Introducing muP

Wenhao Chai

In this blog, we introduce muP [2] (Maximal Update Parametrization), which aims at studying the transfer patterns of hyperparameters across model scales. We revisit the basic concept of muP based on the blog of Jianlin Su* and try to extend the analysis with the extra conditions.

1. Introduction.

Training large language models (LLMs) is computationally expensive, making hyperparameter tuning on large models impractical. A natural solution is to tune hyperparameters on smaller models and transfer them to larger ones. However, this requires understanding how optimal hyperparameters scale with model size. muP [2] (Maximal Update Parametrization) aims to find that how should we adjust the learning rate when we train the model with larger size (scaling law). The principle of muP is derived to ensure that forward propagation, backward propagation, and loss increments remain **stable** across changes in model scale.

Model Assumptions. Consider a MLP model with bias $f : \mathbb{R}^d \mapsto \mathbb{R}^d$. Since muP focuses on hyperparameter scaling laws with respect to model width (d), all constant terms (e.g., initialization variance $1/d$) are treated as $\mathcal{O}(1)$. For each layer, the relation between input X and output Y is

$$Y = \phi(XW_k + B_k), \tag{1}$$

where $X, Y \in \mathbb{R}^{b\times d}, W_k \in \mathbb{R}^{d\times d}, B_k \in \mathbb{R}^{1\times d}$, ϕ is the activation function.

Backward Propagation. The gradients for W, B, and X are derived as follows:

$$\frac{\partial \mathcal{L}}{\partial W} = X^T \left(\frac{\partial \mathcal{L}}{\partial Y} \odot \phi'(XW + B)\right), \tag{2}$$

$$\frac{\partial \mathcal{L}}{\partial B} = \mathbf{1}^T \left(\frac{\partial \mathcal{L}}{\partial Y} \odot \phi'(XW + B)\right), \tag{3}$$

$$\frac{\partial \mathcal{L}}{\partial X} = \left(\frac{\partial \mathcal{L}}{\partial Y} \odot \phi'(XW + B)\right) W^T. \tag{4}$$

If $\phi'(\cdot)$ is bounded by a scale-independent constant (*e.g.*, ReLU), we approximate:

$$\frac{\partial \mathcal{L}}{\partial W} \sim X^T \frac{\partial \mathcal{L}}{\partial Y}, \quad \frac{\partial \mathcal{L}}{\partial X} \sim \frac{\partial \mathcal{L}}{\partial Y} W^T, \quad \frac{\partial \mathcal{L}}{\partial B} \sim \mathbf{1}^T \frac{\partial \mathcal{L}}{\partial Y}. \tag{5}$$

*Blog: https://kexue.fm/archives/10770

Quick Visit on Loss Term. With the groundwork laid for forward and backward propagation, we now analyze the increment in the loss function. Consider the change in the loss function when $W \to W + \Delta W$:

$$\Delta \mathcal{L}_W = \mathcal{L}(W + \Delta W) - \mathcal{L}(W) \approx \left\langle \frac{\partial \mathcal{L}}{\partial W}, \Delta W \right\rangle_F, \tag{6}$$

where $\langle \cdot, \cdot \rangle_F$ denotes the Frobenius inner product, which is equivalent to flattening the matrices into vectors and computing their inner product. For gradient descent, $\Delta W = -\eta \frac{\partial \mathcal{L}}{\partial W}$, where η is the learning rate. Combining this with Equation 2, we obtain:

$$\Delta \mathcal{L}_W \approx -\eta \left\| \frac{\partial \mathcal{L}}{\partial W} \right\|_F^2 \sim -\eta \left\| X^T \frac{\partial \mathcal{L}}{\partial Y} \right\|_F^2. \tag{7}$$

Here, $\left\| X^T \frac{\partial \mathcal{L}}{\partial Y} \right\|_F^2 \in \mathbb{R}^{d \times d}$ and if we assume that each element in matrix is scale-independent, which can be noted as $\mathcal{O}(1)$, the total term is $\mathcal{O}(d^2)$. This basically told us that when we scale the model size d, the delta loss term will increase a lot by $\mathcal{O}(d^2)$. Simply scaling down learning rate η to $\frac{1}{d^2}$ can be a naive solution. However, the case that each element in matrix in $\mathcal{O}(1)$ doesn't always hold.

Similarly, to analyze the stability of the loss function with respect to the bias term B, we consider the change in the loss function when $B \to B + \Delta B$:

$$\Delta \mathcal{L}_B = \mathcal{L}(B + \Delta B) - \mathcal{L}(B) \approx \left\langle \frac{\partial \mathcal{L}}{\partial B}, \Delta B \right\rangle_F, \tag{8}$$

$$\Delta \mathcal{L}_B \approx -\eta \left\| \frac{\partial \mathcal{L}}{\partial B} \right\|_F^2 \sim -\eta \left\| \mathbf{1}^T \frac{\partial \mathcal{L}}{\partial Y} \right\|_F^2. \tag{9}$$

This will draws similar conclusion as W.

