How to Estimate the Spectral Norm of a Matrix More Scientifically

Jianlin Su

2026-05-04

The spectral norm is a core concept in matrix analysis and also plays an important role in deep learning—from the Lipschitz constraints required in the WGAN era, to the training stability of today's LLMs, and the emerging Muon optimizer, all of which are closely related to the spectral norm of matrix parameters. Therefore, how to efficiently and accurately estimate the spectral norm has become increasingly important and worthy of in-depth study.

As is well known, power iteration is the standard scheme for estimating the spectral norm, but there is still much room for improvement. This article briefly reviews some ideas for estimating the spectral norm, including improving the convergence speed of power iteration and how to estimate a strict upper bound of the spectral norm, etc.

1 Spectral Norm

The definition of the spectral norm is

$$\|\boldsymbol{W}\|_2 = \max_{\|\boldsymbol{x}\|_2=1} \|\boldsymbol{W}\boldsymbol{x}\|_2 \tag{1}$$

where $\boldsymbol{W} \in \mathbb{R}^{n\times m}, \boldsymbol{x} \in \mathbb{R}^m$, and without loss of generality we assume $n \geq m$. From the definition, the spectral norm represents the "expansion rate" of a linear layer—after passing through the linear layer, the length of the output vector expands by at most $\|\boldsymbol{W}\|_2$ times. The spectral norm is also equal to the largest singular value of the matrix (for a proof, see "The Road to Low-Rank Approximation (Part 2): SVD"), so we will not dwell on these basics.

The author first encountered the spectral norm during the heyday of GANs, when WGAN appeared and emphasized the necessity of the discriminator satisfying a Lipschitz constraint. The Lipschitz constant of a linear layer $\boldsymbol{W}\boldsymbol{x}$ is precisely the spectral norm of the matrix $\boldsymbol{W}$, which led to techniques such as spectral normalization and spectral regularization; interested readers can refer to "Lipschitz Constraints in Deep Learning: Generalization and Generative Models".

In recent years, the gradually popular Muon optimizer is also closely related to the spectral norm; it is usually regarded as steepest descent under the spectral norm. At the same time, in "Above MuP: 4. Adhering to Parameter Stability", we also proposed constraining the spectral norm of parameters to ensure training stability. All of these contents impose demands on the accurate calculation of the spectral norm.

2 Power Iteration

Computing the spectral norm via SVD is the most direct method, but obviously too expensive. We usually use power iteration

$$\boldsymbol{v}^{(t)} = \frac{\boldsymbol{W}^\top \boldsymbol{W} \boldsymbol{v}^{(t-1)}}{\|\boldsymbol{W}^\top \boldsymbol{W} \boldsymbol{v}^{(t-1)}\|_2} \tag{2}$$

which converges to the right principal singular vector $\boldsymbol{v}_1$ at a rate of $(\sigma_2/\sigma_1)^{2t}$; after T steps we have

$$\sigma_1 \approx \|\boldsymbol{W} \boldsymbol{v}^{(t)}\|_2 \tag{3}$$

The proof can be found in "From the Gradient of the Spectral Norm to Thoughts on a New Weight Decay". In practice, if we only care about the spectral norm and not the singular vectors, the convergence speed is often more optimistic than $(\sigma_2/\sigma_1)^{2t}$. Moreover, the result of power iteration is easily affected by the initial value $\boldsymbol{v}^{(0)}$ and can fluctuate; we can consider computing k paths in parallel and then taking the maximum.

If we simultaneously compute k paths and replace the L2 normalization at each step with orthogonalization of these k vectors, i.e.

$$\boldsymbol{V}^{(t)} = \mathrm{QR}(\boldsymbol{W}^\top \boldsymbol{W} \boldsymbol{V}^{(t-1)}) \tag{4}$$

then the result will converge to the first k right singular vectors, from which the first k singular values can be obtained; the principle can be found in "Streaming Power Iteration Based Muon Implementation: 4. Principles". However, we will not expand on this extension here, and keep our attention focused on estimating the spectral norm—that is, the largest singular value.

