Linearizing Softmax Attention into Gated DeltaNet

Su Jianlin

July 21, 2026

In the article "Taylor Expansion of LogSumExp and Softmax", we introduced the approximate expansion of LogSumExp and Softmax, and thereby obtained a simple scheme for linearizing Softmax Attention. However, in that article, the resulting linear attention was merely in the form of Vanilla Linear Attention.

In this article, we go a step further and attempt to linearize Softmax Attention into a linear attention variant with the Delta Rule. Surprisingly, the final result is not the basic DeltaNet, but directly yields Gated DeltaNet (GDN).

Problem Background

Consider the variation pattern of the Attention output with respect to Key and Value, given a fixed Query $\boldsymbol{q}$. Introducing the notation

$$\boldsymbol{o}_t = \sum_{i=1}^t \alpha_{t,i}\boldsymbol{v}_i, \qquad \alpha_{t,i} = \frac{e^{\boldsymbol{q}\cdot\boldsymbol{k}_i}}{Z_t}, \qquad Z_t = \sum_{i=1}^t e^{\boldsymbol{q}\cdot\boldsymbol{k}_i} \tag{1}$$

Using the Softmax expansion introduced in "Taylor Expansion of LogSumExp and Softmax", we get $\alpha_{t,i} \approx \frac{1}{t}(1 + \boldsymbol{q}\cdot(\boldsymbol{k}_i - \bar{\boldsymbol{k}}_t))$. Substituting this into the above equation yields

$$\boldsymbol{o}_t \approx \frac{1}{t}\sum_{i=1}^t (1 + \boldsymbol{q}\cdot(\boldsymbol{k}_i - \bar{\boldsymbol{k}}_t))\boldsymbol{v}_i = \bar{\boldsymbol{v}}_t + \left(\frac{1}{t}\sum_{i=1}^t \boldsymbol{v}_i\boldsymbol{k}_i^\top - \bar{\boldsymbol{v}}_t\bar{\boldsymbol{k}}_t^\top\right)\boldsymbol{q} \tag{2}$$

where $\bar{\boldsymbol{k}}_t, \bar{\boldsymbol{v}}_t$ are the means of the first t Key and Value vectors, respectively. It can be seen that the final formula is a linear attention, and it is a variant of the earliest Vanilla Linear Attention (hereafter referred to as "VaLA").

From "A Brief History of Linear Attention: From Imitation, Innovation to Feedback", we know that after VaLA, there came the Delta Rule and the corresponding DeltaNet, as well as subsequent extensions like GDN and KDA, which are theoretically stronger linear attention mechanisms than VaLA. So, can we directly linearize Softmax Attention into DeltaNet, GDN, etc., to obtain a better approximation?

Recursive Form

In fact, Softmax Attention itself can be written in the form of a (non-linear) RNN, which we discussed in "Chapter of Space and Time: Treating Attention as a Quadratic Complexity RNN". This recursive form is not difficult to understand:

$$\boldsymbol{o}_t = \sum_{i=1}^{t} \alpha_{t,i}\boldsymbol{v}_i = \frac{\sum_{i=1}^{t} e^{\boldsymbol{q}\cdot\boldsymbol{k}_i}\boldsymbol{v}_i}{Z_t} = \frac{(\sum_{i=1}^{t-1} e^{\boldsymbol{q}\cdot\boldsymbol{k}_i}\boldsymbol{v}_i) + e^{\boldsymbol{q}\cdot\boldsymbol{k}_t}\boldsymbol{v}_t}{Z_t} = \frac{Z_{t-1}\boldsymbol{o}_{t-1} + e^{\boldsymbol{q}\cdot\boldsymbol{k}_t}\boldsymbol{v}_t}{Z_t} \tag{3}$$

Rearranging gives

$$\boldsymbol{o}_t = \boldsymbol{o}_{t-1} + \alpha_{t,t}(\boldsymbol{v}_t - \boldsymbol{o}_{t-1}) \tag{4}$$

Sharp-eyed readers may have already noticed that the increment on the right-hand side is in the form of $\boldsymbol{v}_t - \boldsymbol{o}_{t-1}$, which is "the difference between the new observation and the old prediction". This already faintly resembles the Delta Rule. Note that so far we have not made any approximations; this equation is completely exact, which implies a natural connection between Softmax Attention and the Delta Rule.

