Beyond MuP: 4. Maintaining Parameter Stability

Su Jianlin (Jianlin Su)

2026-04-24

Through the derivations and computations in the previous few articles, we can see that the three stability metrics proposed in the first article, *Beyond MuP: 1. Three Characteristics of a Good Model*, can generally be divided into two parts: "parameter stability" and "incremental stability." In *Beyond MuP: 2. Linear Layers and Steepest Descent* and *Beyond MuP: 3. Special Treatment for Special Cases*, we demonstrated the process of combining incremental stability with steepest descent to obtain new update rules (optimizers).

However, for parameter stability, we previously stopped at initialization. The task of this article is precisely to explore how to maintain parameter stability throughout the entire training process, completing the practical side of the theory.

1 Problem Background

Taking *Beyond MuP: 2. Linear Layers and Steepest Descent* as an example, the three stability metrics are:

$$\text{Forward stability:} \quad \max_{\|\boldsymbol{x}\|_{RMS}=1} \|\boldsymbol{x}\boldsymbol{W}\|_{RMS} = \sqrt{\frac{d_{in}}{d_{out}}}\|\boldsymbol{W}\|_2 \tag{1}$$

$$\text{Dependency stability:} \quad \max_{\|\boldsymbol{x}_1\|_{RMS}=\|\boldsymbol{x}_2\|_{RMS}=1} \|\boldsymbol{x}_1\boldsymbol{W} - \boldsymbol{x}_2\boldsymbol{W}\|_{RMS} = 2\sqrt{\frac{d_{in}}{d_{out}}}\|\boldsymbol{W}\|_2 \tag{2}$$

$$\text{Update stability:} \quad \max_{\|\boldsymbol{x}\|_{RMS}=1} \|\boldsymbol{x}(\boldsymbol{W}+\Delta\boldsymbol{W}) - \boldsymbol{x}\boldsymbol{W}\|_{RMS} = \sqrt{\frac{d_{in}}{d_{out}}}\|\Delta\boldsymbol{W}\|_2 \tag{3}$$

where $\boldsymbol{W} \in \mathbb{R}^{d_{in}\times d_{out}}$ is the parameter of the linear layer. We would like all three metrics to be $\Theta(1)$, which means we want the parameter and its increment to satisfy $\|\boldsymbol{W}\|_2 = \Theta(\sqrt{d_{out}/d_{in}})$ and $\|\Delta\boldsymbol{W}\|_2 = \Theta(\sqrt{d_{out}/d_{in}})$, respectively. In *Beyond MuP: 3. Special Treatment for Special Cases* we performed computations for Embedding, LM Head, and similar layers; the conclusion is similar, except that the corresponding norms differ.

We take the incremental condition as a stability metric and, based on the "seek speed through stability" steepest-descent principle, derive the theoretically optimal update rule; for linear layers the result is the Muon optimizer:

$$\mathop{\arg\min}_{\|\Delta\boldsymbol{W}\|_2\leq\eta\sqrt{\frac{d_{out}}{d_{in}}}} \mathrm{tr}(\boldsymbol{G}^\top\Delta\boldsymbol{W}) \qquad \Rightarrow \qquad \Delta\boldsymbol{W} = -\eta\sqrt{\frac{d_{out}}{d_{in}}}\,\mathrm{msign}(\boldsymbol{G}) \tag{4}$$

As for the parameter stability part, we previously only required that the parameter's initialization satisfy $\|\boldsymbol{W}\|_2 = \Theta(\sqrt{d_{out}/d_{in}})$; how to guarantee that the model maintains the same parameter stability throughout the entire training process remained unknown.

2 Initial Thoughts

How can we guarantee that $\boldsymbol{W}$ maintains $\|\boldsymbol{W}\|_2 = \Theta(\sqrt{d_{out}/d_{in}})$? More generally, given a parameter $\boldsymbol{\omega}$—which may be a vector, a matrix, or even a higher-order tensor—together with a norm $\|\cdot\|$, usually a metric induced by forward stability or dependency stability, and finally a scale τ, the question is: how can we make $\boldsymbol{\omega}$ maintain $\|\boldsymbol{\omega}\| = \Theta(\tau)$ during training?