3 Gradient

First, here is a JAX-based reference implementation of power iteration:

```
import jax, jax.lax as lax
import jax.numpy as jnp

@jax.jit(static_argnums=(1,))
def l2_normalize(x, axis=-2):
```

```
    return x / jnp.linalg.vector_norm(x, axis=axis, keepdims=True)

@jax.jit(static_argnums=(1, 2, 3))
def spec_norm_v1(w, T=10, k=1, key=42):
    v_shape = w.shape[:-2] + w.shape[-1:] + (k,)
    v_init = jax.random.normal(jax.random.PRNGKey(key), v_shape)
    v_step = lambda i, v: l2_normalize(w.mT @ (w @ v))
    v = lax.fori_loop(0, T, v_step, v_init)
    return jnp.linalg.vector_norm(w @ v, axis=-2).max(axis=-1)
```

Obviously, the complexity of power iteration is $\mathcal{O}(mnTk)$, which is already relatively ideal in terms of complexity alone. If we only use it for monitoring, the above implementation is sufficient; but if we use it for spectral normalization or spectral regularization, we need the gradient of the spectral norm. According to the above implementation, it would need to backpropagate through the loop of power iteration step by step, which is relatively expensive in computational complexity.

In fact, in "From the Gradient of the Spectral Norm to Thoughts on a New Weight Decay", we have already derived the gradient of the spectral norm $\nabla_{\boldsymbol{W}}\sigma_1 = \boldsymbol{u}_1\boldsymbol{v}_1^\top$. Therefore, we can simply customize its gradient as $\nabla_{\boldsymbol{W}}\sigma_1 = \boldsymbol{u}_1\boldsymbol{v}_1^\top$ to reduce the complexity of backpropagation. In addition, we can also use $\sigma_1 = \boldsymbol{u}_1^\top\boldsymbol{W}\boldsymbol{v}_1$, and directly output in the forward pass

$$\sigma_1 = \text{sg}[\boldsymbol{u}_1^\top]\boldsymbol{W}\text{sg}[\boldsymbol{v}_1] \tag{5}$$

where $\text{sg}[]$ is the stop-gradient operator. Then during automatic differentiation, only $\boldsymbol{W}$ will be differentiated, and the result is exactly $\boldsymbol{u}_1\boldsymbol{v}_1^\top$, avoiding backpropagation into the interior of the power iteration for $\boldsymbol{u}_1, \boldsymbol{v}_1$.

4 No Waste

Through T steps of power iteration, we obtain the vector sequence $\boldsymbol{v}^{(1)}, \boldsymbol{v}^{(2)}, \cdots, \boldsymbol{v}^{(T)}$, but in the end we only use $\boldsymbol{v}^{(T)}$ to estimate σ_1. This seems rather "wasteful," so some people think about utilizing all of them to form a better estimate of the spectral norm.

Barring accidents, $\boldsymbol{v}^{(1)}, \boldsymbol{v}^{(2)}, \cdots, \boldsymbol{v}^{(T)}$ are distinct but increasingly close to $\boldsymbol{v}_1$; we can imagine them as "$\boldsymbol{v}_1$ plus some kind of noise." If we put them together to "denoise," it is indeed possible to obtain a more accurate result. Specifically, we consider using a linear combination of $\boldsymbol{v}^{(1)}, \boldsymbol{v}^{(2)}, \cdots, \boldsymbol{v}^{(T)}$ to build a better approximation of $\boldsymbol{v}_1$; the subspace spanned by these vectors is called the "Krylov subspace."

For simplicity, we first perform an orthogonalization to obtain an equivalent orthonormal basis $\boldsymbol{Q} = [\boldsymbol{q}_1, \boldsymbol{q}_2, \cdots, \boldsymbol{q}_T] \in \mathbb{R}^{m\times T}$; next, we seek coefficients $\boldsymbol{x} = [x_1, x_2, \cdots, x_T]^\top \in \mathbb{R}^T$ such that the vector $\sum_{i=1}^T x_i\boldsymbol{q}_i = \boldsymbol{Q}\boldsymbol{x}$ satisfies

$$\boldsymbol{W}^\top\boldsymbol{W}\boldsymbol{Q}\boldsymbol{x} \approx \sigma_1^2\boldsymbol{Q}\boldsymbol{x} \tag{6}$$

as closely as possible. Multiplying both sides by $\boldsymbol{Q}^\top$ gives $\boldsymbol{Q}^\top\boldsymbol{W}^\top\boldsymbol{W}\boldsymbol{Q}\boldsymbol{x} \approx \sigma_1^2\boldsymbol{x}$, which indicates that $\sigma_1^2, \boldsymbol{x}$ are respectively an eigenvalue and eigenvector of $(\boldsymbol{W}\boldsymbol{Q})^\top\boldsymbol{W}\boldsymbol{Q}$. This means we only need to perform an eigenvalue decomposition on it, obtain its largest eigenvalue, and then take the square root to get a better estimate. Since this is only a $T \times T$ matrix, when T is relatively small, performing an eigenvalue decomposition on it is not a costly matter, so we can directly call existing functions.

