

Lecture 7: Optimization II — Muon

Prof. Noah Golowich

Today we will use the matrix structure of neural network parameters to motivate Muon and derive its Newton–Schulz update.

1 From coordinatewise signs to matrix orthogonalization

In deep learning, most parameters are entries of matrices: for example, the query, key, value, and output projections in attention, and the weight matrices in MLPs. Although we can concatenate their entries into a vector $\theta \in \mathbb{R}^d$, each matrix acts as a linear map between spaces of activations. We would like an optimizer to account for this structure.

Adam without averaging is sign gradient descent. Recall the batch gradient $g_t = \nabla f_t(\theta_t)$ and step size $\alpha_t > 0$ from Lecture 6. Suppose that we use Adam but set $\beta_1 = \beta_2 = 0$ (i.e., omit momentum), and omit the numerical stabilizer. Then $m_t = g_t$ and $v_t = g_t^2$, where the square is coordinatewise, so

$$\theta_{t+1,i} = \theta_{t,i} - \alpha_t \frac{g_{t,i}}{\sqrt{g_{t,i}^2}} = \theta_{t,i} - \alpha_t \text{sign}(g_{t,i}). \quad (1)$$

At $g_{t,i} = 0$ we define the update to be zero. Thus Adam with no averaging reduces to *sign gradient descent* [2].

For a weight matrix $W \in \mathbb{R}^{m \times n}$, this rule takes the sign of each entry of its gradient matrix. But entrywise signs do not describe a matrix’s action on vectors: that action is governed by its singular vectors and singular values. For example,

$$G = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} \quad \implies \quad \text{sign}(G) = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}.$$

The original matrix acts nontrivially on both $(1, 1)^\top$ and $(1, -1)^\top$, with eigenvalues 3 and 1, whereas the sign matrix annihilates the second direction. Entrywise normalization can therefore discard an entire direction of a full-rank gradient.

The natural analogue for matrices is to retain the singular vectors and normalize the singular values. This operation is called *orthogonalization*; *Newton–Schulz iteration* is an efficient way to approximate it. We first define the target operation precisely.

Definition 1.1 (Orthogonalization). Let $G \in \mathbb{R}^{m \times n}$ have rank $r := \min\{m, n\}$. A matrix $O \in \mathbb{R}^{m \times n}$ is *semi-orthogonal* if

$$O^\top O = I_n \quad \text{when } m \geq n, \quad \text{or} \quad OO^\top = I_m \quad \text{when } m \leq n.$$

Here I_k denotes the $k \times k$ identity matrix. Define the orthogonalization of G to be its closest semi-orthogonal matrix in Frobenius norm:

$$\text{Ortho}(G) := \arg \min_{O \text{ semi-orthogonal}} \|O - G\|_F, \quad \|A\|_F := \left(\sum_{i,j} A_{ij}^2 \right)^{1/2}. \quad (2)$$

Equivalently, write a thin singular value decomposition (SVD)

$$G = U\Sigma V^\top, \quad U \in \mathbb{R}^{m \times r}, \quad V \in \mathbb{R}^{n \times r}, \quad U^\top U = V^\top V = I_r, \quad \Sigma = \text{diag}(\sigma_1, \dots, \sigma_r),$$

where each $\sigma_i > 0$. Then

$$\text{Ortho}(G) = UV^\top. \quad (3)$$

For a square matrix, semi-orthogonality is ordinary orthogonality.

Equivalence of the two definitions. Every feasible O has $\|O\|_F^2 = r$, so expanding the square gives

$$\|O - G\|_F^2 = r + \|G\|_F^2 - 2\text{tr}(O^\top G).$$

Write u_i, v_i for the columns of U, V . Since a semi-orthogonal matrix satisfies $\|Ox\|_2 \leq \|x\|_2$ for every x , Cauchy–Schwarz gives

$$\text{tr}(O^\top G) = \sum_{i=1}^r \sigma_i u_i^\top O v_i \leq \sum_{i=1}^r \sigma_i.$$

The matrix $UV^\top$ is feasible and attains equality because $UV^\top v_i = u_i$. This proves that it minimizes (2). For uniqueness, equality requires $u_i^\top O v_i = 1$ for every i , since $\sigma_i > 0$. If $m \geq n$, this forces $O v_i = u_i$, and the v_i form a basis of $\mathbb{R}^n$, determining $O = UV^\top$. If $m \leq n$, apply the same argument to $O^\top$ and the basis $u_1, \dots, u_m$. $\square$

Thus $Gv_i = \sigma_i u_i$ is replaced by $\text{Ortho}(G)v_i = u_i$: the update treats all singular directions equally. In the 2×2 example above, $\text{Ortho}(G) = I_2$, so both directions are retained. Moreover, for orthogonal changes of coordinates $P \in \mathbb{R}^{m \times m}$ and $Q \in \mathbb{R}^{n \times n}$,

$$\text{Ortho}(PGQ^\top) = P \text{Ortho}(G) Q^\top,$$

as follows by using the SVD $(PU)\Sigma(QV)^\top$. Entrywise signs do not generally have this property. This motivates orthogonalization when we want an update adapted to the matrix's Euclidean input and output geometry. We will discuss this geometric motivation in more detail next lecture.

2 Muon

Muon combines momentum with approximate orthogonalization of each matrix update [1]. Its name abbreviates *MomentUm Orthogonalized by Newton–Schulz*.