A naive idea is to directly enforce $\|\boldsymbol{\omega}\| = \tau$ (τ can also be replaced by a constant multiple of itself, but this does not affect the following discussion). The simplest implementation is to rescale the norm back to τ via normalization after every optimization step (e.g., Hyperball and Nemotron-Flash). Another line of thought is to directly replace the parameters in the original model with normalized ones, i.e., change $\boldsymbol{f}(\boldsymbol{x};\boldsymbol{\omega})$ to $\boldsymbol{f}(\boldsymbol{x};\tau\boldsymbol{\omega}/\|\boldsymbol{\omega}\|)$; in theory this achieves a similar effect.

A more advanced approach combines the steepest-descent idea to adjust the update rule, as discussed in the articles *Steepest Descent on Manifolds: 1. SGD + Hypersphere*, *Steepest Descent on Manifolds: 4. Muon + Spectral Sphere*, and the paper *Controlled LLM Training on Spectral Sphere*. Methodologically this is more elegant, but in practice it is more complicated, usually requiring the solution of a nonlinear equation to obtain the exact update.

However, should we really strictly control some norm of a parameter to a specific value? Intuitively, the norm of a parameter should be determined by the training process itself; at most we set a prior range for it. Although some works show that, when set appropriately, fixing parameter norms to preset values does not hurt performance, this still disrupts the original training dynamics and may require considerably more effort to understand and adapt to it.

Therefore, the viewpoint proposed in this article is that we only need to guarantee $\|\boldsymbol{\omega}\| = \mathcal{O}(\tau)$; specifically, we find a way to guarantee $\|\boldsymbol{\omega}\| \leq \tau$ at every step. As for what the exact value is, and whether $\Theta(\tau)$ can be achieved, we leave to the training algorithm itself to decide, without further intervention.

3 Post Clip

The next question is naturally: how do we implement $\|\boldsymbol{\omega}\| \leq \tau$? More concretely, suppose the original update rule for $\boldsymbol{\omega}$ is

$$\boldsymbol{\omega}_t = \boldsymbol{\omega}_{t-1} - \eta\boldsymbol{\phi}_t \tag{5}$$

Then how should we modify it so that $\boldsymbol{\omega}_t$ always satisfies $\|\boldsymbol{\omega}_t\| \leq \tau$? There are of course many methods; for example, the normalization mentioned in the previous section also counts as one. Given that, we want to choose the scheme with the *least impact* on the optimization process; i.e., given a parameter $\boldsymbol{\omega}$ and a norm $\|\cdot\|$, we wish to make its norm not exceed τ with the minimal modification, formally defined as

$$\lfloor\boldsymbol{\omega}\rfloor_{\|\cdot\|\leq\tau} = \mathop{\arg\min}_{\|\tilde{\boldsymbol{\omega}}\|\leq\tau} \|\boldsymbol{\omega} - \tilde{\boldsymbol{\omega}}\|_{RMS} \tag{6}$$

Readers familiar with convex optimization should easily recognize this as the projection of $\boldsymbol{\omega}$ onto the ball of the given norm with radius not exceeding τ. The key point here is that we want to achieve the goal of the norm not exceeding τ, while minimizing the impact on the original parameter $\boldsymbol{\omega}$; hence we minimize the difference metric $\|\boldsymbol{\omega} - \tilde{\boldsymbol{\omega}}\|_{RMS}$, which induces a specific projection, or clipping, operation.

