A New Understanding of Momentum: Approximating Gradient Descent at the Feature Level

Su Jianlin

2026-08-23

An optimizer that uses momentum as its state variable has the following basic form:

$$\begin{aligned}\boldsymbol{M}_t &= \beta \boldsymbol{M}_{t-1} + (1-\beta)\boldsymbol{G}_t \\ \boldsymbol{W}_t &= \phi(\boldsymbol{W}_{t-1}, \boldsymbol{M}_t, \boldsymbol{G}_t, t)\end{aligned} \tag{1}$$

The differences between various optimizers mainly lie in the update function ϕ, such as SGDM, SignSGD, Muon, etc. New explorations also basically revolve around ϕ, because the momentum in the first equation is too simple—so simple that people do not think there is any room for modification.

However, the protagonist of this article is precisely momentum. We will bring a new perspective of understanding to the momentum mechanism: momentum can not only be viewed as an average of gradients, but it can also be regarded as the solution to an online regression problem. From this, we can naturally derive some recent works.

1 Basic Concepts

Like Muon, this article mainly considers the matrix parameters of linear layers. Suppose there is a linear layer $\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{W}$, where $\boldsymbol{X} \in \mathbb{R}^{b\times d_{in}}$ is the input, $\boldsymbol{W} \in \mathbb{R}^{d_{in}\times d_{out}}$ is the weight, and $\boldsymbol{Y} \in \mathbb{R}^{b\times d_{out}}$ is the output. Denoting the loss function as $\mathcal{L}(\boldsymbol{Y}) = \mathcal{L}(\boldsymbol{X}\boldsymbol{W})$, we have

$$\boldsymbol{G} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{W}} = \boldsymbol{X}^\top \frac{\partial \mathcal{L}}{\partial \boldsymbol{Y}} \tag{2}$$

The simplest optimizer is (parameter-level) gradient descent

$$\boldsymbol{W} \quad \leftarrow \quad \boldsymbol{W} - \eta \frac{\partial \mathcal{L}}{\partial \boldsymbol{W}} = \boldsymbol{W} - \eta \boldsymbol{X}^\top \frac{\partial \mathcal{L}}{\partial \boldsymbol{Y}} \tag{3}$$

However, according to the idea in *Why Do We Prefer Isotropy? An Understanding Based on Steepest Descent*, we believe that parameters are essentially by-products of the model, and

changes at the model feature level are what most correlate with the model's performance. Ideally, we hope to achieve gradient descent at the feature level

$$\boldsymbol{Y} \quad \leftarrow \quad \boldsymbol{Y} - \eta \frac{\partial \mathcal{L}}{\partial \boldsymbol{Y}} \tag{4}$$

The problem is that $\boldsymbol{Y}$ is not a variable that can be arbitrarily modified; we can only directly modify $\boldsymbol{W}$, so we have to find a way to indirectly achieve this effect by modifying $\boldsymbol{W}$.

2 Regression Target

How to achieve this indirectly? Let the final update rule be $\boldsymbol{W} \leftarrow \boldsymbol{W} - \eta \boldsymbol{\Phi}$, then $\boldsymbol{Y} \leftarrow \boldsymbol{Y} - \eta \boldsymbol{X}\boldsymbol{\Phi}$. We hope it can approximate the effect of Equation (4) as closely as possible, i.e., we hope $\boldsymbol{X}\boldsymbol{\Phi} \approx \frac{\partial \mathcal{L}}{\partial \boldsymbol{Y}}$, so we consider minimizing

$$\min_{\boldsymbol{\Phi}} \frac{1}{2} \left\| \boldsymbol{X}\boldsymbol{\Phi} - \frac{\partial \mathcal{L}}{\partial \boldsymbol{Y}} \right\|_F^2 + \frac{\lambda}{2} \|\boldsymbol{\Phi}\|_F^2 \tag{5}$$

where $\lambda > 0$ is the regularization coefficient. In fact, this is just a linear regression problem, which can be solved directly as

$$\boldsymbol{\Phi}^* = (\boldsymbol{X}^\top \boldsymbol{X} + \lambda \boldsymbol{I})^{-1} \boldsymbol{X}^\top \frac{\partial \mathcal{L}}{\partial \boldsymbol{Y}} \tag{6}$$

