Besides Cross Entropy, What Other Choices Are There for LM Loss?

Su Jianlin

2026-08-09

For a long time, Cross Entropy has been the standard loss function for LLM pre-training and fine-tuning. Can this "standard" be changed? If we want to change it, what choices do we have? And what impacts will it bring after the change?

Many readers may have never carefully scrutinized these questions. On one hand, cross entropy is simple and effective, backed by an information-theoretic interpretation, making us feel it is very "natural" and gladly accept it. On the other hand, changing the loss function is a matter where "a single pull affects the whole body"—changing it means all comparisons based on the loss are no longer valid, and we can only compare downstream task performances, which requires too much engineering effort.

However, "natural" does not mean "the only choice". Thinking through the underlying principles clearly not only helps us better understand the model's optimization process, but may also bring new improvement perspectives for enhancing effectiveness.

Analysis

More precisely, cross entropy is the standard loss function for classification problems, and LLM training looks like a token-by-token classification problem, so cross entropy loss is adopted. From this perspective, can we simply switch to other classification loss functions?

Unfortunately, no. Unlike conventional classification problems, natural language rules are one-to-many. For example, after "Baiqie" (plain sliced), it can be followed by "Ji" (chicken), but also by "Ya" (duck), "Gou" (dog), "Yang" (sheep), etc. Therefore, what we need to model is the complete distribution, not just predicting a single label. In other words, we need to estimate the probabilities of "Ji", "Ya", "Gou", and "Yang" following "Baiqie", rather than just giving a single correct answer—the correct answer is not unique.

The core challenge here is that the training corpus is fed into the model "sparsely". This time we get "Baiqie Ji", next time we might get "Baiqie Ya", and later we might get "Baiqie Ji" again. We cannot pre-compute the complete frequency distribution, which requires the loss function to have the ability to transform into a sampled form. Mathematically speaking, this requires the loss function to be linear with respect to the target distribution $\boldsymbol{p}$, i.e.,

$$L(\boldsymbol{p},\boldsymbol{q}) = \sum_{i=1}^n p_i S(\boldsymbol{q}, i) = \mathbb{E}_{i\sim\boldsymbol{p}}[S(\boldsymbol{q}, i)] \qquad \text{s.t.} \qquad \boldsymbol{p} = \operatorname*{argmin}_{\boldsymbol{q}\in\Delta^{n-1}} L(\boldsymbol{p},\boldsymbol{q}) \tag{1}$$

where $\boldsymbol{p} = (p_1, p_2, \cdots, p_n), \boldsymbol{q} = (q_1, q_2, \cdots, q_n)$ represent the target distribution and the predicted distribution respectively. The condition after s.t. indicates that after fixing $\boldsymbol{p}$, the minimizer of $L(\boldsymbol{p},\boldsymbol{q})$ is $\boldsymbol{q}^* = \boldsymbol{p}$, which is a basic requirement for a loss function. The linearity constraint excludes many common probability measures, such as Total Variation:

$$TV(\boldsymbol{p},\boldsymbol{q}) = \sum_{i=1}^n |p_i - q_i| = \sum_{i=1}^n p_i \left|1 - \frac{q_i}{p_i}\right| = \mathbb{E}_{i\sim\boldsymbol{p}}\left[\left|1 - \frac{q_i}{p_i}\right|\right] \tag{2}$$

To estimate it via sampling, we need to compute $|1 - q_i/p_i|$, but we cannot know p_i in advance, so Total Variation cannot be adapted into a form used for LLM training.

Derivation

Now let us solve the objective (1). According to the basic requirement of a loss function, after fixing $\boldsymbol{p}$, the minimum value of $L(\boldsymbol{p}, \boldsymbol{q})$ is $H(\boldsymbol{p}) \triangleq L(\boldsymbol{p}, \boldsymbol{p})$, so we can write

$$L(\boldsymbol{p}, \boldsymbol{q}) \geq H(\boldsymbol{p}) \tag{3}$$

Now we focus our attention on the variable $\boldsymbol{p}$. Obviously, $L(\boldsymbol{p}, \boldsymbol{q})$ is linear with respect to $\boldsymbol{p}$, so when we fix $\boldsymbol{q}$, the left side describes a hyperplane, and the right side describes a hypersurface; they intersect at $\boldsymbol{p} = \boldsymbol{q}$. If we further assume or require that the minimizer is unique, then the intersection is actually a tangency, meaning $L(\boldsymbol{p}, \boldsymbol{q})$ is the tangent plane of $H(\boldsymbol{p})$ at $\boldsymbol{p} = \boldsymbol{q}$!