As for how $\lfloor\boldsymbol{\omega}\rfloor_{\|\cdot\|\leq\tau}$ is computed, it must be analyzed case by case for the specific norm; we will expand on this later. With this operation available, one scheme we can consider is, after each

update, to truncate the parameter's norm via this operation, i.e., change Equation (5) to

$$\boldsymbol{\omega}_t = \lfloor \boldsymbol{\omega}_{t-1} - \eta\boldsymbol{\phi}_t \rfloor_{\|\cdot\|\leq\tau} \tag{7}$$

We provisionally call this scheme "Post Clip." It is simple and intuitive, but may give a feeling of being "non-smooth." This is not hard to understand: suppose the radius of our initialization is smaller than τ, and then at the start of training the parameter radius slowly increases; once it reaches τ, the clipping "suddenly" triggers. The process is continuous but not smooth, similar to the function $\max(x, 0)$.

4 Pre Decay

If this non-smoothness bothers you, you can consider imitating weight decay and spreading the penalty over every update step. Starting again from the update rule (5), assume $\boldsymbol{\phi}_t$ satisfies $\|\boldsymbol{\phi}_t\| \leq \tau$; then by the triangle inequality, $\|\boldsymbol{\omega}_t\| = \|\boldsymbol{\omega}_{t-1} - \eta\boldsymbol{\phi}_t\| \leq \|\boldsymbol{\omega}_{t-1}\| + \eta\tau$. That is, in the extreme case the norm increases by $\eta\tau$ per step, and over long accumulation it will "run away."

To prevent this, before $-\eta\boldsymbol{\phi}_t$ we can preprocess $\boldsymbol{\omega}_{t-1}$ slightly so that its norm decreases, just enough to offset the growth brought by the update. Following the experience of weight decay, we can consider

$$\boldsymbol{\omega}_t = \lfloor \boldsymbol{\omega}_{t-1} \rfloor_{\|\cdot\|\leq(1-\eta)\|\boldsymbol{\omega}_{t-1}\|} - \eta\boldsymbol{\phi}_t \tag{8}$$

That is, first reduce the norm of $\boldsymbol{\omega}_{t-1}$ to $1-\eta$ times its original value, and then perform the update. With this,

$$\|\boldsymbol{\omega}_t\| \leq (1-\eta)\|\boldsymbol{\omega}_{t-1}\| + \eta\tau \leq \max(\|\boldsymbol{\omega}_{t-1}\|, \tau) \tag{9}$$

Carrying this forward, $\|\boldsymbol{\omega}_t\| \leq \max(\|\boldsymbol{\omega}_{t-1}\|, \tau) \leq \cdots \leq \max(\|\boldsymbol{\omega}_0\|, \tau)$; i.e., as long as the initialization satisfies $\|\boldsymbol{\omega}_0\| \leq \tau$, the whole update chain automatically satisfies $\|\boldsymbol{\omega}_t\| \leq \tau$. This conclusion is independent of which particular norm is used; it relies only on the triangle inequality of norms. And the minimal-modification operation for reducing the norm is exactly the clipping operator defined in Equation (6), so using it to reduce the norm is natural.

We call this scheme "Pre Decay." The difference from "Post Clip" is that the latter's threshold is the static τ, so the clipping is not necessarily triggered, whereas the former's threshold is the dynamic $(1-\eta)\|\boldsymbol{\omega}_{t-1}\|$, and the clipping is always triggered. This process is smoother, so we call it decay rather than clipping; it is a general generalization of weight decay.

5 A Simple Example

So far, we have established a general framework for constraining parameter norms, divided into the two schemes of "Post Clip" and "Pre Decay." The core operation is the clipping operator $\lfloor \boldsymbol{\omega} \rfloor_{\|\cdot\|\leq\tau}$ defined in Equation (6), for which we currently have only a formal definition; in practice it must be computed according to the specific norm.

