OPT2024: 16th Annual Workshop on Optimization for Machine Learning 1–19

Old Optimizer, New Norm: An Anthology

Jeremy Bernstein JBERNSTEIN@MIT.EDU
Laker Newhouse LAKERN@MIT.EDU
MIT CSAIL, United States

Abstract

Deep learning optimizers are often motivated through a mix of convex and approximate second-order theory. We select three such methods—Adam, Shampoo and Prodigy—and argue that each method can instead be understood as a squarely first-order method without convexity assumptions. In fact, after switching off exponential moving averages, each method is equivalent to *steepest descent* under a particular *norm*. By generalizing this observation, we chart a new design space for training algorithms. Different operator norms should be assigned to different tensors based on the role that the tensor plays within the network. For example, while linear and embedding layers may have the same weight space of $\mathbb{R}^{m\times n}$, these layers play different roles and should be assigned different norms. We hope that this idea of carefully metrizing the neural architecture might lead to more stable, scalable and indeed faster training.

Prologue

Deep learning optimizers are often motivated from the perspectives of convex and approximate second-order theory. These theoretical frameworks have been used to inspire algorithmic ideas, as well as providing means to analyse the convergence of various optimizers. However, we believe—and will attempt to demonstrate—that there is a wealth of untapped algorithmic opportunity in the simpler realm of exact first-order theory without convexity assumptions.

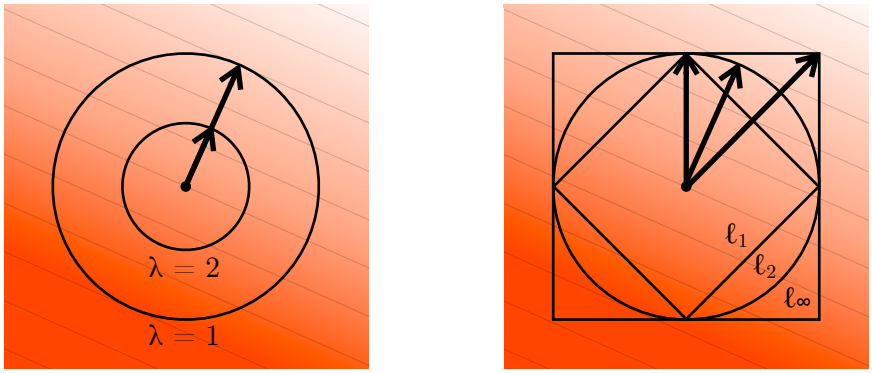
To make our case, we choose three optimizers that were originally analysed under convex or approximate second-order theory: Adam, Shampoo and Prodigy. After disabling their exponential moving averages (EMA), we show that each algorithm admits a parsimonious theoretical explanation as a variant of *steepest descent* under a certain norm. EMA can then be thought of as "smoothing out" the algorithm, or making it more robust to mini-batch noise, although nailing down the precise role of EMA is perhaps still an open problem.

By steepest descent, we mean the procedure of choosing a weight update $\Delta \boldsymbol{w}$ to minimise a local quadratic model of the loss function $\mathcal{L}$ of the form $\mathcal{L}(\boldsymbol{w}) + \nabla_{\boldsymbol{w}}\mathcal{L}(\boldsymbol{w})^\top \Delta \boldsymbol{w} + \frac{\lambda}{2}\cdot\|\Delta \boldsymbol{w}\|^2$, visualized in Figure 1. Crucially, the *sharpness parameter* λ and *norm* $\|\cdot\|$ are chosen a priori, without touching an (approximate) Hessian during training. As such, we consider steepest descent to be a squarely first-order method and not an (approximate) second-order method.

Throughout the anthology, we rely on a dual description of steepest descent:

Proposition 1 (Steepest descent) *For any $\boldsymbol{g} \in \mathbb{R}^n$ thought of as "the gradient" and any $\lambda \geqslant 0$ thought of as "the sharpness", and for any norm $\|\cdot\| : \mathbb{R}^n \to \mathbb{R}$ with dual norm $\|\cdot\|^\dagger$:*

$$\arg\min_{\Delta \boldsymbol{w}\in\mathbb{R}^n}\left[\boldsymbol{g}^\top \Delta \boldsymbol{w} + \frac{\lambda}{2}\|\Delta \boldsymbol{w}\|^2\right] = -\frac{\|\boldsymbol{g}\|^\dagger}{\lambda}\cdot\arg\max_{\|\boldsymbol{t}\|=1}\boldsymbol{g}^\top \boldsymbol{t}. \tag{1}$$



a) varying sharpness λ b) varying choice of norm $\|\cdot\|$

Figure 1: Steepest descent considers the problem of minimizing a linear functional under a quadratic penalty: $\arg\min_{\Delta \boldsymbol{w} \in \mathbb{R}^n} \left[\boldsymbol{g}^\top \Delta \boldsymbol{w} + \frac{\lambda}{2} \|\Delta \boldsymbol{w}\|^2 \right]$ for $\boldsymbol{g} \in \mathbb{R}^n$. Here we show how the solution varies with the sharpness $\lambda > 0$ and the choice of norm $\|\cdot\|$. We overlay different norm balls on top of a linear color gradient, and use arrows to denote the solution, meaning the member of the norm ball that "minimizes the color". a) Increasing the sharpness decreases the size of the solution vector. b) Changing the norm can change the direction of the solution vector. For different ℓ_p norms, the solution direction changes because the gradient is not axis-aligned. In practice, we should pick the sharpness and norm to fit the geometry of our loss.

Equation (1) separates the solution of the steepest descent problem into two pieces: first computing the *step size* as the dual norm of the gradient divided by the sharpness, and second solving for the *step direction* as the unit vector that maximizes the inner product with the gradient. The proof of this proposition is given in Appendix B.

Of course, the art of steepest descent lies in choosing a norm $\|\cdot\|$ and a sharpness λ suited to the optimization problem at hand. While it may be possible to turn this art into a science (Large et al., 2024), that ambition is beyond the scope of this anthology. Here we point out that past methods do implicitly make decisions about norms, and in a somewhat haphazard manner. In fact, they implicitly assign different *induced matrix norms* to the network layers:

Definition 1 (Induced operator norm) *Given a matrix $\boldsymbol{M} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ and two normed vector spaces $(\mathbb{R}^{d_{\text{in}}}, \|\cdot\|_\alpha)$ and $(\mathbb{R}^{d_{\text{out}}}, \|\cdot\|_\beta)$, the "$\alpha$ to β" induced operator norm is given by:*

$$\|\boldsymbol{M}\|_{\alpha \to \beta} = \max_{\boldsymbol{x} \in \mathbb{R}^{d_{\text{in}}}} \frac{\|\boldsymbol{M}\boldsymbol{x}\|_\beta}{\|\boldsymbol{x}\|_\alpha}. \tag{2}$$

Definition 1 tells us that by varying the choice of vector norms $\|\cdot\|_\alpha$ and $\|\cdot\|_\beta$, we can induce a large family of matrix norms. In turn, this implies a correspondingly large family of steepest descent optimizers. By foregrounding this issue, we hope that algorithm designers may develop more suitable optimizers by becoming more intentional about their choice of norm.

Story I. Adam as Steepest Descent under the Max-of-Max Norm

Adam is a widely used deep learning optimizer: the original paper of Kingma and Ba (2015) now has well over 100,000 citations. Adam has been motivated in various ways, including through convex analysis (Kingma and Ba, 2015) and as an approximate second-order method (Sun and Spall, 2021). However, there have been efforts to build a more direct understanding of Adam: for instance, with exponential moving averages (EMA) switched off, Adam is just *sign gradient descent* (Balles and Hennig, 2018; Bernstein et al., 2018), which is equivalent to steepest descent under the infinity norm (Carlson et al., 2015a). In this story, we connect Adam to a certain "max-of-max" norm, showing how Adam respects the tensor structure of a neural network in a very particular way.

To begin, we review how Adam connects to sign gradient descent. Ignoring bias corrections and numerical stabilizations, Adam is given by the following system of updates:

$$\boldsymbol{m}_t = \beta_1 \cdot \boldsymbol{m}_{t-1} + (1-\beta_1) \cdot \boldsymbol{g}_t, \tag{3}$$

$$\boldsymbol{v}_t = \beta_2 \cdot \boldsymbol{v}_{t-1} + (1-\beta_2) \cdot \boldsymbol{g}_t^2, \tag{4}$$

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta \cdot \boldsymbol{m}_t / \sqrt{\boldsymbol{v}_t}, \tag{5}$$

where t denotes the time step, $\boldsymbol{g}_t \in \mathbb{R}^n$ the gradient vector and $\eta > 0$ the step size. The EMA time scales of the first gradient moment $\boldsymbol{m}_t$ and second moment $\boldsymbol{v}_t$ are set by $0 \leqslant \beta_1, \beta_2 < 1$. All operations are conducted entry-wise. If we switch off EMA by setting $\beta_1 = \beta_2 = 0$, the Adam updates reduce to just sign gradient descent:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta \cdot \boldsymbol{g}_t / \sqrt{\boldsymbol{g}_t^2} \tag{6}$$

$$= \boldsymbol{w}_t - \eta \cdot \operatorname{sign}(\boldsymbol{g}_t). \tag{7}$$

This connection to sign descent should not be surprising since Adam, published in 2015, builds on the RMSprop optimizer that Tieleman and Hinton (2012) already called "the mini-batch version of just using the sign of the gradient". And RMSprop itself built on the RPROP optimizer (Riedmiller and Braun, 1993), which also uses gradient signs.