It should also be pointed out that Equation (4) is an RNN with a fixed $\boldsymbol{q}$, but in reality, each position has a different $\boldsymbol{q}_1, \boldsymbol{q}_2, \cdots, \boldsymbol{q}_t$. These t different $\boldsymbol{q}$'s must be substituted in, and they respectively run for $1, 2, \cdots, t$ steps to obtain their own outputs. The total complexity is $1 + 2 + \cdots + t = \mathcal{O}(t^2)$, which is the meaning of a "quadratic complexity RNN".

The fundamental reason is that for different $\boldsymbol{q}$'s, the iteration processes are independent, so each $\boldsymbol{q}$ must run a complete round, making the total complexity quadratic. To linearize it, we must find a way to separate the computation of $\boldsymbol{q}$, letting the iteration proceed purely within $\boldsymbol{k}, \boldsymbol{v}$, and then compute the output using the state variables of the iteration process together with $\boldsymbol{q}$. In this way, the iteration process does not depend on $\boldsymbol{q}$, and only needs to be executed once, reducing the complexity to linear.

Linear Approximation

Next, we apply the first-order approximation of Softmax to $\alpha_{t,t}$, yielding

$$\begin{aligned}\boldsymbol{o}_t &\approx \boldsymbol{o}_{t-1} + \frac{1}{t}(1 + \boldsymbol{q}\cdot(\boldsymbol{k}_t - \bar{\boldsymbol{k}}_t))(\boldsymbol{v}_t - \boldsymbol{o}_{t-1}) \\ &= \left(1 - \frac{1}{t}\right)\boldsymbol{o}_{t-1} + \frac{1}{t}\boldsymbol{v}_t + \frac{1}{t}(\boldsymbol{v}_t - \boldsymbol{o}_{t-1})(\boldsymbol{k}_t - \bar{\boldsymbol{k}}_t)^\top \boldsymbol{q}\end{aligned} \tag{5}$$

Some readers might wonder: didn't we just say that the first-order approximation of Softmax leading to VaLA is not accurate enough? Why are we still using a first-order approximation now? The difference is that previously we needed to approximate $\alpha_{t,i}$, where the subscripts cover all (t, i) pairs, but now we only need to approximate $\alpha_{t,t}$, which means we are only approximating the diagonal part. The approximation burden is greatly reduced, and the approximation accuracy subsequently increases.

There is another perspective that can help us better understand this point. First, let's rewrite $\alpha_{t,i}$ as

$$\alpha_{t,i} = \frac{e^{\boldsymbol{q}\cdot\boldsymbol{k}_i}}{Z_t} = \frac{e^{\boldsymbol{q}\cdot\boldsymbol{k}_i}}{Z_i}\frac{Z_i}{Z_{i+1}}\cdots\frac{Z_{t-1}}{Z_t} = \alpha_{i,i}(1-\alpha_{i+1,i+1})\cdots(1-\alpha_{t,t}) \tag{6}$$

This equation tells us that for $t > i$, $\alpha_{t,i}$ can be decomposed into the product of $\alpha_{i,i}$ and a series of $1-\alpha_{j,j}$. Directly making a first-order approximation for $\alpha_{t,i}$ versus making separate approximations for $\alpha_{i,i}$ and $\alpha_{j,j}$ is equivalent to the difference between $e^{a+b} \approx 1+a+b$ and $e^a e^b \approx (1+a)(1+b) = 1+a+b+ab$. The latter can introduce a cross-term, achieving higher precision.