In this section we first compute a simple example, where the norm chosen is $\|\cdot\|_{RMS}$: for vectors it is equivalent to the L2 norm, and for matrices it is equivalent to the Frobenius norm. It is not hard to obtain

$$\lfloor \boldsymbol{\omega} \rfloor_{\|\cdot\|_{RMS}\leq\tau} = \mathop{\arg\min}_{\|\tilde{\boldsymbol{\omega}}\|_{RMS}\leq\tau} \|\boldsymbol{\omega} - \tilde{\boldsymbol{\omega}}\|_{RMS} = \min\left(1, \frac{\tau}{\|\boldsymbol{\omega}\|_{RMS}}\right)\boldsymbol{\omega} \tag{10}$$

The proof is left to the reader (if you really cannot figure it out, you can ask Kimi). In particular, substituting $\tau = (1-\eta)\|\omega\|_{RMS}$ gives

$$\lfloor \boldsymbol{\omega} \rfloor_{\|\cdot\|_{RMS}\leq(1-\eta)\|\omega\|_{RMS}} = \min\left(1, \frac{(1-\eta)\|\omega\|_{RMS}}{\|\boldsymbol{\omega}\|_{RMS}}\right)\boldsymbol{\omega} = (1-\eta)\boldsymbol{\omega} \tag{11}$$

and then substituting into Equation (8) gives

$$\boldsymbol{\omega}_t = (1-\eta)\boldsymbol{\omega}_{t-1} - \eta\boldsymbol{\phi}_t \tag{12}$$

It is easy to see that this is precisely the usual Weight Decay. In other words, Pre Decay under the RMS norm is the weight decay we commonly use; it is the Pre Decay scheme with minimal modification to the original parameters, while maintaining the RMS norm constraint (equivalently, the L2 norm of a vector or the Frobenius norm of a matrix).

6 Singular Value Clipping

Now we come to the "main feature" of this article: matrix parameters and Muon. Here we switch back to the notation $\boldsymbol{W}$ and write Muon's original update rule as

$$\boldsymbol{W}_t = \boldsymbol{W}_{t-1} - \eta\lambda\boldsymbol{\Phi}_t, \quad \boldsymbol{\Phi}_t = \frac{1}{\lambda}\sqrt{\frac{d_{out}}{d_{in}}}\,\mathrm{msign}(\boldsymbol{G}_t) \tag{13}$$

Let $\tau = \frac{1}{\lambda}\sqrt{\frac{d_{out}}{d_{in}}}$; then $\|\boldsymbol{\Phi}_t\|_2 = \tau$. The two schemes for making $\boldsymbol{W}_t$ satisfy $\|\boldsymbol{W}_t\|_2 \leq \tau$ are:

$$\text{Post Clip:} \quad \boldsymbol{W}_t = \lfloor \boldsymbol{W}_{t-1} - \eta\lambda\boldsymbol{\Phi}_t \rfloor_{\|\cdot\|_2\leq\tau} \tag{14}$$

$$\text{Pre Decay:} \quad \boldsymbol{W}_t = \lfloor \boldsymbol{W}_{t-1} \rfloor_{\|\cdot\|_2\leq(1-\eta\lambda)\|\boldsymbol{W}_{t-1}\|_2} - \eta\lambda\boldsymbol{\Phi}_t \tag{15}$$

The next task is to compute $\lfloor \boldsymbol{W} \rfloor_{\|\cdot\|_2\leq\tau}$. By the equivalence between RMS and the Frobenius norm, it also equals

$$\lfloor \boldsymbol{W} \rfloor_{\|\cdot\|_2\leq\tau} = \mathop{\arg\min}_{\|\tilde{\boldsymbol{W}}\|_2\leq\tau} \|\boldsymbol{W} - \tilde{\boldsymbol{W}}\|_F \tag{16}$$

The optimal solution of this problem should be familiar to some readers: it is the "Singular Value Clipping (SVC)" we mentioned in *Higher-Order MuP: Simpler yet Wiser Spectral Conditioned Scaling*, called mclip in *Computing Singular Value Clipping mclip via msign (Part 1)* and *Computing Singular Value Clipping mclip via msign (Part 2)*:

$$\lfloor \boldsymbol{W} \rfloor_{\|\cdot\|_2\leq\tau} = \mathrm{mclip}(\boldsymbol{W};\tau) = \boldsymbol{U}\min(\boldsymbol{\Sigma},\tau)\boldsymbol{V}^\top \tag{17}$$