Still, *why should using the sign of the gradient be a good idea in deep learning?* In search of a motivation, we might consider that sign descent solves the problem of steepest descent under the vector ℓ_∞ norm, $\|\boldsymbol{v}\|_\infty := \max_i |\boldsymbol{v}_i|$ (Carlson et al., 2015a, 2016; Xie and Li, 2024):

Proposition 2 (Sign descent as steepest descent under the infinity norm) *For any gradient vector $\boldsymbol{g} \in \mathbb{R}^n$ and sharpness $\lambda > 0$, it holds that:*

$$\arg\min_{\Delta \boldsymbol{w} \in \mathbb{R}^n} \left[\boldsymbol{g}^\top \Delta \boldsymbol{w} + \frac{\lambda}{2} \|\Delta \boldsymbol{w}\|_\infty^2 \right] = -\frac{\|\boldsymbol{g}\|_1}{\lambda} \operatorname{sign}(\boldsymbol{g}). \tag{8}$$

In words, the vector that minimizes a linear functional under an infinity norm penalty is a scalar multiple of a sign vector. The proof is given in Appendix B.

While this connection between Adam, sign descent and steepest descent is perhaps cute, it does not answer a basic question: *Why does the vector ℓ_∞ norm have anything to do with neural network training?* In particular, taking the weight space to be $\mathbb{R}^n$ equipped with

This is because the infintiy norm is just the maxmimum vector direction, esentially ignoring (or not caring) the directions of all the other vectors in the feature space. (aka it's reductionist to the max element change)

the simple infinity norm seems to "throw away" the fact that the weight space of a neural network is built in a structured way out of layers of matrices (and perhaps other tensors).

To resolve this conundrum, we suggest that in fact the vector ℓ_∞ norm on the flattened weight space doesn't have anything to do with deep learning. Instead, there is a coincidence at play. The ℓ_∞ norm enjoys a special property summarized by the slogan "a max of a max is a max". To see this, consider a neural network with a list of L weight matrices $\boldsymbol{W}_1, \ldots, \boldsymbol{W}_L$. Let $\text{row}_r(\boldsymbol{W}_l)$ denote the rth row of the lth weight matrix, and let $\boldsymbol{w} = \text{flatten}(\boldsymbol{W}_1, \ldots, \boldsymbol{W}_L) \in \mathbb{R}^n$ denote the full flattened weight vector. Then we have that:

$$\|\boldsymbol{w}\|_\infty = \max_l \max_r \|\text{row}_r(\boldsymbol{W}_l)\|_\infty = \max_l \|\boldsymbol{W}_l\|_{\ell_1 \to \ell_\infty}, \tag{9}$$

where the second equality follows via Proposition 8. In words, the infinity norm of the flattened weight vector coincides with the largest ℓ_1 to ℓ_∞ operator norm of the layers. So Equation (9) connects the *unstructured* space of the flattened weight vector to the *structured* space of the list of weight matrices. We refer to the object $\max_l \|\boldsymbol{W}_l\|_{\ell_1 \to \ell_\infty}$ as the "max-of-max norm". And sign descent emerges as steepest descent under this norm:

Proposition 3 (Sign descent as steepest descent under the max-of-max norm) *For any list of gradient matrices $\boldsymbol{G}_1, ..., \boldsymbol{G}_L$ and any sharpness $\lambda > 0$, consider the problem:*

$$\underset{\Delta \boldsymbol{W}_1, \ldots, \Delta \boldsymbol{W}_L}{\arg\min} \left[\sum_{l=1}^{L} \langle \boldsymbol{G}_l, \Delta \boldsymbol{W}_l \rangle + \frac{\lambda}{2} \max_{l=1}^{L} \|\Delta \boldsymbol{W}_l\|_{\ell_1 \to \ell_\infty}^2 \right], \tag{10}$$

where $\langle \cdot, \cdot \rangle$ denotes the Frobenius inner product, and $\Delta \boldsymbol{W}_l$ has the same shape as $\boldsymbol{G}_l$. For step size $\eta = \frac{1}{\lambda} \sum_{l=1}^{L} \|\boldsymbol{G}_l\|_{\ell_1 \to \ell_\infty}^\dagger$, where $\dagger$ denotes the dual norm, Equation (10) is solved by:

$$\Delta \boldsymbol{W}_l = -\eta \cdot \text{sign}(\boldsymbol{G}_l) \qquad \textit{for each layer } l = 1, ..., L. \tag{11}$$

In words, the matrix-aware steepest descent problem of Equation (10) is solved by layerwise sign descent as given in Equation (11). This observation—that sign descent updates are implicitly doing *per-matrix gradient normalization*—may be a major reason that Adam, sign descent and Lion (Chen et al., 2023) outperform vanilla gradient descent in large language model training (Zhao et al., 2024; Large et al., 2024). The proof is given in Appendix B.

All told, this story has shown that Adam without EMA is sign descent and that, coincidentally, sign descent solves two different steepest descent problems: one on the flattened weight space, and one that is aware of the matrix structure of neural architecture. But, at the end of this story, questions linger. *Why does the ℓ_1 to ℓ_∞ induced operator norm rear its head? What does it have to do with deep learning? Aren't there other induced operator norms on matrices we could equally well consider?* For answers to these questions, dear reader, you'll have to wait for our next story... a story about Shampoo!

Story II. Shampoo as Steepest Descent under the Spectral Norm

Now, dear reader, we turn our attention to Shampoo (Gupta et al., 2017, 2018). A variant of the Shampoo optimizer won the external tuning track of the *2024 AlgoPerf: Training Algorithms* competition (Dahl et al., 2023). While the method was originally motivated as a generalization of the AdaGrad convex optimizer (Duchi et al., 2011) to tensor spaces, more recent work casts Shampoo as an approximate second-order method (Anil et al., 2020; Morwani et al., 2024). We will show that Shampoo—with accumulation disabled—is steepest descent under the max spectral norm over layers.

Meaning they make the feature space semi-orthogonal

To begin, we show that Shampoo updates, without accumulation, are semi-orthogonal matrices. At time step t and for each layer, Shampoo collects the gradient matrix $\boldsymbol{G}_t$ and makes the following update to the weight matrix $\boldsymbol{W}_t$:

$$\boldsymbol{L}_t = \boldsymbol{L}_{t-1} + \boldsymbol{G}_t\boldsymbol{G}_t^T, \tag{12}$$

$$\boldsymbol{R}_t = \boldsymbol{R}_{t-1} + \boldsymbol{G}_t^T\boldsymbol{G}_t, \tag{13}$$

$$\boldsymbol{W}_{t+1} = \boldsymbol{W}_t - \eta \cdot \boldsymbol{L}_t^{-1/4}\boldsymbol{G}_t\boldsymbol{R}_t^{-1/4}. \tag{14}$$

All operations, including the inverse fourth roots, are matrix operations. The accumulators $\boldsymbol{L}_t$ and $\boldsymbol{R}_t$ are referred to as the "left and right pre-conditioners". Practitioners usually replace the simple sums in Equations (12) and (13) with EMAs (Shi et al., 2023). If we disable the accumulation, setting $\boldsymbol{L}_t = \boldsymbol{G}_t\boldsymbol{G}_t^\top$ and $\boldsymbol{R}_t = \boldsymbol{G}_t^\top\boldsymbol{G}_t$, Shampoo reduces to:

$$\boldsymbol{W}_{t+1} = \boldsymbol{W}_t - \eta \cdot (\boldsymbol{G}_t\boldsymbol{G}_t^\top)^{-1/4}\boldsymbol{G}_t(\boldsymbol{G}_t^\top\boldsymbol{G}_t)^{-1/4} \tag{15}$$

$$= \boldsymbol{W}_t - \eta \cdot \boldsymbol{U}_t\boldsymbol{V}_t^\top, \tag{16}$$

where Equation (16) is reached by substituting the reduced singular value decomposition (SVD) of the gradient $\boldsymbol{G}_t = \boldsymbol{U}_t\boldsymbol{\Sigma}_t\boldsymbol{V}_t^\top$ into Equation (15). Notice that there is a direct parallel between Equations (6) and (7) for Adam and Equations (15) and (16) for Shampoo. So, Shampoo without accumulation makes a *semi-orthogonal weight update*. In fact:

Proposition 4 (Projection to the closest semi-orthogonal matrix) *Consider the semi-orthogonal matrices* $\mathcal{O}_{m\times n} := \{\boldsymbol{A} \in \mathbb{R}^{m\times n} : \boldsymbol{A}\boldsymbol{A}^\top = \mathbf{I}_m \text{ or } \boldsymbol{A}^\top\boldsymbol{A} = \mathbf{I}_n\}$ *and let* $\|\cdot\|_F$ *denote the Frobenius norm. For any matrix* $\boldsymbol{G} \in \mathbb{R}^{m\times n}$ *with reduced SVD* $\boldsymbol{G} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$*:*

$$\operatorname*{arg\,min}_{\boldsymbol{A}\in\mathcal{O}_{m\times n}} \|\boldsymbol{A} - \boldsymbol{G}\|_F = \boldsymbol{U}\boldsymbol{V}^\top, \tag{17}$$

where the minimizer $\boldsymbol{U}\boldsymbol{V}^\top$ *is unique if and only if the matrix* $\boldsymbol{G}$ *has full rank.*

So, Shampoo without accumulation projects the gradient matrix to the *closest semi-orthogonal matrix in Frobenius norm*. The proof is in Appendix B. *Why might this be a good idea*, you ask? Well, for one thing, it's steepest descent—this time under the maximum *spectral norm* $\|\cdot\|_{\ell_2\to\ell_2}$ (Definition 1) over all the matrices in the network:

Proposition 5 (Shampoo as steepest descent under the spectral norm) *For any list of gradient matrices* $\boldsymbol{G}_1, ..., \boldsymbol{G}_L$ *and any sharpness* $\lambda > 0$*, consider the problem:*

$$\operatorname*{arg\,min}_{\Delta\boldsymbol{W}_1,\ldots,\Delta\boldsymbol{W}_L} \left[\sum_{l=1}^{L}\langle \boldsymbol{G}_l, \Delta\boldsymbol{W}_l\rangle + \frac{\lambda}{2}\max_{l=1}^{L}\|\Delta\boldsymbol{W}_l\|_{\ell_2\to\ell_2}^2\right], \tag{18}$$

where $\langle \cdot, \cdot \rangle$ *denotes the Frobenius inner product and* $\Delta \boldsymbol{W}_l$ *has the same shape as* $\boldsymbol{G}_l$*. Suppose that* $\boldsymbol{G}_l$ *has reduced SVD given by* $\boldsymbol{G}_l = \boldsymbol{U}_l \boldsymbol{\Sigma}_l \boldsymbol{V}_l^\top$ *for each* $l = 1, ..., L$*. Then Equation (18) is solved with a step size* $\eta = \frac{1}{\lambda} \sum_{l=1}^{L} \operatorname{tr} \boldsymbol{\Sigma}_l$ *and an update:*

$$\Delta \boldsymbol{W}_l = -\eta \cdot \boldsymbol{U}_l \boldsymbol{V}_l^\top \quad \text{for each } l = 1, ..., L. \tag{19}$$

This solution for $\Delta \boldsymbol{W}_l$ *is unique if and only if the matrix* $\boldsymbol{G}_l$ *is of full rank.*

The proof is given in Appendix B. A novelty of this proposition in contrast to prior work on stochastic spectral descent (Carlson et al., 2015a, 2016) is our use of a max norm over layers to handle the multi-layer case. However, our main contribution here is to draw the connection between Proposition 5 and Shampoo as in Equations (15) and (16).

So, Shampoo without accumulation is steepest descent under the spectral norm. *Why might this be a good idea in deep learning?* The idea that we wish to advance is that one can derive upper bounds on the loss of machine learning models in terms of spectral norms. Here we present the simplest possible example: a linear model and the square loss.

Proposition 6 (Bounding the square loss of a linear predictor) *Consider a matrix* $\boldsymbol{W} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ *that we shall think of as a linear predictor mapping an input* $\boldsymbol{x} \in \mathbb{R}^{d_{\text{in}}}$ *to an output* $\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x} \in \mathbb{R}^{d_{\text{out}}}$*. Given a dataset of* n *samples* $\mathcal{D} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), ..., (\boldsymbol{x}_n, \boldsymbol{y}_n)\}$*, where the* i*th input is normalized such that* $\|\boldsymbol{x}_i\|_2 = \sqrt{d_{\text{in}}}$*, we can construct the "square loss":*

$$\mathcal{L}(\boldsymbol{W}) := \frac{1}{2n} \sum_{i=1}^{n} \frac{1}{d_{\text{out}}} \|\boldsymbol{y}_i - \boldsymbol{W}\boldsymbol{x}_i\|_2^2. \tag{20}$$

Then, for any matrix $\Delta \boldsymbol{W} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ *thought of as a weight update, it holds that:*

$$\mathcal{L}(\boldsymbol{W} + \Delta \boldsymbol{W}) \leqslant \mathcal{L}(\boldsymbol{W}) + \langle \nabla_{\boldsymbol{W}} \mathcal{L}(\boldsymbol{W}), \Delta \boldsymbol{W} \rangle + \tfrac{1}{2} \cdot \tfrac{d_{\text{in}}}{d_{\text{out}}} \cdot \|\Delta \boldsymbol{W}\|_{\ell_2 \to \ell_2}^2, \tag{21}$$

where $\langle \cdot, \cdot \rangle$ *is the Frobenius inner product.*

In words: the square loss of a linear predictor admits an upper bound that is quadratic in the spectral norm of the weight perturbation. Choosing the weight perturbation to minimize this upper bound is precisely steepest descent under the spectral norm! The proof is given in Appendix B. This optimizer design pattern, which starts by deriving an upper bound on the loss (as in Proposition 6) and then minimizes it (as in Proposition 5), is known generally as majorization-minimization (Lange, 2016). It is an exact and first-principles design pattern, without Hessian approximations or appeals to convex theory. This design pattern is used extensively by Carlson et al. (2015a, 2016) to design optimizers for restricted Boltzmann machines and discrete graphical models. Generalizing the pattern to arbitrary network architectures and loss functions requires more advanced machinery (Bernstein et al., 2023; Streeter, 2023; Large et al., 2024).

And so, dear reader, we have reached the end of our second story. We have shown that Shampoo without accumulation corresponds to projecting the gradient matrix to the closest

Domain	Norm	Solution	Optimizer	Cousin
$\mathbb{R}^n$	Euclidean ℓ_2	$\Delta \boldsymbol{w} = -\frac{\|\boldsymbol{g}\|_2}{\lambda} \frac{\boldsymbol{g}}{\|\boldsymbol{g}\|_2}$	vanilla gradient descent	SGD
$\mathbb{R}^n$	infinity ℓ_∞	$\Delta \boldsymbol{w} = -\frac{\|\boldsymbol{g}\|_1}{\lambda} \operatorname{sign}(\boldsymbol{g})$	sign descent	Adam
$\mathbb{R}^{m \times n}$	Frobenius S_2	$\Delta \boldsymbol{W} = -\frac{\|\boldsymbol{G}\|_F}{\lambda} \frac{\boldsymbol{G}}{\|\boldsymbol{G}\|_F}$	vanilla gradient descent	SGD
$\mathbb{R}^{m \times n}$	spectral S_∞	$\Delta \boldsymbol{W} = -\frac{\operatorname{tr} \boldsymbol{\Sigma}}{\lambda} \boldsymbol{U} \boldsymbol{V}^\top$	spectral descent	Shampoo

Table 1: Popular optimizers are related to steepest descent under different norms. For vector-valued optimization problems, we consider the steepest descent problem $\arg\min_{\Delta \boldsymbol{w}} \boldsymbol{g}^\top \Delta \boldsymbol{w} + \frac{\lambda}{2} \cdot \|\Delta \boldsymbol{w}\|^2$. For matrix-valued problems, we consider $\arg\min_{\Delta \boldsymbol{W}} \langle \boldsymbol{G}, \Delta \boldsymbol{W} \rangle + \frac{\lambda}{2} \cdot \|\Delta \boldsymbol{W}\|^2$, where $\langle \cdot, \cdot \rangle$ is the Frobenius inner product. We list the solution for different vector ℓ_p norms and Schatten S_p norms. The Schatten S_p norm of a matrix returns the ℓ_p norm of its vector of singular values. Finally, $\boldsymbol{G} = \boldsymbol{U} \boldsymbol{\Sigma} \boldsymbol{V}^\top$ is the reduced singular value decomposition of the gradient.

semi-orthogonal matrix, which solves the problem of steepest descent under the spectral norm. And we showed how steepest descent under the spectral norm emerges from upper bounding the square loss of a linear predictor. This perspective, of viewing Shampoo as a (smoothed out) projection to the space of semi-orthogonal matrices, grounds the algorithm in a prior literature on spectral descent (Carlson et al., 2015a, 2016; Fan, 2017). And in Appendix A, we discuss how it might unlock new means for computing the Shampoo updates.

We summarize our first two stories in Table 1. And we still have one more left to tell...