Separating the Iteration

However, even with the approximation, this recursion is still quadratic, and we have not yet achieved the goal of separating out $\boldsymbol{q}$. To get closer to it, we look for a solution of the following form

$$\boldsymbol{o}_t \approx \boldsymbol{A}_t\boldsymbol{q} + \boldsymbol{b}_t \tag{7}$$

where $\boldsymbol{A}_t, \boldsymbol{b}_t$ are independent of $\boldsymbol{q}$, and we agree on the initial conditions $\boldsymbol{A}_0 = \boldsymbol{0}, \boldsymbol{b}_0 = \boldsymbol{0}$. Of course, the exact solution definitely does not look like this; we are purposefully looking for a solution of the above form to serve as an approximation of the exact solution. Substituting this into the aforementioned recursive form gives

$$\boldsymbol{A}_t\boldsymbol{q} + \boldsymbol{b}_t \approx \left(1-\frac{1}{t}\right)(\boldsymbol{A}_{t-1}\boldsymbol{q} + \boldsymbol{b}_{t-1}) + \frac{1}{t}\boldsymbol{v}_t + \frac{1}{t}(\boldsymbol{v}_t - \boldsymbol{b}_{t-1} - \boldsymbol{A}_{t-1}\boldsymbol{q})(\boldsymbol{k}_t - \bar{\boldsymbol{k}}_t)^\top\boldsymbol{q} \tag{8}$$

An intuitive idea is to separate the iterations of $\boldsymbol{A}_t, \boldsymbol{b}_t$ according to the order of $\boldsymbol{q}$, which would yield recursive formulas for $\boldsymbol{A}_t, \boldsymbol{b}_t$. However, the left-hand side has at most a linear term in $\boldsymbol{q}$, while the right-hand side features a quadratic term like $\boldsymbol{A}_{t-1}\boldsymbol{q}(\boldsymbol{k}_t - \bar{\boldsymbol{k}}_t)^\top\boldsymbol{q}$, so the separation cannot be carried out completely. We can only obtain

$$\boldsymbol{b}_t = \left(1-\frac{1}{t}\right)\boldsymbol{b}_{t-1} + \frac{1}{t}\boldsymbol{v}_t \tag{9}$$

$$\boldsymbol{A}_t = \left(1-\frac{1}{t}\right)\boldsymbol{A}_{t-1} + \frac{1}{t}(\boldsymbol{v}_t - \boldsymbol{b}_{t-1} - \boldsymbol{A}_{t-1}\boldsymbol{q})(\boldsymbol{k}_t - \bar{\boldsymbol{k}}_t)^\top \tag{10}$$

Clearly, the iteration of $\boldsymbol{b}_t$ is essentially finding the cumulative average of $\boldsymbol{v}_t$, so we can directly write $\boldsymbol{b}_t = \bar{\boldsymbol{v}}_t$. The problem is that $\boldsymbol{q}$ appears on the right-hand side of Equation (10), meaning the assumption that $\boldsymbol{A}_t$ does not depend on $\boldsymbol{q}$ fails to hold.

Simple Discard

A simple approach is to assume that $\boldsymbol{q}$ is a small quantity and directly discard it, yielding

$$\boldsymbol{A}_t = \left(1 - \frac{1}{t}\right)\boldsymbol{A}_{t-1} + \frac{1}{t}(\boldsymbol{v}_t - \bar{\boldsymbol{v}}_{t-1})(\boldsymbol{k}_t - \bar{\boldsymbol{k}}_t)^\top \tag{11}$$

This also takes the form of a cumulative average, and solving it gives

$$\boldsymbol{A}_t = \frac{1}{t}\sum_{i=1}^{t}(\boldsymbol{v}_i - \bar{\boldsymbol{v}}_{i-1})(\boldsymbol{k}_i - \bar{\boldsymbol{k}}_i)^\top \quad \Rightarrow \quad \boldsymbol{o}_t \approx \bar{\boldsymbol{v}}_t + \left(\frac{1}{t}\sum_{i=1}^{t}(\boldsymbol{v}_i - \bar{\boldsymbol{v}}_{i-1})(\boldsymbol{k}_i - \bar{\boldsymbol{k}}_i)^\top\right)\boldsymbol{q} \tag{12}$$