where $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ is the SVD of $\boldsymbol{W}$, and $\min(\boldsymbol{\Sigma},\tau)$ clips the singular values so that they do not exceed τ; we demonstrate the proof in the next section. With this notation, the two schemes can be written respectively as

$$\text{Post Clip:} \quad \boldsymbol{W}_t = \mathrm{mclip}(\boldsymbol{W}_{t-1} - \eta\lambda\boldsymbol{\Phi}_t;\tau) \tag{18}$$

$$\text{Pre Decay:} \quad \boldsymbol{W}_t = \mathrm{mclip}(\boldsymbol{W}_{t-1};(1-\eta\lambda)\|\boldsymbol{W}_{t-1}\|_2) - \eta\lambda\boldsymbol{\Phi}_t \tag{19}$$

7 Derivation

In this section we prove the conclusion (17). Let the SVD of $\boldsymbol{W}$ be $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$, with $\boldsymbol{U} \in \mathbb{R}^{d_{in}\times d_{in}}$, $\boldsymbol{\Sigma} \in \mathbb{R}^{d_{in}\times d_{out}}$, $\boldsymbol{V} \in \mathbb{R}^{d_{out}\times d_{out}}$; then

$$\|\boldsymbol{W} - \tilde{\boldsymbol{W}}\|_F = \|\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top - \tilde{\boldsymbol{W}}\|_F = \|\boldsymbol{U}(\boldsymbol{\Sigma} - \boldsymbol{U}^\top\tilde{\boldsymbol{W}}\boldsymbol{V})\boldsymbol{V}^\top\|_F = \|\boldsymbol{\Sigma} - \boldsymbol{U}^\top\tilde{\boldsymbol{W}}\boldsymbol{V}\|_F \tag{20}$$

The last equality holds because orthogonal matrices do not change the Frobenius norm. Meanwhile, orthogonal matrices do not change the spectral norm either, so letting $\tilde{\boldsymbol{\Sigma}} = \boldsymbol{U}^\top\tilde{\boldsymbol{W}}\boldsymbol{V}$, the objective (16) can be equivalently simplified to

$$\operatorname*{arg\,min}_{\|\tilde{\boldsymbol{\Sigma}}\|_2\le\tau} \|\boldsymbol{\Sigma} - \tilde{\boldsymbol{\Sigma}}\|_F \tag{21}$$

Note that here $\boldsymbol{\Sigma}$ is a diagonal matrix with diagonal elements $\sigma_1, \sigma_2, \cdots \ge 0$, but $\tilde{\boldsymbol{\Sigma}}$ is for now undetermined; for the proof we must take it as a general matrix. Writing in component form:

$$\|\boldsymbol{\Sigma} - \tilde{\boldsymbol{\Sigma}}\|_F^2 = \sum_i \sigma_i^2 + \sum_{i,j}\tilde{\Sigma}_{i,j}^2 - 2\sum_i \sigma_i\tilde{\Sigma}_{i,i} \ge \sum_i \sigma_i^2 + \sum_i(\tilde{\Sigma}_{i,i}^2 - 2\sigma_i\tilde{\Sigma}_{i,i}) \tag{22}$$

Term by term, $\tilde{\Sigma}_{i,i}^2 - 2\sigma_i\tilde{\Sigma}_{i,i}$ is merely a univariate quadratic function in $\tilde{\Sigma}_{i,i}$, whose minimum is attained at $\tilde{\Sigma}_{i,i} = \sigma_i$. But we also have the constraint $\|\tilde{\boldsymbol{\Sigma}}\|_2 \le \tau$; since the spectral norm is at least the absolute value of any entry of the matrix, we have at least the constraint $\tilde{\Sigma}_{i,i} \le \tau$. Under this constraint, the minimum of $\tilde{\Sigma}_{i,i}^2 - 2\sigma_i\tilde{\Sigma}_{i,i}$ is attained at $\tilde{\Sigma}_{i,i}^* = \min(\sigma_i, \tau)$.