Story III. Prodigy: Automatically Computing the Escape Velocity

For our final story, we speak of Prodigy (Mishchenko and Defazio, 2023). The Prodigy optimizer falls amid a series of recent works (Defazio and Mishchenko, 2023; Khaled et al., 2023; Ivgi et al., 2023) that attempt to apply convex theory to design and analyse deep learning optimizers that do not require tuning. In contrast, we argue that Prodigy (without EMA) is but another example of steepest descent, where instead of using the step size $\eta = \|\boldsymbol{g}\|^{\dagger}/\lambda$ from Proposition 1, Prodigy uses a heuristic to automatically warm up to a good step size. This demonstrates the value of Proposition 1 for disentangling the optimizer design problem. If one knows a good norm $\|\cdot\|$ but is ignorant of the sharpness parameter λ, then one may obtain the step direction by solving $\arg\max_{\|\boldsymbol{t}\|=1} \boldsymbol{g}^{\top}\boldsymbol{t}$ from Proposition 1, while using another means to find a good step size.

Then let us make our case. We focus on Algorithm 3 in the Prodigy paper, since this is the version used in their experiments. We first show that with EMA switched off, Prodigy implements sign gradient descent with a step size that warms up automatically. Ignoring the numerical stabilization and learning rate schedule, Prodigy is given by:

$$\boldsymbol{m}_t = \beta_1 \cdot \boldsymbol{m}_{t-1} + (1-\beta_1)\cdot \eta_t\, \boldsymbol{g}_t, \tag{22}$$

$$\boldsymbol{v}_t = \beta_2 \cdot \boldsymbol{v}_{t-1} + (1-\beta_2)\cdot \eta_t^2\, \boldsymbol{g}_t^2, \tag{23}$$

$$r_t = \sqrt{\beta_2}\cdot r_{t-1} + (1-\sqrt{\beta_2})\cdot \eta_t^2\, \boldsymbol{g}_t^{\top}(\boldsymbol{w}_0 - \boldsymbol{w}_t), \tag{24}$$

$$\boldsymbol{s}_t = \sqrt{\beta_2}\cdot \boldsymbol{s}_{t-1} + (1-\sqrt{\beta_2})\cdot \eta_t^2\, \boldsymbol{g}_t, \tag{25}$$

$$\eta_{t+1} = \max\left(\eta_t, \tfrac{r_t}{\|\boldsymbol{s}_t\|_1}\right), \tag{26}$$

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta_t \cdot \boldsymbol{m}_t/\sqrt{\boldsymbol{v}_t}, \tag{27}$$

where t denotes the time step and $\boldsymbol{g}_t \in \mathbb{R}^n$ the gradient vector. While this system of updates may seem intimidating, if we switch off EMA by setting $\beta_1 = \beta_2 = 0$, the Prodigy updates simplify dramatically to just sign gradient descent with a dynamical step size as follows:

$$\eta_{t+1} = \max\left(\eta_t, \tfrac{\boldsymbol{g}_t^{\top}(\boldsymbol{w}_0 - \boldsymbol{w}_t)}{\|\boldsymbol{g}_t\|_1}\right), \tag{28}$$

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta_t \cdot \operatorname{sign}(\boldsymbol{g}_t). \tag{29}$$

But Proposition 2 showed that sign descent is steepest descent under the infinity norm. Therefore Equations (28) and (29) prove our claim that Prodigy without EMA is steepest descent, although with a dynamically chosen step size denoted η_t.

All that remains is to understand the dynamical rule, given by Equation (28), for choosing the step size η_t. We shall argue that this dynamical rule can be understood to approximate a heuristic algorithm for achieving, but not exceeding, what we shall call *escape velocity*:

- Choose a very small initial step size η_0—small enough to be *a priori* sure that $\eta_0 \ll \eta_\star$, where $\eta_\star$ denotes *escape velocity*: the unknown but optimal initial step size;
- At each step, check if the weights $\boldsymbol{w}_t$ have escaped the linearization of the loss around the initial weights $\boldsymbol{w}_0$—if not, double the step size according to $\eta_{t+1} = 2 \times \eta_t$;
- Once the weights $\boldsymbol{w}_t$ have escaped the initial linearization, stop increasing the step size. We say that the step size η_t has reached escape velocity $\eta_\star$.

The rationale behind this procedure is that if we knew the optimal initial step size $\eta_\star$, then the weights should escape the initial linearization of the loss in a single step. Formally, the directional derivative $(\boldsymbol{w}_1 - \boldsymbol{w}_0)^\top \boldsymbol{g}_1$ must vanish if the step size is chosen optimally (Cauchy, 1847). If the directional derivative in the direction of the first weight update is still negative $(\boldsymbol{w}_1 - \boldsymbol{w}_0)^\top \boldsymbol{g}_1 < 0$, then we could have taken a larger step. Said another way, we can use the *angle* that the gradient $\boldsymbol{g}_1$ makes with the change in weights $\boldsymbol{w}_1 - \boldsymbol{w}_0$ to tell us whether or not we should increase the step size. Notice that procedure has no reliance on convexity.

With this in mind, let us massage Prodigy's step size update (Equation (28)) as follows:

$$\eta_{t+1} = \max\left(\eta_t, \frac{\boldsymbol{g}_t^\top(\boldsymbol{w}_0 - \boldsymbol{w}_t)}{\|\boldsymbol{g}_t\|_1}\right) = \max\left(\eta_t, \frac{\|\boldsymbol{g}_t\|_2}{\|\boldsymbol{g}_t\|_1} \times \|\boldsymbol{w}_t - \boldsymbol{w}_0\|_2 \times \cos\theta\right), \tag{30}$$

where θ denotes the angle between the gradient $\boldsymbol{g}_t$ and the difference in weights $\boldsymbol{w}_0 - \boldsymbol{w}_t$. To help make sense of this expression, we make two assumptions:

1. The gradient is a "dense" vector in $\mathbb{R}^n$, meaning that $\|\boldsymbol{g}_t\|_2/\|\boldsymbol{g}_t\|_1 \approx 1/\sqrt{n}$;
2. $\boldsymbol{w}_t$ is still close enough to the initialization $\boldsymbol{w}_0$ that $\cos\theta \approx 1$.

Under these assumptions, Equation (30) becomes just $\eta_{t+1} \approx \max(\eta_t, \|\boldsymbol{w}_t - \boldsymbol{w}_0\|_{\text{RMS}})$, where the root mean square (RMS) norm is defined via $\|\cdot\|_{\text{RMS}} := \frac{1}{\sqrt{n}}\|\cdot\|_2$. Combined with Equation (29), this allows us to estimate the size of the weight change at step $t+1$:

$$\|\boldsymbol{w}_{t+2} - \boldsymbol{w}_{t+1}\|_{\text{RMS}} = \eta_{t+1} \cdot \|\text{sign}(\boldsymbol{g}_t)\|_{\text{RMS}} \approx \max(\eta_t, \|\boldsymbol{w}_t - \boldsymbol{w}_0\|_{\text{RMS}}) \geqslant \|\boldsymbol{w}_t - \boldsymbol{w}_0\|_{\text{RMS}},$$

where we have used the fact that a sign vector has unit RMS norm. In words, while assumptions (1) and (2) hold, the step size at time $t+1$ is equivalent to the whole progress up to step t. This suggests exponential growth in the step size that continues until assumption (2) breaks, which we think of as the step size reaching the escape velocity $\eta_\star$.

Now we wish to point out that this procedure is just one amongst a space of line search methods that one might consider (Armijo, 1966; Riedmiller and Braun, 1993; Kenneweg et al., 2024). For instance, Prodigy's decision to only let η_t increase and never decrease could be sub-optimal. And the decision to measure the angle between the gradient and the weight difference $\boldsymbol{w}_t - \boldsymbol{w}_0$ has alternatives. One could instead use the most recent weight difference $\boldsymbol{w}_t - \boldsymbol{w}_{t-1}$. Lastly, in place of relying on the norm ratio $\|\boldsymbol{g}\|_2/\|\boldsymbol{g}_1\|$ to implicitly convert the ℓ_2 norm $\|\boldsymbol{w}_t - \boldsymbol{w}_0\|_2$ into the RMS norm $\|\boldsymbol{w}_t - \boldsymbol{w}_0\|_{\text{RMS}}$, one could consider a more explicit method. For instance, we found a rule akin to $\eta_{t+1} = \eta_t \times (1 + \cos\theta)$ to work well in some preliminary experiments.

Our time grows short, dear reader, and our third story draws to an end. We have argued that Prodigy without EMA is sign descent—an example of steepest descent—with a particular mechanism for warming up the step size. Starting with a tiny initial step size, Prodigy multiplicatively increases the step size until the weights escape the initial locally linear region of the loss. Prodigy's step size adjustment is based on the angle between the gradient and the total weight change. This is a form of online line search. This highlights that once one has chosen a norm, the steepest descent framework allows freedom to estimate the step size in various different ways.

Epilogue

This anthology has presented new ways of understanding old optimizers. Proposition 1 decouples the optimizer design problem into two pieces: first choosing a norm and second finding a step size. This design space is already broad. We have argued that Adam chooses the infinity norm (Proposition 2) or equivalently the max-of-max norm (Proposition 3), which respects a layered matrix structure. Shampoo chooses the spectral norm (Proposition 5). Prodigy chooses the same norm as Adam, and then uses a heuristic to automatically warm up to a good step size, as in Equation (28), which we term *reaching escape velocity*.