Formulate the Gradient. Start with W, combining Equation 2 and 7, we get

$$\frac{\partial \mathcal{L}}{\partial W} = \underbrace{X^T}_{\partial Z / \partial W} \cdot \underbrace{\left(\frac{\partial \mathcal{L}}{\partial Y} \odot \phi'(Z) \right)}_{\partial \mathcal{L} / \partial Z}, \tag{10}$$

where $Z = WX + B$.

2. On SGD

We initialize weights following Kaiming init [1] as:

$$W = \frac{\Theta}{d}, \quad \Theta \sim \mathcal{N}(0, 2)$$

This ensures forward pass stability:

$$\mathbb{E}[(XW)_{ij}^2] = \mathcal{O}(1)$$

The gradient for Θ becomes:

$$\frac{\partial \mathcal{L}}{\partial \Theta} = \frac{1}{\sqrt{d}} \cdot X^T \left(\frac{\partial \mathcal{L}}{\partial Y} \odot \phi'(Z) \right)$$

Under muP assumptions:

- X and $\frac{\partial \mathcal{L}}{\partial Y}$ have $\mathcal{O}(1)$ entries
- $\|\frac{\partial \mathcal{L}}{\partial \Theta}\|_F^2 \sim \mathcal{O}(d)$

The parameter update for W:

$$\Delta W = -\frac{\eta}{\sqrt{d}} \cdot \frac{\partial \mathcal{L}}{\partial \Theta} \implies \|\Delta W\|_F^2 \sim \eta^2$$

For stable training, η remains **constant** with respect to d. This preserves:

$$\|\Delta W\|_F^2 = \mathcal{O}(1) \quad \forall d$$

Conclusion

- Weight initialization scales as $1/d$
- Gradient magnitudes naturally adapt to model width
- Learning rate η requires no width-dependent scaling

3. On Adam

Weights are parameterized as:

$$W = \frac{\Theta}{\sqrt{d}}, \quad \Theta \sim \mathcal{N}(0,1) \implies \mathrm{Var}(W_{ij}) = \frac{1}{d}$$

This ensures forward/backward signal stability:

$$\mathbb{E}[(XW)_{ij}^2] = \mathcal{O}(1), \quad \mathbb{E}\left[\left\|\frac{\partial \mathcal{L}}{\partial X}\right\|_F^2\right] = \mathcal{O}(1)$$

The gradient for Θ becomes:

$$\frac{\partial \mathcal{L}}{\partial \Theta} = \frac{1}{\sqrt{d}} \cdot \underbrace{X^T \left(\frac{\partial \mathcal{L}}{\partial Y} \odot \phi'(Z)\right)}_{\mathcal{O}(d)} \implies \left\|\frac{\partial \mathcal{L}}{\partial \Theta}\right\|_F^2 \sim \mathcal{O}(d)$$

Adam's first and second moments scale as:

$$m_t \sim \mathcal{O}(\sqrt{d}), \quad v_t \sim \mathcal{O}(d)$$

The normalized update term:

$$\frac{m}{\sqrt{v}+\epsilon} \sim \frac{\sqrt{d}}{\sqrt{d}+\epsilon} = \mathcal{O}(1)$$

The update for Θ:

$$\Delta\Theta = -\eta \cdot \frac{m}{\sqrt{v}+\epsilon} \implies \|\Delta\Theta\|_F^2 \sim \eta^2 d$$

Translating to W:

$$\Delta W = \frac{\Delta\Theta}{\sqrt{d}} \implies \|\Delta W\|_F^2 \sim \frac{\eta^2 d}{d} = \eta^2$$

To maintain $\|\Delta W\|_F^2 = \mathcal{O}(1)$:

$$\eta = \frac{\eta_0}{d}$$

where η_0 is a width-independent base learning rate.

The stable update rule becomes:

$$W \leftarrow W - \frac{\eta_0}{d} \cdot \frac{1}{\sqrt{d}} \cdot \frac{m}{\sqrt{v} + \epsilon}$$

Adam's $\sqrt{v}$ normalization cancels the inherent d-scaling from gradients, but the learning rate still requires explicit $1/d$ scaling to stabilize weight updates.

Conclusion

- **Initialization**: Weight variance scales as $1/d$
- **Learning Rate**: $\eta \propto 1/d$
- **Adam Adaptation**: Moment estimates naturally absorb d-dependence through v_t

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[2] Greg Yang, Edward J Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tensor programs v: Tuning large neural networks via zero-shot hyperparameter transfer. *arXiv preprint arXiv:2203.03466*, 2022.