This looks like a new VaLA variant, but it is actually completely equivalent to Equation (2)! To prove this, we only need to verify

$$t\bar{\boldsymbol{v}}_t\bar{\boldsymbol{k}}_t^\top - (t-1)\bar{\boldsymbol{v}}_{t-1}\bar{\boldsymbol{k}}_{t-1}^\top = \underbrace{t\bar{\boldsymbol{v}}_t}_{(t-1)\bar{\boldsymbol{v}}_{t-1}+\boldsymbol{v}_t}\bar{\boldsymbol{k}}_t^\top - \bar{\boldsymbol{v}}_{t-1}\underbrace{(t-1)\bar{\boldsymbol{k}}_{t-1}^\top}_{t\bar{\boldsymbol{k}}_t^\top - \boldsymbol{k}_t^\top} = -\bar{\boldsymbol{v}}_{t-1}\bar{\boldsymbol{k}}_t^\top + \boldsymbol{v}_t\bar{\boldsymbol{k}}_t^\top + \bar{\boldsymbol{v}}_{t-1}\boldsymbol{k}_t^\top \tag{13}$$

And thus

$$\sum_{i=1}^{t}\boldsymbol{v}_i\boldsymbol{k}_i^\top - t\bar{\boldsymbol{v}}_t\bar{\boldsymbol{k}}_t^\top = \sum_{i=1}^{t}\boldsymbol{v}_i\boldsymbol{k}_i^\top - \sum_{i=1}^{t}(-\bar{\boldsymbol{v}}_{i-1}\bar{\boldsymbol{k}}_i^\top + \boldsymbol{v}_i\bar{\boldsymbol{k}}_i^\top + \bar{\boldsymbol{v}}_{i-1}\boldsymbol{k}_i^\top) = \sum_{i=1}^{t}(\boldsymbol{v}_i - \bar{\boldsymbol{v}}_{i-1})(\boldsymbol{k}_i - \bar{\boldsymbol{k}}_i)^\top \tag{14}$$

This transforms Equation (2) identically into Equation (12).

Error Principle

Another, more accurate approach is to replace $\boldsymbol{q}$ with some appropriate quantity $\boldsymbol{c}$. Intuitively, replacing $\boldsymbol{q}$ with $\boldsymbol{q}_t$ might seem like a natural and good choice, but that is not the case. Let us return to Equation (4); writing $\boldsymbol{q}$ explicitly gives

$$\boldsymbol{o}_t(\boldsymbol{q}) = \boldsymbol{o}_{t-1}(\boldsymbol{q}) + \alpha_{t,t}(\boldsymbol{q}) \cdot (\boldsymbol{v}_t - \boldsymbol{o}_{t-1}(\boldsymbol{q})) \tag{15}$$

Our goal is to replace the last $\boldsymbol{o}_{t-1}(\boldsymbol{q})$ with some $\boldsymbol{o}_{t-1}(\boldsymbol{c})$, so that no quadratic term of $\boldsymbol{q}$ will appear during the subsequent approximate expansion. The error introduced by this is

$$\alpha_{t,t}(\boldsymbol{q}) \cdot (\boldsymbol{o}_{t-1}(\boldsymbol{q}) - \boldsymbol{o}_{t-1}(\boldsymbol{c})) \tag{16}$$

Why might $\boldsymbol{c} = \boldsymbol{q}_t$ not be good? Because it only guarantees the lowest error for the single position where $\boldsymbol{q} = \boldsymbol{q}_t$, but the state variable at this moment also has to serve the subsequent $\boldsymbol{q}_{t+1}, \boldsymbol{q}_{t+2}, \cdots$. Therefore, the ideal goal is "to minimize the error for any possible $\boldsymbol{q}$ that might appear". This is the "minimum error principle" we use to search for $\boldsymbol{c}$.