Through the lens of steepest descent, the decisions that Adam, Shampoo and Prodigy make may seem arbitrary. In fact, we think that they *are* somewhat arbitrary. And there may be more principled ways to make these decisions. To demonstrate this point, we now introduce a tool called the modular norm (Large et al., 2024) and its corresponding steepest descent algorithm. The modular norm generalizes the norms that appeared in Proposition 3 for Adam and Proposition 5 for Shampoo. Formally:

Proposition 7 (Steepest descent under the modular norm) *Given scalar coefficients $s_1, \ldots, s_L > 0$ and norms $\|\cdot\|_1, \ldots, \|\cdot\|_L$, we define the modular norm as the mapping:*

$$\boldsymbol{W}_1, \ldots, \boldsymbol{W}_L \mapsto \max\left\{s_1\|\boldsymbol{W}_1\|_1, \ldots, s_L\|\boldsymbol{W}_L\|_L\right\}. \tag{31}$$

The corresponding steepest descent problem is given by:

$$\underset{\Delta\boldsymbol{W}_1,\ldots,\Delta\boldsymbol{W}_L}{\arg\min}\left[\sum_{l=1}^{L}\langle\boldsymbol{G}_l, \Delta\boldsymbol{W}_l\rangle + \frac{\lambda}{2}\max_{l=1}^{L} s_l^2\|\Delta\boldsymbol{W}_l\|_l^2\right], \tag{32}$$

where $\langle\cdot,\cdot\rangle$ denotes the Frobenius inner product, and for each $l = 1, ..., L$ the two matrices $\Delta\boldsymbol{W}_l$ and $\boldsymbol{G}_l$ are of the same shape. If we define the global step size $\eta = \frac{1}{\lambda}\sum_{k=1}^{L}\frac{1}{s_k}\|\boldsymbol{G}_k\|_k^\dagger$, then the solution to Equation (32) is given by:

$$\Delta\boldsymbol{W}_l = -\frac{\eta}{s_l}\cdot\underset{\|\boldsymbol{T}_l\|_l=1}{\arg\max}\langle\boldsymbol{G}_l, \boldsymbol{T}_l\rangle \quad \text{for each layer } l = 1, ..., L. \tag{33}$$

In words, steepest descent under the modular norm updates each layer in a direction informed by that layer's norm and with a global step size computed as a weighted sum of the dual norms of the gradients over layers. The proof of this proposition is given in Appendix B.

When confronted with the modular norm, it's natural to ask how one should assign norms to layers. And there are so many norms to choose from! Beyond the familiar $\ell_2 \to \ell_2$ spectral norm, many other induced operator norms are computationally tractable:

Proposition 8 ($\ell_1 \to \ell_p$ and $\ell_p \to \ell_\infty$ induced operator norms are tractable) *For a matrix $\boldsymbol{M} \in \mathbb{R}^{m\times n}$ with m rows $\{\text{row}_i(\boldsymbol{M})\}_{i=1}^m$ and n columns $\{\text{col}_j(\boldsymbol{M})\}_{j=1}^n$, and $1 \leqslant p \leqslant \infty$:*

$$\|\boldsymbol{M}\|_{\ell_1\to\ell_p} = \max_j\|\text{col}_j(\boldsymbol{M})\|_p; \qquad \|\boldsymbol{M}\|_{\ell_p\to\ell_\infty} = \max_i\|\text{row}_i(\boldsymbol{M})\|_{\frac{p}{p-1}}. \tag{34}$$

In words, the $\ell_1 \to \ell_p$ operator norm is the largest ℓ_p norm of the columns; the $\ell_p \to \ell_\infty$ operator norm is the largest dual ℓ_p norm over the rows. The proof is given in Appendix B.

To assign a norm to a layer, we believe that one should consider the role that layer plays in the neural network. For instance, since linear layers are typically used to map to and from vectors with roughly unit RMS norm, it is appropriate to equip linear layers with the induced RMS to RMS operator norm (Yang et al., 2023), which resolves to a rescaled spectral norm. And since embedding layers map from one-hot vectors to vectors with roughly unit RMS norm, it is appropriate to equip embedding layers with the ℓ_1 to RMS operator norm, which resolves to a rescaled ℓ_1 to ℓ_2 operator norm. So embedding layers and linear layers should be equipped with different norms despite the weight space being a matrix space in both cases. In short, the algorithm designer has freedom to choose input and output norms for layers that capture differences in how the layers are used; inducing the corresponding operator norm on the layer's weights provides control over how the optimizer learns representations.

We believe that picking the right norms could improve the speed and scalability of neural network training. We are seeing evidence that equipping neural network layers with better norms can lead to learning rate transfer across scale (Yang et al., 2023; Large et al., 2024). And since Shampoo won the external tuning track of the 2024 AlgoPerf competition (Dahl et al., 2023), it is garnering interest as a fast training method. The second story in our anthology shows that Shampoo is closely connected to the spectral norm.

In conclusion, this work highlights a perspective on optimizer design as choosing two things: a *norm* and a *step size*. We have shown that three popular methods—Adam, Shampoo and Prodigy—fit within this perspective. We hope that researchers can design improved training algorithms by choosing norms and step sizes more intentionally.

"Though this be madness, yet there is method in't."
Hamlet

Acknowledgements

We are grateful to Tim Large and Phillip Isola for invaluable discussions on the stories in this anthology. We also thank Jack Gallagher, Keller Jordan, Tongzhou Wang and Victor Butoi for very helpful conversations.

References

Rohan Anil, Vineet Gupta, Tomer Koren, Kevin Regan, and Yoram Singer. Scalable second order optimization for deep learning. *arXiv:2002.09018*, 2020. Cited on pages 5 and 15.

Larry Armijo. Minimization of functions having Lipschitz continuous first partial derivatives. *Pacific Journal of Mathematics*, 1966. Cited on page 9.

Lukas Balles and Philipp Hennig. Dissecting Adam: The sign, magnitude and variance of stochastic gradients. In *International Conference on Machine Learning*, 2018. Cited on page 3.

Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. signSGD: Compressed optimisation for non-convex problems. In *International Conference on Machine Learning*, 2018. Cited on page 3.

Jeremy Bernstein, Chris Mingard, Kevin Huang, Navid Azizan, and Yisong Yue. Automatic Gradient Descent: Deep Learning without Hyperparameters. *arXiv:2304.05187*, 2023. Cited on page 6.

Åke Björck and C. Bowie. An iterative algorithm for computing the best estimate of an orthogonal matrix. *SIAM Journal on Numerical Analysis*, 1971. Cited on page 15.

David Carlson, Volkan Cevher, and Lawrence Carin. Stochastic spectral descent for Restricted Boltzmann Machines. In *International Conference on Artificial Intelligence and Statistics*, 2015a. Cited on pages 3, 6, and 7.

David Carlson, Edo Collins, Ya-Ping Hsieh, Lawrence Carin, and Volkan Cevher. Preconditioned spectral descent for deep learning. In *Neural Information Processing Systems*, 2015b. Cited on page 15.

David Carlson, Ya-Ping Hsieh, Edo Collins, Lawrence Carin, and Volkan Cevher. Stochastic spectral descent for discrete graphical models. *Selected Topics in Signal Processing*, 2016. Cited on pages 3, 6, and 7.

Augustin-Louis Cauchy. Méthode générale pour la résolution des systèmes d'équations simultanées. *Comptes Rendus Hebdomadaires des Séances de l'Académie des Sciences*, 1847. Cited on page 9.

Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, and Quoc V Le. Symbolic discovery of optimization algorithms. In *Neural Information Processing Systems*, 2023. Cited on page 4.

George E. Dahl, Frank Schneider, Zachary Nado, Naman Agarwal, Chandramouli Shama Sastry, Philipp Hennig, Sourabh Medapati, Runa Eschenhagen, Priya Kasimbeg, Daniel Suo, Juhan Bae, Justin Gilmer, Abel L. Peirson, Bilal Khan, Rohan Anil, Mike Rabbat, Shankar Krishnan, Daniel Snider, Ehsan Amid, Kongtao Chen, Chris J. Maddison, Rakshith Vasudev, Michal Badura, Ankush Garg, and Peter Mattson. Benchmarking neural network training algorithms. *arXiv:2306.07179*, 2023. Cited on pages 5 and 11.

Aaron Defazio and Konstantin Mishchenko. Learning-rate-free learning by D-adaptation. In *International Conference on Machine Learning*, 2023. Cited on page 8.

John C. Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal Machine Learning Research*, 2011. Cited on page 5.

Kai Fan. Unifying the stochastic spectral descent for Restricted Boltzmann Machines with Bernoulli or Gaussian inputs. *arXiv:1703.09766*, 2017. Cited on page 7.

Vladimir Feinberg, Xinyi Chen, Y. Jennifer Sun, Rohan Anil, and Elad Hazan. Sketchy: Memory-efficient adaptive regularization with frequent directions. In *Neural Information Processing Systems*, 2023. Cited on page 15.

Vineet Gupta, Tomer Koren, and Yoram Singer. A unified approach to adaptive regularization in online and stochastic optimization. Technical report, Google Brain, 2017. Cited on page 5.

Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In *International Conference on Machine Learning*, 2018. Cited on page 5.

Nicholas J. Higham. *Functions of Matrices*. Society for Industrial and Applied Mathematics, 2008. Cited on page 15.

Maor Ivgi, Oliver Hinder, and Yair Carmon. DoG is SGD's best friend: A parameter-free dynamic step size schedule. In *International Conference on Machine Learning*, 2023. Cited on page 8.

Philip Kenneweg, Tristan Kenneweg, and Barbara Hammer. Improving line search methods for large scale neural network training. In *International Conference on Artificial Intelligence, Computer, Data Sciences and Applications*, 2024. Cited on page 9.

Ahmed Khaled, Konstantin Mishchenko, and Chi Jin. DoWG unleashed: An efficient universal parameter-free gradient descent method. In *Neural Information Processing Systems*, 2023. Cited on page 8.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. Cited on page 3.

Zdislav Kovarik. Some iterative methods for improving orthonormality. *SIAM Journal on Numerical Analysis*, 1970. Cited on page 15.

Slobodan Lakić. On the computation of the matrix k-th root. *Journal of Applied Mathematics and Mechanics*, 1998. Cited on page 15.

Kenneth Lange. *MM Optimization Algorithms*. Society for Industrial and Applied Mathematics, 2016. Cited on page 6.

Tim Large, Yang Liu, Minyoung Huh, Hyojin Bahng, Phillip Isola, and Jeremy Bernstein. Scalable optimization in the modular norm. *arXiv:2405.14813*, 2024. Cited on pages 2, 4, 6, 10, and 11.

Per-Gunnar Martinsson and Joel A. Tropp. Randomized numerical linear algebra: Foundations and algorithms. *Acta Numerica*, 2020. Cited on page 15.

Konstantin Mishchenko and Aaron Defazio. Prodigy: An expeditiously adaptive parameter-free learner. *arXiv:2306.06101*, 2023. Cited on page 8.

Depen Morwani, Itai Shapira, Nikhil Vyas, Eran Malach, Sham Kakade, and Lucas Janson. A new perspective on Shampoo's preconditioner. *arXiv:2406.17748*, 2024. Cited on page 5.

Martin Riedmiller and Heinrich Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *International Conference on Neural Networks*, 1993. Cited on pages 3 and 9.

Hao-Jun Michael Shi, Tsung-Hsien Lee, Shintaro Iwasaki, Jose Gallego-Posada, Zhijing Li, Kaushik Rangadurai, Dheevatsa Mudigere, and Michael Rabbat. A distributed data-parallel PyTorch implementation of the distributed Shampoo optimizer for training neural networks at-scale. *arXiv:2309.06497*, 2023. Cited on page 5.

Matthew Streeter. Universal majorization-minimization algorithms. *arXiv:2308.00190*, 2023. Cited on page 6.

Shiqing Sun and James C. Spall. Connection of diagonal Hessian estimates to natural gradients in stochastic optimization. In *Information Sciences and Systems*, 2021. Cited on page 3.

Tijmen Tieleman and Geoffrey Hinton. RMSprop. *Coursera: Neural Networks for Machine Learning*, Lecture 6.5, 2012. Cited on page 3.

Shuo Xie and Zhiyuan Li. Implicit bias of AdamW: ℓ_∞-norm constrained optimization. In *International Conference on Machine Learning*, 2024. Cited on page 3.

Greg Yang, James B. Simon, and Jeremy Bernstein. A spectral condition for feature learning. *arXiv:2310.17813*, 2023. Cited on page 11.

Rosie Zhao, Depen Morwani, David Brandfonbrener, Nikhil Vyas, and Sham Kakade. Deconstructing what makes a good optimizer for language models. *arXiv:2407.07972*, 2024. Cited on page 4.

Appendix A. Computational Strategies for Shampoo

Let $\boldsymbol{G} \in \mathbb{R}^{m \times n}$ be a gradient matrix with reduced SVD $\boldsymbol{G} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$. By Equations (15) and (16), the corresponding Shampoo update (with EMA disabled) is given by:

$$\Delta \boldsymbol{W} = -\eta \cdot (\boldsymbol{G}\boldsymbol{G}^\top)^{-1/4} \boldsymbol{G} (\boldsymbol{G}^\top \boldsymbol{G})^{-1/4} = -\eta \cdot \boldsymbol{U}\boldsymbol{V}^\top. \tag{35}$$

Here we list every means we know of computing or approximating this equation. First, we mention that $(\boldsymbol{G}\boldsymbol{G}^\top)^{-1/4} \boldsymbol{G} (\boldsymbol{G}^\top \boldsymbol{G})^{-1/4} = (\boldsymbol{G}\boldsymbol{G}^\top)^{-1/2} \boldsymbol{G} = \boldsymbol{G} (\boldsymbol{G}^\top \boldsymbol{G})^{-1/2}$, so if one is willing to compute inverse matrix roots, one need only compute either $(\boldsymbol{G}\boldsymbol{G}^\top)^{-1/2}$ or $(\boldsymbol{G}^\top \boldsymbol{G})^{-1/2}$, whichever has smaller dimension. With that said, to compute Equation (35), one may:

1. **Do the SVD.** Apply an SVD routine to compute $\boldsymbol{U}$, $\boldsymbol{\Sigma}$ and $\boldsymbol{V}^\top$ and just discard $\boldsymbol{\Sigma}$.

2. **Do sketching.** Sketching is a randomized method (Martinsson and Tropp, 2020) that can be used to approximate the SVD. See, for instance, Sketchy (Feinberg et al., 2023) and spectral descent for deep learning (Carlson et al., 2015b).

3. **Do Newton iteration for inverse pth roots.** Inverse matrix roots such as $(\boldsymbol{G}\boldsymbol{G}^\top)^{-1/2}$ can be computed via Newton iteration (Lakić, 1998). This is discussed in Chapter 7 of Higham (2008)'s book. And see Anil et al. (2020)'s paper.

4. **Do Newton-Schulz iteration.** We developed a "Newton-Schulz iteration" for computing $\boldsymbol{U}\boldsymbol{V}^\top$, adapted from Equation 5.22 in Higham (2008)'s book. In short, if we set $\boldsymbol{X}_0 = \boldsymbol{G}/\|\boldsymbol{G}\|_{\ell_2 \to \ell_2}$ (or alternatively $\boldsymbol{X}_0 = \boldsymbol{G}/\|\boldsymbol{G}\|_F$) and iterate:

$$\boldsymbol{X}_{t+1} = \frac{3}{2} \cdot \boldsymbol{X}_t - \frac{1}{2} \cdot \boldsymbol{X}_t \boldsymbol{X}_t^\top \boldsymbol{X}_t, \tag{36}$$

   then as $t \to \infty$, the sequence $\boldsymbol{X}_t \to \boldsymbol{U}\boldsymbol{V}^\top$. To see this, one should plot the univariate cubic function $f(x) := \frac{3}{2} \cdot x - \frac{1}{2} \cdot x^3$ and see that, for $0 < x < \sqrt{3}$, iterating this cubic will push x closer and closer to $+1$. The final step is to realize that the effect of the iteration in Equation (36) is to apply this cubic $f(x)$ to each singular value of $\boldsymbol{X}_t$. This also shows that the spectral normalization $\boldsymbol{X}_0 = \boldsymbol{G}/\|\boldsymbol{G}\|_{\ell_2 \to \ell_2}$ is stronger than what is required: we need only ensure that $\boldsymbol{X}_0$ has all singular values greater than zero and less than $\sqrt{3}$ in order for the iteration to converge.

   There are in fact a family of degree $2n+1$ polynomial iterations of the form

$$\boldsymbol{X}_{t+1} = a \cdot \boldsymbol{X}_t + b \cdot \boldsymbol{X}_t \boldsymbol{X}_t^\top \boldsymbol{X}_t + c \cdot (\boldsymbol{X}_t \boldsymbol{X}_t^\top)^2 \boldsymbol{X}_t + \ldots + z \cdot (\boldsymbol{X}_t \boldsymbol{X}_t^\top)^n \boldsymbol{X}_t \tag{37}$$

   for suitable $a, b, c, \ldots, z$ that could be used instead of Equation (36). One should choose coefficients $a, b, c, \ldots, z$ so that the univariate polynomial $g(x) = a \cdot x + b \cdot x^3 + c \cdot x^5 + \ldots + z \cdot x^{2n+1}$ is a suitable approximation to $\operatorname{sign}(x)$. The coefficients can be tuned graphically to achieve the fastest convergence.

   After posting the first version of this paper on arXiv, we found out that the iteration, at least for fixed coefficients, is classical (Kovarik, 1970; Björck and Bowie, 1971).

Which of these methods is most useful in practice may depend on factors such as the condition number of the matrix $\boldsymbol{G}$ or the nature of the available computational resources.

