MoE Travelogue: 9. The Gate Normalization Debate

Su Jianlin

2026-06-17

Tracing back the history of MoE, we can find that in the early years, when the MoE Router served as a Gate multiplied onto an Expert, it basically used Softmax activation; to this day it remains one of the standard forms of MoE. However, to accommodate Loss-Free load balancing, DeepSeek changed the activation function to Sigmoid, and proved that this is also a highly competitive scheme, which has triggered more in-depth reflection and experimentation on the form of the Router.

Even within Softmax, there are two slightly different practices: should we first apply Softmax and then select Top-k, or first select Top-k and then apply Softmax? The latter can also be understood as performing an additional normalization on the selected Top-k, i.e., Re-Norm. Thus, whether the Gate's activation function should be normalized, and if so, whether to normalize before selecting Top-k or to Re-Norm, is the topic to be discussed in this article.

1 Problem Description

We know that the general form of MoE is

$$\boldsymbol{y} = \sum_{i\in \text{argtop}_k \boldsymbol{\rho}} \rho_i \boldsymbol{e}_i \tag{1}$$

Here $\boldsymbol{\rho}$ actually plays two roles: when it is used to select the Top-k Experts, its role is Router; when it is used to multiply onto the Expert, its role is Gate. From the design of MoE, the core role of $\boldsymbol{\rho}$ is obviously that of the Router; the role of the Gate is to provide gradients for it during training.

The problem we want to discuss can also be understood as how to construct $\boldsymbol{\rho} = (\rho_1, \rho_2, \cdots, \rho_n)$ more scientifically, so that the Router can obtain better gradients. For a long time, the standard answer has been Softmax, namely

$$\rho_i = \frac{e^{s_i}}{\sum_{j=1}^n e^{s_j}} \tag{2}$$

where $\boldsymbol{s} = (s_1, s_2, \cdots, s_n)$ are the Logits directly projected by the linear layer. However, although this answer is "standard," the author has not found any explanation for it; it seems that everyone simply accepted it and kept using it, which once made the author very confused about the training mechanism of MoE.

2 Other Choices

As mentioned at the beginning, DeepSeek tried Sigmoid activation for Loss-Free load balancing and later used it in DeepSeek-V3; its success shows that non-Softmax activations can also achieve

good effects. This has inspired people to try more general approaches. For example, ReMoE used ReLU activation, while from the geometric perspective presented in MoE Travelogue: 1. Starting from Geometric Meaning, arbitrary non-negative activation functions are allowed.

Furthermore, regarding the form of MoE, there is another option of Re-Norm, which modifies (1) to

$$\boldsymbol{y} = \frac{\sum\limits_{i\in\operatorname{argtop}_k \boldsymbol{\rho}} \rho_i \boldsymbol{e}_i}{\sum\limits_{i\in\operatorname{argtop}_k \boldsymbol{\rho}} \rho_i} \tag{3}$$

That is, re-performing a normalization on the selected Top-k ρ_i. For Softmax, this is equivalent to selecting the Top-k using $\boldsymbol{s}$, setting the unselected ones to $-\infty$, and then applying Softmax again. The benefit of Re-Norm is that it makes the forward computation numerically more stable, but note that when using Re-Norm, k must be at least greater than 1; otherwise $\boldsymbol{\rho}$ will have no gradients at all and thus cannot be trained.

Synthesizing current practices from various parties, the effects of these MoE variants are all more or less the same, with no single one being significantly superior. Since practice cannot distinguish a winner, we will discuss theoretically which form is more scientific.

3 Design Principles

Our goal is to find a first principle that is closer to the essence, and use it to derive the current gating mechanism of MoE.

