

Configuring Different Learning Rates, Can LoRA Gain a Bit More?

Su Jianlin

February 27, 2024

LoRA (**Low-Rank Adaptation**) is currently one of the most popular parameter-efficient fine-tuning methods for Large Language Models (LLMs). We previously had a brief discussion in "[LoRA from a Gradient Perspective: Introduction, Analysis, Conjectures, and Extensions](#)". In this article, we will learn about a new conclusion regarding LoRA:

By assigning different learning rates to the two matrices of LoRA, the performance of LoRA can be further improved.

This conclusion comes from the recent paper "[LoRA+: Efficient Low Rank Adaptation of Large Models](#)" (hereinafter referred to as "LoRA+"). At first glance, this conclusion might not seem particularly special, as configuring different learning rates is equivalent to introducing new hyperparameters; generally, introducing and fine-tuning hyperparameters leads to improvements. The uniqueness of "LoRA+" lies in the fact that it justifies this necessity from a theoretical perspective and determines that the optimal solution must involve the learning rate of the right matrix being greater than that of the left matrix. In short, "LoRA+" is a classic example of theory guiding training that proves effective in practice, making it well worth studying.

Brief Analysis of the Conclusion

Assume the pre-trained parameters are $W_0 \in \mathbb{R}^{n \times m}$. If full-parameter fine-tuning is used, the increment is also an $n \times m$ matrix. To reduce the number of parameters, LoRA constrains the update to a low-rank matrix, i.e., $W = W_0 + AB$, where $A \in \mathbb{R}^{n \times r}$, $B \in \mathbb{R}^{r \times m}$ and $r \ll \min(n, m)$. The original model parameters are replaced by the new W , W_0 is kept fixed, and only A and B are updated during training, as shown below:

$$\begin{array}{c} \boxed{W_0 \in \mathbb{R}^{n \times m}} \end{array} + \begin{array}{c} \boxed{A \in \mathbb{R}^{n \times r}} \end{array} \times \begin{array}{c} \boxed{B \in \mathbb{R}^{r \times m}} \end{array}$$

Note that LoRA is typically used for Dense layers. While the original paper's analysis is based on weights left-multiplying the input, in implementation, it is almost always the input right-multiplying the weights. To avoid confusion, the notation in this article aligns with the implementation: assume the layer input is $X \in \mathbb{R}^{b \times n}$, and the layer operation is $XW = X(W_0 + AB)$. Since the conclusion of "LoRA+" is independent of the pre-trained weights, we can set $W_0 = 0$ without loss of generality, simplifying the layer operation to $Y = XAB \in \mathbb{R}^{b \times m}$.

The conclusion of "LoRA+" is:

To make the effect of LoRA as close to optimal as possible, the learning rate of weight B should be greater than the learning rate of weight A .

Note that to ensure the initial model is equivalent to the original pre-trained model, LoRA usually initializes one of A or B to all zeros. I initially thought this conclusion was due to the zero initialization and should depend on the position of the zero initialization. However, after careful reading, I found that the claim in "LoRA+" is independent of zero initialization. That is, while A and B appear symmetric on the surface, they possess an inherent asymmetry such that regardless of whether A or B is initialized to zero, the conclusion remains that the learning rate of B should be greater than that of A . This is quite interesting.

However, it must be said that the explanation in the original "LoRA+" paper is quite difficult to follow. Therefore, the following is a simplified derivation based on my own reasoning. Broadly, it is based on two assumptions:

1. **Numerical Stability:** The output values of each layer in the model should be numerically stable and independent of the network width.
2. **Equal Contribution:** To make LoRA optimal, the two matrices A and B should contribute equally to the performance.

Next, we analyze and quantify these two assumptions one by one.

Numerical Stability

First, numerical stability means that each component of X, XA, XAB should be of the order $\mathcal{O}(1)$, independent of the network widths n, m . Here, $\mathcal{O}(1)$ primarily describes that its order relative to the network width is zero; it does not mean its absolute value is close to 1. This assumption is likely uncontroversial; it is hard to imagine a numerically unstable network having good predictive performance. However, some readers might question the necessity of " XA being $\mathcal{O}(1)$ " because X is the input and XAB is the output. While it is reasonable to require their stability, XA is just an intermediate variable—must it also be stable?

From the perspective of forward propagation alone, the numerical stability of XA is indeed not strictly necessary. However, if XA is unstable while XAB is stable, two cases arise: XA is too large and B is too small, which leads to A 's gradient being too small and B 's gradient being too large; or conversely, XA is too small and B is too large, leading to A 's gradient being too large and B 's gradient being too small. In short, the numerical instability of XA leads to instability in the gradients of A and B , thereby increasing optimization difficulty. Thus, it is better to include the numerical stability of XA as a condition.

