q] = \underbrace{(-1) \cdot \left[- \sum_x p(x) \log p(x) \right]}_{H(p) = \text{entropy of reality}} + \underbrace{(-1) \cdot \left[- \sum_x p(x) \log q(x) \right]}_{H_p(q) = \text{cross-entropy of } q \text{ under } p} \tag{5}$$

Therefore:

$$\boxed{\mathrm{KL}[p||q] = H_p(q) - H(p)} \tag{6}$$

KL divergence can also be thought of as the regret, or "surprise tax" you pay for using the wrong distribution q when the true distribution is p: it is the gap between the code length you actually incur (cross-entropy $H_p(q)$) and the optimal code length you could have achieved if you had known p (entropy $H(p)$).

1.2 KL ≥ 0

$H(p)$ is the optimal average code length when the world is p. $H_p(q)$ is the average code length you get when you insist the world looks like q. In expectation, you can't beat the optimal code, and you only match it when you guessed perfectly ($p = q$). The difference $KL[p\|q]$ should therefore always be ≥ 0.

This is also equivalent to Gibbs' Inequality, which we'll succintly derive now. We begin with:

Lemma 1. *For all $x > 0$,*

$$\log x \leq x - 1,$$

with equality if and only if $x = 1$.

This is true because $\log x$ is concave. Now apply this to KL. Start with:

$$KL[p\|q] = \sum_x p(x) \log \frac{p(x)}{q(x)}.$$

Let

$$u(x) = \frac{q(x)}{p(x)}.$$

and thus

$$\log \frac{p(x)}{q(x)} = -\log u(x).$$

Since probability values cannot be negative, $p(x) \geq 0, q(x) \geq 0$, the assumption $u(x) \geq 0$ from the lemma holds for all values. From $\log u \leq u - 1$ for all $u > 0$, we get

$$-\log u(x) \;\geq\; 1 - u(x).$$

Multiply both sides by $p(x)$:

$$p(x) \log \frac{p(x)}{q(x)} \;\geq\; p(x)\big(1 - u(x)\big) = p(x) - q(x).$$

Now sum over all x:

$$\begin{aligned} \sum_x p(x) \log \frac{p(x)}{q(x)} \;&\geq\; \sum_x \big(p(x) - q(x)\big) \\ &\sum_x p(x) - \sum_x q(x) \\ &= 1 - 1 = 0 \end{aligned}$$

since both p and q are probability distributions and thus each sum to 1. The left-hand side is exactly $\mathrm{KL}[p\|q]$, so we conclude

$$\mathrm{KL}[p\|q] \geq 0,$$

with equality if and only if $\log u(x) = u(x) - 1$ for all x, i.e. $u(x) = 1$ for all x, which means $p(x) = q(x)$ everywhere.

1.3 Log Likelihood

Suppose the real world has some unknown distribution $p(x)$, and we build a model $q_\theta(x)$ with parameters θ to approximate it. In practice, we fit θ by maximizing the log-likelihood of the observed data:

$$\max_\theta \; \mathbb{E}_{x\sim p}[\log q_\theta(x)].$$

This has a close relationship with KL Divergence. Start from the forward KL:

$$\begin{aligned} \mathrm{KL}[p\|q_\theta] &= \sum_x p(x) \log \frac{p(x)}{q_\theta(x)} \qquad (7) \\ &= \sum_x p(x) \log p(x) - \sum_x p(x) \log q_\theta(x). \qquad (8) \end{aligned}$$

The first term,

$$\sum_x p(x) \log p(x)$$

is $H(p)$ and depends only on the true distribution p, which we do not control. So, as a function of θ,

$$KL[p\|q_\theta] = \text{constant} - \mathbb{E}_{x\sim p}[\log q_\theta(x)].$$

Therefore, when optimizing:

$$\theta^\star = \arg\max_\theta \mathbb{E}_{x\sim p}[\log q_\theta(x)] \tag{9}$$

$$= \arg\min_\theta \mathrm{KL}[p\|q_\theta]. \tag{10}$$

Maximum likelihood training is choosing the model whose predictions make the observed world least surprising on average. Phrased yet another way: among all q_θ, we pick the one that wastes the fewest extra bits compared to the (unknowable) true compressor for p. The closer q_θ is to p, the better the model is able to compress its data distribution, and the more it "understands".

2 Forward vs Reverse KL Divergence

Where forward KL is KL[reality||guess], reverse KL is KL[guess||reality]. While their equations look near identical, the behavior of a policy iterating under either KL could not be more different.

Forward KL is mode-covering: in a multi-modal distribution, it tries to split the difference and cover as much as possible. Imagine $p(x) = 0.01$, and $q(x) = 0.0$, then $\log(p(x)/q(x)) = \infty$. Even a tiny bit of probability in p, when q says "impossible", makes KL divergence infinite, so forward KL spreads the distribution out.

Reverse KL is mode-seeking: picks one mode of p and matches it perfectly. When $q(x) = 0$, there is no penalty. But when $p(x) = 0$ and $q(x) > 0$, then KL $= \infty$. So, Reverse KL says "You can ignore regions where p is small, but you absolutely cannot claim something is possible when it is actually impossible."

Forward KL is used by default in many contexts. Reverse KL is used in generative models like VAEs, where we'd like clear, sharp faces from specific ethnicity/ages, rather than blurry "average" human faces. We also use reverse KL in model distillation, where a large model might say an answer "could be A, B, or C" and you want your small model to model "definitely A" instead of being unable to parse the nuance of A/B/C and being unable to learn at all.