So, the first question is naturally what this "principle" is. For simplicity, consider $k = 1$ first. We know that the most important characteristic of MoE is Sparse: first a Router determines which Experts to activate, and then only these Experts are computed, thereby increasing the number of parameters while controlling the amount of computation. If this is the only idea, then the naive model should be

$$\boldsymbol{f}\left(\boldsymbol{e}_{\operatorname{argmax}\boldsymbol{\rho}}\right) \tag{4}$$

That is, from the Router $\boldsymbol{\rho}$, pick the one with the highest score and activate the corresponding Expert. This form is completely fine for inference, but during training, the Router will not receive any gradients and thus cannot be updated. Therefore, we must find a way to design gradients for the Router. How to design gradients for the Router? To answer this question, we must first clarify: what kind of Router do we need?

Since only 1 Expert can be activated, we naturally hope that this Expert is the one with the best effect. If we use ℓ to denote the loss function, then our expectation can be written as

$$\operatorname{argmax}\boldsymbol{\rho} = \operatorname{argmin}\left[\ell(\boldsymbol{e}_1), \ell(\boldsymbol{e}_2), \cdots, \ell(\boldsymbol{e}_n)\right] \tag{5}$$

This is the design principle we seek.

4 Target Transformation

However, target (5) is not yet a loss function that can be directly used for training; it needs further transformation. To this end, we construct two distributions. The first is the target distribution $\boldsymbol{q} = (q_1, q_2, \cdots, q_n)$ built from the loss function, defined as

$$q_i = \frac{e^{-\ell(\boldsymbol{e}_i)/\tau}}{\sum_{j=1}^n e^{-\ell(\boldsymbol{e}_j)/\tau}} \tag{6}$$

This distribution has nothing to do with the Router; for Router learning it is the "target distribution." The second is the predictive distribution $\boldsymbol{p}$ built from $\boldsymbol{\rho}$; here there are many possibilities. For example, $\boldsymbol{\rho}$ itself could be the distribution $\boldsymbol{p}$ (if $\boldsymbol{\rho}$ is already normalized), or $\boldsymbol{p}$ could be the Softmax of $\boldsymbol{\rho}$ (at this point $\boldsymbol{\rho}$ is the Logits), or some normalization method other than Softmax. In short, $\boldsymbol{p}$ is some probabilistic distribution representation of the Router; let its parameters be $\boldsymbol{\theta}$.

We transform target (5) into pulling the distance between $\boldsymbol{p}$ and $\boldsymbol{q}$ closer, thereby providing gradients for $\boldsymbol{\theta}$. To this end, we consider minimizing the KL divergence

$$KL(\boldsymbol{p}\|\boldsymbol{q}) = \sum_{i=1}^n p_i \log \frac{p_i}{q_i} \tag{7}$$

After slight rearrangement we get

$$KL(\boldsymbol{p}\|\boldsymbol{q}) = -\mathcal{H}(\boldsymbol{p}) + \frac{1}{\tau}\sum_{i=1}^n p_i \ell(\boldsymbol{e}_i) - \log \sum_{i=1}^n e^{-\ell(\boldsymbol{e}_i)/\tau} \tag{8}$$

It can be seen that this objective has three terms. The first term is the negative entropy $-\mathcal{H}(\boldsymbol{p})$; minimizing it means maximizing entropy, which actually encourages the model to explore fully. This can be considered to be already playing a similar role as load balancing, so we will ignore it for now. The third term has nothing to do with $\boldsymbol{p}$, i.e., with $\boldsymbol{\theta}$, so the equivalent loss function is $\mathcal{L} = \sum_{i=1}^n p_i \ell(\boldsymbol{e}_i)$.

5 Straight-Through Estimator

Taking the gradient of the equivalent loss, we get

$$\nabla_{\boldsymbol{\theta}}\mathcal{L} = \sum_{i=1}^n \nabla_{\boldsymbol{\theta}} p_i \cdot \ell(\boldsymbol{e}_i) = \sum_{i=1}^n p_i \nabla_{\boldsymbol{\theta}} \log p_i \cdot \ell(\boldsymbol{e}_i) = \mathbb{E}_{i\sim\boldsymbol{p}}[\nabla_{\boldsymbol{\theta}} \log p_i \cdot \ell(\boldsymbol{e}_i)] \tag{9}$$