This idea is equivalent to a simplified version of the "Lanczos algorithm" in modern SVD solvers; actual SVD involves more refined processing. In addition, this acceleration idea is also related to Randomized SVD: random SVD usually projects with a Gaussian random matrix, while here we project with an orthonormal basis in the Krylov subspace.

5 Acceleration Techniques

The reference code for power iteration incorporating Krylov subspace acceleration is as follows:

```
@jax.jit(static_argnums=(1, 2))
def spec_norm_v2(w, T=10, key=42):
    v_shape = w.shape[:-2] + w.shape[-1:] + (1,)
    v_init = jax.random.normal(jax.random.PRNGKey(key), v_shape)
    v_step = lambda v, _: (l2_normalize(w.mT @ (w @ v)),) * 2
    v = lax.scan(v_step, v_init, length=T)[1].swapaxes(0, -1)[0]
    v = jnp.linalg.qr(v)[0]
    return jnp.linalg.eigvalsh((u := w @ v).mT @ u)[..., -1]**0.5
```

Experiments show that the subspace acceleration effect is very obvious: at $T = 5$ its accuracy already exceeds that of the original power iteration at $T = 10$, and it is faster. At $T = 10$, roughly two-thirds of the time of this function is due to power iteration, one-third is due to the QR decomposition, and the eigenvalue decomposition takes a very small proportion.

Readers who have read the "Streaming Power Iteration" series may think of using the Cholesky QR introduced there to accelerate the QR decomposition. Unfortunately, because the results of power iteration become increasingly collinear, the condition number of the matrix to be QR-decomposed will continuously deteriorate, which is exactly the region where Cholesky QR is "powerless"; the failure rate is extremely high, so it basically cannot be used to accelerate.

A more effective acceleration means is: only use the last few steps of the power iteration results, such as the last three steps $\boldsymbol{v}^{(T-2)}, \boldsymbol{v}^{(T-1)}, \boldsymbol{v}^{(T)}$, to build the Krylov subspace acceleration. This already captures most of the benefit, and reduces the computational cost of the QR decomposition and eigenvalue decomposition, thereby achieving acceleration.

6 Upper Bound Estimation

Strictly speaking, power iteration and its accelerated version can only obtain a lower bound of the spectral norm; of course this is often sufficient. But if in certain scenarios we must obtain an upper bound—for example, if we want to ensure via spectral normalization that the norm of some matrix is strictly less than 1—then we need another approach.

At this point a basic scheme is to compute the Schatten norm, which is precisely an upper bound of the spectral norm. Let the SVD of matrix $\boldsymbol{W}$ be $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$, with singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_m$; then its Schatten-$p$ norm is defined as

$$\|\boldsymbol{W}\|_{S,p} = \sqrt[p]{\sum_{i=1}^m \sigma_i^p} \quad \geq \quad \sigma_1 \tag{7}$$

The larger p is, the closer to the true value. When p is even, computing the Schatten-p norm is relatively feasible, mainly using the following identities:

$$\|\boldsymbol{W}\|_{S,2k} = \sqrt[2k]{\sum_{i=1}^m \sigma_i^{2k}} = \sqrt[2k]{\mathrm{tr}(\boldsymbol{V}\boldsymbol{\Sigma}^{2k}\boldsymbol{V}^\top)} = \sqrt[2k]{\mathrm{tr}((\boldsymbol{W}^\top\boldsymbol{W})^k)} \tag{8}$$

$$\|\boldsymbol{W}\|_{S,4k} = \sqrt[4k]{\sum_{i=1}^m \sigma_i^{4k}} = \sqrt[2k]{\|\boldsymbol{V}\boldsymbol{\Sigma}^{2k}\boldsymbol{V}^\top\|_{S,2}} = \sqrt[2k]{\|(\boldsymbol{W}^\top\boldsymbol{W})^k\|_{S,2}} \tag{9}$$

It is not difficult to see that the Schatten-2 norm is actually the Frobenius norm, which can be obtained by summing elementwise squares and taking the square root. Therefore, the above two identities indicate that as long as we compute $(\boldsymbol{W}^\top\boldsymbol{W})^k$, we can obtain $\|\boldsymbol{W}\|_{S,2k}$ and $\|\boldsymbol{W}\|_{S,4k}$ at relatively low cost.