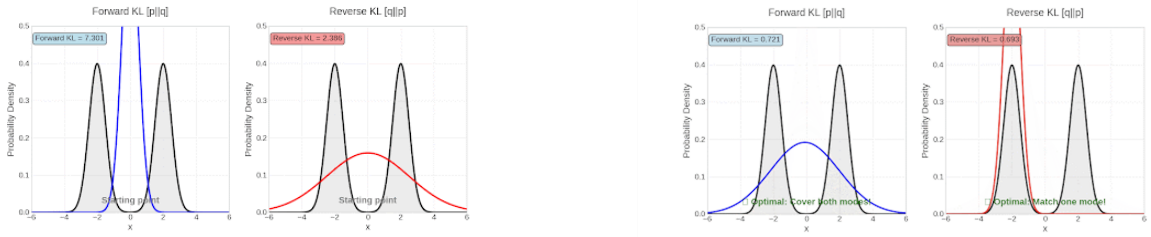

Figure 1: Evolution of q under forward KL (mode-covering) versus reverse KL (mode-seeking) optimization. **Left:** Initial configuration with q starting between two modes of p. **Right:** Final convergence after optimization—forward KL spreads to cover both peaks while reverse KL commits to matching a single mode perfectly.

3 KL Divergence Estimators

This section is heavily inspired by `http://joschu.net/blog/kl-approx.html`. *It is slightly lighter on mathematical theory than the source, and puts slightly more effort into motivating the various estimators— all flaws my own.*

In RLHF, KL Divergence is used to prevent models from going completely off the rails. We have a fixed reference model π_{ref} and the updating policy π. We define $\pi(x_t|x_{<t})$ as the distribution for position t conditioned on all previous tokens up to t. The full KL penalty would be:

$$\text{KL penalty} = \text{KL}[\pi(x_t|x_{<t}) \;||\; \pi_{\text{ref}}(x_t|x_{<t})] \tag{11}$$

$$= \sum_{v \in \text{vocab}} \pi(v|x_{<t}) \log \frac{\pi(v|x_{<t})}{\pi_{\text{ref}}(v|x_{<t})} \tag{12}$$

Computing this exactly would require evaluating all probabilities $\pi(v|x_{<t})$ and $\pi_{\text{ref}}(v|x_{<t})$ for every token v in the vocabulary, at every position t in the sequence, for every sequence in the batch. With typical values (vocab size = 50,000, sequence length = 2,048, batch size = 32), this would be roughly 3.3 billion probability evaluations per batch— which can be too memory or computationally inefficient.

A good estimator is unbiased (it has the same mean as the original) and preferably has low variance. A naive estimator would be:

$$k_1 = -\log \frac{p(x)}{q(x)} = -\log r \tag{13}$$

$$\hat{\text{KL}} = \mathbb{E}(K_1) = \mathbb{E}(-\log r) \tag{14}$$

It is unbiased, but it has very high variance. This value can often be negative, even though KL ≥ 0.

We can sample from q, and for each sample x we can compute $\log r(x) = \log \frac{p(x)}{q(x)}$. Any estimator we build has to be some function $g(\log r)$. So our goal is to pick g such that $\mathbb{E}_q[g(\log r)]$ approximates KL well.

Let $t = \log r$ to keep things clean. When $p = q$, we have $r = 1$. Ideally, our estimator should have the following properties:

1. When $p = q$, KL is zero. So we want $g(0) = 0$.
2. Locally matches KL when p and q are close to each other (in practice, p and q are often close). When p and q are close, small perturbations don't matter. Thus, $g'(0) = 0$.
3. Has lower per-sample variance than k_1. Here, in order to avoid a dive into Fisher information theory, I'll ask us to assume that the naive estimator $(-\log r)$ has second derivative 1 in the right coordinates. To measure distance on the same scale, we want $g''(0) = 1$ (Read the original blog if you want to dig further in).

Looking at the Taylor expansion of g around 0:

$$g(t) = a_0 + a_1 t + \tfrac{1}{2} a_2 t^2 + O(t^3)$$

Knowing our above constraints: $a_0 = g(0) = 0$, $a_1 = g'(0) = 0$ and $a_2 = g''(0) = 1$. So

$$g(t) = \tfrac{1}{2} t^2 + O(t^3)$$

And given we want the simpliest g, we simplify drop the higher-order value to get

$$g(\log r) = \frac{1}{2} (\log r)^2$$

Some nice things fall out of this estimation— it's always positive (like our KL value), it measures a distance between p and q and is lower variance than our naive estimator. We have also confirmed with our derivation above that it is

$$\mathbb{E}(K_2) = KL(q||p) + O(\delta^3)$$

As you will note, this is definitionally *not* unbiased— though the bias is small in practice. We can be quite happy with our k_2 estimator.

But we can yet do better! Is there a way to make an unbiased estimator with lower variance? Quoting the original blog, *"The general way to lower variance is with a control variate— take k_1 and add something that has expectation 0 but is negatively correlated with k_1."*

What do we know that might have expectation zero? Well, we know that $\mathbb{E}(r) = 1$. And so $(r - 1)$ is guaranteed to have zero expectation. If we can find a λ such that $-\log r + \lambda(1 - r)$ has lower variance, we'll have a lower var, unbiased estimator.

Calculating the optimal λ is hard, but we can estiamte a reasonable value of λ to be 1 (see again original doc for an explanation for why). This is the k_3 estimator

$$k_3 = (r - 1) - \log r$$

This is an example of a Bregman Divergence— the gap between a convex curve and the tangent line drawn from the curve at some point x.

4 Further Readings of Note

For the curious reader who wants to pursue an even deeper understanding, this document lacks coverage on the following:

- The relationship between log-likelihood and KL
- f-divergences and Bregman divergences
- Local geometry and Fisher information