Noticing that $\boldsymbol{X}^\top \frac{\partial \mathcal{L}}{\partial \boldsymbol{Y}}$ is exactly $\boldsymbol{G} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{W}}$, and $(\boldsymbol{X}^\top \boldsymbol{X} + \lambda \boldsymbol{I})^{-1}$ is a preconditioner based on the input data. To synthesize the contributions of different batches, we consider applying EMA to both $\boldsymbol{X}^\top \boldsymbol{X}$ and $\boldsymbol{X}^\top \frac{\partial \mathcal{L}}{\partial \boldsymbol{Y}}$, which will yield an SGDM variant:

$$\begin{aligned} \boldsymbol{M}_t &= \beta \boldsymbol{M}_{t-1} + (1-\beta)\boldsymbol{G}_t \\ \boldsymbol{Z}_t &= \beta \boldsymbol{Z}_{t-1} + (1-\beta)(\boldsymbol{X}_t^\top \boldsymbol{X}_t + \lambda \boldsymbol{I}) \\ \boldsymbol{W}_t &= \boldsymbol{W}_{t-1} - \eta \boldsymbol{Z}_t^{-1} \boldsymbol{M}_t \end{aligned} \tag{7}$$

3 Switching Perspective

Now we know that through the corrected gradient $(\boldsymbol{X}^\top \boldsymbol{X} + \lambda \boldsymbol{I})^{-1} \boldsymbol{G}$, we can achieve gradient descent at the feature level. Then we can view it as some sort of "more reliable gradient". Similarly, the corresponding $\boldsymbol{Z}_t^{-1} \boldsymbol{M}_t$ can be viewed as a "more reliable momentum".

Under this perspective, we can try to replace the momentum in various momentum-based optimizers with $\boldsymbol{Z}_t^{-1} \boldsymbol{M}_t$, such as Muon:

$$\begin{aligned} \boldsymbol{M}_t &= \beta \boldsymbol{M}_{t-1} + (1-\beta)\boldsymbol{G}_t \\ \boldsymbol{Z}_t &= \beta \boldsymbol{Z}_{t-1} + (1-\beta)(\boldsymbol{X}_t^\top \boldsymbol{X}_t + \lambda \boldsymbol{I}) \\ \boldsymbol{W}_t &= \boldsymbol{W}_{t-1} - \eta \,\mathrm{msign}(\boldsymbol{Z}_t^{-1} \boldsymbol{M}_t) \end{aligned} \tag{8}$$

This is roughly the Newton-Muon optimizer. The reason for saying "roughly" is that the original Newton-Muon actually uses a scheme of correcting first and then applying EMA (i.e., applying EMA to $(\boldsymbol{X}^\top\boldsymbol{X}+\lambda\boldsymbol{I})^{-1}\boldsymbol{G}$ as the momentum), which in principle can save one set of state variables. As for matrix inversion, one can refer to *Efficient Computation of the r-th Root and Inverse r-th Root of a Matrix*.

If the input is isotropic, then we can expect $\boldsymbol{Z}_t = \sigma^2\boldsymbol{I}$. At this point, Newton-Muon degenerates into Muon. In other words, Muon can be understood as feature-level gradient descent under the isotropy assumption, which returns again to the conclusion of the article *Why Do We Prefer Isotropy? An Understanding Based on Steepest Descent*. In addition, when $\lambda\to\infty$, Newton-Muon also degenerates into Muon, so we can adjust λ to control its degree of approximation to Muon.

4 Incremental Update

A more universal and playable approach is: instead of trying to find an analytical solution, directly use gradient descent to optimize the objective (5)! First, the gradient of Equation (5) with respect to $\boldsymbol{\Phi}$ can be found as:

$$(\boldsymbol{X}^\top\boldsymbol{X}+\lambda\boldsymbol{I})\boldsymbol{\Phi}-\boldsymbol{X}^\top\frac{\partial\mathcal{L}}{\partial\boldsymbol{Y}} = (\boldsymbol{X}^\top\boldsymbol{X}+\lambda\boldsymbol{I})\boldsymbol{\Phi}-\boldsymbol{G} \tag{9}$$