The first step $\boldsymbol{W}^\top\boldsymbol{W}$ has complexity $\mathcal{O}(nm^2)$; afterwards each repeated squaring step has complexity $\mathcal{O}(m^3)$. Thus with complexity $\mathcal{O}(nm^2 + Tm^3)$ we can obtain $(\boldsymbol{W}^\top\boldsymbol{W})^{2^T}$. Theoretically, this complexity far exceeds that of power iteration; even when n, m are relatively large, the first step $\boldsymbol{W}^\top\boldsymbol{W}$ alone already far exceeds power iteration. Therefore, if we only want to estimate the spectral norm and do not require an upper bound, power iteration should still be the priority.

However, matrix multiplication can usually be fully parallelized, so when n, m are not large or $n \gg m$, the Schatten-p norm will not encounter a computational bottleneck; in such cases we can consider it. Alternatively, when we absolutely must have a strict upper bound, this seems to be the most direct route.

7 Preserving Numerical Stability

To compute $\|\boldsymbol{W}\|_{S,2^{T+2}}$, a naive implementation starts from $\boldsymbol{M} = \boldsymbol{W}^\top\boldsymbol{W}$ and repeatedly executes $\boldsymbol{M} \leftarrow \boldsymbol{M}^2$ for T times to obtain $(\boldsymbol{W}^\top\boldsymbol{W})^{2^T}$, and then substitutes into formula

(9). However, because it is a super-exponential operation, it quickly explodes to NaN or collapses to zero.

In general, $\|\boldsymbol{W}\|_{S,2^{T+2}}$ itself does not explode numerically; the problem lies in explicitly computing $(\boldsymbol{W}^\top\boldsymbol{W})^{2^T}$. The solution is to re-normalize after each squaring $[\boldsymbol{M} \leftarrow \boldsymbol{M}^2/\operatorname{tr}(\boldsymbol{M}^2)]$; this can prevent numerical explosion and also ensure the scaling is tight enough to avoid collapsing to zero. At the same time, we accumulate the normalization factor at each step in the log domain for final computation of $\|\boldsymbol{W}\|_{S,2^{T+2}}$.

The computational procedure is summarized as follows:

Compute $\|\boldsymbol{W}\|_{S,2^{T+2}}$ as an upper bound of the spectral norm
1 : Initialize $\log S = \log\operatorname{tr}(\boldsymbol{W}^\top\boldsymbol{W})$, $\boldsymbol{M} = \frac{\boldsymbol{W}^\top\boldsymbol{W}}{\operatorname{tr}(\boldsymbol{W}^\top\boldsymbol{W})}$
2 : **For** $t = 1, 2, \cdots, T$ **do**
3 : $\quad\log S \leftarrow 2\log S + \log\operatorname{tr}(\boldsymbol{M}^2)$
4 : $\quad\boldsymbol{M} \leftarrow \frac{\boldsymbol{M}^2}{\operatorname{tr}(\boldsymbol{M}^2)}$
5 : Output $\exp\left(\frac{\log S + \log\|\boldsymbol{M}\|_F}{2^{T+1}}\right)$

A simple reference implementation:

```
@jax.jit
def tr(w):
    return w.trace(axis1=-1, axis2=-2)[..., None, None]

@jax.jit(static_argnums=(1,))
def spec_norm_v3(w, T=5):
    m = (m := w.mT @ w) / (s := tr(m))
    ms_step = lambda i, ms: ((m := ms[0] @ ms[0]) / (s := tr(m)), 2 * ms[1] + jnp.
        log(s))
    m, logs = lax.fori_loop(0, T, ms_step, (m, jnp.log(s)))
    logf = 0.5 * jnp.log((m**2).sum(axis=[-1, -2], keepdims=True))
    return jnp.exp((logs + logf) / 2**(T + 1))
```

8 Multi-Order Moments

To compute $\|\boldsymbol{W}\|_{S,2^{T+2}}$, we must in some way compute $\boldsymbol{W}^\top\boldsymbol{W}, \cdots, (\boldsymbol{W}^\top\boldsymbol{W})^{2^{T-1}}, (\boldsymbol{W}^\top\boldsymbol{W})^{2^T}$. This means we can simultaneously obtain $\|\boldsymbol{W}\|_{S,2}, \cdots, \|\boldsymbol{W}\|_{S,2^{T+1}}, \|\boldsymbol{W}\|_{S,2^{T+2}}$, but in the end we only use $\|\boldsymbol{W}\|_{S,2^{T+2}}$, which again seems somewhat "wasteful."