Since the above inequality holds constantly, this is equivalent to saying that $H(\boldsymbol{p})$ is always below its tangent plane, which is exactly the definition of a concave function! So we can determine that $H(\boldsymbol{p})$ is a concave function. Conversely, given any concave function $H(\boldsymbol{p})$ with respect to $\boldsymbol{p}$, its tangent plane at $\boldsymbol{p} = \boldsymbol{q}$ is

$$H(\boldsymbol{q}) + (\boldsymbol{p} - \boldsymbol{q}) \cdot \nabla_{\boldsymbol{q}} H(\boldsymbol{q}) = \boldsymbol{p} \cdot \big[H(\boldsymbol{q}) + \nabla_{\boldsymbol{q}} H(\boldsymbol{q}) - \boldsymbol{q} \cdot \nabla_{\boldsymbol{q}} H(\boldsymbol{q})\big] \tag{4}$$

where the equality utilizes the constraint that $\boldsymbol{p}, \boldsymbol{q} \in \Delta^{n-1}$ (the sum of components is 1), "$\cdot$" denotes the inner product, and the addition of a vector and a scalar is handled element-wise. According to the previous derivation, the above equation is the desired $L(\boldsymbol{p}, \boldsymbol{q})$, so we can directly read off

$$S(\boldsymbol{q}, i) = H(\boldsymbol{q}) + \partial_i H(\boldsymbol{q}) - \boldsymbol{q} \cdot \nabla_{\boldsymbol{q}} H(\boldsymbol{q}) \tag{5}$$

$$= H(\boldsymbol{q}) + (\boldsymbol{e}_i - \boldsymbol{q}) \cdot \nabla_{\boldsymbol{q}} H(\boldsymbol{q}) \tag{6}$$

This is the general form of $S(\boldsymbol{q}, i)$ (allowing the addition or subtraction of a constant and multiplication by a positive constant), where $\partial_i H(\boldsymbol{q})$ denotes the i-th component of $\nabla_{\boldsymbol{q}} H(\boldsymbol{q})$, and $\boldsymbol{e}_i$ is the one-hot vector with 1 at the i-th position. It is not difficult to see that $S(\boldsymbol{q}, i)$ is linear with respect to $H(\boldsymbol{q})$, and since the linear interpolation of two concave functions is still concave, the linear interpolation of two scoring functions is also a scoring function.

Scoring

The above result actually has a specific name, called Proper Scoring Rules. We will not trace the origin of this name here; instead, let us give a few classic examples:

Name	$H(\boldsymbol{p})$	$S(\boldsymbol{q}, i)$
Logarithmic Score (Cross Entropy)	$-\sum_i p_i \log p_i$	$-\log q_i$
Brier Score (Square Loss)	$1 - \sum_i p_i^2$	$\|\boldsymbol{q} - \boldsymbol{e}_i\|^2$
Tsallis Score ($\alpha > 0$)	$\frac{1-\sum_i p_i^\alpha}{\alpha-1}$	$\sum_j q_j^\alpha - \frac{\alpha}{\alpha-1} q_i^{\alpha-1} + \frac{1}{\alpha-1}$
Spherical Score ($\alpha > 0$)	$\frac{1-\|\boldsymbol{p}\|_\alpha}{\alpha-1}$	$\frac{1}{\alpha-1}\left(1 - \frac{q_i^{\alpha-1}}{\|\boldsymbol{q}\|_\alpha^{\alpha-1}}\right)$
Rényi Score ($0 < \alpha < 1$)	$\frac{1}{1-\alpha} \log \sum_i p_i^\alpha$	$\frac{1}{1-\alpha}\left(\log \sum_j q_j^\alpha + \alpha \frac{q_i^{\alpha-1}}{\sum_j q_j^\alpha} - \alpha\right)$

It is worth noting that the latter three scoring functions all degenerate into the logarithmic score when $\alpha \to 1$, which means they are all some kind of generalization of cross entropy.