Appendix B. Proofs

Proposition 1: Steepest descent

First, let's study the minimization under the change of variables $\Delta \boldsymbol{w} = c \cdot \boldsymbol{t}$, where $c \geqslant 0$ encodes the "magnitude" and $\boldsymbol{t}$ is a unit vector ($\|\boldsymbol{t}\| = 1$) encoding the "direction":

$$\min_{\Delta \boldsymbol{w} \in \mathbb{R}^n} \left[\boldsymbol{g}^\top \Delta \boldsymbol{w} + \frac{\lambda}{2} \|\Delta \boldsymbol{w}\|^2 \right] = \min_{c \geqslant 0} \min_{\boldsymbol{t} \in \mathbb{R}^n : \|\boldsymbol{t}\| = 1} \left[c \cdot \boldsymbol{g}^\top \boldsymbol{t} + \frac{\lambda}{2} c^2 \|\boldsymbol{t}\|^2 \right] \tag{38}$$

$$= \min_{c \geqslant 0} \left[c \cdot \min_{\boldsymbol{t} \in \mathbb{R}^n : \|\boldsymbol{t}\| = 1} \left[\boldsymbol{g}^\top \boldsymbol{t} \right] + \frac{\lambda}{2} c^2 \right] \tag{39}$$

$$= \min_{c \geqslant 0} \left[-c \cdot \|\boldsymbol{g}\|^\dagger + \frac{\lambda}{2} c^2 \right], \tag{40}$$

Inspecting Equation (39), we see that the minimizer for the direction $\boldsymbol{t}$ is given by:

$$\boldsymbol{t} = \arg\min_{\boldsymbol{t} \in \mathbb{R}^n : \|\boldsymbol{t}\| = 1} \left[\boldsymbol{g}^\top \boldsymbol{t} \right] = - \arg\max_{\boldsymbol{t} \in \mathbb{R}^n : \|\boldsymbol{t}\| = 1} \left[\boldsymbol{g}^\top \boldsymbol{t} \right] \tag{41}$$

And similarly, by inspecting Equation (40), the minimizer for the magnitude c is given by:

$$c = \arg\min_{c \geqslant 0} \left[-c \cdot \|\boldsymbol{g}\|^\dagger + \frac{\lambda}{2} c^2 \right] = \frac{\|\boldsymbol{g}\|^\dagger}{\lambda}. \tag{42}$$

Multiplying these expressions, we obtain the minimizer for $\Delta \boldsymbol{w}$, yielding the result. ■

Proposition 2: Sign descent as steepest descent under the infinity norm

The result follows by applying Proposition 1. We just need that $\arg\max_{\|\boldsymbol{t}\|_\infty = 1} \boldsymbol{g}^\top \boldsymbol{t} = \text{sign}(\boldsymbol{g})$, and also that the dual norm $\|\boldsymbol{g}\|_\infty^\dagger := \max_{\|\boldsymbol{t}\|_\infty = 1} \boldsymbol{g}^\top \boldsymbol{t} = \boldsymbol{g}^\top \text{sign}(\boldsymbol{g}) = \|\boldsymbol{g}\|_1$. ■

Proposition 3: Sign descent as steepest descent under the max-of-max norm

The result follows from Proposition 7 by setting all the scalars $s_1, ..., s_L$ to one and all the norms $\|\cdot\|_1, ..., \|\cdot\|_L$ to the ℓ_1 to ℓ_∞ operator norm. All we need is to show that the argmax at each matrix space $l = 1, ..., L$ satisfies:

$$\arg\max_{\|\boldsymbol{T}_l\|_{\ell_1 \to \ell_\infty} = 1} \text{tr}(\boldsymbol{G}_l^\top \boldsymbol{T}_l) = \text{sign}(\boldsymbol{G}_l). \tag{43}$$

But this holds because, by Proposition 8, $\|\boldsymbol{T}\|_{\ell_1 \to \ell_\infty} = \max_i \|\text{col}_i(\boldsymbol{T})\|_\infty = \max_{ij} |\boldsymbol{T}_{ij}|$, and therefore all components in the argmax must be of unit size and gradient aligned. ■

Proposition 4: Projection to the closest semi-orthogonal matrix

To begin, we observe that the minimizer over semi-orthogonal matrices of the "distance" $\|\boldsymbol{A}-\boldsymbol{G}\|_F$ is the same as the maximizer over semi-orthogonal matrices of the "alignment" $\langle \boldsymbol{A}, \boldsymbol{G}\rangle$, where $\langle\cdot,\cdot\rangle$ denotes the Frobenius inner product. This is because:

$$\|\boldsymbol{A}-\boldsymbol{G}\|_F^2 = \|\boldsymbol{A}\|_F^2 - 2\cdot\langle \boldsymbol{A}, \boldsymbol{G}\rangle + \|\boldsymbol{G}\|_F^2, \tag{44}$$

and the term $\|\boldsymbol{A}\|_F^2$ is fixed at $\|\boldsymbol{A}\|_F^2 = \min(m,n)$ for a semi-orthogonal matrix $\boldsymbol{A}\in\mathcal{O}_{m\times n}$.

Now, let $\boldsymbol{G}=\sum_i \sigma_i\,\boldsymbol{u}_i\boldsymbol{v}_i^\top$ denote the SVD of $\boldsymbol{G}$. Then the alignment satisfies:

$$\langle \boldsymbol{A}, \boldsymbol{G}\rangle = \operatorname{tr}\sum_i \sigma_i\,\boldsymbol{v}_i\boldsymbol{u}_i^\top\boldsymbol{A} = \sum_i \sigma_i\,\boldsymbol{u}_i^\top\boldsymbol{A}\boldsymbol{v}_i \leqslant \sum_i \sigma_i, \tag{45}$$

where the second equality follows by the cyclic property of the trace, and the inequality is since $\boldsymbol{A}$ being semi-orthogonal means that $\boldsymbol{u}^\top\boldsymbol{A}\boldsymbol{v}\leqslant 1$ for any two unit vectors $\boldsymbol{u}$ and $\boldsymbol{v}$.

Next, observe that for the semi-orthogonal matrix $\boldsymbol{A}_\star=\sum_i \boldsymbol{u}_i\boldsymbol{v}_i^\top$, we have that:

$$\langle \boldsymbol{A}_\star, \boldsymbol{G}\rangle = \sum_i \sigma_i \sum_j \boldsymbol{u}_i^\top\boldsymbol{u}_j\boldsymbol{v}_j^\top\boldsymbol{v}_i = \sum_i \sigma_i, \tag{46}$$

since the $\{\boldsymbol{u}_i\}$ and $\{\boldsymbol{v}_i\}$ are orthonormal. Comparing against Equation (45), we see that $\boldsymbol{A}_\star$ indeed maximizes the alignment, since it achieves the upper bound of $\sum_i\sigma_i$. And $\boldsymbol{A}_\star$ therefore also minimizes the distance $\|\boldsymbol{A}-\boldsymbol{G}\|_F$ amongst semi-orthogonal matrices $\boldsymbol{A}$. Note that if $\boldsymbol{U}$ is the matrix that has the $\{\boldsymbol{u}_i\}$ as columns, and likewise for $\boldsymbol{V}$ and the $\{\boldsymbol{v}_i\}$, then this solution may equivalently be expressed as $\boldsymbol{A}_\star=\boldsymbol{U}\boldsymbol{V}^\top$.

All that remains is to explore the uniqueness of this solution:

- If $\boldsymbol{G}$ is full rank, the solution $\boldsymbol{A}_\star$ is unique. $\boldsymbol{G}$ being full rank means that all the singular values σ_i are positive. In this case, we see from Equation (45) that to maximize the alignment the semi-orthogonal matrix $\boldsymbol{A}$ must satisfy $\boldsymbol{u}_i^\top\boldsymbol{A}\boldsymbol{v}_i=1$ for all i. Since $\boldsymbol{A}$ has spectral norm one, in turn this requires that $\boldsymbol{A}\boldsymbol{v}_i=\boldsymbol{u}_i$ and $\boldsymbol{A}^\top\boldsymbol{u}_i=\boldsymbol{v}_i$ for all i. These conditions uniquely pick out $\boldsymbol{A}=\sum_i\boldsymbol{u}_i\boldsymbol{v}_i^\top$.
- If $\boldsymbol{G}$ is not full rank then the solution $\boldsymbol{A}_\star$ is not unique. This solution is just as good:

$$\boldsymbol{A}_\dagger = \sum_{i:\sigma_i>0}\boldsymbol{u}_i\boldsymbol{v}_i^\top + \sum_{i:\sigma_i=0}\boldsymbol{u}_i(-\boldsymbol{v}_i)^\top. \tag{47}$$

This completes the proof. ■

Proposition 5: Shampoo as steepest descent under the spectral norm

First, we apply Proposition 7 with scalars $s_1,\ldots,s_L$ set to one and all norms set to $\|\cdot\|_{\ell_2\to\ell_2}$. This tells us that the solution is given by $\Delta\boldsymbol{W}_l = -\eta\cdot\arg\max_{\|\boldsymbol{T}_l\|_l=1}\operatorname{tr}(\boldsymbol{G}_l^\top\boldsymbol{T}_l)$ for each $l=1,\ldots,L$ and with $\eta=\frac{1}{\lambda}\sum_{k=1}^L\|\boldsymbol{G}_k\|^\dagger_{\ell_2\to\ell_2}$. We just need to resolve the dual norm and evaluate the argmax.