The key here is to use $\nabla_{\boldsymbol{\theta}} p_i = p_i \nabla_{\boldsymbol{\theta}} \log p_i$ to isolate a term p_i, so that the summation can be transformed into an expectation, and then sampling can be used to achieve the sparse computation goal of MoE. Astute readers may have already recognized that this is precisely REINFORCE in policy gradients! (refer to Looking at Optimization from Sampling: A Unified Perspective of Differentiable and Non-Differentiable Optimization and Policy Gradient and Zeroth-Order Optimization: Different Paths, Same Destination)

The problem with REINFORCE is high variance. Intuitively, this is because it places p_i outside the loss function ℓ; if possible, we would prefer a "reparameterized" form where p_i is inside ℓ. To derive such a form, we use the invariance of REINFORCE to subtracting a baseline to obtain

$$\begin{aligned}
\mathbb{E}_{i\sim\boldsymbol{p}}[\nabla_{\boldsymbol{\theta}} \log p_i \cdot \ell(\boldsymbol{e}_i)] &= \mathbb{E}_{i\sim\boldsymbol{p}}[\nabla_{\boldsymbol{\theta}} \log p_i \cdot (\ell(\boldsymbol{e}_i) - \ell(\boldsymbol{0}))] \\
&\approx \mathbb{E}_{i\sim\boldsymbol{p}}[\nabla_{\boldsymbol{\theta}} \log p_i \cdot \langle \nabla_{\boldsymbol{e}_i}\ell(\boldsymbol{e}_i), \boldsymbol{e}_i - \boldsymbol{0}\rangle] \\
&= \mathbb{E}_{i\sim\boldsymbol{p}}[\nabla_{\boldsymbol{\theta}} \langle \nabla_{\boldsymbol{e}_i}\ell(\boldsymbol{e}_i), \log p_i \cdot \boldsymbol{e}_i\rangle] \\
&= \mathbb{E}_{i\sim\boldsymbol{p}}[\nabla_{\boldsymbol{\theta}} \ell((\log p_i + [1 - \log p_i]_{\text{sg}}) \cdot \boldsymbol{e}_i)] \\
&= \nabla_{\boldsymbol{\theta}}\mathbb{E}_{i\sim\boldsymbol{p}}[\ell((\log p_i + [1 - \log p_i]_{\text{sg}}) \cdot \boldsymbol{e}_i)]
\end{aligned} \tag{10}$$

where the approximate equal sign $\approx$ is a first-order Taylor approximation at $\boldsymbol{e}_i$, and $[\,\cdot\,]_{\text{sg}}$ denotes Stop Gradient. Ultimately we obtain a Straight-Through Estimator (STE) with "forward propagation using 1 and backward propagation using $\log p_i$" to provide gradients for the Router.

6 Final Form

Although STE can provide a feasible training scheme, due to the inconsistency between forward and backward propagation, it often only yields suboptimal results. At this point, a very magical improvement is—changing each Expert to $p_i \boldsymbol{e}_i$! Repeating the above derivation, we have

$$\begin{aligned}
\mathbb{E}_{i\sim\boldsymbol{p}}[\nabla_{\boldsymbol{\theta}} \log p_i \cdot \ell(p_i \boldsymbol{e}_i)] &= \mathbb{E}_{i\sim\boldsymbol{p}}[\nabla_{\boldsymbol{\theta}} \log p_i \cdot (\ell(p_i \boldsymbol{e}_i) - \ell(\boldsymbol{0}))] \\
&\approx \mathbb{E}_{i\sim\boldsymbol{p}}[\nabla_{\boldsymbol{\theta}} \log p_i \cdot \langle \nabla_{p_i \boldsymbol{e}_i} \ell(p_i \boldsymbol{e}_i), p_i \boldsymbol{e}_i - \boldsymbol{0}\rangle] \\
&= \mathbb{E}_{i\sim\boldsymbol{p}}[\nabla_{\boldsymbol{\theta}} \langle \nabla_{p_i \boldsymbol{e}_i} \ell(p_i \boldsymbol{e}_i), p_i \boldsymbol{e}_i\rangle] \\
&= \mathbb{E}_{i\sim\boldsymbol{p}}[\nabla_{\boldsymbol{\theta}} \ell(p_i \boldsymbol{e}_i)] \\
&= \nabla_{\boldsymbol{\theta}} \mathbb{E}_{i\sim\boldsymbol{p}}[\ell(p_i \boldsymbol{e}_i)]
\end{aligned} \tag{11}$$