Considering that all equalities should hold simultaneously, we obtain $\tilde{\Sigma}_{i,j}^* = 0$ for $i \ne j$. Hence $\tilde{\boldsymbol{\Sigma}}^*$ is also a diagonal matrix, which can be written concisely as $\tilde{\boldsymbol{\Sigma}}^* = \min(\boldsymbol{\Sigma}, \tau)$, which in turn corresponds to $\tilde{\boldsymbol{W}}^* = \boldsymbol{U}\min(\boldsymbol{\Sigma}, \tau)\boldsymbol{V}^\top$. This completes the proof of conclusion (17).

8 Clipping the Top Component

So how can mclip be computed efficiently? Performing an SVD at every training step is obviously too expensive. In *Computing Singular Value Clipping mclip via msign (Part 1)* and *Computing Singular Value Clipping mclip via msign (Part 2)*, we actually explored this problem systematically; the formulation at the time used msign, but it required 2–3 applications of msign, at considerable cost. For example, an identity found in the second part is

$$\operatorname{mclip}(\boldsymbol{W}; \tau) = \frac{1}{2}\Big\{\boldsymbol{W} + \tau\operatorname{msign}(\boldsymbol{W}) - (\tau\boldsymbol{I} - \boldsymbol{W}\operatorname{msign}(\boldsymbol{W})^\top)\operatorname{msign}(\tau\operatorname{msign}(\boldsymbol{W}) - \boldsymbol{W})\Big\} \tag{23}$$

This requires two applications of msign. Since parameters are usually computed in FP32, executing msign twice is still rather expensive, so this is not particularly practical yet.

Here we mainly consider the entrywise clipping approach discussed in *Streaming Power Iteration Based Muon Implementations: 5. Extensions*. Specifically, mclip turns all singular values greater than τ into τ; the essential operation is to turn the top singular value into τ (if it is greater than τ). After clipping the top singular value, if there are still singular values greater than τ, the largest among them becomes the new top singular value. So, by repeatedly "clipping the top singular value to τ," one can implement mclip.

Since the top singular value and top singular vectors can be found efficiently via power iteration (denoted SVD1), clipping the top singular value can be considered efficient. Furthermore, we assume the training is sufficiently gentle, so that performing only one top-singular-value clipping

per step suffices, which approximately achieves the same effect. Based on this strategy, the two schemes for constraining the singular values can be further written as

$$\text{Post Clip:}\quad \boldsymbol{W}_t = \tilde{\boldsymbol{W}}_t - \max(\sigma_1 - \tau, 0)\boldsymbol{u}_1\boldsymbol{v}_1^\top,\quad \sigma_1, \boldsymbol{u}_1, \boldsymbol{v}_1 = \text{SVD1}(\tilde{\boldsymbol{W}}_t),\quad \tilde{\boldsymbol{W}}_t = \boldsymbol{W}_{t-1} - \eta\boldsymbol{\Phi}_t \tag{24}$$

$$\text{Pre Decay:}\quad \boldsymbol{W}_t = \boldsymbol{W}_{t-1} - \lambda\eta\sigma_1\boldsymbol{u}_1\boldsymbol{v}_1^\top - \eta\boldsymbol{\Phi}_t,\quad \sigma_1, \boldsymbol{u}_1, \boldsymbol{v}_1 = \text{SVD1}(\boldsymbol{W}_{t-1}) \tag{25}$$

The "Pre Decay" version is exactly the spectral weight decay introduced in *Thoughts on Spectral-Norm Gradients and a New Style of Weight Decay*; more than a year later, we have arrived at the same result again via another route. In practice, since only one singular value is clipped per step, there may exist some rather "ambitious" matrices whose spectral norm noticeably deviates from the set threshold; this is normal, and during the LR decay phase it will gradually come down near the set threshold.