In particular, if we further require $S(\boldsymbol{q}, i)$ to depend only on q_i, i.e., $S(\boldsymbol{q}, i) = S(q_i)$, then there is only one choice: cross entropy $-\log q_i$. This is not difficult to prove: at this time $H(\boldsymbol{q}) = \sum_i q_i S(q_i)$, substituting into equation (5) gives

$$\cancel{S(q_i)} = \cancel{S(q_i)} + q_i S'(q_i) - \sum_j q_j^2 S'(q_j) \tag{7}$$

where the term $\sum_j q_j^2 S'(q_j)$ is effectively a constant for a single q_i, so this equation is equivalent to $q_i S'(q_i) = -c$, which is easily solved as $S(q_i) = -c \log q_i$, yielding the logarithmic score.

> **Note:** The argument here is actually slightly lacking in rigor. Due to the constraint $\sum_i q_i = 1$, the notion of "depending only on q_i" is itself not so naive. For example, $S(q_n) = S(1 - q_1 - \cdots - q_{n-1})$, so we cannot simply claim that $S(q_n)$ depends only on q_n.
>
> A similar confusion appears in the argument about "whether $\sum_j q_j^2 S'(q_j)$ depends on q_i". A relatively rigorous formulation here is to only require the first $n-1$ variables q_i to satisfy $q_i S'(q_i) = \sum_j q_j^2 S'(q_j)$, with q_n eliminated using $q_n = 1 - q_1 - \cdots - q_{n-1}$. This leaves only $n-1$ relatively independent variables. The equation $q_i S'(q_i) = \sum_j q_j^2 S'(q_j)$ means that the first $n-1$ terms $q_i S'(q_i)$ equal the same expression containing $q_1, \cdots, q_{n-1}$. But $q_i S'(q_i)$ depends at most on q_i, so this expression can only be a constant, leading to $q_i S'(q_i) = -c$.

These scoring functions can all be generalized to continuous distributions, simply by replacing the discrete variable i with the continuous variable $\boldsymbol{x}$, and replacing the summation with an integral. However, the difficulty with continuous distributions is often the inability to compute the normalization factor. These scoring functions all require an explicit probability density, so they are usually not so "user-friendly". In such scenarios, scoring functions that do not rely on the normalization factor are more needed (usually requiring gradients, which is unique to continuous distributions), but we will not expand on this here.

Gradient

As just mentioned, the latter three scoring functions are all generalizations of cross entropy. Intuitively, as long as we carefully tune α, we might achieve improvements in downstream tasks? However, things are not that simple.

In general, the model can only predict an unbounded Logits vector $\boldsymbol{z} \in \mathbb{R}^n$. We need to add an activation function to project it into a probability distribution $\boldsymbol{q}$, and the standard choice for the activation function is Softmax. Since we can only use gradient-based optimizers, the convexity and gradient properties of the loss function with respect to $\boldsymbol{z}$ become especially important. Taking the logarithmic score and Brier score as examples, the gradients of the loss with respect to $\boldsymbol{z}$ under the Softmax activation function are respectively

$$\text{Logarithmic Score (Cross Entropy)}: \qquad \nabla_{\boldsymbol{z}} S(\boldsymbol{q}, i) = \boldsymbol{q} - \boldsymbol{e}_i \tag{8}$$

$$\text{Brier Score (Square Loss)}: \qquad \nabla_{\boldsymbol{z}} S(\boldsymbol{q}, i) = 2(\operatorname{diag}(\boldsymbol{q}) - \boldsymbol{q}\boldsymbol{q}^\top)(\boldsymbol{q} - \boldsymbol{e}_i) \tag{9}$$

$$\tag{10}$$

Obviously, the gradient of cross entropy looks "cleaner". The gradient is zero if and only if $\boldsymbol{q} = \boldsymbol{e}_i$, which means that as long as the target is not reached, it can provide an effective gradient, and the further away from the target, the larger the gradient. The square loss has an extra transformation $\operatorname{diag}(\boldsymbol{q}) - \boldsymbol{q}\boldsymbol{q}^\top$; when $\boldsymbol{q} = \boldsymbol{e}_j \neq \boldsymbol{e}_i$, this term also becomes zero, meaning that when the model is "confidently wrong", it will also suffer from vanishing gradients.