The transformation process is extremely exquisite and worth savoring carefully. By changing the Expert from $\boldsymbol{e}_i$ to $p_i \boldsymbol{e}_i$, we eliminated the Stop Gradient, achieving consistency between forward and backward, and theoretically raising the ceiling of model performance.

Now, we can answer the question posed at the beginning:

> If we need a top-down probabilistic derivation, then when the Router acts as a Gate it should be normalized, but should not Re-Norm.

7 To Sample or Not

One detail worth noting is: $\mathbb{E}_{i\sim\boldsymbol{p}}$ means we should sample from $\boldsymbol{p}$, but in practice we usually directly select Top-k. How should we understand this difference?

This is actually a trade-off between diversity and stability. Random sampling encourages the model to explore more fully, but sampling increases gradient variance and introduces additional instability; directly selecting Top-k is more stable, but there is a concern that the model may fall into a suboptimal solution, or even cause model collapse. Fortunately, various load balancing strategies are now quite mature, which to a certain extent already encourage the model to explore comprehensively, so selecting Top-k remains the mainstream for now.

If you want to sample while maintaining stability, you can slightly expand on the basis of Top-k rather than completely free sampling. For example, first select Top-$k + c$, then randomly pick k from the $k + c$ Experts; or add slight noise on top of the Logits of $\boldsymbol{p}$, and then select Top-k. This increases randomness without straying too far from the original Top-k, thus striking a balance with stability.

8 Related Work

It needs to be clarified that the derivation in this article is not new. It is extracted and modified by the author from Teacher Liu Liyuan's article Sparse Backpropagation for MoE Training. This article also has an upper volume Bridging Discrete and Backpropagation: Straight-Through and Beyond and a lower volume GRIN: GRadient-INformed MoE.

Although they are already from around 2023 and 2024, if you want to deepen your understanding of the MoE Router, it is still highly recommended to read this trilogy. They provide a unified probabilistic framework for designing gradients for various discretization operations. Of course,

the probabilistic framework also has its limitations, namely that it is rather formal and a bit "constricting" to operate.

For example, when $k = 2$, if we generalize the previous results in parallel, it should be

$$\mathbb{E}_{i,j\sim\boldsymbol{p}}[\nabla_{\boldsymbol{\theta}} \log p_i p_j \cdot \ell(p_i p_j(\boldsymbol{e}_i + \boldsymbol{e}_j))] \approx \nabla_{\boldsymbol{\theta}} \mathbb{E}_{i,j\sim\boldsymbol{p}}[\ell(p_i p_j(\boldsymbol{e}_i + \boldsymbol{e}_j))] \tag{12}$$

That is, take the joint distribution $p_i p_j$ and the sum of expert pairs $\boldsymbol{e}_i + \boldsymbol{e}_j$ as the basic unit, converting Top-2 into Top-1. However, the MoE we always use is actually in the form of $\ell(p_i \boldsymbol{e}_i + p_j \boldsymbol{e}_j)$, which is not easy to find an exact probabilistic derivation for.

At this point, a more "relaxed" way of understanding might be to treat it directly as something similar to MaxPooling, without insisting on a probabilistic interpretation. Of course, you can also choose to understand it according to the geometric meaning in MoE Travelogue: 1. Starting from Geometric Meaning. In summary, the probabilistic framework only proves the feasibility of a certain scheme, but in principle does not deny the feasibility of other schemes.

9 Summary

This article attempts to start from first principles to discuss the design problem of Router and Gate in MoE, providing a probabilistic explanation for gate normalization.