If we use gradient descent to update $\boldsymbol{\Phi}$, we get

$$\begin{aligned}\boldsymbol{\Phi}_t &= \boldsymbol{\Phi}_{t-1} - \gamma[(\boldsymbol{X}_t^\top\boldsymbol{X}_t+\lambda\boldsymbol{I})\boldsymbol{\Phi}_{t-1}-\boldsymbol{G}_t]\\ &= (1-\gamma\lambda)\boldsymbol{\Phi}_{t-1} + \gamma[\boldsymbol{G}_t - (\boldsymbol{X}_t^\top\boldsymbol{X}_t)\boldsymbol{\Phi}_{t-1}]\end{aligned} \tag{10}$$

where $\gamma>0$ is the learning rate of this "inner" optimization. Continuing the idea from the previous section, if we say $\boldsymbol{Z}_t^{-1}\boldsymbol{M}_t$ is some kind of more reliable momentum, then the above formula is a "more reliable momentum update rule" derived based on the idea of gradient descent! This is DeltaMomentum, which introduces the Delta Rule into momentum. Compared to the old momentum, the difference is like that between GDN and Vanilla Linear Attention, while $\boldsymbol{Z}_t^{-1}\boldsymbol{M}_t$ corresponds to MesaNet.

Of course, to make DeltaMomentum work, some details need to be noted, mainly regarding the normalization of $\boldsymbol{X}$ and the choice of γ. Please refer to the original paper yourself. The author is not particularly convinced by some of the processing in the original paper, and it is recommended that readers deliberate and use it as a reference when reading. But regardless, DeltaMomentum provides a new direction of exploration for the momentum mechanism, and it does not require expensive operations like inversion, making it worth savoring.

5 Related Work

In fact, recently, there have been more and more explorations on constructing preconditioners based on inputs. In addition to Newton-Muon and DeltaMomentum, student Xie Tian previously published another slightly different result in *Steepest Descent in Feature Space*. Using our notation here, it roughly is

$$\begin{aligned}\boldsymbol{M}_t &= \beta \boldsymbol{M}_{t-1} + (1-\beta)\boldsymbol{G}_t \\ \boldsymbol{Z}_t &= \beta \boldsymbol{Z}_{t-1} + (1-\beta)(\boldsymbol{X}_t^\top \boldsymbol{X}_t + \lambda \boldsymbol{I}) \\ \boldsymbol{W}_t &= \boldsymbol{W}_{t-1} - \eta \boldsymbol{Z}_t^{-1/2}\,\mathrm{msign}(\boldsymbol{Z}_t^{-1/2}\boldsymbol{M}_t)\end{aligned} \tag{11}$$

That is, $\boldsymbol{Z}_t$ acts both inside and outside of msign in the form of the $-1/2$ power. Its derivation principle is exactly the steepest descent under the spectral norm constraint at the feature level, which is consistent with the central idea of this article.

There might be similar works, but the author really cannot think of any others. Readers are welcome to supplement in the comment section. From the experimental results, some attempts have indeed achieved positive results. For example, Newton-Muon performs better than Muon on Speedrun (refer here). Therefore, overall, this direction still has some merits.

6 Extended Thoughts

However, this design of input-oriented preconditioners also has some dissatisfying aspects.

The first is at the implementation level: it requires us to record the autocorrelation matrix $\boldsymbol{X}_t^\top \boldsymbol{X}_t$ during the forward calculation, and then pass it into the optimizer. This breaks the independence of the optimizer—the optimizer is no longer a "black box" that can work just by getting the gradients; it needs to be coupled with details at the model definition level. It is hard to call this elegant in implementation, not to mention the additional communication and computation overhead introduced.

The second is at the theoretical level: constructing a preconditioner based on actual inputs may limit the optimizer's exploration to the subspace spanned by the inputs, leading to insufficient exploration in other directions and thus poor performance. A relatively simple mitigation strategy is to increase λ to reduce the preconditioning strength—since $\lambda \to \infty$ corresponds to no preconditioning, choosing an appropriate λ always provides a chance to tune the performance, but this also adds a hyperparameter that needs to be carefully tuned.

Overall, this direction is still in its nascent stage—the prototype has appeared, but the engineering and theoretical problems have not been completely solved, and the problems are no fewer than the conclusions.

7 Article Summary

Starting from the idea of making "parameter gradient descent" approximate "feature gradient descent", this article reinterprets momentum as the solution to an online regression problem, thereby leading to some related works. Interestingly, this route is highly consistent with the evolution of linear attention from Vanilla to DeltaNet and then to MesaNet, and it seems that the two have much to learn from each other.