This error term has a multiplicative form: if $\alpha_{t,t}(\boldsymbol{q})$ is inherently small, then the product will not be large; conversely, if $\alpha_{t,t}(\boldsymbol{q})$ is large, then $\boldsymbol{o}_{t-1}(\boldsymbol{q}) - \boldsymbol{o}_{t-1}(\boldsymbol{c})$ must be small to bring the error down. From this, we derive a "minimax strategy": first find the $\boldsymbol{q}^*$ that maximizes $\alpha_{t,t}(\boldsymbol{q})$, and then simply set $\boldsymbol{c} = \boldsymbol{q}^*$ to achieve $\boldsymbol{o}_{t-1}(\boldsymbol{q}) - \boldsymbol{o}_{t-1}(\boldsymbol{c}) = \boldsymbol{0}$. This balances the error on both sides of large and small $\alpha_{t,t}(\boldsymbol{q})$.

Grand Finale

According to the linear approximation $\alpha_{t,t} \approx \frac{1}{t}(1 + \boldsymbol{q}\cdot(\boldsymbol{k}_t - \bar{\boldsymbol{k}}_t))$, the direction of $\boldsymbol{q}$ that maximizes it is $\frac{\boldsymbol{k}_t - \bar{\boldsymbol{k}}_t}{\|\boldsymbol{k}_t - \bar{\boldsymbol{k}}_t\|}$. However, since the magnitude can be arbitrarily large, $\alpha_{t,t}$ effectively has no maximum. To obtain a finite result, we need to constrain the magnitude of $\boldsymbol{q}$. For simplicity, we assume that the magnitudes of $\boldsymbol{q}, \boldsymbol{k}$ are comparable, so we can directly take

$$\boldsymbol{q}^* = \boldsymbol{k}_t - \bar{\boldsymbol{k}}_t \tag{17}$$

This extremely simple form, when substituted into Equation (10), gives

$$\begin{aligned}\boldsymbol{A}_t &= \left(1 - \frac{1}{t}\right)\boldsymbol{A}_{t-1} + \frac{1}{t}(\boldsymbol{v}_t - \bar{\boldsymbol{v}}_{t-1} - \boldsymbol{A}_{t-1}(\boldsymbol{k}_t - \bar{\boldsymbol{k}}_t))(\boldsymbol{k}_t - \bar{\boldsymbol{k}}_t)^\top \\ &= \boldsymbol{A}_{t-1}\left(\left(1 - \frac{1}{t}\right)\boldsymbol{I} - \frac{1}{t}(\boldsymbol{k}_t - \bar{\boldsymbol{k}}_t)(\boldsymbol{k}_t - \bar{\boldsymbol{k}}_t)^\top\right) + \frac{1}{t}(\boldsymbol{v}_t - \bar{\boldsymbol{v}}_{t-1})(\boldsymbol{k}_t - \bar{\boldsymbol{k}}_t)^\top\end{aligned} \tag{18}$$

This is exactly the appearance of Gated DeltaNet (GDN)! The standard form of GDN is

$$\boldsymbol{S}_t = \boldsymbol{S}_{t-1}(\alpha_t(\boldsymbol{I} - \beta_t \boldsymbol{k}_t\boldsymbol{k}_t^\top)) + \beta_t \boldsymbol{v}_t\boldsymbol{k}_t^\top \tag{19}$$

The difference lies in whether α_t multiplies $\boldsymbol{I} - \beta_t \boldsymbol{k}_t\boldsymbol{k}_t^\top$ or just multiplies $\boldsymbol{I}$, but this difference is not fundamental, and the two can be transformed into each other. At this point, we have completed the approximate transformation from Softmax Attention to Delta Rule-based attention, and our final output is

$$\boldsymbol{o}_t = \boldsymbol{A}_t\boldsymbol{q}_t + \bar{\boldsymbol{v}}_t \tag{20}$$

Article Summary

Starting from the concept that "Softmax Attention is a quadratic complexity RNN", this article first discovered that its recursive increment naturally takes the form of the Delta Rule. Then, by applying a first-order approximation to the diagonal elements of Softmax, and finally finding a substitute for the residual $\boldsymbol{q}$ according to the "minimum error principle", we successfully linearized Softmax Attention into the form of Gated DeltaNet.

(Acknowledgments: This article was completed with the guidance of Kimi K3.)

When reposting, please include the address of this article: *https://kexue.fm/archives/11823*

For more detailed reposting matters, please refer to: *"Science Space FAQ"*