9 Other Details

If we want more precise clipping, we can also use power iteration to simultaneously find the Top-k singular values and singular vectors, clipping at most k singular values per step; the cost is that the L2 normalization of the power iteration must be replaced by QR decomposition, and there are also some acceleration tricks for QR decomposition. The relevant principles can be found in the streaming power iteration series, e.g., *Streaming Power Iteration Based Muon Implementations: 1. First Encounter*.

Besides the spectral norm of linear-layer matrices, in *Beyond MuP: 3. Special Treatment for Special Cases* we encountered other norms for other layers: for Embedding and LM Head these correspond to the maximum row and column RMS respectively, while the gamma parameter of an RMS Norm layer corresponds to the maximum absolute value, also known as the infinity norm of a vector.

Fortunately, the clipping operators $\lfloor\boldsymbol{\omega}\rfloor_{\|\cdot\|\leq\tau}$ under these norms are all easy to compute. For example, the norm of the Embedding layer is the maximum row RMS, and the clipping operator clips the RMS of every row vector to not exceed τ; the LM Head is analogous, except that rows are replaced by columns. As for the gamma parameter, it is even simpler: it is just componentwise clipping, $\text{clip}(\boldsymbol{\gamma}; -\tau, \tau) = \max(\min(\boldsymbol{\gamma}, \tau), -\tau)$.

These conclusions are all quite intuitive, and their proofs are fairly simple, so we will not expand on them; consider them exercises for the reader.

10 The Necessary Guarantee

Some readers may wonder: does it have to be this complicated? Wouldn't ordinary weight decay suffice, as in *Training Deep Learning Models with Norm-Constrained LMOs*? For example,

$$\boldsymbol{W}_t = (1 - \eta\lambda)\boldsymbol{W}_{t-1} - \eta\sqrt{\frac{d_{out}}{d_{in}}}\,\text{msign}(\boldsymbol{G}_t) \tag{26}$$

This can likewise keep the spectral norm within $\tau = \frac{1}{\lambda}\sqrt{\frac{d_{out}}{d_{in}}}$; why not use this simple form?

The answer is: to avoid excessive intervention. From the definition (6) we can see that the clipping operator we defined is the operation that minimally modifies the original parameters while achieving the same effect. For the spectral norm, directly multiplying by $1 - \eta\lambda$ can indeed make

the spectral norm of $\boldsymbol{W}_{t-1}$ not exceed $(1-\eta\lambda)\|\boldsymbol{W}_{t-1}\|_2$, but since it differs from the minimal-modification mclip, it necessarily involves some degree of "excessive intervention."

Excessive intervention has two possible consequences: either, to guarantee performance, a small λ is chosen, in which case τ is too large—i.e., we cannot guarantee the spectral norm stays within the range we expect; or, to guarantee control of the spectral norm, a large λ is chosen, which then clearly degrades performance. For example, in the case $d_{in} = d_{out}$, if we want the spectral norm not to exceed 5, then according to the formula $\lambda = 0.2$; for the Muon of Equation (26), a weight decay coefficient of 0.2 is huge (a typical value is around 0.01).

Note that we have repeatedly emphasized "guarantee"; this is crucial. Suppose we use weight decay with coefficient 0.01; theoretically the spectral norm can reach at most 100, but experiments on small models may find it does not even reach 5—this is very common. However, small models being safe does not mean large models are safe; as we said before, large models are powerful—so powerful that they can amplify any subtle bug. If the theoretical upper bound is 100, small models may never reach it, but large models really might.

Therefore, keeping a reasonable theoretical bound on the key norms of parameters is very necessary; this is the embodiment of "stability" in the "seek speed through stability" principle. And the clipping operator defined in Equation (6) is the "lightest-weight" operation that guarantees boundedness; in other words, it is likely the operation with minimal performance loss while guaranteeing the same bound.

11 Summary

Based on the minimal-modification idea, this article proposed a general framework for maintaining parameter stability during training, comprising two schemes: Post Clip and Pre Decay. Under the spectral norm, these further evolve into singular value clipping and spectral weight decay. These operations aim to keep the key norms of parameters bounded while minimizing the intervention on the training dynamics.