This characteristic is double-edged: in the early to mid-stages, most of the model's predictions are inaccurate, which means learning efficiency with square loss is extremely low; but in the later stages, the model is basically stable, and the cases that are still "confidently wrong" might be extremely difficult or mislabeled samples. Skipping them might be more beneficial for the overall performance. Therefore, cross entropy has higher learning efficiency and should be used as the main loss, but square loss has a better ability to resist noise and can be tried in the later stages.

These can also be understood via the convexity of $S(\boldsymbol{q}, i)$ with respect to $\boldsymbol{z}$. It can be proved that under Softmax activation, cross entropy is convex with respect to $\boldsymbol{z}$, meaning the optimum is unique, and the gradient at any point points towards the target $\boldsymbol{e}_i$; but the square loss is non-convex with respect to $\boldsymbol{z}$, and these nice properties are not guaranteed to exist. The model might fall into a saturation dilemma of being "both wrong and unable to escape".

Of course, even if it is convex with respect to $\boldsymbol{z}$, $\boldsymbol{z}$ still depends on parameters, and in deep models, the loss function is generally non-convex with respect to the parameters. Our requirement for the loss function's convexity with respect to $\boldsymbol{z}$ is mostly out of the consideration of not "adding extra trouble" to the model—deep learning optimization is already hard enough, and there is no need to add obstacles at the final layer.

Reverse Derivation

The results in the previous section all share a premise—Softmax activation—but what if it's not Softmax activation? Or conversely, for a given scoring function, can we derive the "optimal" activation function suitable for that scoring function?

The first question to consider is: how should "optimal" be defined? Referring to the "Softmax + Cross Entropy" combination, it has two advantages: first, convexity with respect to $\boldsymbol{z}$; second, the gradient $\nabla_{\boldsymbol{z}} S(\boldsymbol{q}, i) = \boldsymbol{q} - \boldsymbol{e}_i$ is relatively "clean". In fact, the second point is stronger and more practical, so we start considering from the second point. That is, we hope to find a transformation $\boldsymbol{q} = \sigma(\boldsymbol{z}) \in \Delta^{n-1}$ such that for a given $S(\boldsymbol{q}, i)$, the following holds

$$\nabla_{\boldsymbol{z}} S(\boldsymbol{q}, i) = \boldsymbol{q} - \boldsymbol{e}_i \tag{11}$$

Note that $\boldsymbol{e}_i = \nabla_{\boldsymbol{z}} z_i$, so the above equation can also be written as $\nabla_{\boldsymbol{z}}(S(\boldsymbol{q}, i) + z_i) = \boldsymbol{q}$. This implies that there exists some scalar function $\Phi(\boldsymbol{z})$ independent of i such that

$$\Phi(\boldsymbol{z}) = S(\boldsymbol{q}, i) + z_i, \qquad \boldsymbol{q} = \nabla_{\boldsymbol{z}} \Phi(\boldsymbol{z}) \tag{12}$$

Rearranging gives $\Phi(\boldsymbol{z}) - z_i = S(\boldsymbol{q}, i)$. Multiplying both sides by p_i and summing gives

$$\Phi(\boldsymbol{z}) - \boldsymbol{p} \cdot \boldsymbol{z} = L(\boldsymbol{p}, \boldsymbol{q}) \geq H(\boldsymbol{p}) \tag{13}$$

Continuing to rearrange gives $\Phi(\boldsymbol{z}) \geq \boldsymbol{p} \cdot \boldsymbol{z} + H(\boldsymbol{p})$. This holds for any $\boldsymbol{p}$, so $\Phi(\boldsymbol{z})$ is an upper bound for all $\boldsymbol{p} \cdot \boldsymbol{z} + H(\boldsymbol{p})$. Then substituting $\boldsymbol{p} = \boldsymbol{q}$ into the above equation gives $\Phi(\boldsymbol{z}) - \boldsymbol{q} \cdot \boldsymbol{z} = L(\boldsymbol{q}, \boldsymbol{q}) = H(\boldsymbol{q})$, meaning equality can be achieved when $\boldsymbol{p} = \boldsymbol{q}$. Thus, $\Phi(\boldsymbol{z})$ is the "supremum" of all $\boldsymbol{p} \cdot \boldsymbol{z} + H(\boldsymbol{p})$, i.e.,

$$\Phi(\boldsymbol{z}) = \max_{\boldsymbol{p} \in \Delta^{n-1}} \boldsymbol{p} \cdot \boldsymbol{z} + H(\boldsymbol{p}), \qquad \boldsymbol{q} = \operatorname*{argmax}_{\boldsymbol{p} \in \Delta^{n-1}} \boldsymbol{p} \cdot \boldsymbol{z} + H(\boldsymbol{p}) \tag{14}$$