2.1 The optimizer

Fix one hidden-layer weight matrix $W_t \in \mathbb{R}^{m \times n}$ in the parameter vector θ_t , and let $G_t := \nabla_W f_t(\theta_t)$ denote its batch gradient. Choose a momentum coefficient $\mu \in [0, 1)$ and learning rates $\alpha_t > 0$, and initialize the momentum buffer $B_0 = 0$. The Muon update is

$$\begin{aligned} B_t &= \mu B_{t-1} + G_t, \\ O_t &= \text{NewtonSchulz5}(B_t), \\ W_{t+1} &= W_t - \alpha_t O_t. \end{aligned} \quad (4)$$

The subroutine `NewtonSchulz5`, defined in section 2.2, uses five matrix iterations to approximate orthogonalization. With exact orthogonalization in its place, the update would be $W_{t+1} = W_t - \alpha_t \text{Ortho}(B_t)$. Momentum is accumulated *before* orthogonalizing: B_t retains the magnitudes of the past gradients.

2.2 Computing the update with NewtonSchulz5

An SVD gives an exact formula for orthogonalization, but computing one at every training step is expensive. Newton–Schulz uses matrix multiplication to transform the singular values without explicitly finding the singular vectors.

A matrix iteration becomes a scalar iteration. For coefficients $a, b, c \in \mathbb{R}$, consider the polynomial update

$$X_{k+1} = aX_k + b(X_k X_k^\top)X_k + c(X_k X_k^\top)^2 X_k. \quad (5)$$

Here k counts inner iterations within a single optimizer step. To understand this operation, let $X_k = US_k V^\top$, where U, V have orthonormal columns and S_k is diagonal. Since $V^\top V = U^\top U = I_r$,

$$\begin{aligned} X_k X_k^\top &= US_k^2 U^\top, \\ (X_k X_k^\top)X_k &= US_k^3 V^\top, \\ (X_k X_k^\top)^2 X_k &= US_k^4 U^\top U S_k V^\top = US_k^5 V^\top. \end{aligned}$$

Substituting into (5) gives

$$X_{k+1} = U(aS_k + bS_k^3 + cS_k^5)V^\top = U\varphi(S_k)V^\top, \quad \varphi(x) := ax + bx^3 + cx^5, \quad (6)$$

where φ acts on each diagonal entry. Thus the singular vectors stay fixed while each singular value undergoes the same scalar polynomial map (for the nonnegative scalar trajectories considered here).

For a nonzero input matrix $G = U\Sigma V^\top$, start with

$$X_0 = \frac{G}{\|G\|_F}. \quad (7)$$

Since $\|G\|_F^2 = \sum_i \sigma_i^2$, every initial singular value lies in $[0, 1]$. After K iterations,

$$X_K = U \operatorname{diag}\left(\varphi^{\circ K}\left(\frac{\sigma_1}{\|G\|_F}\right), \dots, \varphi^{\circ K}\left(\frac{\sigma_r}{\|G\|_F}\right)\right) V^\top, \quad (8)$$

where $\varphi^{\circ K}$ means K compositions, not a power. For a full-rank input, convergence of these scalars to 1 gives $X_K \rightarrow UV^\top = \operatorname{Ortho}(G)$. Normalization does not change the target: $\operatorname{Ortho}(sG) = \operatorname{Ortho}(G)$ for every $s > 0$.

Why convergence is possible. The classical cubic iteration takes $a = 3/2$, $b = -1/2$, and $c = 0$, so $\varphi(x) = (3x - x^3)/2$. For $0 < x < 1$,

$$\varphi(x) - x = \frac{x(1 - x^2)}{2} > 0, \quad 1 - \varphi(x) = \frac{(1 - x)^2(x + 2)}{2} > 0.$$

Its iterates increase and stay below 1, so they have a limit $\ell > 0$. Continuity implies $\ell = \varphi(\ell)$, whose only positive solution in $(0, 1]$ is $\ell = 1$. This proves convergence for every positive initial singular value at most 1; a zero singular value stays zero. The quintic family in (6) gives more freedom to shape this scalar map.

The five-step subroutine. For input $G \in \mathbb{R}^{m \times n}$, Muon uses the following procedure [1]. Set

$$(a, b, c) = (3.4445, -4.7750, 2.0315), \quad \epsilon_{\text{NS}} = 10^{-7}.$$

1. Normalize: $X \leftarrow G/(\|G\|_F + \epsilon_{\text{NS}})$. If $m > n$, transpose X . It now has shape $p \times q$, where $p = \min\{m, n\}$ and $q = \max\{m, n\}$.
2. Repeat the following operations five times:

$$A \leftarrow XX^\top, \quad D \leftarrow bA + cA^2, \quad X \leftarrow aX + DX. \quad (9)$$

3. If $m > n$, transpose X back. Return the resulting matrix as $\text{NewtonSchulz5}(G)$.

The stabilizer makes the routine defined at $G = 0$, where it returns zero. Forming $XX^\top$ after the optional transpose keeps the Gram matrix A at the smaller size $p \times p$. Expanding DX in (9) recovers (5).

Tuning the polynomial. The post first considers $(a, b, c) = (2, -1.5, 0.5)$, then tunes the coefficients to accelerate small singular values: near zero, $\varphi(x) \approx ax$. Allowing approximate normalization around $[0.7, 1.3]$ permits a larger a and gives the coefficients above; five steps sufficed in the reported experiments [1].

References

- [1] Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks. 2024. <https://kellerjordan.github.io/posts/muon/>.
- [2] Jeremy Bernstein and Laker Newhouse. Old Optimizer, New Norm: An Anthology. 2024. <https://arxiv.org/abs/2409.20325>.

AI Use. I prepared these lecture notes with the aid of AI assistance; I take full responsibility for all content in the notes.