Is there a way, like the Krylov subspace method for power iteration, to utilize all these results to improve estimation accuracy? There is! "Fast Tight Spectral-Norm Bounds" provides an idea to improve the estimate via nonlinear programming. Let us first look at a simple case. Suppose we have obtained the 2nd and 4th moments of the singular values

$$\sum_{i=1}^m \sigma_i^2 = S_2, \qquad \sum_{i=1}^m \sigma_i^4 = S_4 \tag{10}$$

Then according to the results of the previous two sections, $S_4^{1/4}$ is closer to the spectral norm than $S_2^{1/2}$, so we return $S_4^{1/4}$ as an upper bound of the spectral norm and discard S_2. But is S_2 really useless? Imagine if $m = 2$; then we have exactly two equations and two unknowns, and in principle we can solve for σ_1, σ_2! Although we cannot solve exactly when $m > 2$, we can still narrow the range of σ_1. Specifically, we have

$$\frac{S_2 - \sigma_1^2}{m-1} = \frac{1}{m-1}\sum_{i=2}^{m}\sigma_i^2 \leq \sqrt{\frac{1}{m-1}\sum_{i=2}^{m}\sigma_i^4} = \sqrt{\frac{S_4 - \sigma_1^4}{m-1}} \tag{11}$$

That is, $(S_2 - \sigma_1^2)^2 \leq (m-1)(S_4 - \sigma_1^4)$, which is essentially a quadratic inequality; it is easy to solve

$$\sigma_1 \leq \sqrt{\frac{S_2 + \sqrt{(m-1)(mS_4 - S_2^2)}}{m}} \tag{12}$$

This gives an upper bound on σ_1 expressed in terms of S_2 and S_4, which is a better estimate than $S_4^{1/4}$. "Fast Tight Spectral-Norm Bounds" provides a result using S_2, S_4, S_6, S_8 simultaneously, but it is more complex overall; interested readers can study the original article themselves. Using more moments to improve the estimate is theoretically feasible, but in practice it often requires solving more complex nonlinear systems of equations, which is of limited practical value.

9 Limitations

Theoretically, result (12) can be generalized to use arbitrary S_{2k} and S_{4k} to obtain a more accurate upper bound:

$$\sigma_1 \leq \sqrt[2k]{\frac{S_{2k} + \sqrt{(m-1)(mS_{4k} - S_{2k}^2)}}{m}} \tag{13}$$

However, when k is relatively large, the practical significance of this result is very limited, because it requires explicitly computing S_{2k} and S_{4k}, which will explode or collapse when k is large, which is unrealistic. One possible improvement is to factor out $S_{4k}^{1/4k}$:

$$\sqrt[2k]{\frac{S_{2k} + \sqrt{(m-1)(mS_{4k} - S_{2k}^2)}}{m}} = S_{4k}^{1/4k} \cdot \sqrt[2k]{\frac{S_{2k}/S_{4k}^{1/2} + \sqrt{(m-1)(m - S_{2k}^2/S_{4k})}}{m}} \tag{14}$$

Then let $S_{2k}/S_{4k}^{1/2} = e^{\epsilon}$ and expand approximately in the log domain

$$\sqrt[2k]{\frac{S_{2k}/S_{4k}^{1/2} + \sqrt{(m-1)(m - S_{2k}^2/S_{4k})}}{m}} \approx 1 - \frac{\epsilon^2}{4k(m-1)} \tag{15}$$

However, our goal is to find an upper bound of the spectral norm; how to guarantee the upper-bound property while performing an approximate expansion seems relatively complicated. On the other hand, if we have already computed up to relatively large k, then $S_{4k}^{1/4k}$ itself already has relatively high accuracy, and the significance of further improving the precision with programming ideas is not great.

10 Conclusion

This article briefly reviewed several ideas for estimating the spectral norm. In practical applications, if only an approximate monitoring of the spectral norm is needed, power iteration and its subspace accelerated version are usually sufficient; if a strict upper bound must be guaranteed, then computing the Schatten norm can be considered. The "no waste" ideas extending from the two routes—utilizing the iteration history via Krylov subspaces and utilizing multi-order moment information via nonlinear programming—also provide excellent algorithmic design inspiration for us.