This is exactly the convex conjugate of the convex function $-H(\boldsymbol{p})$, and the contents of these sections combined are exactly the classic Fenchel-Young Losses framework.

Activation

In this section, we will also derive the optimal activation functions corresponding to the scoring functions mentioned earlier. It is not difficult to find that their corresponding $H(\boldsymbol{q})$ all share the same structure $g(\sum_i q_i^\alpha)$, so we can solve them uniformly. Let $t = \sum_i q_i^\alpha$, then $H(\boldsymbol{q}) = g(t)$, where

$$g(t) = \frac{1-t}{\alpha-1}\ \text{(Tsallis/Brier)},\quad g(t) = \frac{1-t^{1/\alpha}}{\alpha-1}\ \text{(Spherical)},\quad g(t) = \frac{\log t}{1-\alpha}\ \text{(Rényi)} \tag{15}$$

Introducing the Lagrangian function $\boldsymbol{q}\cdot\boldsymbol{z} + H(\boldsymbol{q}) - \lambda(\sum_i q_i - 1)$, taking the derivative with respect to q_i and setting it to 0 yields

$$z_i + \alpha g'(t) q_i^{\alpha-1} = \lambda \qquad \Rightarrow \qquad q_i^{\alpha-1} = \frac{\lambda - z_i}{\alpha g'(t)} \tag{16}$$

For an individual component, λ and $\alpha g'(t)$ are shared "constants". We are to adjust these two constants so that $\boldsymbol{q}$ becomes a proper distribution. Note that q_i only has two possibilities: $q_i > 0$ and $q_i = 0$. The latter is trivial, so we only need to analyze the former. Also note that when $\alpha > 1$, $g'(t) < 0$, and when $\alpha < 1$, $g'(t) > 0$. Therefore, we can write

$$q_i = \begin{cases} e^{z_i - \lambda}, & \alpha \to 1 \\ \left[\dfrac{z_i - \lambda}{-\alpha g'(t)}\right]_+^{\frac{1}{\alpha-1}}, & \alpha \neq 1 \end{cases} \tag{17}$$

where $[x]_+ = \max(x, 0)$. But note that when $\alpha < 1$, this truncation is redundant, because the exponent is less than 0, and a negative power of zero is undefined, so the truncation definitely does not take effect. This also indicates that $\alpha > 1$ corresponds to a sparse distribution, while $\alpha < 1$ corresponds to a dense distribution. As for λ, it is determined by the equation $\sum_i q_i = 1$. Obviously, when $\alpha \to 1$, this is exactly the classic Softmax. When $\alpha \neq 1$, for the Tsallis score we have $g'(t) = 1/(1-\alpha)$, so

$$q_i = \left[\frac{\alpha-1}{\alpha}(z_i - \lambda)\right]_+^{\frac{1}{\alpha-1}} \tag{18}$$

λ can be solved using the bisection method under the condition $\sum_i q_i = 1$. When $\alpha = 2$, the result is exactly Sparsemax. The other cases are called Entmax-α. When $\alpha = 2$ and $\alpha = 1.5$, λ has a more efficient exact solution than the bisection method, which we also introduced in *The Path to Probability Distributions: Taking Stock of Softmax and Its Alternatives*. The results for other scores are slightly more complex, so we leave them for the reader to try.

Summary

In this article, from the two perspectives of "learning distributions" and "allowing sampling", we derived the general construction method for LM Loss. Subsequently, combined with the activation function used for the predicted distribution, we calculated the gradients and convexity of these losses to briefly judge their pros and cons. Finally, we attempted to reverse-derive the matching optimal activation function from a given loss, and the optimal activation function for cross entropy is exactly Softmax—this explains why these two almost always appear together.