Let's start with the dual norm. For a matrix $\boldsymbol{G}$ with SVD $\sum_i \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^\top = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ we have:

$$\|\boldsymbol{G}\|^\dagger_{\ell_2\to\ell_2} := \max_{\|\boldsymbol{T}\|_{\ell_2\to\ell_2}=1} \operatorname{tr} \boldsymbol{G}^\top \boldsymbol{T} = \max_{\|\boldsymbol{T}\|_{\ell_2\to\ell_2}=1} \operatorname{tr} \sum_i \sigma_i \boldsymbol{v}_i \boldsymbol{u}_i^\top \boldsymbol{T} \tag{48}$$

$$= \max_{\|\boldsymbol{T}\|_{\ell_2\to\ell_2}=1} \sum_i \sigma_i \boldsymbol{u}_i^\top \boldsymbol{T} \boldsymbol{v}_i \leqslant \sum_i \sigma_i = \operatorname{tr} \boldsymbol{\Sigma}, \tag{49}$$

where the upper bound follows from the spectral norm constraint on $\boldsymbol{T}$. But this upper bound is attained by setting $\boldsymbol{T} = \boldsymbol{U}\boldsymbol{V}^\top$ (also resolving the argmax) and so $\|\boldsymbol{G}\|^\dagger_{\ell_2\to\ell_2} = \operatorname{tr} \boldsymbol{\Sigma}$.

The uniqueness claim follows by the same argument as for Proposition 4. ■

Proposition 6: Bounding the square loss of a linear predictor

First observe that the square loss is quadratic in $\boldsymbol{W}$ so there are no cubic terms or higher. The bound must agree to first-order with the first-order Taylor expansion of $\mathcal{L}(\boldsymbol{W} + \Delta\boldsymbol{W})$, which is precisely $\mathcal{L}(\boldsymbol{W}) + \langle \nabla_{\boldsymbol{W}} \mathcal{L}(\boldsymbol{W}), \Delta\boldsymbol{W} \rangle$, since otherwise the bound would be violated for sufficiently small $\Delta\boldsymbol{W}$. To obtain the second-order piece of the bound, it's easiest just to multiply out $\mathcal{L}(\boldsymbol{W} + \Delta\boldsymbol{W})$ and see that the second-order piece of $\mathcal{L}(\boldsymbol{W} + \Delta\boldsymbol{W})$ satisfies:

$$\frac{1}{2n} \sum_{i=1}^n \frac{1}{d_{\text{out}}} \|\Delta\boldsymbol{W}\boldsymbol{x}^{(i)}\|_2^2 \leqslant \frac{1}{2n} \sum_{i=1}^n \frac{1}{d_{\text{out}}} \|\Delta\boldsymbol{W}\|^2_{\ell_2\to\ell_2} \cdot \|\boldsymbol{x}^{(i)}\|_2^2 = \frac{1}{2} \frac{d_{\text{in}}}{d_{\text{out}}} \|\Delta\boldsymbol{W}\|^2_{\ell_2\to\ell_2}, \tag{50}$$

where the last equality uses the input normalization $\|\boldsymbol{x}^{(i)}\|_2 = \sqrt{d_{\text{in}}}$. We are done. ■

Proposition 7: Steepest descent under the modular norm

For each layer $l = 1, ..., L$, we decompose $\Delta\boldsymbol{W}_l$ into its magnitude and direction: $\Delta\boldsymbol{W}_l = c_l \cdot \boldsymbol{T}_l$, for $c_l \geqslant 0$ and $\|\boldsymbol{T}_l\|_l = 1$. Under this change of variables, the minimization becomes:

$$\min_{\Delta\boldsymbol{W}_1,\ldots,\Delta\boldsymbol{W}_L} \left[\sum_{l=1}^L \langle \boldsymbol{G}_l, \Delta\boldsymbol{W}_l \rangle + \frac{\lambda}{2} \max_{l=1}^L s_l^2 \|\Delta\boldsymbol{W}_l\|_l^2 \right] \tag{51}$$

$$= \min_{c_1,\ldots,c_L \geqslant 0} \left[\sum_{l=1}^L c_l \min_{\|\boldsymbol{T}_l\|_l=1} \langle \boldsymbol{G}_l, \boldsymbol{T}_l \rangle + \frac{\lambda}{2} \max_{l=1}^L s_l^2 c_l^2 \right] \tag{52}$$

$$= \min_{c_1,\ldots,c_L \geqslant 0} \left[-\sum_{l=1}^L c_l \|\boldsymbol{G}_l\|_l^\dagger + \frac{\lambda}{2} \max_{l=1}^L s_l^2 c_l^2 \right] \tag{53}$$

$$= \min_{\eta \geqslant 0} \left[-\sum_{l=1}^L \frac{\eta}{s_l} \|\boldsymbol{G}_l\|_l^\dagger + \frac{\lambda}{2} \eta^2 \right], \tag{54}$$

where Equation (54) follows by observing that at the minimum we must have $s_1 c_1, ..., s_L c_L$ all taking the same value of $\eta \geqslant 0$ (still to be determined), since otherwise we could increase the sum $\sum_l c_l \|\boldsymbol{G}_l\|_l^\dagger$ by increasing any of the slack c_l without paying a penalty in terms of

the max. We can now read off the minimizers from Equations (52) to (54):

$$\boldsymbol{T}_l = \arg\min_{\|\boldsymbol{T}_l\|_l=1} \langle \boldsymbol{G}_l, \boldsymbol{T}_l \rangle = -\arg\max_{\|\boldsymbol{T}_l\|_l=1} \langle \boldsymbol{G}_l, \boldsymbol{T}_l \rangle; \tag{55}$$

$$c_l = \frac{\eta}{s_l}; \tag{56}$$

$$\eta = \frac{1}{\lambda} \sum_{k=1}^{L} \frac{1}{s_k} \|\boldsymbol{G}_k\|_k^\dagger. \tag{57}$$

Combining, we obtain the overall minimizer for each $l = 1, ..., L$ via $\Delta \boldsymbol{W}_l = c_l \cdot \boldsymbol{T}_l = -\frac{\eta}{s_l} \arg\max \langle \boldsymbol{G}_l, \boldsymbol{T}_l \rangle$, where η is given by Equation (57), proving the result. ■

Proposition 8: $\ell_1 \to \ell_p$ and $\ell_p \to \ell_\infty$ induced operator norms are tractable

Let's start with the $\ell_1 \to \ell_p$ operator norm. Here we observe that, in matrix-vector multiplication, each component of an input vector selects and scales a column of the matrix:

$$\|\boldsymbol{M}\|_{\ell_1 \to \ell_p} = \max_{\|\boldsymbol{x}\|_1=1} \|\boldsymbol{M}\boldsymbol{x}\|_p = \max_{\|\boldsymbol{x}\|_1=1} \left\| \sum_j \text{col}_j(\boldsymbol{M}) \boldsymbol{x}_j \right\|_p \leqslant \max_{\|\boldsymbol{x}\|_1=1} \sum_j |\boldsymbol{x}_j| \cdot \|\text{col}_j(\boldsymbol{M})\|_p \tag{58}$$

$$\leqslant \max_{\|\boldsymbol{x}\|_1=1} \|\boldsymbol{x}\|_1 \cdot \max_j \|\text{col}_j(\boldsymbol{M})\|_p \tag{59}$$

$$= \max_j \|\text{col}_j(\boldsymbol{M})\|_p, \tag{60}$$

by the triangle inequality and Hölder's inequality. But the upper bound in Equation (60) is attained by selecting the column index $j_\star = \arg\max_j \|\text{col}_j(\boldsymbol{M})\|_p$ with the largest norm, then setting $\boldsymbol{x}_{j_\star} = 1$ and the other input components to zero. So $\|\boldsymbol{M}\|_{\ell_1 \to \ell_p} = \max_j \|\text{col}_j(\boldsymbol{M})\|_p$.

Next, let's deal with the $\ell_p \to \ell_\infty$ operator norm. Here we break up a matrix-vector product in terms of the dot product between the vector and the matrix rows:

$$\|\boldsymbol{M}\|_{\ell_p \to \ell_\infty} = \max_{\|\boldsymbol{x}\|_p=1} \|\boldsymbol{M}\boldsymbol{x}\|_\infty = \max_{\|\boldsymbol{x}\|_p=1} \max_i |\boldsymbol{x}^\top \text{row}_i(\boldsymbol{M})| \tag{61}$$

$$= \max_i \max_{\|\boldsymbol{x}\|_p=1} |\boldsymbol{x}^\top \text{row}_i(\boldsymbol{M})| \tag{62}$$

$$= \max_i \|\text{row}_i(\boldsymbol{M})\|_p^\dagger. \tag{63}$$

The proof is completed by recalling that the vector ℓ_p norm is dual to the vector ℓ_q norm for $1/p + 1/q = 1$. In other words, $\|\cdot\|_p^\dagger = \|\cdot\|_{\frac{p}{p-1}}$. ■