This numerical stability condition reminds us of [LeCun initialization](#), which states that if $W \in \mathbb{R}^{n \times m}$ is sampled i.i.d. from a distribution with mean 0 and variance $1/n$, then the magnitude of each component of XW is roughly the same as that of X . Following the same strategy, if the input X is already $\mathcal{O}(1)$, then to ensure the components of XA and XAB are $\mathcal{O}(1)$, A and B should be initialized with variances of $1/n$ and $1/r$, respectively (means are assumed to be 0).

Of course, as mentioned, LoRA chooses one of A or B for zero initialization to maintain identity. But this is not very important; we only need to realize that variances of $1/n$ and $1/r$ allow XA and XAB to maintain numerical stability. We can then guess that after training, A and B likely still approximately have variances of $1/n$ and $1/r$. Given $r \ll n$, this is equivalent to saying that the absolute values of the components of A will be significantly smaller than those of B . This is the source of the asymmetry between A and B .

Equal Contribution

Next, let's look at the second assumption: A and B should contribute equally to the performance. This assumption seems reasonable because, in the LLM+LoRA scenario, we usually have $m = n$, meaning the number of parameters in A and B is the same, so it is logical for their contributions to be equal. If $m \neq n$, we can further generalize this assumption to state that the contribution is proportional to the number of parameters. The most basic metric for performance is, of course, the loss function, denoted here as $\mathcal{L}$.

We want to measure the change in the loss function when $A \rightarrow A + \Delta A$ and $B \rightarrow B + \Delta B$:

$$\mathcal{L}(A + \Delta A, B + \Delta B) - \mathcal{L}(A, B) \approx \left\langle \frac{\partial \mathcal{L}}{\partial A}, \Delta A \right\rangle + \left\langle \frac{\partial \mathcal{L}}{\partial B}, \Delta B \right\rangle \quad (1)$$

Here, a first-order linear approximation is used, where $\frac{\partial \mathcal{L}}{\partial A}, \frac{\partial \mathcal{L}}{\partial B}$ are the gradients of A and B , and $\langle \cdot, \cdot \rangle$ is the (Frobenius) inner product. The two terms on the right can be understood as the respective contributions of A and B . Note that the validity of the linear approximation depends on the increments $\Delta A, \Delta B$ being small, but for trained weights, the increment relative to the original weights might not be small. Therefore, we refine the "equal contribution" assumption to: " A and B should contribute equally to the performance in each update step." Since the step-wise update is usually small, the linear approximation holds well.

Considering the update amount per step leads us to the optimizer. Currently, the mainstream optimizers for pre-training and fine-tuning are Adam. We will use Adam as our primary object of analysis. We know that the Adam optimizer has two sets of moving average states and corresponding hyperparameters β_1, β_2 , making precise analysis difficult. However, for our purposes, we only need an order-of-magnitude estimate. We consider an extreme case and assume it yields the same order-of-magnitude result as the general case. This case is $\beta_1 = \beta_2 = 0$, where Adam reduces to [SignSGD](#):

$$\Delta A = -\eta_A \text{sign} \left(\frac{\partial \mathcal{L}}{\partial A} \right), \quad \Delta B = -\eta_B \text{sign} \left(\frac{\partial \mathcal{L}}{\partial B} \right) \quad (2)$$

where η_A, η_B are the respective learning rates. The conclusion of "LoRA+" is $\eta_B \gg \eta_A$.

Substituting the SignSGD increments (2) back into equation (1), we get:

$$\mathcal{L}(A + \Delta A, B + \Delta B) - \mathcal{L}(A, B) \approx \underbrace{-\eta_A \left\| \frac{\partial \mathcal{L}}{\partial A} \right\|_1}_{\Delta \mathcal{L}_A} - \underbrace{\eta_B \left\| \frac{\partial \mathcal{L}}{\partial B} \right\|_1}_{\Delta \mathcal{L}_B} \quad (3)$$

where $\| \cdot \|_1$ is the L_1 norm (the sum of the absolute values of all components). "Equal contribution" means we want $\Delta \mathcal{L}_A$ and $\Delta \mathcal{L}_B$ to be of the same order of magnitude.

Fast Derivation

Further analysis requires the specific form of the gradients. Setting $Y = XAB$ again, we can derive:

$$\frac{\partial \mathcal{L}}{\partial A} = X^\top \frac{\partial \mathcal{L}}{\partial Y} B^\top, \quad \frac{\partial \mathcal{L}}{\partial B} = A^\top X^\top \frac{\partial \mathcal{L}}{\partial Y} \quad (4)$$

Readers unfamiliar with matrix calculus might be confused by these results. In fact, I am not very familiar with it either, but there is a simple trick. For $\frac{\partial \mathcal{L}}{\partial A}$, we know it is an $n \times r$ matrix (same shape as A). Similarly, $\frac{\partial \mathcal{L}}{\partial Y}$ is a $b \times m$ matrix. According to the chain rule, $\frac{\partial \mathcal{L}}{\partial A}$ should be the product of $\frac{\partial \mathcal{L}}{\partial Y}$, X , and B . We just need to figure out how to multiply these three matrices to get an $n \times r$ matrix following the rules of matrix multiplication.

After finding the specific forms of $\frac{\partial \mathcal{L}}{\partial A}$ and $\frac{\partial \mathcal{L}}{\partial B}$, we have a quick way to understand LoRA+. First, $\Delta \mathcal{L}_A$ is proportional to $\left\| \frac{\partial \mathcal{L}}{\partial A} \right\|_1$, which is the sum of nr absolute values. If each component

is comparable, this means $\Delta\mathcal{L}_A$ is roughly proportional to nr . Since $\frac{\partial\mathcal{L}}{\partial A}$ is linear in B , we can assume the magnitude of each component of $\frac{\partial\mathcal{L}}{\partial A}$ is proportional to the magnitude of B 's components. Combined, $\Delta\mathcal{L}_A$ is proportional to both nr and the magnitude of B . Similarly, $\Delta\mathcal{L}_B$ is roughly proportional to mr and the magnitude of A . As discussed in the "Numerical Stability" section, for forward stability, the magnitude of B should be larger than that of A (proportional to their approximate standard deviations $\sqrt{1/r}$ and $\sqrt{1/n}$). Thus, for $\Delta\mathcal{L}_A$ and $\Delta\mathcal{L}_B$ to be comparable:

$$\eta_A \times nr \times \sqrt{1/r} \approx \eta_B \times mr \times \sqrt{1/n} \Rightarrow \frac{\eta_B}{\eta_A} \approx \frac{n}{m} \sqrt{\frac{n}{r}} \quad (5)$$

Considering that in practice $m = n$ and $r = \mathcal{O}(1)$, this can be simplified to:

$$\frac{\eta_B}{\eta_A} = \mathcal{O}(\sqrt{n}) \quad (6)$$

But we are not done yet; we need to check if the result is self-consistent, as one of the conditions we used—"forward numerical stability"—is still just an ideal assumption. How do we make the assumption hold as much as possible? By introducing another assumption:

In the Adam optimizer, if the ratio of the learning rates of two parameters is λ , then after long-term training, the ratio of the magnitudes of these two parameters will also be λ .

According to the SignSGD approximation (2), the magnitude of the increment per step is indeed proportional to the learning rate, but the total update is not a simple summation of each step. So this assumption feels like it "makes some sense, but not complete sense." But that's okay; assumptions are often like that—as long as they make some sense, the rest relies on empirical validation. Under this assumption, if we train with a learning rate ratio of $\frac{\eta_B}{\eta_A} = \mathcal{O}(\sqrt{n})$, then the ratio of the magnitudes of parameters B and A will also be $\mathcal{O}(\sqrt{n})$. We previously expected them to have approximate standard deviations of $\sqrt{1/r}$ and $\sqrt{1/n}$, the ratio of which is exactly $\mathcal{O}(\sqrt{n})$. The result is perfectly self-consistent!

The result in the original paper is slightly different, giving $\mathcal{O}(n)$. This is because the original paper considers ΔA and ΔB having equal increments on Y . However, Y is only the output of a model layer and does not represent the final performance, so this approach is slightly flawed. Although the original paper attempts to link the increment of Y to the increment of $\mathcal{L}$, it does not carry out the full calculation, leading to a deviation. Furthermore, the derivation in the original paper technically only applies to the special case of $b = 1, r = 1, m = n$; the general case of $b > 1, r > 1$ is simply extrapolated, meaning the analysis is not sufficiently general.

Of course, whether it is $\mathcal{O}(n)$ or $\mathcal{O}(\sqrt{n})$ is not critically important; in practice, tuning is still required. However, LoRA+ conducted experiments on models of various sizes, where r was generally 8 and n ranged from 768 to 4096. They concluded that the recommended default learning rate ratio is $2^4 = 16$, which happens to be close to $\sqrt{n/r}$. Thus, the optimal value is closer to $\mathcal{O}(\sqrt{n})$ than $\mathcal{O}(n)$.

Summary

In this article, we introduced and derived a result called "LoRA+," which supports the inherent asymmetry between the two low-rank matrices A and B in LoRA. Regardless of which matrix is initialized to zero, the learning rate of B should be set higher than that of A to achieve better performance.

*When reprinting, please include the original link: <https://kexue.fm/archives/10001>
For more details on reprinting, please refer to: "Scientific Space FAQ"*